Multimedia Systems
Data Compression

Er. Narayan Sapkota, M.Sc.
narayan.sapkota@eemc.edu.np

Everest Engineering College (Senior Lecturer)
Kathmandu University (Visiting Faculty)

January 7, 2025

Syllabus

Table of Contents

Data Compression and Coding Fundamentals

- Data compression refers to the process of encoding information using fewer bits than the original representation.
- Compression can be lossless or lossy:
  - Lossless Compression: The original data can be perfectly reconstructed from the compressed data. Examples are: ZIP, PNG
  - Lossy Compression: Some data is lost during compression, making it impossible to recover the exact original data. Examples are: MP3, JPEG

Why We Need Data Compression?

- Storage Efficiency: Reduces the storage space required to store data, allowing more data to be stored.
- Transmission Efficiency: Reduces the amount of data to be transmitted over networks, leading to faster data transfer.
- Cost Efficiency: Reduces the cost of data storage and transmission.
- Real-time Processing: Important for applications like video streaming, telecommunication, and image processing.

Storage Space

- Uncompressed Data Requires Substantial Storage
  - Graphics, audio, and video data require significant storage capacity.
  - Even with modern CD and DVD technology, uncompressed video data is not feasible for storage and transmission.
- High Bandwidth Requirement for Data Transfer: Uncompressed video transfer over digital networks requires extremely high bandwidth for point-to-point communication.
- Necessity of Data Compression: To make multimedia systems cost-effective and feasible, compressed video and audio streams are essential.
- Popular Compression Techniques:
  - JPEG: Widely used for compressing single images.
  - H.263 (p$\times$64): A video compression standard for low-bitrate video. For low-resolution video sequences, often used with audio coding techniques for mobile communications.
  - MPEG: Compression standard for both video and audio.

Coding Requirements (1)

Data Storage Requirements

- Images require significantly more storage than text.
- Audio and video data are even more demanding in terms of storage.
- Transmitting continuous media (audio/video) requires high communication data rates.
- The need for compression becomes evident as data transitions from simple text to full-motion video.
- The following specifications are based on a display window of 640×480 pixels:

Text Representation

Text Medium: Two bytes are used for each 8×8 pixel character.

- Characters per screen page $= \frac{640\times480}{8\times8} = 4,800$ characters
- Storage per screen page $= 4,800 \times 2$ bytes $= 9,600$ bytes $= 9.4$ Kbytes

Coding Requirements (2)

Uncompressed Stereo Audio (CD Quality)

- Sampled at 44.1 kHz, quantized with 16 bits.
- Data Rate: $44,100\,\text{s} \times 16\,\text{bits} \times 2\,(\text{stereo}) = 1,411,200\,\text{bits/s}$
- Required Storage per Second:

$$\frac{1,411,200\,\text{bits/s}}{8\,\text{bits/byte}} \times \frac{1}{1024} = 172,800\,\text{bytes/s} \approx 172\,\text{Kbyte}$$

These examples briefly illustrate the increased demands on a computer system in terms of required storage space and data throughput if still images and in particular continuous media are to be processed.

Increased Demands on Storage and Data Throughput: Processing uncompressed video requires storage in the gigabyte range and buffer space in the

Coding Requirements (3)

megabyte range.

Requirements for Compression Techniques

- Quality: Compressed data should be of high quality after decompression.
- Complexity: The compression technique should be simple enough to be cost-effective.
- Processing Time: The decompression time should be minimal.

Requirements for Dialogue Mode (e.g., video conferencing): End-to-end delay should not exceed 150ms, with compression or decompression adding no more than 50ms.

Requirements for Retrieval Mode (e.g., multimedia databases):

Coding Requirements (4)

- Fast Forward/Rewind: Quick access to data, faster than a CD-DA system.
- Random Access: Access to specific images or audio in less than half a second.
- Decompression: Can be done without interpreting all preceding data for random access.

General Requirements for Both Modes

- Standardized Formats: Support for various frame sizes and video frame rates.
- Variable Data Rates: Audio and video compression should support different data rates based on system conditions.
- Synchronization: Audio and video should be precisely synchronized.
- Cross-System Compatibility: Data must be compatible across systems, especially in distributed multimedia systems.

Source, Entropy and Hybrid Coding (1)

Compression techniques can be categorized into three types: Entropy Coding, Source Coding, and Hybrid Coding

Entropy Coding

- A lossless process, works without loss of data.
- Data is treated as a sequence of digital data values, ignoring its semantics.
- Entropy refers to the measure of uncertainty or randomness in the source data. Higher entropy has more unpredictability and requires more bits to represent it efficiently.
- Examples:
  - Run-Length Coding
  - Huffman Coding
  - Arithmetic Coding

Source, Entropy and Hybrid Coding (2)

- For a discrete random variable, entropy is given by:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Where:

- $H(X)$ is the *entropy* of the random variable X, which represents the uncertainty or the average number of bits needed to represent the variable's values.
- $P(x_i)$ is the probability of occurrence of each possible value x_i of the random variable X.
- $\log_2$ is the base-2 logarithm (since entropy is typically measured in *bits*).

Source, Entropy and Hybrid Coding (3)

Properties of Entropy

- Maximum entropy: When all outcomes are equally likely, the entropy is maximized, since the uncertainty is highest.
- Minimum entropy: If one outcome has probability 1 (i.e., the event is certain), the entropy is zero, indicating no uncertainty.

Source Coding

- Often a lossy process. Can cause data loss, but achieves higher compression ratios.
- Source coding is the process of converting information from a source (such as text, audio, or video) into a more compact form that requires fewer bits for representation.

Source, Entropy and Hybrid Coding (4)

- The goal of source coding is to reduce the redundancy in the data to make it more efficient for storage or transmission, without losing important information.
- Source coding takes into account the semantics of the information being encoded.
- Compression Dependency: The degree of compression depends on the medium being encoded.

For Speech Compression

- Data reduction can be achieved by transforming the signal from the time domain to the frequency domain.
- Formants: Maxima in the voice spectrum that are used for reconstruction. Typically, five formants and the fundamental frequency are sufficient for a good reconstruction.

Source, Entropy and Hybrid Coding (5)

- Challenge: Formant tracking in speech analysis can be problematic.

Image Compression:

- In still images, spatial redundancies are exploited using content prediction techniques.
- Transforming the spatial domain into the frequency domain (e.g., using cosine transform) helps compress data.
- Low frequencies (average color) are more important than high frequencies (sharp edges), so they are given more weight.

Examples:

- Prediction(e.g., DPCM, DM)
- Transformation(e.g., FFT, DCT)
- Layred Coding(e.g., Bit Position, Subsampling, Subband Coding)

Source, Entropy and Hybrid Coding (6)

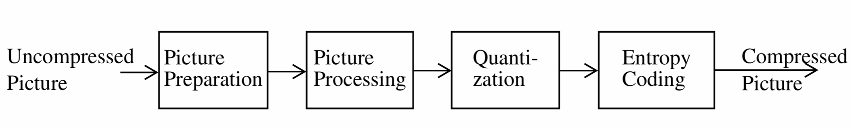
- Vector Quantization

Hybrid Coding

- Combines entropy and source coding, typically without introducing new algorithms. Allowing for more efficient and flexible solutions.
- Examples: JPEG, MPEG, H.263
- Most multimedia systems use hybrid techniques, which are typically combinations of entropy and source coding.
- Hybrid techniques are favored because they offer an optimal balance between compression efficiency and system feasibility.

Major Steps of Data Compression (1)

1. Preparation Step
   - Generates an appropriate digital representation of the medium being compressed.
   - Example: An image may be divided into blocks of 8×8 pixels, each with a fixed number of bits per pixel.



2. Processing Step
   - The first step that applies compression algorithms.
   - Example: Applying Discrete Cosine Transform (DCT) to move from the time domain to the frequency domain.
   - Motion vectors for each 8×8 pixel block can also be calculated in the case of interframe coding.

Major Steps of Data Compression (2)

3. Quantization
   - Quantization simplifies values from the previous step to a specified resolution.
   - This can involve techniques like the μ-law or A-law for audio.
   - The results can be treated differently based on their importance (e.g., using different numbers of bits).
4. Entropy Coding
   - Uses lossless compression to compress the data stream further.
   - Example: Long sequences of zeroes can be compressed by storing the number of occurrences followed by the zero itself.

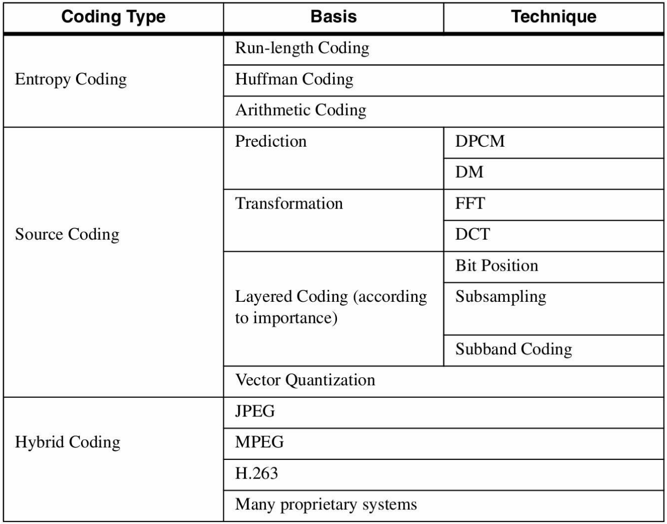
- Iterative Process Picture processing and quantization can be repeated iteratively (e.g., in ADPCM), with feedback or multiple compression techniques applied (e.g., interframe and intraframe coding in MPEG).

Major Steps of Data Compression (3)

- Data Stream After the compression steps, the data is placed in a defined format, which may include the image starting point, compression type, and error correction codes.
- Decompression The inverse of compression, where decoders and coders may differ. Symmetric coding has equal costs for encoding and decoding, while asymmetric coding has a cheaper decoding process, suitable for real-time decompression in applications like audiovisual course modules.

Table of Contents

Basic Data Compression Techniques



Coding Type	Basis	Technique
Entropy Coding	Run-length Coding	
	Huffman Coding	
	Arithmetic Coding	
Source Coding	Prediction	DPCM
		DM
	Transformation	FFT
		DCT
	Layered Coding (according to importance)	Bit Position
		Subsampling
		Subband Coding
	Vector Quantization	
Hybrid Coding	JPEG	
	MPEG	
	H.263	
	Many proprietary systems	

Run-Length Coding

- Run-length coding is used to compress data with repeated sequences of identical bytes. It works by reducing the consecutive repeated characters (runs) into a single character and the number of times it repeats. If a character does not repeat, it is stored as is.
- A special marker, denoted as the `M-byte`, is used to indicate the start of a run-length sequence. The `M-byte` should not occur within the data itself. For example, we define the exclamation mark (`!`) as the `M-byte`. If two consecutive exclamation marks are encountered, it is interpreted as an exclamation mark within the data, not a marker.
- Example: Uncompressed:`ABCCCCCCCCDEFGGG` and Coded:`ABC!8DEFGGG`
- In the example, the character `C` occurs 8 times in a row and is compressed to `C!8`, indicating that `C` repeats 8 times.
- The run-length encoding only applies when there are at least 4 consecutive identical bytes.

Statistical Coding

- Statistical coding is a compression technique where characters are not coded with a fixed number of bits.
- The principle is based on the frequency of character or byte occurrences.
- Frequently occurring characters are coded with shorter bit sequences.
- Rarely occurring characters are assigned longer bit sequences.
- This technique takes advantage of the fact that some symbols appear more often than others in a given dataset.
- Two prominent examples of statistical coding techniques are:
  - Huffman coding
  - Arithmetic coding

Huffman Coding (1)

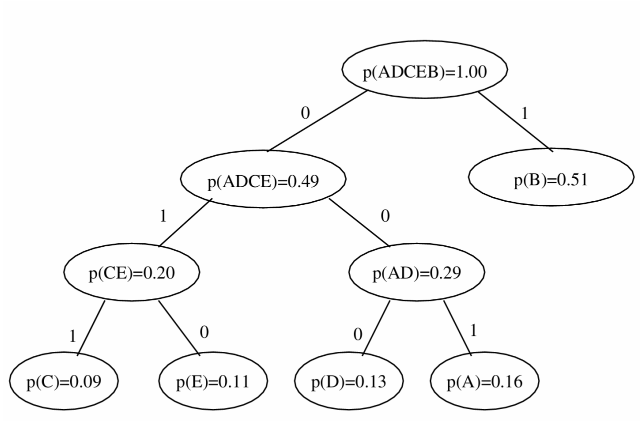
- Huffman coding assigns variable-length code words based on the frequency of characters.
- The most frequent characters receive the shortest code words.
- It minimizes the total number of bits used for encoding a set of characters.
- The algorithm constructs a binary tree based on the relative probabilities of character occurrences.

Steps of Huffman Coding:

1. List characters and their probabilities of occurrence: $p(A) = 0.16, p(B) = 0.51, p(C) = 0.09, p(D) = 0.13, p(E) = 0.11$
2. Combine the two lowest probability nodes: Combine C and E with probability 0.20 and so on.
3. Repeat for the remaining nodes:
    - Combine D and A, $p(AD) = 0.29$.

Huffman Coding (2)

4. Continue combining until only one node remains:



Resulting Huffman Code:

Huffman Coding (3)

- $w(A) = 001$
- $w(B) = 1$
- $w(C) = 011$
- $w(D) = 000$
- $w(E) = 010$

Average Code Length: Entropy :

Arithmetic Coding (1)

- Arithmetic coding is another optimal coding method, similar to Huffman coding.
- Unlike Huffman, arithmetic coding does not encode each symbol separately but considers the entire data stream.
- This approach results in minimal encoded data length, but it does not allow random access (i.e., must read from the beginning).
- Arithmetic coding is especially efficient when symbols occur with very small probabilities.
- It can encode symbols with less than one bit, whereas Huffman coding requires at least one bit per symbol.
- Arithmetic coding assigns fractional bit lengths based on the probability distribution of the symbols.
- Arithmetic coding is more efficient for certain data patterns, such as large areas of uniform color in images.

Arithmetic Coding (2)

- Arithmetic coding is optimal for compression but does not allow random access to the encoded data.
- Huffman coding is simpler and allows faster decoding but may not always be as efficient in some specific cases.

Transformation Coding (1)

- Transformation coding converts data into a mathematical domain that is more suitable for compression.
- The transformation is reversible, meaning that an inverse transformation exists to recover the original data.
- Common transformations
  - Fourier Transform (time to frequency domain)
  - Walsh, Hadamard, Haar, and Slant transforms
- The transformed data may not inherently provide major advantages for compression.
- The most effective transformations for data reduction:
  - Discrete Cosine Transform (DCT)
  - Fast Fourier Transform (FFT)
- Transformation coding is widely used in image and video compression, such as JPEG (using DCT) and MPEG.

Fast Fourier Transform (FFT) (1)

Fourier Transform and Compression

- In general, the Fourier Transform allows a signal to be represented as a sum of sine and cosine waves with different frequencies, amplitudes, and phases.
- The idea is that many types of data (e.g., audio signals, images) contain redundancies that are distributed in the frequency domain.
- By removing frequencies that are less perceptible to the human senses (e.g., in audio compression), or that don't significantly contribute to the signal's overall structure, we can reduce the amount of data required to represent the signal.
- The Fast Fourier Transform (FFT) is a computational technique for efficiently calculating the Discrete Fourier Transform (DFT) of a sequence of data.

Fast Fourier Transform (FFT) (2)

- FFT is useful because it transforms data from the time domain (or spatial domain) to the frequency domain, where data may have a more compact representation, often allowing for better compression.

Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) of a sequence $x[n]$ of length N is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-i\frac{2\pi kn}{N}}, \quad k = 0, 1, 2, \ldots, N-1$$

where:

- $X[k]$ is the Fourier coefficient at frequency k.
- $x[n]$ is the time-domain (or spatial-domain) signal.
- N is the total number of data points.

Fast Fourier Transform (FFT) (3)

- $e^{-i2\pi kn/N}$ is a complex exponential representing the frequency basis functions (sine and cosine waves).
- i is the imaginary unit.

This formula essentially transforms the input signal from the time domain to the frequency domain.

Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) is an optimized algorithm for calculating the DFT efficiently, reducing the computational complexity from $O(N^2)$ (for direct computation) to $O(N \log N)$. This reduction in complexity makes the FFT feasible for real-time applications and large datasets, which is essential in data compression tasks such as image, audio, or video compression.

Fast Fourier Transform (FFT) (4)

Compression Using FFT

- Frequency Domain Representation: The FFT decomposes the signal into frequency components. By analyzing the frequency coefficients, it's often found that most of the information in the data is concentrated in a small number of frequency bins, with the rest contributing little to the overall signal.
- Thresholding: One common approach is to apply a thresholding technique to the frequency coefficients. This means that small coefficients (which correspond to less significant frequencies) can be discarded (set to zero), and only the most significant coefficients are retained.
- Quantization: The retained coefficients can be quantized (compressed further by reducing the precision), leading to even more compression.
- Encoding: Use entropy coding techniques (like Huffman coding or Arithmetic coding) to encode the remaining coefficients.

Fast Fourier Transform (FFT) (5)

The compressed signal can be represented as:

$$\hat{x}[n] = \sum_{k=0}^{N-1} \hat{X}[k] \cdot e^{i2\pi kn/N}$$

where:

- $\hat{X}[k]$ is the compressed Fourier coefficient after thresholding and quantization.
- $\hat{x}[n]$ is the reconstructed signal, which might not be identical to the original but will retain much of the signal's essential information.

Advantages of FFT in Compression

- Efficient Representation: Signals that are sparse in the frequency domain can be compressed more efficiently because only a small number of frequency components are needed to represent the signal.

Fast Fourier Transform (FFT) (6)

- Noise Reduction: High-frequency noise can often be removed in the frequency domain.
- Lossy Compression: Techniques like quantization and thresholding lead to lossy compression, but this is often acceptable in media applications where human perception limits the impact of loss.

Discrete Cosine Transform (DCT) (1)

The Discrete Cosine Transform (DCT) is widely used in signal processing for data compression algorithms like JPEG (for image compression) and MP3 (for audio compression). The DCT transforms a sequence of data from the time (or spatial) domain into the frequency domain. By concentrating the signal's energy in fewer coefficients, it enables efficient compression.

The DCT is closely related to the Discrete Fourier Transform (DFT), but it uses only real-valued cosines instead of complex exponentials. This makes the DCT particularly effective for compressing signals with a lot of redundancy or smoothness, as it provides a compact representation of the data.

The 1D DCT of a sequence of N values $x[n]$ is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right), \quad k = 0, 1, 2, \ldots, N-1$$

Discrete Cosine Transform (DCT) (2)

Where:

- $X[k]$ is the transformed coefficient at frequency index k.
- $x[n]$ is the original data (signal) at index n.
- N is the total number of data points.
- $\cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right)$ is the cosine basis function.

The DCT operates on a set of N input values, $x[n]$, and computes a set of N output values, $X[k]$, which represent the frequency components of the signal.

- Image Compression (JPEG): In JPEG image compression, the image is divided into small blocks (e.g., 8×8 blocks), and the DCT is applied to each block. After transforming the block into the frequency domain, the coefficients are quantized (reducing their precision), and many of the high-frequency components are discarded (since they are often imperceptible to the human eye).

Discrete Cosine Transform (DCT) (3)

- Audio Compression (MP3): For MP3 compression, audio signals are divided into small frames, and the DCT is used to convert them to the frequency domain. The most significant frequency components are retained, and the less important ones are discarded or quantized.

Concepts for Compression

- Energy Compaction: DCT transforms usually pack most of the energy into the first few coefficients, making it easier to compress.
- Thresholding: Small coefficients can be discarded because they contribute very little to the overall signal.
- Quantization: The retained coefficients can be quantized, reducing the precision of the values, which further reduces the data size.

Discrete Cosine Transform (DCT) (4)

- Encoding: The remaining coefficients can be efficiently encoded using methods like Huffman coding or arithmetic coding.

2D DCT for Image Compression

For image compression (like in JPEG), the 2D DCT is applied. For a 2D image with dimensions $M \times N$, the 2D DCT is computed by applying the DCT to each row of the image and then to each column. The formula for the 2D DCT is:

$$X[k_1, k_2] = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} x[n_1, n_2] \cdot \cos\left(\frac{\pi}{M}\left(n_1 + \frac{1}{2}\right) k_1\right) \cdot \cos\left(\frac{\pi}{N}\left(n_2 + \frac{1}{2}\right) k_2\right)$$

Where:

- $X[k_1, k_2]$ is the 2D DCT coefficient at the position (k_1, k_2).

Discrete Cosine Transform (DCT) (5)

- $x[n_1, n_2]$ is the pixel value in the image at position (n_1, n_2).
- M and N are the dimensions of the image.

Table of Contents

JPEG (1)

[1] JPEG (Joint Photographic Experts Group) compression is a widely used method for compressing photographic images, allowing for the reduction of file sizes while maintaining acceptable image quality. It is a *lossy* compression method, meaning some of the image data is discarded to achieve compression. JPEG is used for compressing both color and grayscale still images. It can also be applied to video sequences by quickly encoding and decoding individual still images, a method referred to as Motion JPEG.

1. Requirements

In order to ensure the widespread distribution and application of JPEG, the following requirements were established and fulfilled:

- The standard should be independent of image size.
- It should be applicable to any image aspect ratio and any pixel aspect ratio.

JPEG (2)

- The color space and the number of colors used should be independent of one another.
- Image content may be of any complexity, with any statistical characteristics.
- The JPEG standard should be state-of-the-art (or near) regarding the compression factor and image quality.
- The processing complexity should permit a software solution to run on a large number of available general-purpose processors, and should be drastically reduced with the use of special-purpose hardware.
- Sequential (line by line) and progressive (successive refinement of the whole image) decoding should both be possible. A lossless, hierarchical coding of the same image with different resolutions should also be possible.

JPEG (3)

2. JPEG Overview

- JPEG applications may only require either an encoder or a decoder, depending on the use case.
- The encoded data stream follows a fixed interchange format, which includes encoded image data, parameters, and coding tables for decoding.
- If the encoder and decoder share a common context, an abbreviated format may be used, containing fewer tables.
- In the regular interchange format, all necessary information for decoding is included.
- Figure 1 outlines the fundamental steps of JPEG compression. JPEG defines several image compression modes by selecting different combinations of these steps.

JPEG (4)

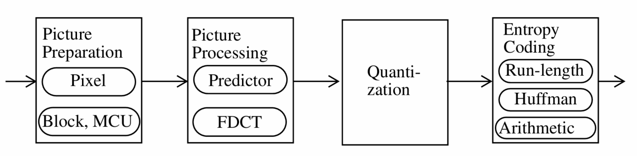


Figure 1: Steps of the JPEG compression technique: summary of the different modes

3. JPEG Modes

JPEG defines four modes, which themselves include additional variations:

- The lossy sequential DCT-based mode (baseline process, base mode) must be supported by every JPEG decoder.
- The expanded lossy DCT-based mode provides a set of further enhancements to the base mode.
- The lossless modes has a low compression ratio and allows a perfect reconstruction of the original image.

JPEG (5)

- The hierarchical mode accommodates images of different resolutions by using algorithms defined for the other three modes.

The baseline process uses the following techniques: block, Minimum Coded Unit (MCU), FDCT, run-length, and Huffman coding.

Image preparation is first presented for all modes. The picture processing, quantization, and entropy encoding steps used in each mode are then described separately for each mode.

[1]Refer Section 7.5 of [1]

JPEG: Image Preparation (1)

1. Image Representation and Components

JPEG supports a flexible image model that allows for varying resolutions, component types, and pixel representations. The model is not tied to specific image characteristics such as the number of pixels (image size) or pixel aspect ratios.

Each image component is a rectangular array of pixels, which can vary in size and resolution. For instance, in color images, there can be multiple components like Y (luminance) and U, V (chrominance), and their resolution might differ.

Common representations include:

- Grayscale images: Usually have a single component (Y).
- RGB images: Have three components (Red, Green, Blue) with the same resolution.

JPEG: Image Preparation (2)

- YUV images with chrominance subsampling: The Y component has higher resolution than the U and V components (often 4:2:2 or 4:2:0 subsampling).

2. Bit Depth and Pixel Precision Each pixel in the image is represented using p bits, where the value ranges from 0 to $2^p - 1$.
JPEG uses a precision of:

- 8 or 12 bits per pixel for lossy compression.
- Between 2 and 12 bits per pixel for lossless compression.

3. Resolution and Subsampling The image resolution is described by the maximum number of pixels in the horizontal (X) and vertical (Y) directions. The horizontal (H) and vertical (V) resolution factors are used to express the relative resolutions between different components. For example:

- Plane 0: $H_0 = 4, V_0 = 1$
- Plane 1: $H_1 = 2, V_1 = 2$

JPEG: Image Preparation (3)

- Plane 2: $H_2 = 1, V_2 = 1$

This means that the second and third planes (usually chrominance components) have lower resolutions compared to the first plane (typically the luminance component).

4. Data Units and Blocks In JPEG compression:

- Lossless mode: Each pixel is treated as a data unit.
- Lossy mode: The image is divided into blocks of 8×8 pixels, because the Discrete Cosine Transform (DCT) operates on blocks of pixels.

Components can be processed sequentially (non-interleaved) or in an interleaved manner (where different components are combined).

5. Interleaved vs Non-Interleaved Data Ordering

JPEG: Image Preparation (4)

- Non-Interleaved: In non-interleaved mode, each component (e.g., Y, U, V) is processed independently. The image is processed one component at a time, which could lead to rendering issues in high-resolution images.
- Interleaved: In interleaved mode, data units of different components are combined into Minimum Coded Units (MCUs). An MCU consists of one data unit from each component. This allows for correct color rendering during decoding, even with partial data.

JPEG: Image Preparation (5)

For images with components of different resolutions, regions of data units are grouped into MCUs.

6. Regions and MCUs In cases where the components have different resolutions, the construction of MCUs becomes more complex. Each component is divided into regions (e.g., 6 regions in the example) that represent sections of the image. The MCU is composed of one region from each component, and within a region, the data units are ordered from top-left to bottom-right.

7. Number of Components JPEG allows for images to have up to 255 components. These components can represent different color channels, such as the RGB channels or YUV/YIQ color models.

JPEG: Image Preparation (6)

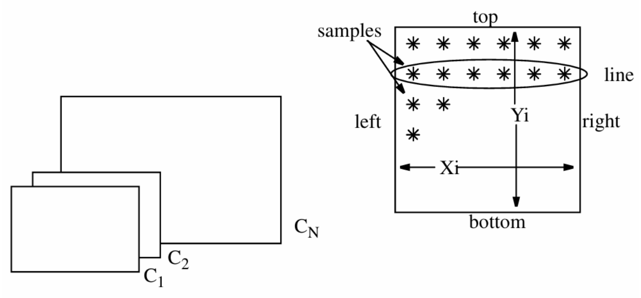


Figure 2: Uncompressed digital image

JPEG: Image Preparation (7)

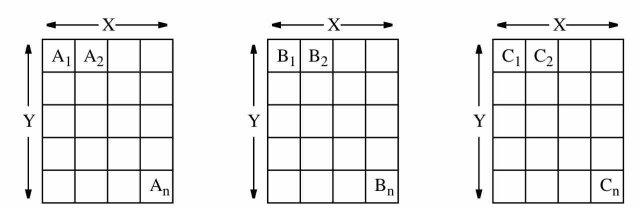


Figure 3: Example of JPEG image preparation with three components having the same resolution

JPEG: Image Preparation (8)

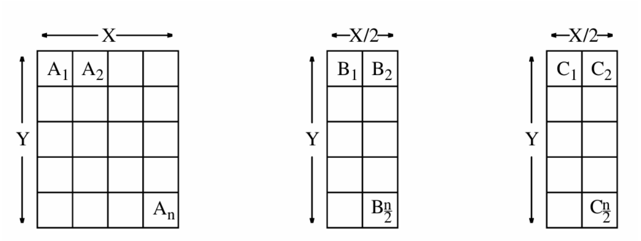


Figure 4: Example of JPEG image preparation with three components having different resolutions

JPEG: Image Preparation (9)

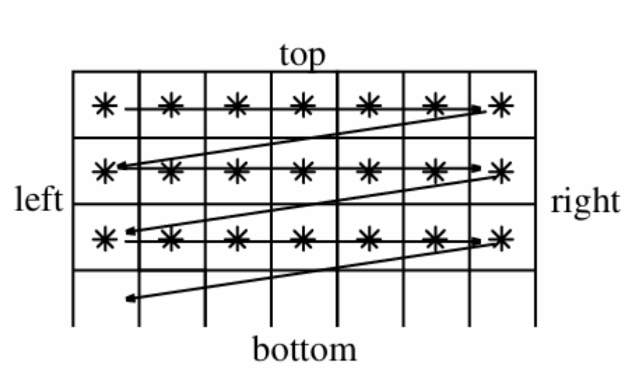


Figure 5: Non interleaved processing order of data units when processing a single component

JPEG: Image Preparation (10)

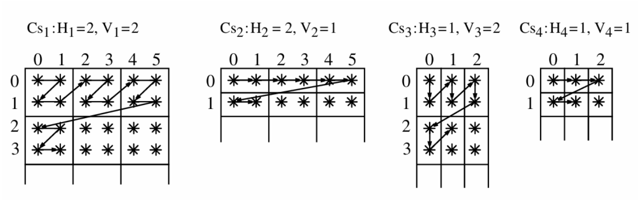


Figure 6: Interleaved processing order of data units

JPEG: Lossy Sequential DCT-Based Mode (1)

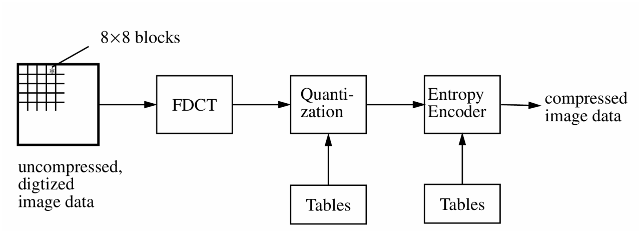


Figure 7: Steps of the lossy sequential DCT-based coding mode, starting with an uncompressed image after image preparation.

1. Input: Uncompressed, Digitized Image Data
   - The input is a digitized image in raw, uncompressed form, represented as pixel intensity values.
   - The image is divided into 8×8 blocks for processing, as shown in the figure.
2. FDCT (Forward Discrete Cosine Transform)

JPEG: Lossy Sequential DCT-Based Mode (2)

- Each 8×8 block is transformed using the Forward Discrete Cosine Transform (FDCT).
- The DCT converts spatial domain data (pixel intensities) into the frequency domain:
  - Low-frequency components, which represent smooth areas, are concentrated in the top-left corner.
  - High-frequency components, which represent details or noise, are distributed across the rest of the block.

3. Quantization
   - The DCT coefficients are quantized using predefined quantization tables.
   - High-frequency components are reduced by dividing them by larger values from the quantization table, followed by rounding.
   - This step introduces *lossiness* by discarding less significant data, which is imperceptible to the human eye.
4. Entropy Encoding
   - After quantization, the remaining coefficients are compressed using entropy encoding techniques such as:

JPEG: Lossy Sequential DCT-Based Mode (3)

  - *Run-Length Encoding (RLE)*: Efficiently encodes consecutive zero values.
  - *Huffman Encoding*: Assigns shorter codes to more frequent values.
- This step is lossless and ensures further reduction in data size.

5. Tables
   - The quantization and entropy encoding tables are essential for decoding the compressed data.
   - These tables can be standardized or customized based on image properties.
6. Output: Compressed Image Data
   - The final output is the compressed image data, which includes:
     - Entropy-encoded coefficients.
     - Metadata such as the quantization table and image dimensions.

Key Characteristics

- Lossy Compression: Quantization discards some data to reduce the file size, introducing minor artifacts.

JPEG: Lossy Sequential DCT-Based Mode (4)

- Block-Based: Processing is done in 8×8 blocks for computational efficiency.
- Human Vision Optimization: Compression exploits the human eye's reduced sensitivity to high-frequency details, retaining low-frequency components.
- Applications: Widely used in JPEG compression to achieve high compression ratios with minimal perceptual quality loss.

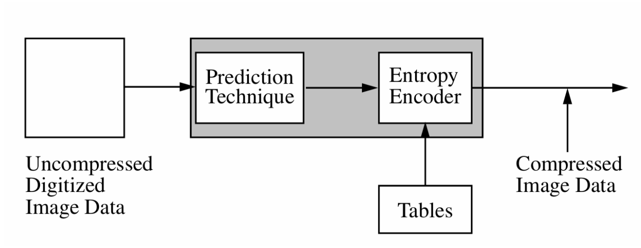
Expanded Lossy DCT-Based Mode (1)

- The Expanded Lossy DCT-Based Mode in JPEG compression refers to an advanced technique that enhances the standard JPEG compression by focusing on greater efficiency while still maintaining its lossy nature.
- This mode leverages the Discrete Cosine Transform (DCT) but allows for more aggressive compression, achieving higher compression ratios with a potentially larger loss in quality.
- In this mode, the image is first transformed using the DCT, which converts spatial domain data (pixel values) into frequency domain data (DCT coefficients).
- These coefficients are then quantized, with more significant coefficients (lower frequencies) preserved while less important ones (higher frequencies) are discarded or heavily reduced.

Expanded Lossy DCT-Based Mode (2)

- The ”expanded” aspect means that additional adjustments are made during quantization and coefficient encoding to exploit further redundancy in the image, allowing more data to be discarded while keeping the image perceptually similar.
- The lossiness comes from quantization, where fine details and high-frequency information are sacrificed for smaller file sizes.
- This mode is particularly effective when aiming for very high compression ratios, especially for applications like web images where file size is critical, and slight degradation in image quality is acceptable.

JPEG: Lossless Mode (1)



Uncompressed Digitized Image Data:

- The input is the raw, uncompressed pixel data of the image, typically represented as intensity or color values.
- Unlike the lossy mode, this process operates directly on the pixel data without converting it into the frequency domain (i.e., no DCT).

Prediction Technique:

JPEG: Lossless Mode (2)

- The core step in lossless JPEG compression involves a prediction-based model.
- Each pixel is predicted based on the values of its neighboring pixels (typically to the left, above, and top-left of the current pixel).
- The prediction technique estimates the value of the current pixel, and the difference (or error) between the actual pixel value and the predicted value (called the residual) is computed.
    - This residual typically has lower entropy (i.e., more zeros or small values), making it easier to compress.

Entropy Encoder:

- The residual values (differences) are fed into an entropy encoder for compression.
- Common entropy encoding techniques include:
    - Huffman Encoding: Assigns shorter codes to frequent residual values.

JPEG: Lossless Mode (3)

  - Arithmetic Encoding: Encodes the entire sequence of residuals more efficiently.
- This step is lossless and ensures further reduction in data size.

Tables:

- Similar to the lossy mode, tables (e.g., Huffman or arithmetic coding tables) are used to guide the entropy encoding process.
- These tables must be stored or transmitted with the compressed image data to enable decoding.

Compressed Image Data:

- The final output consists of the compressed residual values along with any necessary metadata (e.g., prediction mode, coding tables).
- This compressed data can be decompressed later to reconstruct the original image without any loss.

JPEG: Lossless Mode (4)

Key Features of Lossless JPEG Compression

No Data Loss:

- Unlike lossy compression, all pixel values are preserved exactly, ensuring perfect reconstruction of the original image.
- This makes it suitable for applications where precision is critical, such as medical imaging or archival purposes.

Prediction-Based Compression:

- The compression relies on the ability to predict pixel values accurately based on their neighbors.
- A good prediction results in smaller residuals, leading to better compression.

Entropy Encoding:

- The compression efficiency is further enhanced by entropy encoding techniques that exploit the statistical properties of the residuals.

JPEG: Lossless Mode (5)

No Transformations:

- Unlike lossy JPEG, no transformations (e.g., DCT) or quantization steps are used, as they would introduce data loss.

Table of Contents

H.261 (px64) (1)

Recommended Reading

For more details [a]

[a] Refer Section 7.6 of [1]

H.261 is one of the earliest video compression standards, designed for video conferencing and telephony applications over ISDN networks.

Step 1: Input Video Frame

- The video is captured as a sequence of individual frames, representing the motion over time.
- Each frame is divided into Macroblocks (16x16 pixels) to simplify processing and match human visual perception.
- Each macroblock contains:
    - 4 blocks of luminance (Y) and

H.261 (px64) (2)

  - 1 block each of chrominance (U and V) (subsampled in 4:2:0 format).
- Macroblocks are processed independently for intra-frame or inter-frame coding, based on redundancy.

Step 2: Color Space Conversion

- The RGB color format is inefficient for compression due to strong correlations between its components.
- Conversion to YUV color space separates brightness (Y) from color (U, V), taking advantage of the human eye's higher sensitivity to brightness than color.
- Subsampling the U and V components (4:2:0) reduces the amount of chrominance data by 75%, as the human eye is less sensitive to fine color details.

H.261 (px64) (3)

- This step reduces overall data to compress while maintaining visual quality.

Step 3: Motion Compensation

- This step targets temporal redundancy by leveraging the similarity between successive frames in a video.
- The encoder identifies matching macroblocks in the previous frame and calculates a motion vector (displacement in horizontal and vertical directions).
- The motion vector describes how far the macroblock has shifted between frames.
- Instead of encoding the entire macroblock, only the residual (difference) between the predicted macroblock and the actual macroblock is encoded.

H.261 (px64) (4)

- If no suitable match is found, the macroblock is encoded using intra-frame coding (similar to still image compression).
- Motion compensation significantly reduces bitrate, especially for scenes with minimal motion.

Step 4: Discrete Cosine Transform (DCT)

- Each macroblock or residual is transformed from the spatial domain (pixel intensities) to the frequency domain using DCT.
- DCT breaks down the data into:
  - Low-frequency components: Represent large-scale structures like brightness.
  - High-frequency components: Represent fine details or noise.
- Most of the visual information is concentrated in the low-frequency coefficients, while high-frequency coefficients are often negligible.

H.261 (px64) (5)

- This step makes the data more compact and easier to compress.

Step 5: Quantization

- The DCT coefficients are quantized, which involves dividing them by a predefined quantization matrix and rounding the results.
- Low-frequency coefficients (important visual information) are retained with higher accuracy.
- High-frequency coefficients (representing less critical details) are heavily reduced or discarded.
- Quantization reduces precision, enabling compression at the cost of minor visual loss.
- This is the only lossy step in H.261 compression, determining the trade-off between compression efficiency and visual quality.

H.261 (px64) (6)

Step 6: Zig-Zag Scanning

- After quantization, the coefficients are arranged in a zig-zag order, prioritizing low-frequency coefficients (top-left) over high-frequency coefficients (bottom-right).
- Grouped zeros at the end of the sequence are easier to compress using subsequent steps.
- This ordering exploits the natural distribution of energy in the frequency domain.

Step 7: Entropy Encoding

- The reordered coefficients are encoded using entropy encoding to achieve further compression.
- Two common techniques:

H.261 (px64) (7)

- Run-Length Encoding (RLE): Encodes runs of zeros (resulting from quantization) as pairs of values: the length of the run and the next non-zero value.
- Huffman Encoding: Assigns shorter binary codes to frequent symbols (e.g., common residual values or zero runs) and longer codes to less frequent ones.
- These lossless techniques maximize compression by exploiting statistical properties of the data.

Step 8: Multiplexing

- The compressed video data is combined with additional elements to form the final bitstream:
- Motion vectors and prediction mode information for decoding.
- Huffman tables and metadata required to interpret the compressed data.

H.261 (px64) (8)

- Optionally, audio streams (if applicable) are multiplexed for synchronized playback.
- The multiplexed bitstream adheres to the H.261 standard format.

Step 9: Transmission

- The final compressed bitstream is transmitted at bitrates of 64 kbps or multiples of 64 kbps (hence the name px64).
- It is designed to work efficiently within the bandwidth limitations of ISDN-based systems.
- Error resilience mechanisms ensure the stream can handle packet losses or network inconsistencies.

H.261 (px64) (9)

Key Features of H.261 Compression

- Bitrate Control: Supports fixed bitrates (64 kbps to 2 Mbps), making it suitable for ISDN-based systems.
- Redundancy Reduction: Combines intra-frame coding (removing spatial redundancy) and inter-frame coding (removing temporal redundancy).
- Scalability: Offers adjustable compression levels through quantization, enabling trade-offs between quality and bandwidth.
- Efficient Compression: Exploits human visual system limitations (e.g., reduced sensitivity to color and high-frequency details).
- Application: Primarily used for early video conferencing and telecommunication systems.

Table of Contents

MPEG (1)

Recommended Reading

For more detail [a]

[a] Refer Section 7.7 of [1]

MPEG was developed and defined by ISO/IEC JTC1/SC 29/WG 11 to cover motion video as well as audio coding. In light of the state of the art in CD technology, the goal was a compressed stream data rate of about 1.2 Mbit/s. MPEG specifies a maximum data rate of 1,856,000 bit/s, which should not be exceeded. The data rate for each audio channel can be chosen between 32 and 448 Kbit/s in increments of 16 Kbit/s. This data rate enables video and audio compression of acceptable quality. Since 1993, MPEG has been an International Standard (IS).

MPEG (2)

MPEG explicitly takes into account developments in other standardization activities:

- JPEG: Since a video sequence can be regarded as a sequence of still images and the JPEG standard development was always ahead of the MPEG standardization, the MPEG standard makes use of the results of the JPEG work.
- H.261: Since the H.261 standard already existed during the MPEG work, the MPEG group strived to achieve at least a certain compatibility between the two standards in some areas. This should simplify implementations of MPEG that also support H.261. Technically, MPEG is the more advanced technique. Conversely, H.263 borrowed techniques from MPEG.

MPEG (3)

Although mainly designed for asymmetric compression, a suitable MPEG implementation can also meet symmetric compression requirements. Asymmetric coding requires considerably more effort for encoding than for decoding. Compression is carried out once, whereas decompression is performed many times. A typical application area is retrieval systems. Symmetric compression is characterized by comparable effort for the compression and decompression processing, which is required for interactive dialogue applications.

MPEG provides a system definition that describes the combination of individual data streams into a common stream.

MPEG (4)

Video Encoding

Image Preparation

An image must consist of three components: luminance (Y) and two color difference signals (C_r and C_b), similar to the YUV format. The luminance component has twice as many samples in the horizontal and vertical axes as the other components (color subsampling). The resolution of the luminance component should not exceed 768×576 pixels, and the pixel depth is eight bits for each component.

An MPEG data stream also contains information not part of a JPEG-compressed stream, such as the pixel aspect ratio. MPEG supports 14 different pixel aspect ratios. The most important are:

- 1:1 (square pixel): Suitable for most computer graphics systems.
- 16:9 (European HDTV): Defined for a 625-line image.

MPEG (5)

- 16:9 (U.S. HDTV): Defined for a 525-line image.
- 4:3 (702×575 and 711×487 pixel images).

The image refresh frequency is encoded in the data stream. Eight frequencies are defined: 23.976 Hz, 24 Hz, 25 Hz, 29.97 Hz, 30 Hz, 50 Hz, 59.94 Hz, and 60 Hz.

Image Processing

MPEG supports four types of image coding to balance efficient coding and random access:

- I frames (intra-coded pictures): Coded without reference to other frames, using intraframe coding similar to JPEG. These serve as anchors for random access.

MPEG (6)

- P frames (predictive-coded pictures): Require information about previous I and/or P frames for encoding and decoding, achieving a higher compression ratio than I frames.
- B frames (bidirectionally predictive-coded pictures): Use information from previous and following I and/or P frames, yielding the highest compression ratio but cannot serve as reference frames.
- D frames (DC-coded pictures): Contain only low-frequency components for efficient fast-forward.

MPEG (7)

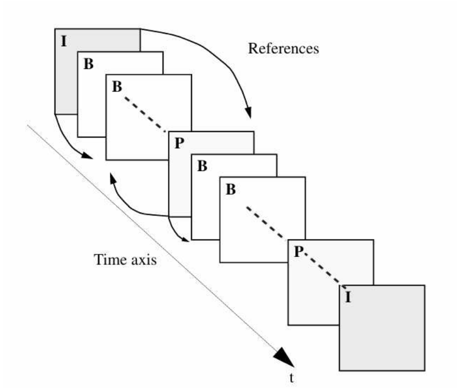


Figure 8: Types of individual images in MPEG: I, B, and P frames

Quantization Quantization adjusts dynamically based on frame type and data rate. If the data rate increases, quantization becomes coarser, and vice versa.

MPEG (8)

Audio Coding

MPEG audio coding is compatible with Compact Disc Digital Audio (CD-DA) and Digital Audio Tape (DAT). Sampling rates of 44.1 kHz, 48 kHz, and 32 kHz at 16 bits per sample are supported. Compression rates vary between 64 and 192 Kbit/s.

MPEG (9)

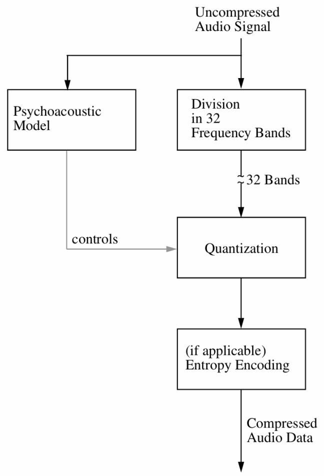

Uncompressed Audio Signal
Psychoacoustic Model
Division in 32 Frequency Bands
~32 Bands
controls
Quantization
(if applicable) Entropy Encoding
Compressed Audio Data

Figure 9: MPEG audio encoding

MPEG (10)

Encoding Layers

Three quality layers are defined:

- Layer 1: Low complexity with fixed bit rates up to 448 Kbit/s.
- Layer 2: Supports bit rates up to 384 Kbit/s.
- Layer 3: Allows variable bit rates and supports Huffman coding.

Audio coding employs the Fast Fourier Transform (FFT) to divide the spectrum into 32 subbands. These are quantized based on psychoacoustic models that assess noise levels. Joint stereo mode further exploits redundancies between channels for higher compression.

Data Streams Audio Stream An audio stream comprises frames made up of audio access units, which are divided into slots. Frames consist of a fixed number of samples, and each audio access unit can be independently decoded. Play time varies based on sampling rates (e.g., 8 ms for 48 kHz). Video Stream A video stream consists of six hierarchical layers:

MPEG (11)

1. Sequence Layer: Handles data buffering and constant bit rate.
2. Group of Pictures Layer: Contains at least one I frame for random access.
3. Picture Layer: Contains an entire still image with temporal reference.
4. Slice Layer: Includes macro blocks and scaling information for quantization.
5. Macro Block Layer: Describes the structure of macro blocks.
6. Block Layer: Defines individual 8×8 pixel blocks.

Table of Contents

DVI (1)

See Textbook for more details

R. Steinmetz and K. Nahrstedt.
Multimedia: Computing, Communications and Applications.
Pearson, 2012.