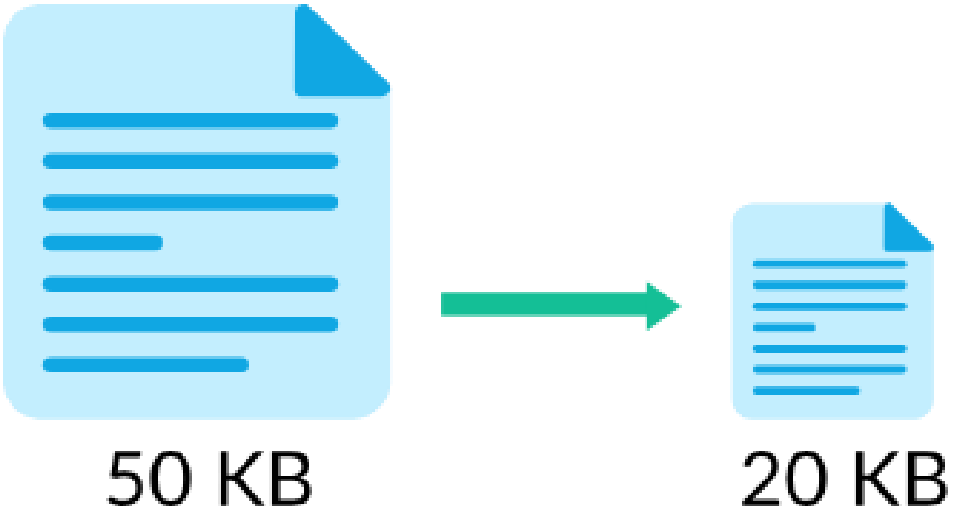
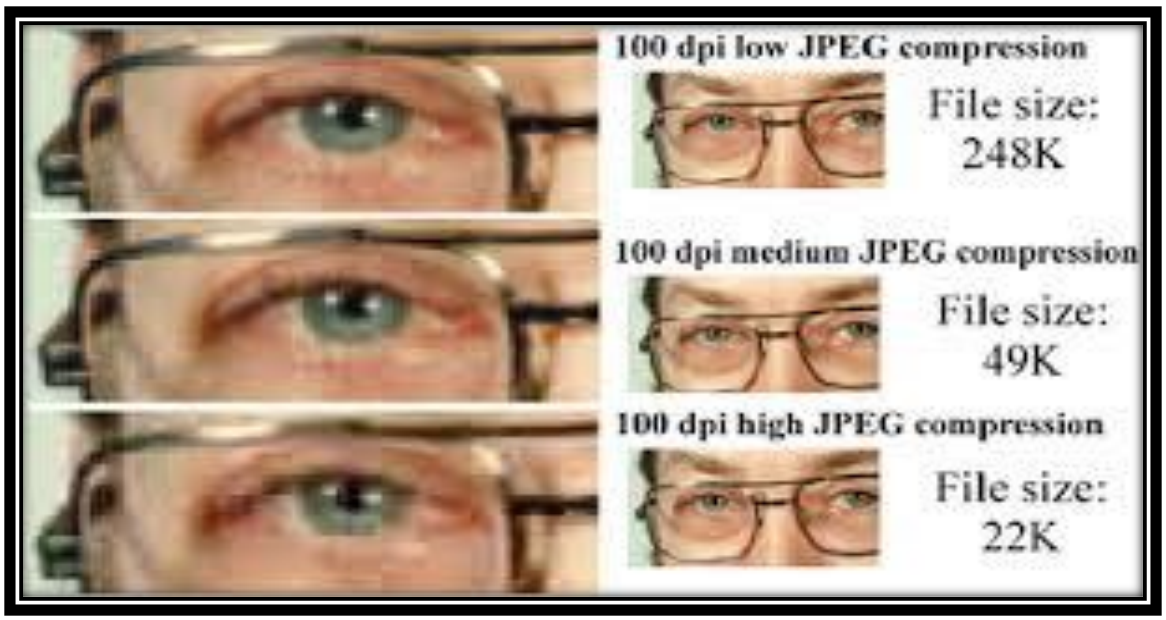
Image Compression

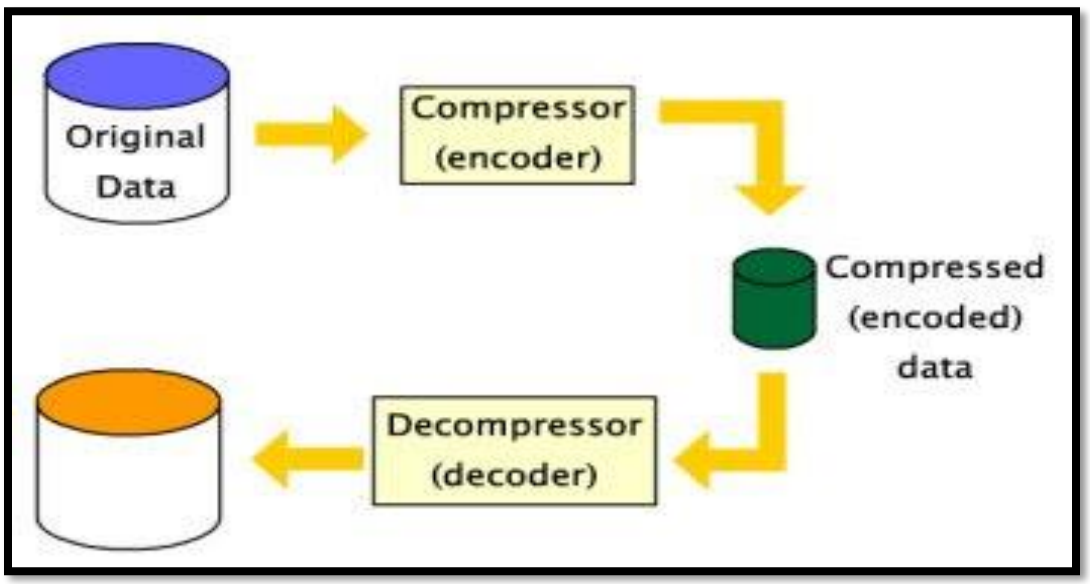
Topics

- **Coding**
- **Interpixel and Psychovisual Redundancy**
- **Image Compression models**
- **Lossless and Lossy Compressions**

Sanjeev Thapa :- RHCSA, RHCE, RHCSA-Openstack, DevOps Engineer, MTCNA,MTCTCE,USRS,3CX CE,HE IPv6 ;)
CCT-BCIS,6th Sem, Spring 2024, maximus.innase@gmail.com


50 KB
20 KB


100 dpi low JPEG compression
File size:
248K
100 dpi medium JPEG compression
File size:
49K
100 dpi high JPEG compression
File size:
22K


Original
Data
Compressor
(encoder)
Compressed
(encoded)
data
Decompressor
(decoder)

Original PNG - 12 MB

Compressed JPEG - 2.5 MB

© TechTerms.com

What is Compression Technology?

- Compression denotes compact representation of data.
- Examples for the kind of data you typically want to compress are e.g.
  - text
  - source-code
  - arbitrary files
  - images
  - video
  - audio data
  - speech
- Obviously, these data are fairly different in terms of data volume, data structure, intended usage etc. For that reason, it is plausible to develop specific compression technologies for different data and application types, respectively.
- The idea of a general-purpose compression technique for all thinkable application scenarios has to be entirely abandoned. Instead, we find a large number of very different techniques with respect to target data types and target application environments (e.g. data transmission in networks like streaming, storage and long-term archiving applications, interactive multimedia applications like gaming etc.).

- Given this vast amount of different techniques, there are different ways how to classify compression techniques:
    - with respect to the type of data to be compressed
    - with respect to the target application area
    - with respect the fundamental building blocks of the algorithms used.

Consider you have a 48-megapixel camera on a mobile phone to capture an image, how much storage space would it require? If you have 512 MB disk, how many images will be stored? Now you share this image via internet of 5 Mbps upload, how much time would it take?

Solution

A 48-megapixel image has 48,000,000 pixels. (may be 8000 * 6000) Total number of bits = 48,000,000 pixels × 3 bpp = 144,000,000 bits = 144,000,000 bits / 8 = 18 000 000 bytes = 18 000 000 bytes / 1024 * 1024 = 17.18 MB
Number of images = 512 MB / 17.18 = 29.8 = 29 images
Convert the file size from megabytes to megabits: Since 1 byte = 8 bits, File size = 17.18 MB × 8 =137.44 megabits (Mb) Time to upload (sec) = File Size (megabits) / Upload Speed (Mbps) =137.44 Mb / 5 Mbps=27.488 sec = 0.458 min
Conclusion ***Storage space required for one 48-megapixel image at 3 bpp: 17.18 MB.*** ***Number of images that can be stored on a 512 MB disk: 29 images.*** ***Time to upload/download the image via a 5 Mbps connection: 27.49 sec***

Bits per Pixel (bpp)	File Size (MB)	Number of Images on 512 MB Disk	Upload Time (seconds)
3	17.18	29	27.49
8	45.78	11	73.25
24	137.33	3	219.73

Why do we still need compression?

- **Reduced Storage Requirements**: Compression reduces the size of files or data, allowing more efficient use of storage resources. This is particularly valuable in environments where storage space is limited or costly.
- **Faster Transmission**: Smaller file sizes resulting from compression lead to faster transmission times over networks. This is critical for internet speeds, where reducing the amount of data transferred can improve overall performance.
- **Bandwidth Conservation**: Compression helps conserve bandwidth, making it possible to transmit more data within the same bandwidth limitations. This is crucial for telecommunications and network infrastructure efficiency.
- **Cost Savings**: By reducing storage and bandwidth requirements, compression can lead to significant cost savings, both in terms of infrastructure costs and operational expenses.
- **Improved Performance**: Smaller file sizes resulting from compression can lead to quicker loading times and improved performance in applications, websites, and other digital platforms.
- **Archiving and Backup**: Compression facilitates faster backup processes and reduces the amount of storage space needed for archival purposes, making data management more efficient.

Why is it possible to compress data?

- Compression is enabled by statistical and other properties of most data types.

- **Redundancy in Data**: Many types of data contain redundancy, which means there are patterns, repetitions, or predictable elements within the data. Compression algorithms exploit these redundancies to represent the data more efficiently.

- **Statistical Redundancy**: Data often exhibits statistical properties where certain symbols or sequences are more likely to occur than others. Compression algorithms use statistical models to encode frequent symbols with shorter codes and infrequent symbols with longer codes, thereby reducing the average length of the representation.

- **Lossless vs. Lossy Compression**:
  - **Lossless Compression**: Preserves all original data without any loss. It removes redundancy by identifying and eliminating repetitive patterns or using techniques like dictionary-based encoding (e.g., Huffman coding, Lempel-Ziv-Welch).

- **Lossy Compression**: Sacrifices some data quality to achieve higher compression ratios. It is used for media data (images, audio, video) where minor loss in quality may not be noticeable or acceptable (e.g., JPEG for images, MP3 for audio).

- **Entropy and Information Theory**: Entropy measures the average information content of data. Compression algorithms leverage concepts from information theory to identify and remove predictable patterns, thereby reducing entropy and achieving compression.

- **Compression Algorithms**: There are various algorithms designed to exploit different types of redundancies in data:
  - **Dictionary-based methods**: Capture repetitive sequences and replace them with references to a dictionary.
  - **Statistical methods**: Use probability distributions to assign shorter codes to more frequent symbols.
  - **Transform-based methods**: Transform data into a different domain (e.g., Fourier, wavelet) where it can be more efficiently compressed.

- **Advancements in Computing**: Increasing computational power allows for more sophisticated compression algorithms and faster processing, making compression feasible and practical for large datasets and real-time applications.

Definition: *"Image compression deals with reducing the amount of data required to represent a digital image by removing of redundant data".*

A standard definition (SD) TV movie using 720 x 480 x 24-bit pixel arrays, with 30 fps, so what SD digital video data must be accessed? Adjust the data for compression if needed.

Solution

Number of bytes per frame = width × height × bytes per pixel

= 720 × 480 × 3 = 1,036,800 bytes/frame

Calculate the data rate = Number of bytes per frame × frames per second

= 1,036,800 × 30=31,104,000 bytes/second

Calculate the total data = Data rate × seconds per hour × hours

(for a two-hour movie) = 31,104,000 bytes/second × 3600 seconds/hour × 2 hours

= 31,104,000 × 7200 = 223,948,800,000 bytes

Total data in GB = 223,948,800,000 / 1024 * 1024 * 1024 = 208.51 GB

Given the substantial data requirements for standard definition (SD) TV movies, using image and video compression techniques can significantly reduce the amount of data needed, making storage and transmission more efficient.

Uncompressed Data Requirements

- **Resolution**: 720 x 480 pixels
- **Color Depth**: 24 bits per pixel (3 bytes per pixel)
- **Frame Rate**: 30 frames per second (fps)
- **Duration**: 2 hours (7200 seconds)

Applying Compression Techniques

Lossless Compression:

- Techniques like Huffman coding and Run-Length Encoding (RLE) can be used to reduce data size without losing any information.
- Typical compression ratios for lossless compression range from 2:1 to 3:1.

- **Estimated Data with Lossless Compression**:
  - Lossless Compression (2:1) ≈208.51 / 2 = 104.28 GB
  - Lossless Compression (3:1) ≈ 208.51 / 3 = 69.5 GB

Lossy Compression:

- Techniques like JPEG (for images) and MPEG (for video) offer higher compression ratios by removing some data, which may result in some loss of quality.
- Typical compression ratios for lossy compression can range from 10:1 to 50:1, depending on the quality requirements.
- **Estimated Data with Lossy Compression**
  - Lossy Compression (10:1) ≈ 208.51 / 10 = 20.851 GB
  - Lossy Compression (20:1) ≈ 208.51 / 20 = 10.425 GB
  - Lossy Compression (50:1) ≈ 208.51 / 50 = 4.1702 GB

Conclusion

- By utilizing compression techniques, both lossless and lossy, the data requirements for a two-hour SD TV movie can be significantly reduced.
- This makes it feasible to store and transmit video data more efficiently.

  - **Without Compression**: 208 GB
  - **With Lossless Compression (2:1 - 3:1)**: 69 - 104 GB
  - **With Lossy Compression (10:1 - 50:1)**: 20.8 – 4.17 GB

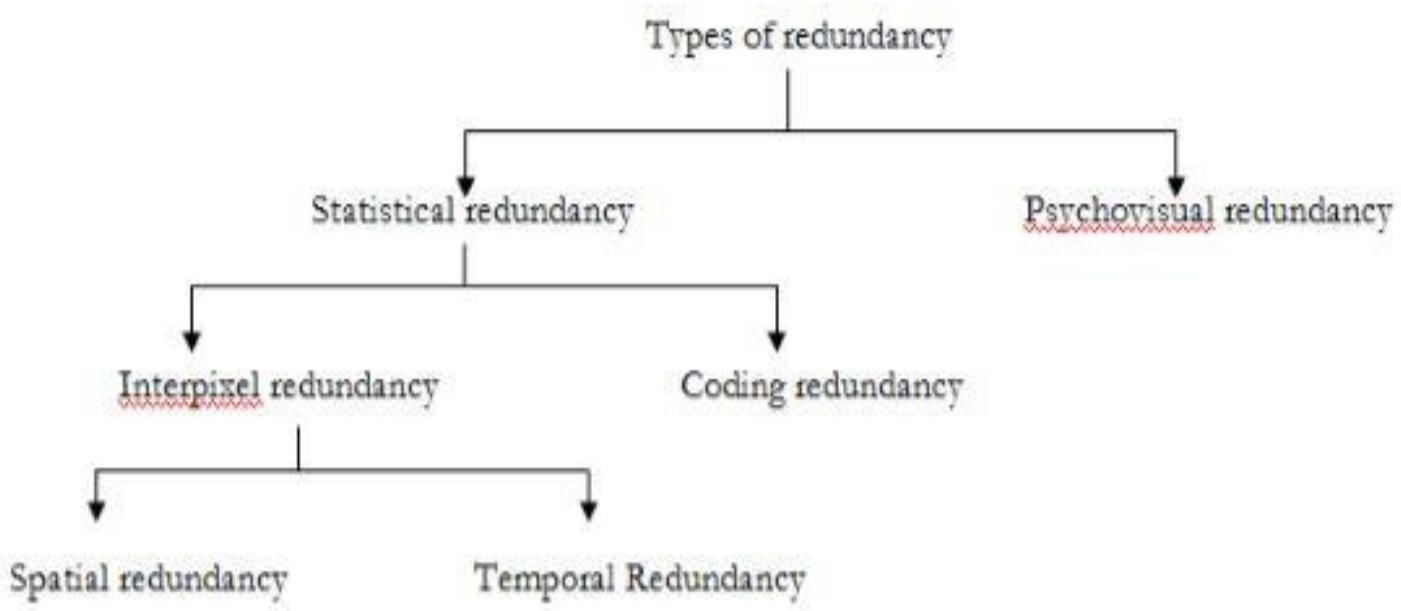
- This reduction in data size is critical for practical applications, ***allowing for more efficient use of storage space and bandwidth while maintaining acceptable levels of quality for various use cases.***

Fundamentals:

- Data computer refers to the process of reducing the amount of data required to represent a given quantity of information.
- If we have b & b' denote the no. bits is two representation of the same information, the relative data redundancy R of representation with b bits is R = 1 – (1 / C)
- where C is the compression representation, & C = b / b'
- Eg, c=10 (written as 10 : 1), the larger representation has 10 bits of data for every 1 bit of data in the smaller representation, i.e, for the larger representation, R=0.9 or the 90% of the data is redundant.

Fundamentals of visual data compression/ Types of Redundancy

- **Redundancy** in the context of image compression refers to the presence of unnecessary or repetitive information within image data that does not contribute to the perceptual quality of the image.
- Reducing redundancy is a key goal in image compression, as it allows for a more efficient representation of the image, thereby reducing the file size without significantly affecting the visual quality.



- **Statistical Redundancy:** Exists within the data itself, based on statistical patterns.
  - **Inter-pixel Redundancy:** Correlation between neighboring pixels.
    - **Spatial Redundancy:** Correlation between pixels in the same image frame (e.g., smooth color gradients)
    - **Temporal Redundancy:** Correlation between pixels in adjacent frames of a video sequence.
  - **Coding Redundancy:** Inefficient ways of representing the data.

- **Psychovisual Redundancy:** Information that the human visual system is less sensitive to.
  - Exploited in lossy compression to remove data with minimal impact on perceived quality.

Statistical Redundancy: This redundancy exists due to predictable patterns and statistical regularities within the data.

- **Example**: In an image with a blue sky, there may be many consecutive pixels with similar color values due to the smooth gradient. This correlation can be exploited to reduce the amount of data needed to represent the image.

Inter-pixel Redundancy: Because of the pixels of most, 2-D intensity arrays are correlated spatially (i.e., each pixel is similar or dependent on neighboring pixels), information is unnecessarily replicated in the correlated pixels. In a video sequence, temporally correlated pixels (nearly frames) also duplicate information. This refers to correlations between neighboring pixels in an image or video.

- **Spatial Redundancy**: Refers to correlations between pixels within the same image frame.
  - **Example**: In a photograph of a clear blue sky, adjacent pixels are likely to have similar color values, as the sky's color changes gradually.

- **Temporal Redundancy**: Refers to correlations between pixels in successive frames of a video sequence.
  - **Example**: In a video of a person walking, the background may remain largely unchanged between consecutive frames, leading to redundancy that can be exploited for compression.

Coding Redundancy: A code is a system of symbols (letters, numbers, etc.) used to represent a body of information. Each piece of information is assigned a sequence of code symbols, called a **code word**. The number of symbols in each code word is its length. [eg 8 bit codes]. This redundancy arises from inefficient methods of representing data.

- **Example**: Using 8 bits (256 levels) to represent an image when the actual number of distinct colors is far fewer. This inefficiency can be minimized using variable-length coding techniques like Huffman coding to assign shorter codes to frequently occurring symbols.

Psychovisual Redundancy: This redundancy involves aspects of data that the human visual system is less sensitive to.

- **Example**: In an image with fine details, removing very subtle color variations that are imperceptible to the human eye can significantly reduce file size without noticeable loss in quality. This is often used in JPEG compression where high-frequency components are quantized more aggressively.

Rahul Basyal wishes to download the Spiderman movie in 720p (1280 x 720) format. The movie is of 2 hours duration and was recorded at 60 fps (frames per second).

1. **Calculate the amount of storage it would require to store this movie.**
2. **How many dual-layer Blu-ray disks would be required to store this movie?**
3. **How much time would it take to transmit this video over a broadband connection having a bandwidth capacity of 100 Mbps?**

Solution

Given Information:

- **Resolution**: 1280 x 720 pixels
- **Duration**: 2 hours
- **Frame rate**: 60 frames per second (fps)
- **Color depth**: 24 bits per pixel (3 bytes per pixel)

Calculate the amount of storage required to store this movie.

Number of bytes per frame = width × height × bytes per pixel
= 1280×720×3=2,764,800 bytes/frame

Data rate = Number of bytes per frame × frames per second
= 2,764,800×60
=165,888,000 bytes/second

Total data = Data rate × seconds per hour × hours
(two-hour movie) = 165,888,000 × 3600 × 2
= 1,194,393,600,000 bytes
= 1,194,393,600,000 / 1,024 * 1,024 * 1,024
= 1,112.38 GiB

*(1 GiB = 1,024 * 1,024 * 1,024 bytes)*

How many dual-layer Blu-ray disks would be required to store this movie?

Assume: If Storage capacity of a single dual-layer Blu-ray disk: 50 GB

Number of disks = Total data in GB / Storage capacity per disk
= 1,112.38 GiB / 50GiB per disk
= 22.24 disk
= 22 disk

Since you can't use a fraction of a disk, you would need **22 dual-layer Blu-ray disks** to store the movie.

How much time would it take to transmit this video over a broadband connection having a bandwidth capacity of 100 Mbps?

Total data (Mb) = 1112.38 GB * (8*1024) = 9,112,616.96 MB
Transmission time = Total data in Mb / Bandwidth in Mbps
= 9,112,616.96 / 100
= 91,126.1696 sec
= 91,126.1696 / 3600 hrs
= 25.31 hrs

Summary of the results:

1. Amount of storage required**: 1112.38 GB**
2. Number of dual-layer Blu-ray disks required**: 22 disks**
3. Time to transmit the video over a 100 Mbps broadband connection**: 25.31 hours**

Compression	**Ratio**	**Transmission time**	**Storage Size**
H.264 (AVC)	1112.38/7	3.79 hrs	170.63 GB
H.265 (HEVC)	15	1.77	79.63 GB
VP9	15	1.77	79.63 GB
AV1	20	1.33	59.72 GB

Conclusion: **Best Compression Technique**: **AV1** is recommended due to its superior compression efficiency and future-proof characteristics. It results in significantly reduced storage requirements and transmission times while maintaining high video quality.

Coding Techniques:

A Error-Free Compression
1. Variable-Length Coding
    1. Huffman Coding
    2. Other Near Optimal Variable Length Codes
    3. Arithmetic Coding
2. LZW Coding
3. Bit-Plane Coding
    1. Bit-Plane Decomposition
    2. Constant Area Coding
    3. One-Dimensional Run-Length Coding
    4. Two-Dimensional Run-Length Coding
4. Lossless Predictive Coding

B. Lossy Compression
1. Lossy Predictive Coding

A. Error-Free Compression:

- Devise an alternative representation of the image in which its inter pixel redundancies are reduced.
- Code the representation to eliminate the coding redundancies.

A1. Variable-Length Coding:

- Reduces only coding redundancy.
- Assign the shortest possible code words to the most probable gray levels.

A1.1. Huffman Coding

- Yields the smallest possible number of code symbols per source symbol.
- Creates a series of source reductions by ordering the probabilities of the symbols.
- Combines the lowest probability symbols into a single symbol that replaces them in the next source reduction.
- Codes each reduced source, starting with the smallest source and working back to the original source.
- This operation is repeated for each reduced source until the original source is reached.

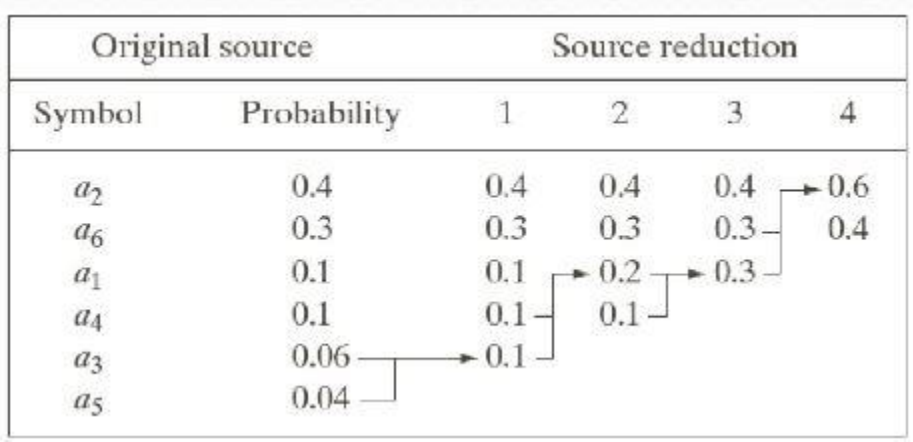


Fig Huffman Source Reduction

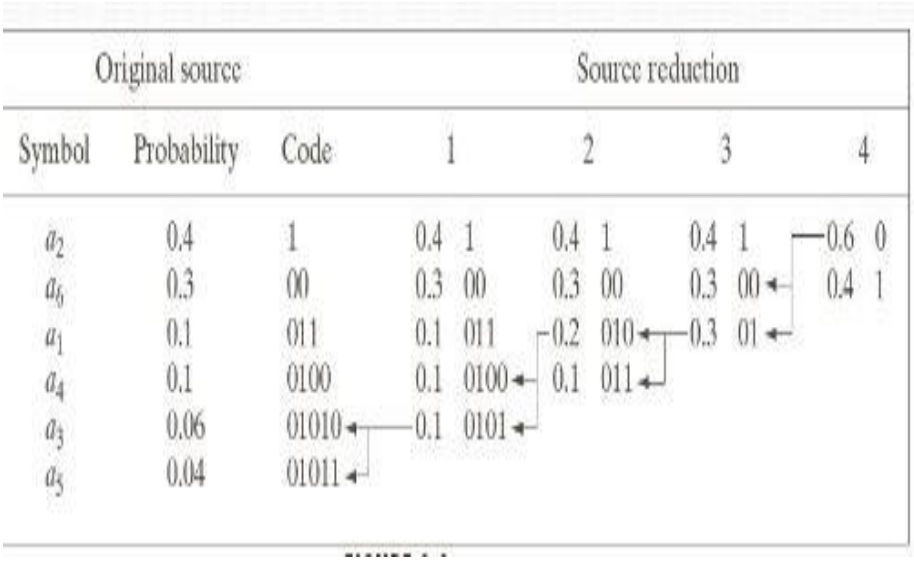


Fig: Huffman Code assignment procedure.

Huffman's procedure creates optimal code which is an instantaneous uniquely decodable block code.

- ***Block code*** – each source symbol is mapped into a fixed sequence of code symbols.
- ***Instantaneous*** – each code word can be decoded without referencing succeeding symbols.
- ***Uniquely decodable*** – any string of code symbols can be coded in only one way.

HUFFMAN CODING

Say, Image size: 10 x 10 (5 bit image)
Frequency:
a2 = 40 a6 = 30 a1 = 10 a4 = 10 a3 = 6 a5 = 4
P(a2) = 40/100 = 0.4

Symbols (like intensity levels)	Probabilities (sorted)	Source Reduction (do till two values are left) (Maintain in sorted order here as well)			
		1	2	3	4
a2	0.4	0.4	0.4	0.4	0.6
a6	0.3	0.3	0.3	0.3	0.4
a1	0.1	0.1	0.2	0.3	
a4	0.1	0.1	0.1		
a3	0.06	0.1			
a5	0.04				

Symbols (like intensity levels)	Probabilities (sorted)	Source Reduction (do till two values are left) (Maintain in sorted order here as well)			
		1	2	3	4
a2	0.4 1	0.4	0.4	0.4	0.6 0
a6	0.3 00	0.3	0.3	0.3 00	0.4 1
a1	0.1 011	0.1	0.2 010	0.3 01	
a4	0.1 0100	0.1 0100	0.1 011		
a3	0.06 01010	0.1 0101			
a5	0.04 01011				

Encoded String: 010100111100
Decoding : a3 a1a2 a2 a6
Parameters:
1. Average length of code
   Lavg = 0.4 * 1 + 0.3 * 2 + 0.1 * 3 + 0.1 * 4 + 0.06 * 5 + 0.04 * 5 = 2.2 bits/symbol
2. Total no. of bits to be transmitted
   10 * 10 * 2.2 = 220 bits
3. Entropy = 2.1396
4. How much you saved = $\frac{10 * 10 * 5 - 10 * 10 * 2.2}{10 * 10 * 5} = 0.56 = 56\%$

Average Length of Code:

$$Lavg=(0.4\times1)+(0.3\times2)+(0.1\times3)+(0.1\times4)+(0.06\times5)+(0.04\times5)=2.2 \text{ bits/symbol}$$

Entropy Calculation

$$H = -\sum_i \; p(x_i) \;\; \log p(x_i)$$

$H=-[0.4\log 0.4 \; +0.3\log 0.3 \; +0.1\log 0.1 \; +0.1\log 0.1+0.06\log 0.06+0.04\log 0.04]$

$=-[0.4\cdot(-1.32193)+0.3\cdot(-1.73697)+0.1\cdot(-3.32193)+0.1\cdot(-3.32193)+0.06\cdot(-4.39232)+0.04\cdot(-4.64386)]$

$=-[-0.528772-0.521091-0.332193-0.332193-0.263539-0.185755]$
$= 2.163543$

Efficiency η = H / Lavg =2.164 / 2.2 = 0.9836

Total bits required:

The average length of code Lavg is 2.2 bits/symbol.

Let's encode a message with the symbols and calculate the number of bits used.

Assume the message consists of the symbols in the same proportions as their probabilities over 100 symbols:

- a2a2a2: 40 symbols
- a6a6a6: 30 symbols
- a1a1a1: 10 symbols
- a4a4a4: 10 symbols
- a3a3a3: 6 symbols
- a5a5a5: 4 symbols

Using the Huffman codes, calculate the total number of bits:

Bits for a2=40×1=40 bits
Bits for a6=30×2=60 bits
Bits for a1=10×3=30 bits
Bits for a4=10×4=40 bits
Bits for a3=6×5=30 bits
Bits for a5=4×5=20 bits

Total bits required: 40+60+30+40+30+20= 220 bits

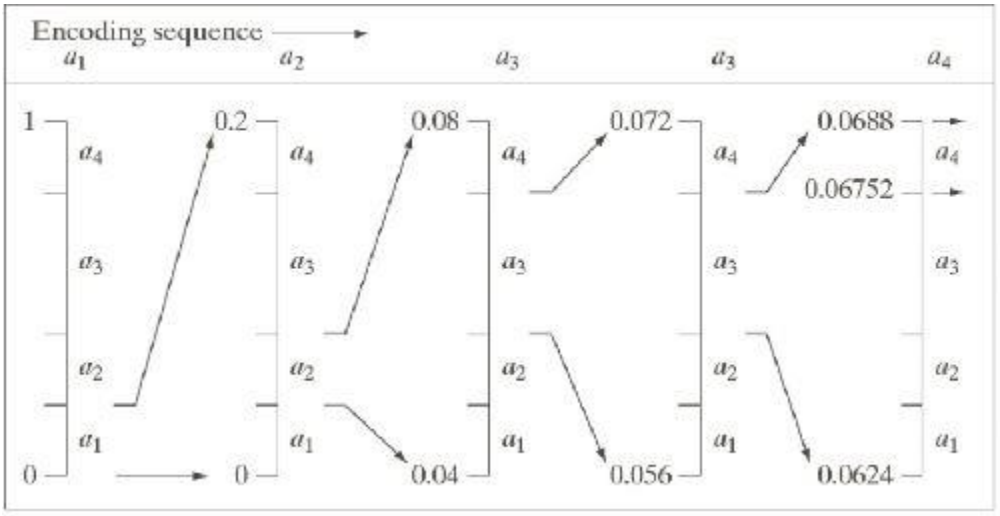
A.1.3 Arithmetic Coding

- Generates non-block codes.
- An entire sequence of source symbols is assigned a single arithmetic code word.
- The code word defines an interval of real numbers between0 and 1.
- As the number of symbols in the message increases, the interval used to represent it becomes smaller and the number of information units required to represent the interval becomes larger.

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

Fig: Arithmetic Coding Example

Fig : Arithmetic Coding procedure



A.2 Lempel-Ziv-Welch (LZW) Coding :

- Assigns fixed length code words to variable length sequences of source symbols.
- Requires no apriority knowledge of the probability of occurrence of the symbols to be encoded.
- A codebook or dictionary containing the source symbols to be coded is constructed while the data are being encoded.
- An LZW decoder builds an identical decompression dictionary during decoding.

Dictionary Location	Entry
0	0
1	1
⋮	⋮
255	255
256	—
⋮	⋮
511	—

Fig : LZW Coding

Fig : LZW Coding Example

Currently Recognized Sequence	Pixel Being Processed	Encoded Output	Dictionary Location (Code Word)	Dictionary Entry
	39			
39	39	39	256	39-39
39	126	39	257	39-126
126	126	126	258	126-126
126	39	126	259	126-39
39	39			
39-39	126	256	260	39-39-126
126	126			
126-126	39	258	261	126-126-39
39	39			
39-39	126			
39-39-126	126	260	262	39-39-126-126
126	39			
126-39	39	259	263	126-39-39
39	126			
39-126	126	257	264	39-126-126
126		126		

A.3 Bit-Plane Coding

- Reduces intermixed redundancies.
- Decomposes a multilevel image into a series of binary images.
- Compresses each image using one of the binary compression methods.

$$m\text{-}\ bit\ \ grayscale\ \ image$$

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + ... + a_1 2^1 + a_0 2^0$$

A.3.2 Constant Area Coding

- Use special code words to identify large areas of contiguous 1's or 0's.
- Image is divided into blocks of size pxq pixels, which are classified as all white, all black or mixed intensity.
- Most probable or frequently occurring category is assigned 1-bit code word 0, the other two are assigned 2-bit codes 10 and 11.

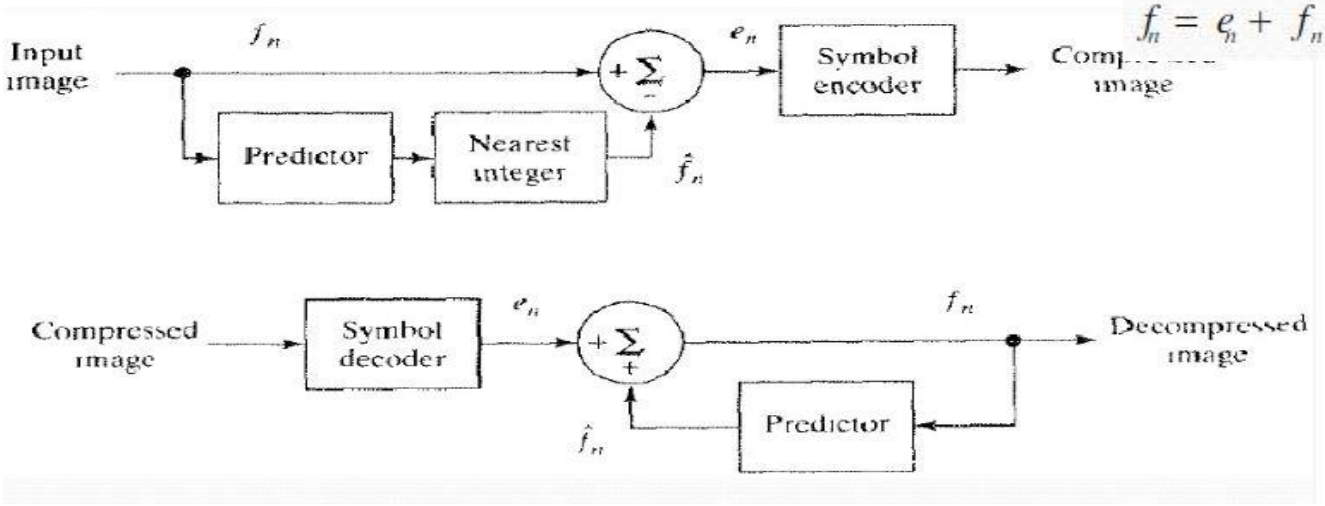
A.3.3 One-dimensional run-length coding

- Represents each row of an image or bit plane by a sequence of lengths that describe successive runs of black and white pixels.

- Code each contiguous group of 0's or 1 's encountered in a left to right scan of a row by its length and specify the value of the first run of each row.
- Black and white run lengths may be coded separately using variable-length codes.

A.4 Lossless Predictive Coding

- Based on eliminating the inter pixel redundancies of closely spaced pixels by extracting and coding only the new information in each pixel.
- The new information of a pixel is defined as the difference between the actual and predicted value of that pixel.
- System consists of an encoder and a decoder each containing an identical predictor. $e_n = f_n - \hat{f}_n$
- The prediction error is coded using variable-length code. $f_n = e_n + \hat{f}_n$



- The decoder reconstructs the error from the received variable-length code words and performs the inverse operation.

- Prediction is usually formed by a linear combination of ***m*** previous pixels.

$$\hat{f}_n = round\left[\sum_{i=1}^{m} \alpha_i f_i\right]$$

In 1-D linear predictive coding,

$$\hat{f}_n(x, y) = round\left[\sum_{i=1}^{m} \alpha_i f(x, y-i)\right]$$

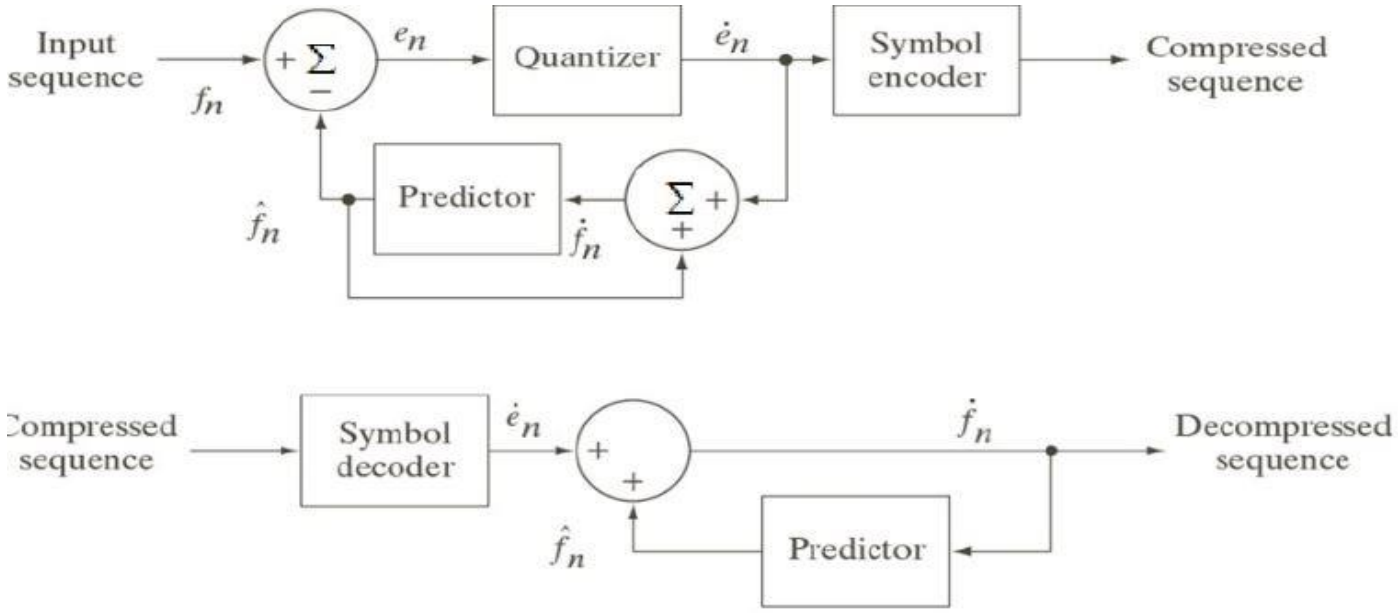
B. Lossy Compression:

- Lossy compression is most commonly used to compress multimedia data (audio, video, and images), especially in applications such as streaming media and internet telephony.
- By contrast, lossless compression is typically required for text and data files, such as bank records and text articles.

- In many cases it is advantageous to make a master lossless file which is to be used to produce new compressed files;
- for example, a multi-megabyte file can be used at full size to produce a full-page advertisement in a glossy magazine, and a 10 kilobyte lossy copy can be made for a small image on a web page.

B.1 Lossy Predictive Coding

- Quantizer absorbs the nearest-integer function of the error- free encoder and is inserted between the symbol encoder and the point at which the prediction error is formed.
- It maps prediction error into a limited range of outputs and establishes the amount of compression and distortion associated with lossy predictive coding.
- The lossy encoder's predictor is placed within a feedback loop, where its input $\mathbf{f_n}$, is generated as a function of past predictions and the corresponding quantized errors.



$$f_n = e_n + \hat{f}_n$$

$$\hat{f}_n = round\left[\sum_{i=1}^{m} \alpha_i \, f_{n-i}\right]$$

- Output of the decoder is also given by $\dot{f}_n = \dot{e}_n + \hat{f}_n$
- Optimum predictors minimize the mean square prediction error,

$$E\{\dot{e}_n^2\} = E\left\{\left[f_n - \hat{f}_n\right]^2\right\}, \quad \text{subject to the constraints}$$

$$\dot{f}_n = \dot{e}_n + \hat{f}_n \approx e_n + \hat{f}_n = f_n \quad \text{and} \quad \hat{f}_n = \sum_{i=1}^{m} \alpha_i \, f_{n-i}$$

A. Continuous Tone Still Image Compression Standards

- **JPEG:** Widely used, adjustable quality levels, lossy compression.
  - Lossy Baseline Coding System
  - Extended Coding System
  - Lossless Independent Coding System
- **JPEG 2000:** Improved quality, wavelet transform, supports both lossy and lossless compression.
- **HEIF:** Modern, efficient, supports advanced features.
- **WebP:** Hybrid compression, supports transparency, efficient for web use.
- **JPEG XR:** HDR support, efficient compression, better quality at lower bitrates.

B. Video Compression Standards

- **Based principally on lossy transform coding techniques.**
- **Original DCT-based JPEG standard**
- **Wavelet-based JPEG 2000 standard**

- **JPEG-LS standard**
- **H.265 / HEVC** (High Efficiency Video Coding)
- **VP8 and VP9:** This video compression standards developed by Google, primarily used in web applications.
- **AV1 (AOMedia Video 1):** AV1 is a newer video compression standard developed by the Alliance for Open Media (AOMedia).
- **MPEG-2**: MPEG-2 is an older video compression standard, still in use for specific applications.
- **H.263:** H.263 is a video compression standard developed for low-bitrate applications.

JPEG

- Three different coding systems:
- *Joint Photographic Experts Group*
- Lossy baseline coding system – based on DCT
- Extended coding system – for higher precision
- Lossless independent coding system for reversible compression
- It is a standardised image compression mechanism. **JPEG** is designed for compressing either **full**-colour (24 bit) or grey-scale digital images of "natural" (real-world) scenes.

JPEG 2000

- Extends the initial JPEG standard to provide increased flexibility.
- Portions of JPEG 2000 compressed image can be extracted for retransmission, storage, display and editing.
- Based on wavelet coding.
- Quantized coefficients are arithmetically coded on bit-plane basis.
- **JPEG** 2000 (JP2) is an image compression standard and coding system. It was created by the Joint Photographic Experts Group committee in 2000 with the intention of superseding their original discrete cosine transform-based**JPEG** standard (created in 1992) with a newly designed, wavelet-based method.