Multimedia Systems
Lecture – 14

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Types of Video Signals

Composite Video

- Composite video is also called baseband video or RCA video. It is the analog waveform that conveys the image data in the conventional NTSC, PAL and SECAM television signal.
- Composite video contains both chrominance (color) and luminance (brightness) information, along with synchronization and blanking pulses, all together in a single signal.
- This was done to reduce bandwidth and achieve real-time transmission.
- However, in composite video, interference between the chrominance and luminance information is inevitable and tends to worsen when the signal is weak.
- This is why fluctuating colors, false colors, and intensity variations are seen when a distant NTSC television station sends signals that are weak and not properly captured at home with old-fashioned “rabbit ears,” or outdoor “aerial” antennae.

S-Video

- S-Video (*Super-Video*, sometimes referred to as *Y/C Video*) is a video signal transmission in which the luminance signal and the chrominance signal are transmitted separately to achieve superior picture clarity.
- The luminance signal (*Y*) carries brightness information, and the chrominance signal (*C*) carries color information.
- Here, the chrominance signal (*C*) is formed by combining the two chrominance signals *U* and V into one signal along with their respective synchronization data, so at display time, the *C* signal can be separated into *U* and *V* signals.
- Separating the *Y* and *C* channels and sending them separately reduces problems caused by interference between the luminance and chrominance signals and yields a superior visual quality.

Component Video

- Component video strives to go a step further than S-Video by keeping all three *Y, U, V* (or equivalent) components separate.
- Consequently, the bandwidth required to broadcast component video is more than the composite or S-Video and, correspondingly, so is the visual quality.
- The separation of these components prevents artifacts due to intersignal interference.

Connectors for typical analog display interfaces. From left to right:, Composite video, S-video, and Component video

Digital Video

The advantages of digital representation for video are many. It permits

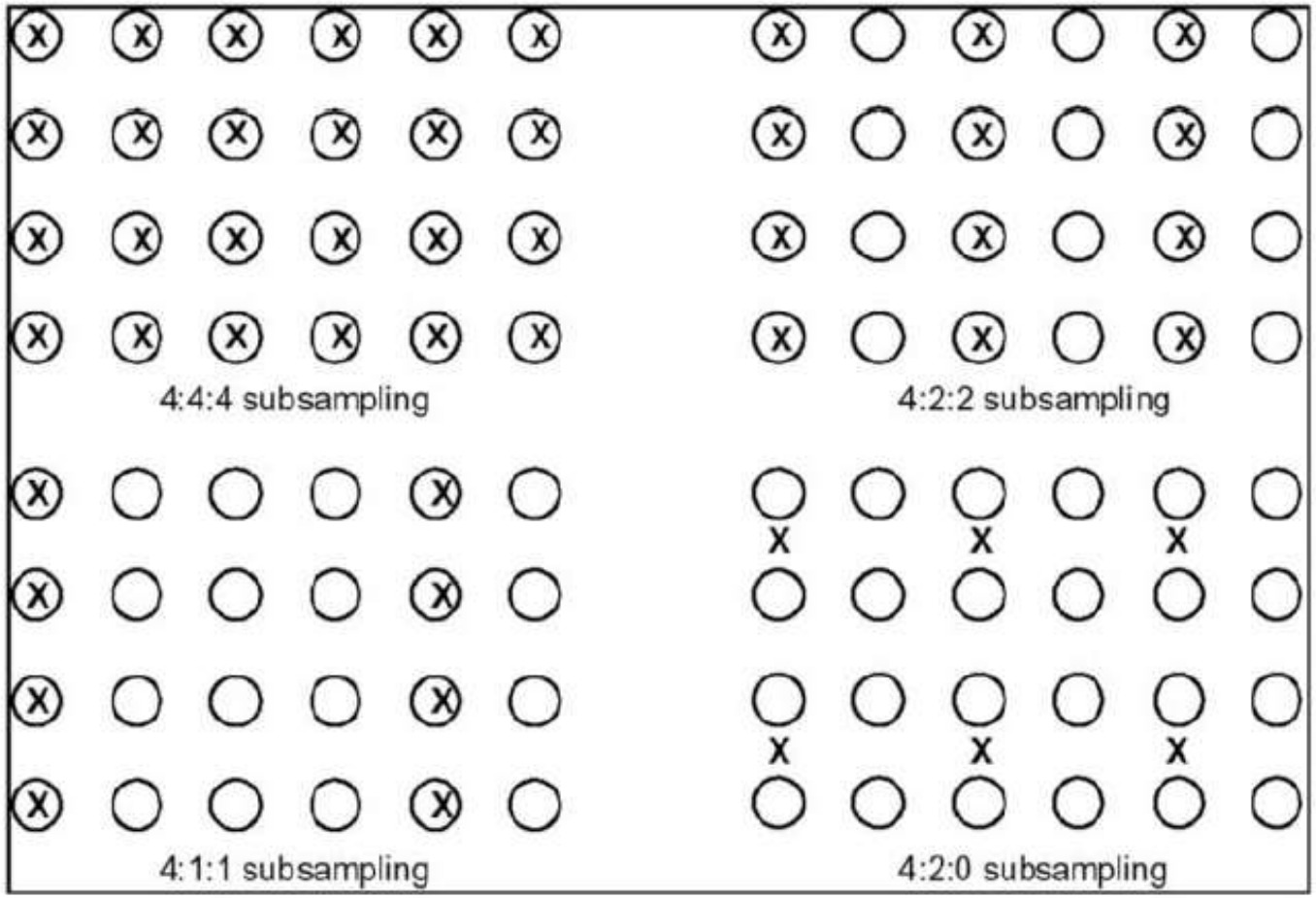
- Storing video on digital devices or in memory, ready to be processed (noise removal, cut and paste, and so on) and integrated into various multimedia applications.
- Direct access, which makes nonlinear video editing simple.
- Repeated recording without degradation of image quality.
- Ease of encryption and better tolerance to channel noise.

YUV Subsampling Schemes

- Video signals captured by digital cameras are represented in the RGB color space, which is also used to render video frames on a display device.
- However, for transmission and other intermediary processing, the YUV space is commonly used.
- The YUV space separates the color and luminance information.
- The color information (UV) is then further subsampled to gain more bandwidth.
- In analog video, subsampling is achieved by allocating half as much bandwidth to chrominance as to luminance.
- In digital video, subsampling can be done by reducing the number of bits used for the color channels on average.

- Depending on the way subsampling is done, a variety of subsampling ratios can be achieved.
- The circles represent pixel information.
- Potentially, we could store 1 byte each for Y, U, and V components, resulting in 24 bits per pixel.
- In subsampling, the luminance component Y is left untouched—that is, 1 byte is reserved for the luminance data per pixel.
- An X at a pixel position suggests that we also store the chrominance components for this position.

YUV subsampling schemes used in video



- In the 4:4:4 scheme, each pixel has luminance (8 bits) and chrominance (8 bits for U and 8 bits for V), resulting in 24 bits per pixel.
- In the 4:2:2 subsampling scheme, chrominance information is stored for every other pixel bringing the equivalent bits per pixel down to 16.
- In the 4:1:1 subsampling scheme, chrominance is stored every fourth pixel in a row.
- Whereas in the 4:2:0 scheme, the average of the U values for a 2×2 pixel area is stored, and similarly for the V values.
- Since there is only 1 U and 1 V sample for every four luminance samples, the equivalent bits per pixel is brought down to 12 bits per pixel.

Multimedia Systems
Lecture – 15

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

CCIR and ITU-R Standards for Digital Video

- CCIR - *Consultative Committee for International Radio*
- One of the most important standards it has produced is *CCIR-601* for component digital video.
- Later renamed as ITU-R Rec. 601 (International Telecommunication Union – Radiocommunication sector), a standard for professional video applications for encoding interlaced analog video signals in digital video form.
- It includes methods of encoding 525-line 60 Hz (NTSC) and 625-line 50 Hz (PAL) signals. The color subsampling used is 4:2:2.
  - The NTSC version has 525 scan lines, each having 858 pixels. Because it uses 4:2:2, each pixel can be represented with two bytes (8 bits for *Y* and 8 bits alternating between *U* and *V*). The Rec. 601 (NTSC) data rate is thus approximately 216Mbps.

- The CIF format (Common Interchange Format) was established for a progressive digital broadcast television.
- It consists of VHS quality resolutions whose width and height are divisible by 8—a requirement for digital encoding algorithms.
- The Quarter Common Interchange Format (QCIF) was established for digital videoconferencing over ISDN lines.
- CIF is a compromise between NTSC and PAL, in that it adopts the NTSC frame rate and half the number of active lines in PAL.
- When played on existing TV sets, NTSC TV will first need to convert the number of lines, whereas PAL TV will require frame rate conversion.

ITU-R digital video specifications

	Rec. 601 525/60 NTSC	Rec. 601 625/50 PAL/SECAM	CIF	QCIF
Luminance resolution	720 × 480	720 × 576	352 × 288	176 × 144
Chrominance resolution	360 × 480	360 × 576	176 × 144	88 × 72
Color subsampling	4:2:2	4:2:2	4:2:0	4:2:0
Aspect ratio	4:3	4:3	4:3	4:3
Fields/sec	60	50	30	30
Interlaced	Yes	Yes	No	No

High-Definition TV (HDTV)

- The usual NTSC analog TV signal in the United States has 525 scan lines, with 480 actually visible. The usual TV has an effective picture resolution of about 210,000 pixels.
- Today, consumers are accustomed to better resolutions such as 1024×768 and even higher, which are now commonly supported by most graphics hardware that come with computers.
- A class of digital television called HDTV supports a higher resolution display format along with surround sound.
- The visual formats used in HDTV are as follows:
  - 720*p*—1280×720 pixels progressive
  - 1080*i*—1920×1080 pixels interlaced
  - 1080*p*—1920×1080 pixels progressive

- They use the MPEG2-based video compression format with a 17 Mbps bandwidth.
- Although HDTV signals can be stored and transmitted effectively using MPEG-2 technology, a lot of bandwidth is required to transmit numerous channels.
- The aspect ratio of HDTV is 16:9 (1.78:1), which is closer to the ratios used in theatrical movies, typically 1.85:1 or 2.35:1.
- The increased resolution provides for a clearer, more detailed picture.
- In addition, progressive scan and higher frame rates result in a picture with less flicker and better rendering of fast motion.

Ultra High Definition TV (UHDTV)

- UHDTV is a new development—a new generation of HDTV.
- The standards announced in 2012 support 4K UHDTV: 2160P (3,840×2,160, progressive scan) and 8K UHDTV: 4320P (7,680 × 4,320, progressive scan).
- The aspect ratio is 16:9. The bit-depth can be up to 12 bits, and the chroma subsampling can be 4:2:0 or 4:2:2.
- The supported frame rate has been gradually increased to 120 fps.
- The UHDTV will provide superior picture quality, comparable to IMAX movies, but it will require a much higher bandwidth and/or bitrate.

Digital Display Interfaces

- Given the rise of digital video processing and the monitors that directly accept digital video signals, there is a great demand toward video display interfaces that transmit digital video signals.

- The most widely used digital video interfaces include
  - Digital Visual Interface (DVI)
  - High- Definition Multimedia Interface (HDMI), and
  - DisplayPort

Connectors of different digital display interfaces. From left to right: VGA, DVI, HDMI, DisplayPort

Digital Visual Interface (DVI)

- Digital Visual Interface (DVI) was developed by the *Digital Display Working Group* (DDWG) for transferring digital video signals, particularly from a computer's video card to a monitor.
- It carries uncompressed digital video and can be configured to support multiple modes, including DVI-D (digital only), DVI-A (analog only), or DVI-I (digital and analog).
- The support for analog connections makes DVI backward compatible with VGA (Video Graphics Array).
- It allows a maximum 16:9 screen resolution of 1,920×1,080 at 60 Hz.

- DVI's digital video transmission format is based on *PanelLink*, a high-speed serial link technology using *transition minimized differential signaling* (TMDS).
- Through DVI, a source, e.g., video card, can read the display's *extended display identification data* (EDID), which contains the display's identification, color characteristics, and table of supported video modes.
- When a source and a display are connected, the source first queries the display's capabilities by reading the monitor's EDID block.
- A preferred mode or native resolution can then be chosen.
- In a single-link mode, the maximum pixel clock frequency of DVI is 165MHz, which supports a maximum resolution of 2.75megapixels at the 60Hz refresh rate.
- This allows a maximum 16:9 screen resolution of 1,920×1,080 at 60 Hz.
- In dual link mode, it can achieve higher resolutions up to 2, 560 × 1, 600 at 60 Hz.

High-Definition Multimedia Interface (HDMI)

- HDMI is a newer digital audio/video interface developed to be backward-compatible with DVI.
- Its electrical specifications, in terms of TMDS and VESA/DDC links, are identical to those of DVI.
- HDMI, however, differs from DVI in the following aspects:
  - HDMI does not carry analog signal and hence is not compatible with VGA.
  - DVI is limited to the RGB color range (0–255). HDMI supports both RGB and YUV 4:4:4 or 4:2:2. The latter are more common in application fields other than computer graphics.
  - HDMI supports digital audio, in addition to digital video.
- The maximum pixel clock rate for HDMI 1.0 is 165MHz, HDMI 1.3 increases that to 340MHz while the latest HDMI 2.0 supports 4K resolution at 60 fps.

DisplayPort

- DisplayPort is the first display interface that uses packetized data transmission, like the Internet or Ethernet.
- Specifically, it is based on small data packets known as *micro packets*, which can embed the clock signal within the data stream.
- DisplayPort can achieve a higher resolution yet with fewer pins than the previous technologies.
- The use of data packets also allows DisplayPort to be extensible.
- DisplayPort can be used to transmit audio and video simultaneously, or either of them.
- It has a much higher video bandwidth, enough for four simultaneous 1080P 60Hz displays, or 4K video at 60 Hz.

- Compared with HDMI, DisplayPort has slightly more bandwidth, which also accommodates multiple streams of audio and video to separate devices.
- It is royalty-free, while HDMI charges an annual fee to manufacturers. These points make DisplayPort a strong competitor to HDMI in the consumer electronics market

Multimedia Systems
Lecture – 16

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

3D Video and TV

- Three-dimensional (3D) pictures and movies have been in existence for decades.
- Increasingly, it is in movie theaters, broadcast TV (e.g., sporting events), personal computers, and various handheld devices.
- The main advantage of the 3D video is that it enables the experience of immersion— be there, and really Be there.
- We will see fundamentals of 3D vision or 3D percept, emphasizing stereo vision (or stereopsis) since most modern 3D video and 3D TV are based on stereoscopic vision.

Cues for 3D Percept

- The human vision system is capable of achieving a 3D percept by utilizing multiple cues.
- They are combined to produce optimal depth estimates.
- When the multiple cues agree, this enhances the 3D percept.
- When they conflict with each other, the 3D percept can be hindered.

Monocular Cues

- The monocular cues that do not necessarily involve both eyes include:
  - *Shading*—depth perception by shading and highlights
  - *Perspective scaling*—converging parallel lines with distance and at infinity
  - *Relative size*—distant objects appear smaller compared to known same-size objects not in distance
  - *Texture gradient*—the appearance of textures change when they recede in distance
  - *Blur gradient*—objects appear sharper at the distance where the eyes are focused, whereas nearer and farther objects are gradually blurred
  - *Haze*—due to light scattering by the atmosphere, objects at distance have lower contrast and lower color saturation
  - *Occlusion*—a far object occluded by nearer object(s)
  - *Motion parallax*—induced by object movement and head movement, such that nearer objects appear to move faster.
- Among the above monocular cues, it has been said that Occlusion and Motion parallax are more effective.

Binocular Cues

- The human vision system utilizes effective binocular vision, i.e., *stereo vision*.
- Our left and right eyes are separated by a small distance, on average approximately 2.5 inches, or 65mm. This is known as the *interocular distance*.
- As a result, the left and right eyes have slightly different views, i.e., images of objects are shifted horizontally.
- The amount of the shift, or *disparity*, is dependent on the object's distance from the eyes, i.e., its *depth*, thus providing the binocular cue for the 3D percept.
- The horizontal shift is also known as *horizontal parallax*.
- The fusion of the left and right images into single vision occurs in the brain, producing the 3D percept.

3D Movie and TV Based on Stereo Vision

- **3D Movie Using Colored Glasses**
- **3D Movies Using Circularly Polarized Glasses**
- **3D TV with Shutter Glasses**

3D Movie Using Colored Glasses

- In the early days, most movie theaters offering a 3D experience provided glasses tinted with complementary colors, usually *red* on the left and *cyan* on the right. This technique is called *Anaglyph 3D*.
- Anaglyph 3D images contain two differently filtered colored images, one for each eye. The left image is filtered to remove Blue and Green, and the right image is filtered to remove Red.
- When viewed through the "color-coded" "anaglyph glasses", each of the two images reaches the eye it's intended for, revealing an integrated stereoscopic image.
- The visual cortex of the brain fuses this into the perception of a three-dimensional scene or composition.
- The Anaglyph 3D movies are easy to produce. However, due to the color filtering, the ***color quality is not necessarily the best***.

Anaglyph 3d glasses

An Anaglyph image

3D Movies Using Circularly Polarized Glasses

- Nowadays, the dominant technology in 3D movie theaters is the RealD Cinema System.
- Movie-goers are required to wear polarized glasses in order to see the movie in 3D.
- Basically, the lights from the left and right pictures are polarized in different directions. They are projected and superimposed on the same screen.
- The left and right polarized glasses that the audience wear are polarized accordingly, which allows one of the two polarized pictures to pass through while blocking the other.
- Circularly polarized glasses are used so the users can tilt their heads and look around a bit more freely without losing the 3D percept.

Polarized 3D systems

3D TV with Shutter Glasses

- Most TVs for home entertainment, however, use *Shutter Glasses*.
- Basically, the liquid crystal layer on the glasses that the user wears becomes opaque (behaving like a shutter) when some voltage is applied. It is otherwise transparent.
- The glasses are actively (e.g., via Infra-Red) synchronized with the TV set that alternately shows left and right images (e.g., 120Hz for the left and 120Hz for the Right) in a Time Sequential manner.
- 3D vision with shutter glasses can readily be realized on desktop computers or laptops with a modest addition of specially designed hardware and software. The NVIDIA GeForce 3D Vision Kit is such an example.

A pair of Crystal Eyes shutter glasses

Multimedia Systems
Lecture – 17

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Audio

- Audio information is crucial for multimedia presentations and, in a sense, is the simplest type of multimedia data.

- However, some important differences between audio and image information cannot be ignored.

- For example, while it is customary and useful to occasionally drop a video frame from a video stream, to facilitate viewing speed, we simply cannot do the same with sound information or all sense will be lost from that dimension.

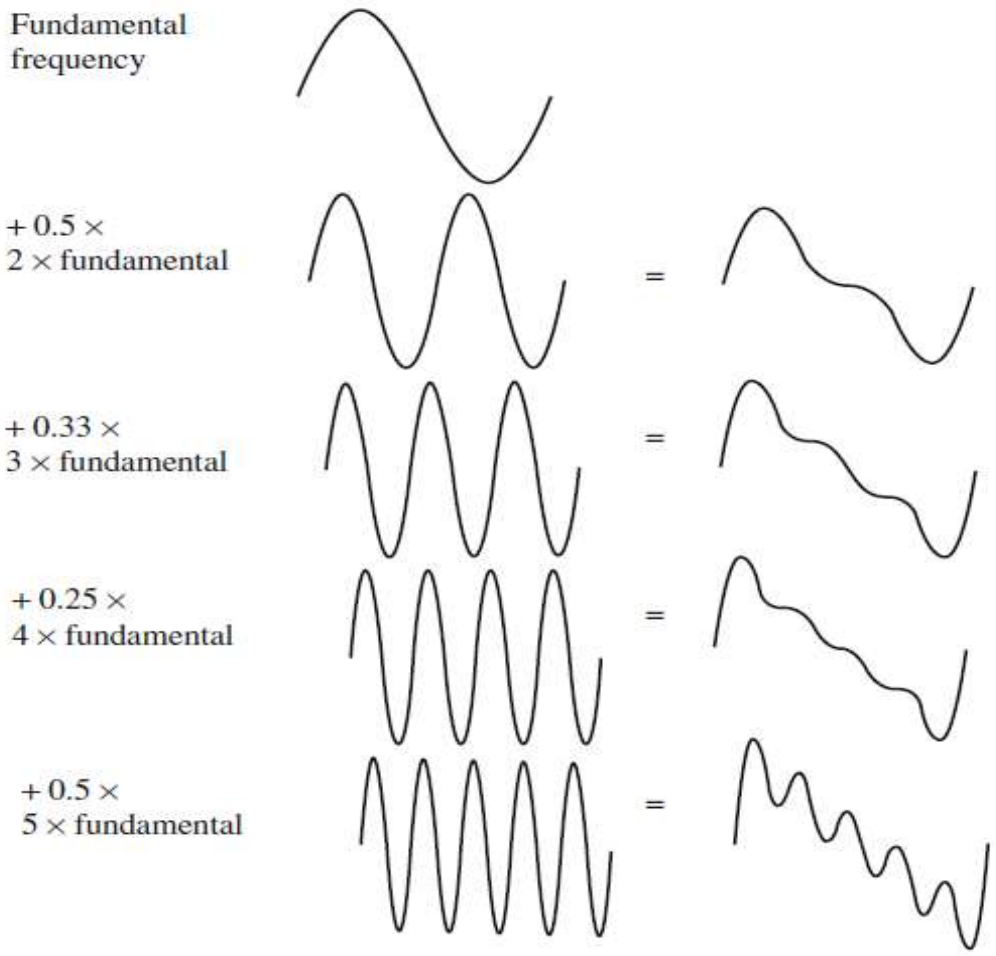
Digitization of Sound

What is Sound?

- Sound is a wave phenomenon like light, but it is macroscopic and involves *molecules of air being compressed and expanded under the action of some physical device*.
- For example, a *speaker in an audio system* vibrates back and forth and produces a longitudinal pressure wave that we perceive as sound.
- *Without air there is no sound—for example, in space*.
- Since sound is a pressure wave, it takes on continuous values, as opposed to digitized ones with a finite range.
- Nevertheless, if we wish to use a digital version of sound waves, we must form digitized representations of audio information.

- Although such pressure waves are longitudinal, they still have ordinary wave properties and behaviors, such as reflection (bouncing), refraction (change of angle when entering a medium with a different density), and diffraction (bending around an obstacle). This makes the design of “surround sound” possible.
- In general, any signal can be decomposed into a sum of sinusoids, if we are willing to use enough sinusoids.
- A weighted sinusoids can build up quite a complex signal.

Building up a complex signal by superposing sinusoids

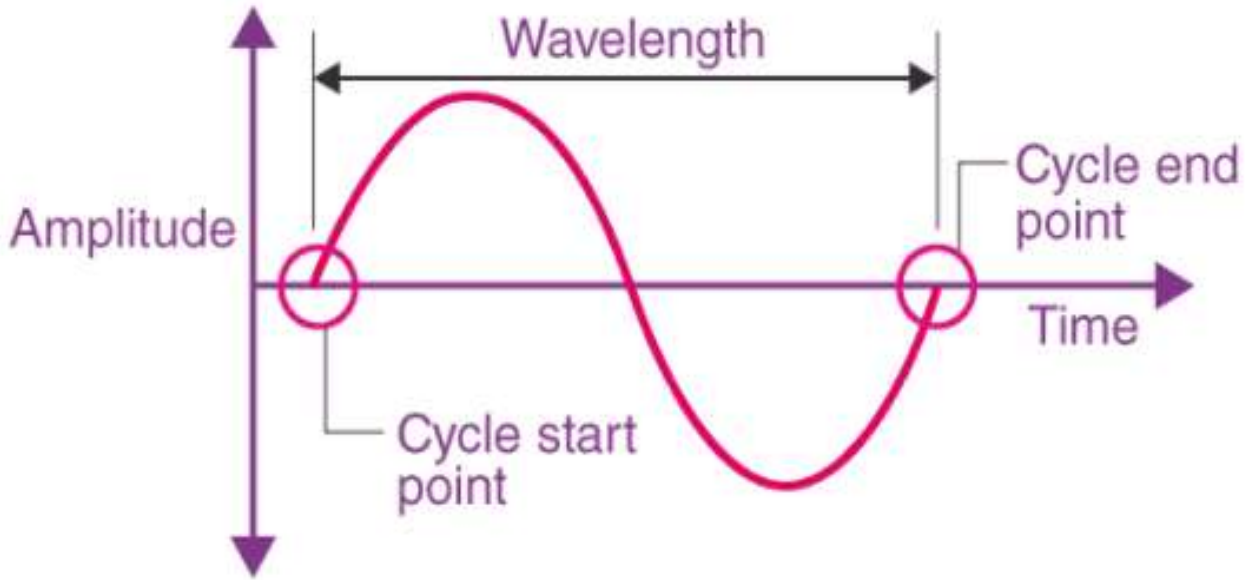


Characteristics of Sound

- The sound wave is having the following characteristics
- Amplitude:
  - It refers to the distance of the maximum vertical displacement of the wave from its mean position.
  - In sound, amplitude refers to the *magnitude of compression and expansion* experienced by the medium the sound wave is travelling through.
  - This amplitude is perceived by our ears as loudness. High amplitude is equivalent to loud sounds.
- Wavelength:
  - A sound wave is made of areas of high pressure alternated by an area of low pressure.
  - The high-pressure areas are represented as the peaks of the graph. The low-pressure areas are represented as troughs of the graph.
  - The *physical distance between two consecutive peaks in a sound wave is referred to as the wavelength of the sound wave*.

- Frequency/ Pitch of the Sound Waves
  - Frequency in a sound wave refers to the rate of the vibration of the sound travelling through the air. This parameter decides whether a sound is perceived as high pitched or low pitched.
  - In sound, the frequency is also known as **Pitch**.
  - The frequency of the vibrating source of sound is calculated in *cycles per second (Hertz)*.

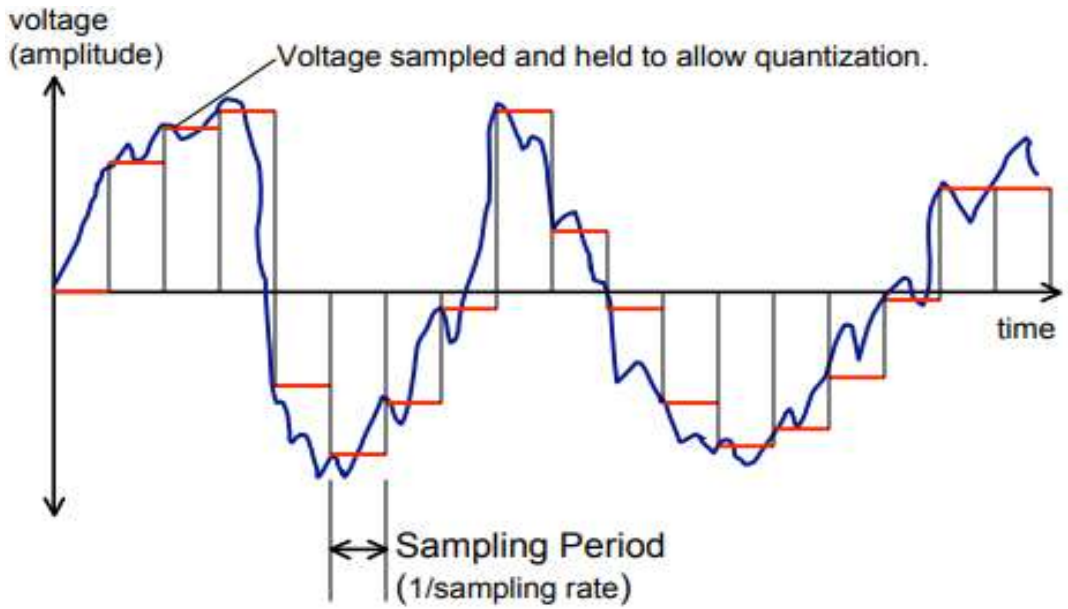
A depiction of Sound Waves in Waveform

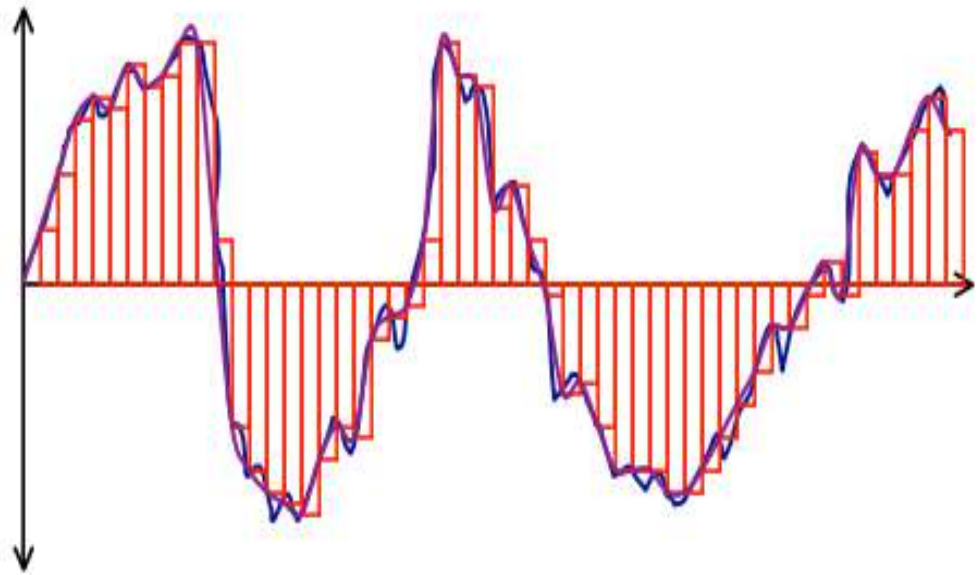


Digitization

- Since there is only one independent variable in sound i.e. time, we call this a **1D signal**.
- The amplitude value is a continuous quantity. Digitization means conversion to a stream of numbers.
- To fully digitize the sound signal, we have to ***sample in time and in amplitude.***
- ***Sampling*** means measuring the quantity we are interested in, usually at evenly spaced intervals.
- The first kind of sampling—using measurements only at evenly spaced *time* intervals—is simply called *sampling* and the rate at which it is performed is called the ***sampling rate*** or ***sampling frequency***.

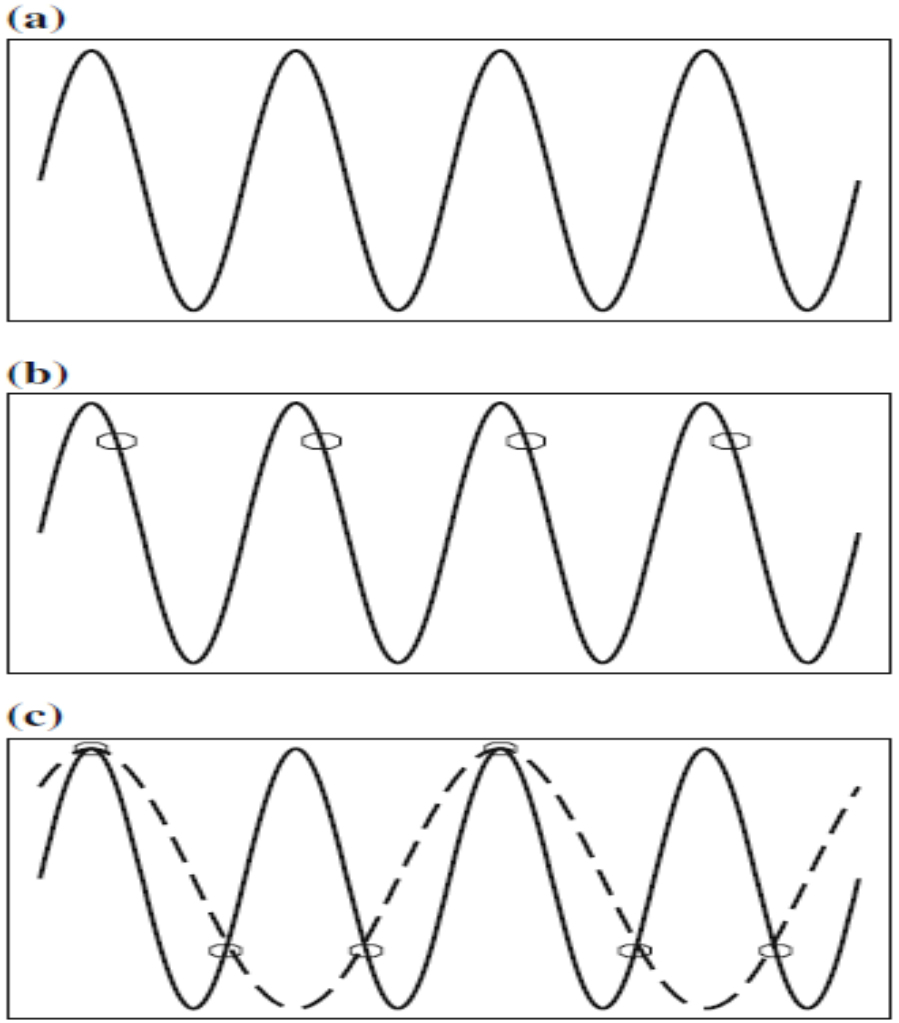
Sampling


voltage
(amplitude)
Voltage sampled and held to allow quantization.
time
Sampling Period
(1/sampling rate)



- For audio, typical sampling rates are from ***8 kHz*** (8,000 samples per second) to ***48 kHz***.
- The human ear can hear from about *20Hz to as much as 20kHz*; above this level, we enter the range of ultrasound.
- The human voice can reach approximately 4 kHz.
- *Nyquist sampling rate :*
  - To preserve the full information in the signal, it is necessary to sample at twice the maximum frequency of the signal. This is known as the ***Nyquist rate***.
  - If we sample the signal at a frequency that is lower that the Nyquist rate, when the signal is converted back into a continuous time signal, it will exhibit a phenomenon called ***aliasing***. Aliasing is the presence of unwanted components in the reconstructed signal.

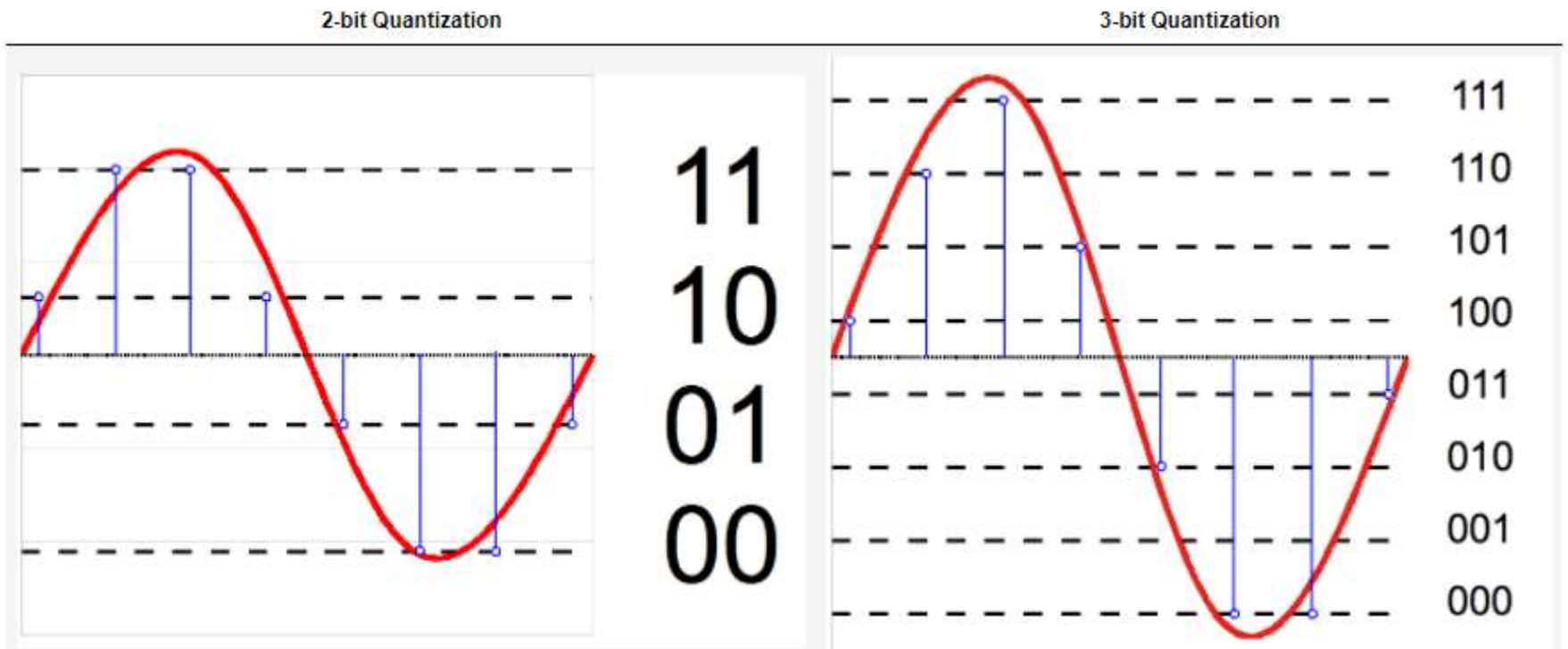
Aliasing: **a)** a single frequency; **b)** sampling at exactly the frequency produces a constant; **c)** sampling at 1.5 times per cycle produces an *alias* frequency that is perceived



Quantization:

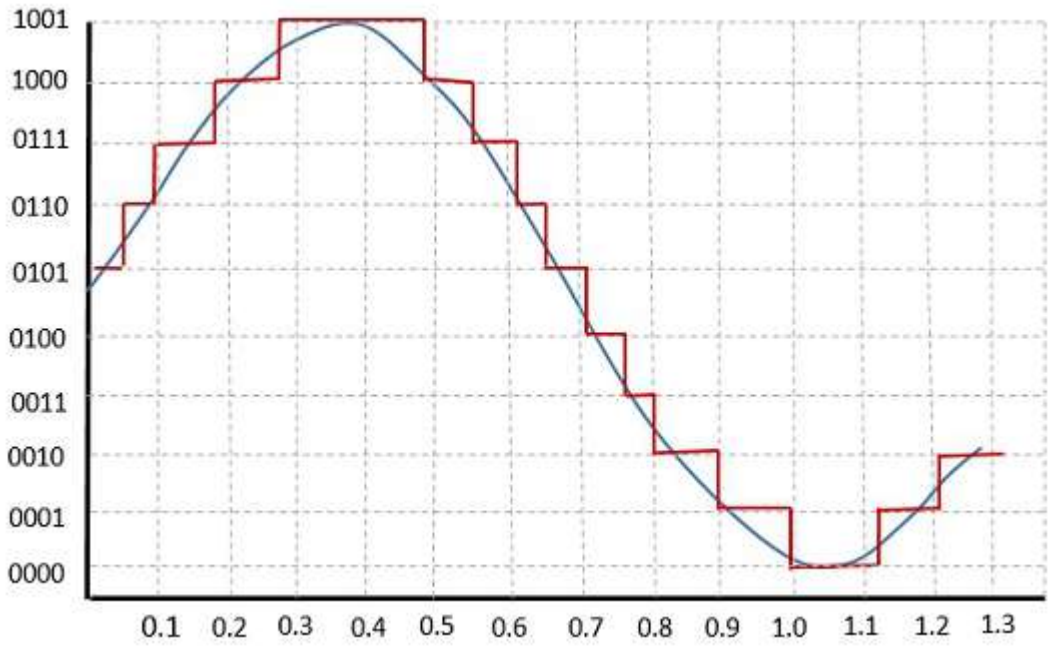
- Sampling in the amplitude dimension is called ***quantization*** *or It* refers to the process of *transforming a sampled analog signal, to a digital signal*, which has a discrete set of values.
- While we have discussed only uniform sampling, with equally spaced sampling intervals, non-uniform sampling is possible. This is not used for sampling in time but is used for quantization.
- Typical uniform quantization rates are 8-bit and 16-bit; *8-bit quantization divides the vertical axis into 256 levels*, and *16-bit divides it into 65,536 levels*.
- Quantization Error: The difference between an input value and its quantized value is called a ***Quantization Error***.
- A digitized sample can have a maximum error of one-half the discretization step size.

2-bit and 3-bit Quantization



Linear and Nonlinear/ Uniform and Non-uniform Quantization

- Samples are typically stored as uniformly quantized values. This is called *linear or uniform quantization.*



- There are two types of uniform quantization. They are *Mid-Rise* type and *Mid-Tread* type.
- The **Mid-Rise** type is so called because the origin lies in the middle of a raising part of the stair-case like graph. The quantization levels in this type are even in number.
- The **Mid-tread** type is so called because the origin lies in the middle of a tread of the stair-case like graph. The quantization levels in this type are odd in number.

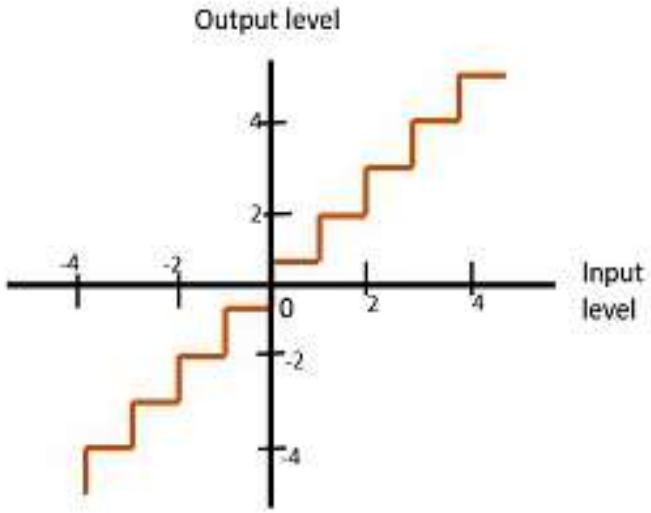


Fig 1 : Mid-Rise type Uniform Quantization

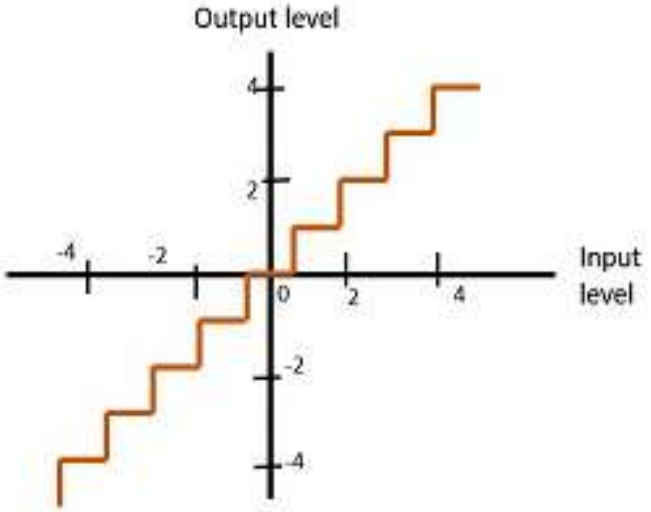


Fig 2 : Mid-Tread type Uniform Quantization

Multimedia Systems
Lecture – 18

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

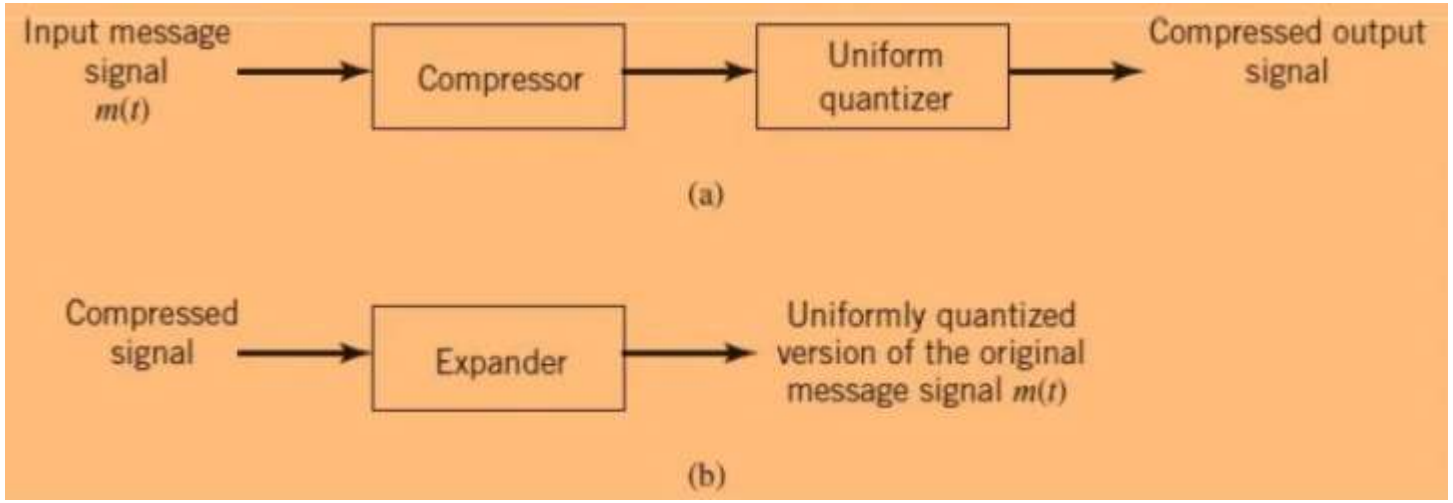
Non-uniform Quantization

- With a limited number of bits available, it may be more sensible to try to take into account the properties of human perception and set up nonuniform quantization levels that pay more attention to the frequency range over which humans hear best.
- If the quantization characteristic is nonlinear then the step size is not constant and quantization is known as non-uniform quantization.
- It is mostly used in case of speech or music as here the variation in amplitude is high which is expressed as crest factor and is given by

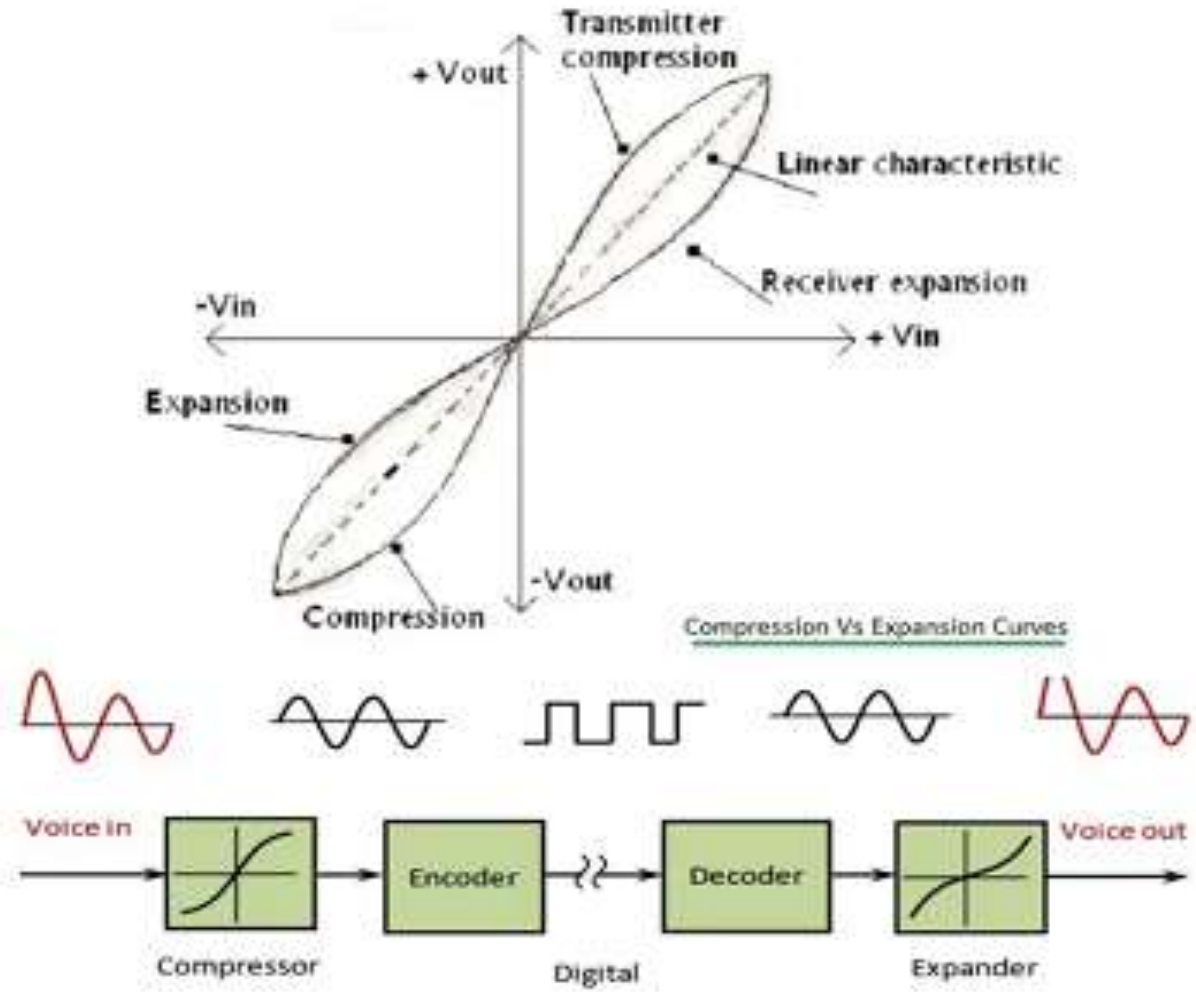
  crest factor = peak value of signal/rms value of signal

- Non-uniform quantization is achieved using companding.

- Companding:
- It is derived from two words, ***Compressing*** and ***Expanding***.
- The desired form of non-uniform quantization can be achieved by using compressor followed by a uniform quantizer.



Companding Process



μ-law and A-law companding

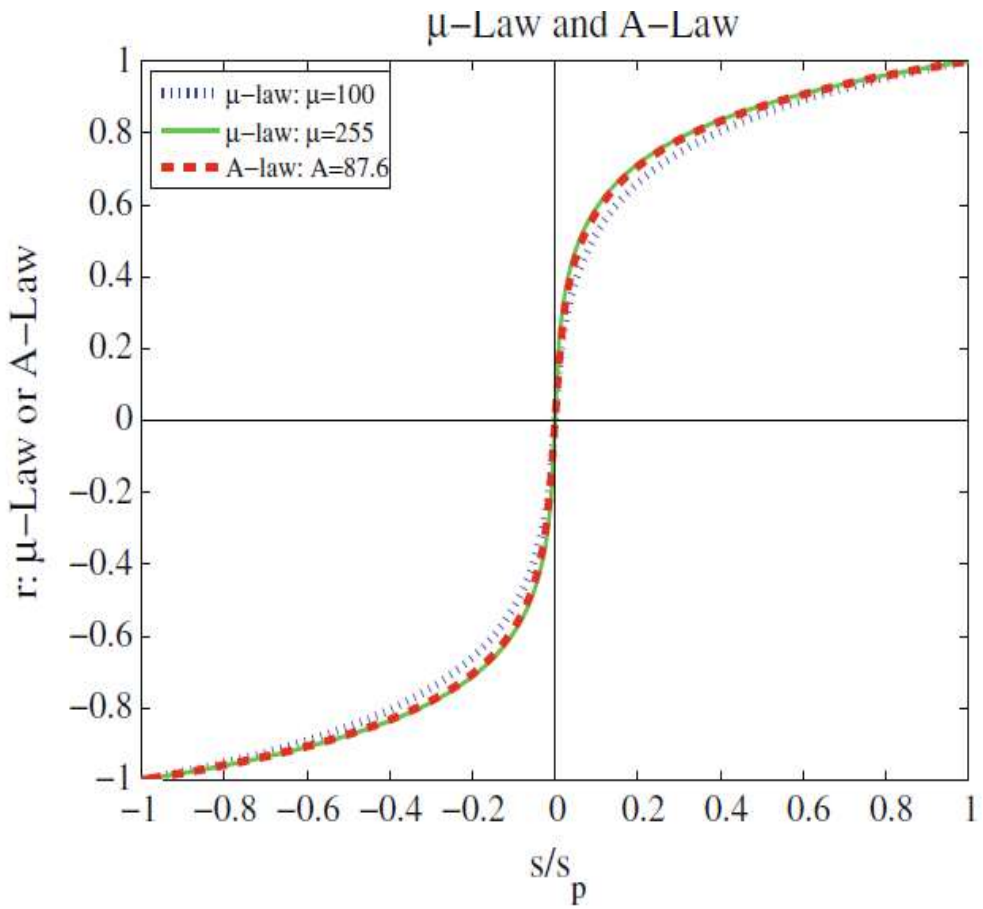
- μ-law is popular technique used in USA and Japan.
- Here the input and output relationship is given by

$$r = \frac{\text{sign}(s)}{\ln(1+\mu)} \ln\left\{1 + \mu\left|\frac{s}{s_p}\right|\right\}, \qquad \left|\frac{s}{s_p}\right| \le 1$$

- A very similar rule, called *A-law*, is used in telephony in Europe.

$$r = \begin{cases} \frac{A}{1+\ln A}\left(\frac{s}{s_p}\right), & \left|\frac{s}{s_p}\right| \le \frac{1}{A} \\ \frac{\text{sign}(s)}{1+\ln A}\left[1 + \ln A\left|\frac{s}{s_p}\right|\right], & \frac{1}{A} \le \left|\frac{s}{s_p}\right| \le 1 \end{cases}$$

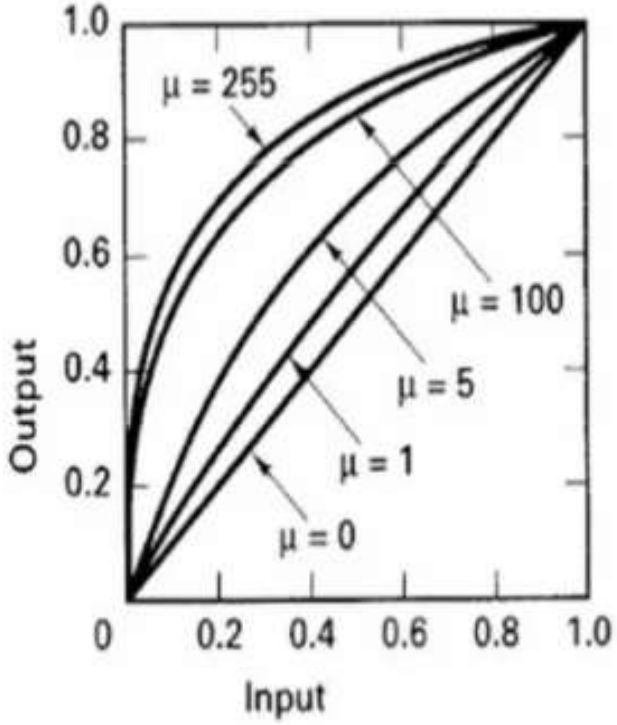
$$\text{where } \text{sign}(s) = \begin{cases} 1 & \text{if } s > 0, \\ -1 & \text{otherwise} \end{cases}$$

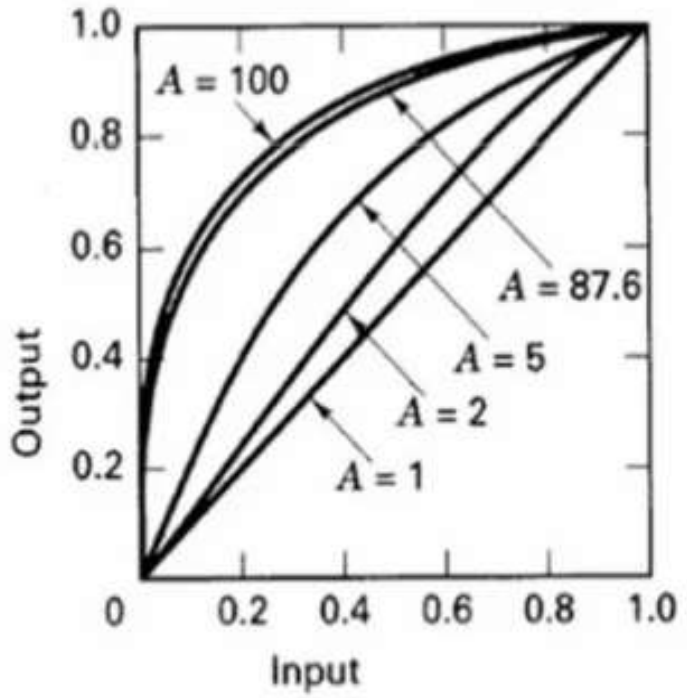


For steps near the low end of the signal, quantization steps are effectively more concentrated on the s axis, whereas for large values of s, one quantization step in r encompasses a wide range of s values.

s_p is the peak signal value and s is the current signal value.

μ-law and A-law Compression Characteristics

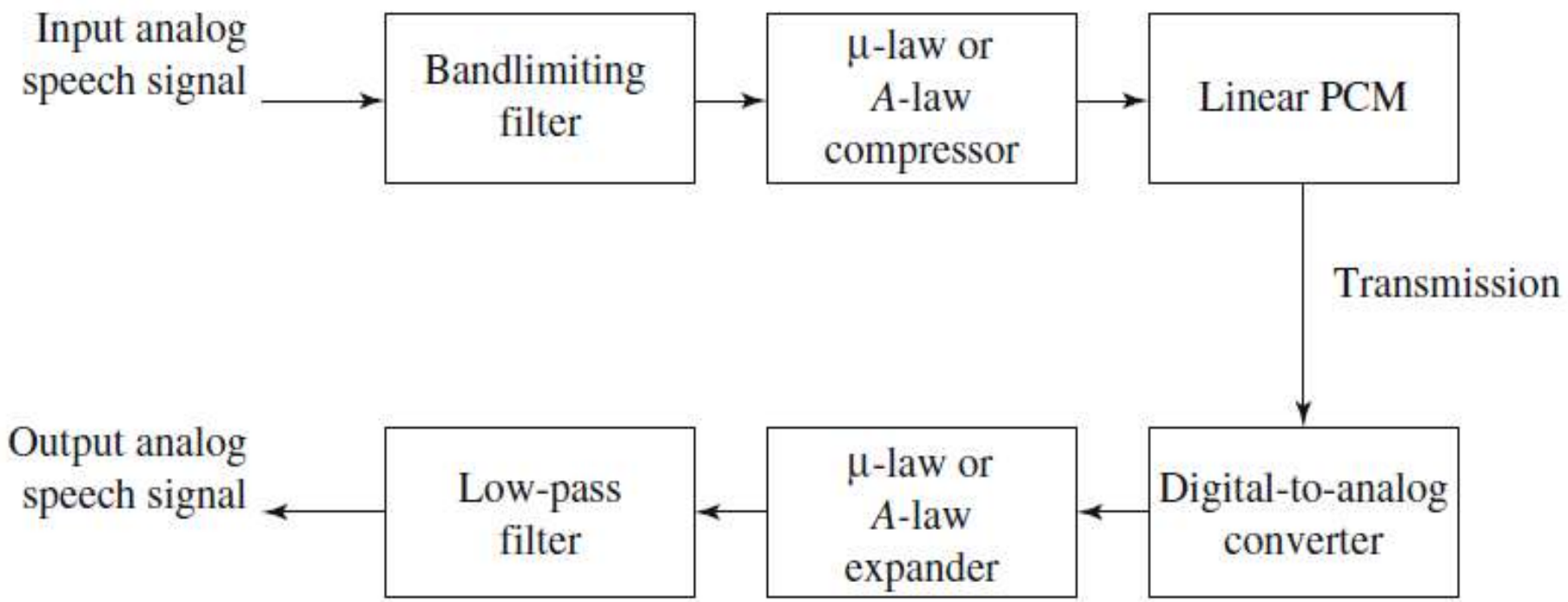


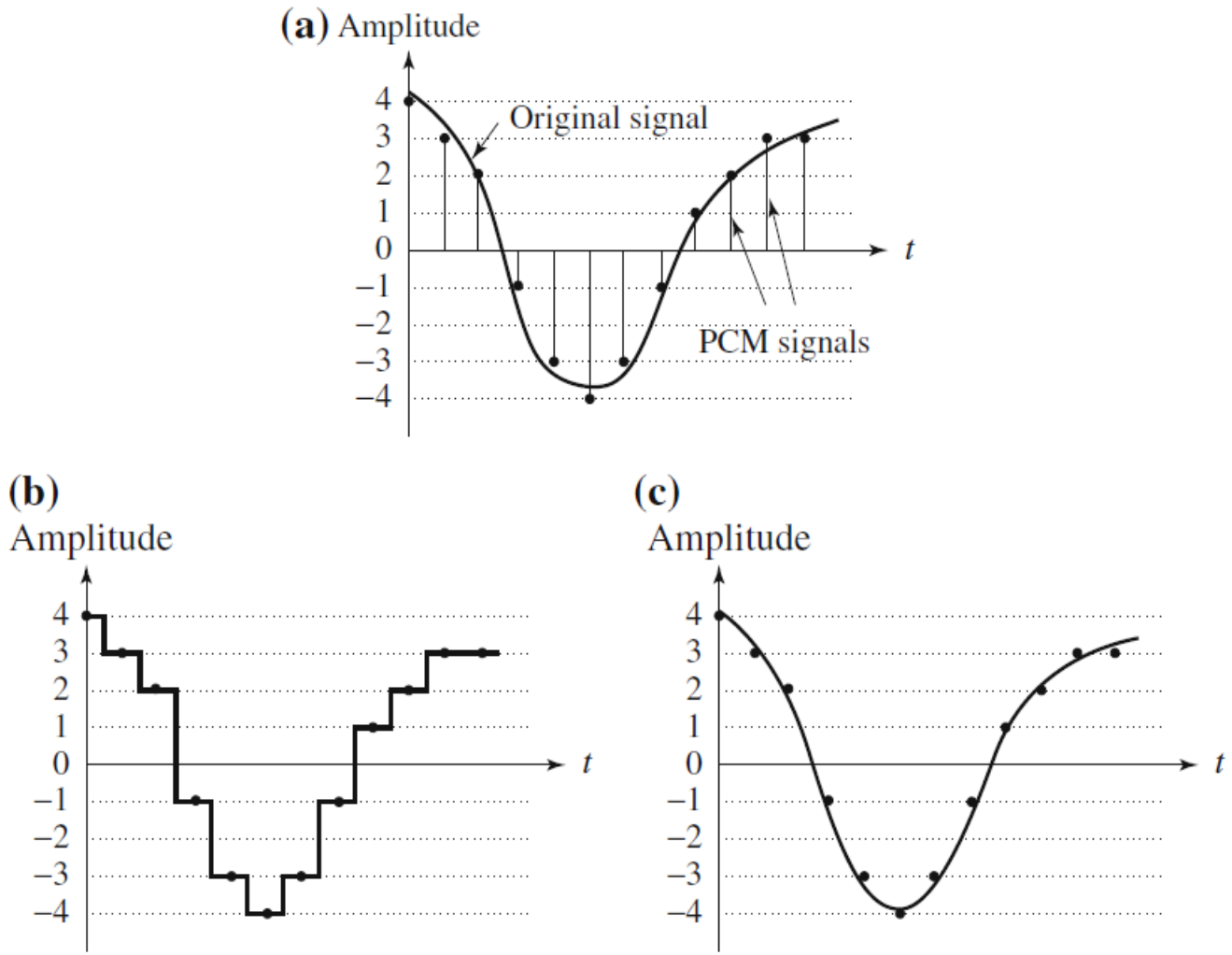


Pulse Code Modulation (PCM)

- **Modulation** is the process of varying one or more parameters of a carrier signal in accordance with the instantaneous values of the message signal.
- There are many modulation techniques, which are classified according to the type of modulation employed. Of them all, the digital modulation technique used is **Pulse Code Modulation.**
- We know that the basic techniques for creating digital signals from analog ones consist of *sampling* and *quantization*.
- Pulse Code Modulation, is a formal term for the sampling and quantization we have already been using.
- *Pulse* comes from an engineer's point of view that the resulting digital signals can be thought of as infinitely narrow vertical "pulses."

Basic Elements of PCM





a original analog signal and its corresponding PCM signals; **b** decoded staircase signal; **c** reconstructed signal after low-pass filtering

Multimedia Systems
Lecture – 19

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Differential Pulse Code Modulation (DPCM)

- It is based on **differential predictive coding**.
- Audio is often stored not in simple PCM but in a form that ***exploits differences***.
- Generally, if a time-dependent signal has some consistency over time (*temporal redundancy*), the difference signal—subtracting the current sample from the previous one—will have a more ***peaked histogram***, with a maximum around zero.
- For a start, differences will generally be smaller numbers and hence offer the possibility of using fewer bits to store.

- Suppose our integer sample values are in the range 0 .. 255. Then differences could be as much as −255 .. 255. So we have unfortunately increased our *dynamic range* (ratio of maximum to minimum) by a factor of two.
- Let's formalize our statement of what we are doing by defining the integer signal as the set of values f_n.
- Then we *predict* $\hat{f}_n$ values as simply the previous value, and we define the error e_n as the difference between the actual and predicted signals:

$$\hat{f}_n = f_{n-1}$$
$$e_n = f_n - \hat{f}_n$$

- We certainly would like our error value e_n to be as small as possible.
- Therefore, we would wish our prediction $\hat{f}_n$ to be as close as possible to the actual signal f_n.

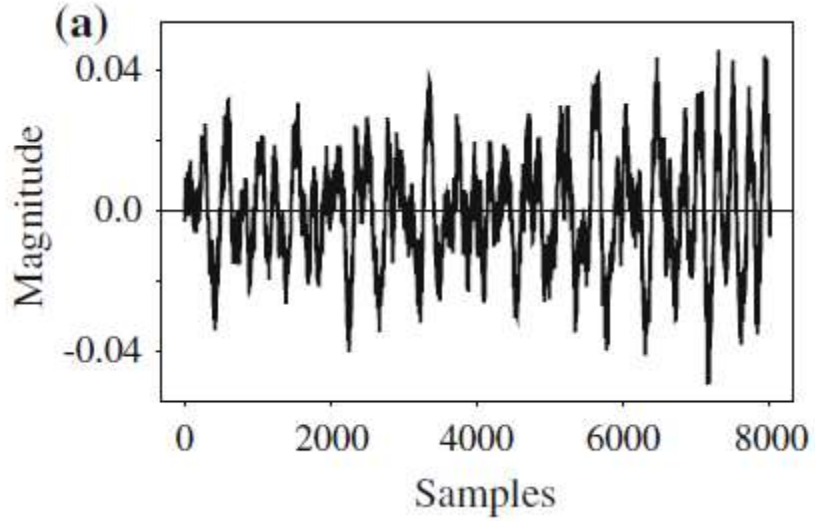
- But for a particular sequence of signal values, some *function* of a few of the previous values, f_{n-1}, f_{n-2}, f_{n-3}, etc., may provide a better prediction of f_n.

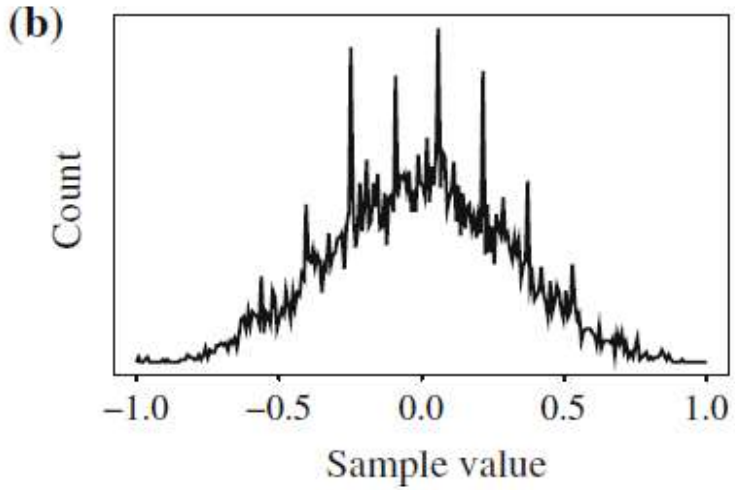
$$\hat{f}_n = \sum_{k=1}^{2 \text{ to } 4} a_{n-k} f_{n-k}$$

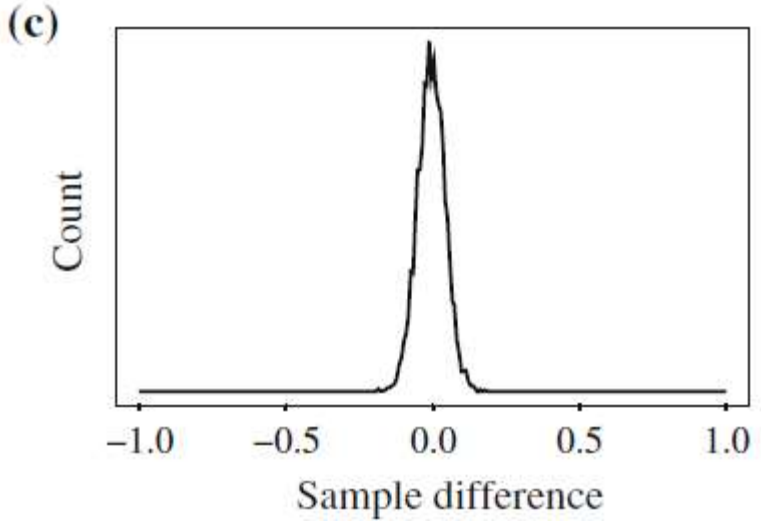
- The idea of forming differences is to make the histogram of sample values more peaked.

- ***So we can assign short codes to prevalent values like zeros here and long codewords to rarely occurring ones.***

Differencing concentrates the histogram: **a** digital speech signal; **b** histogram of digital speech signal values; **c** histogram of digital speech signal differences







- Example: suppose we devise a predictor for $\hat{f}_n$ as follows:

$$\hat{f}_n = \lfloor \frac{1}{2}(f_{n-1} + f_{n-2}) \rfloor$$
$$e_n = f_n - \hat{f}_n$$

- Then the error e_n (or a codeword for it) is what is actually transmitted. Suppose we wish to code the sequence

f1, f2, f3, f4, f5 = 21, 22, 27, 25, 22

- For the purposes of the predictor, we'll invent an extra signal value f_0, equal to $f1 = 21$, and first transmit this initial value, uncoded; after all, every coding scheme has the extra expense of some header information.

- Then the first error, $e1$, is zero, and subsequently

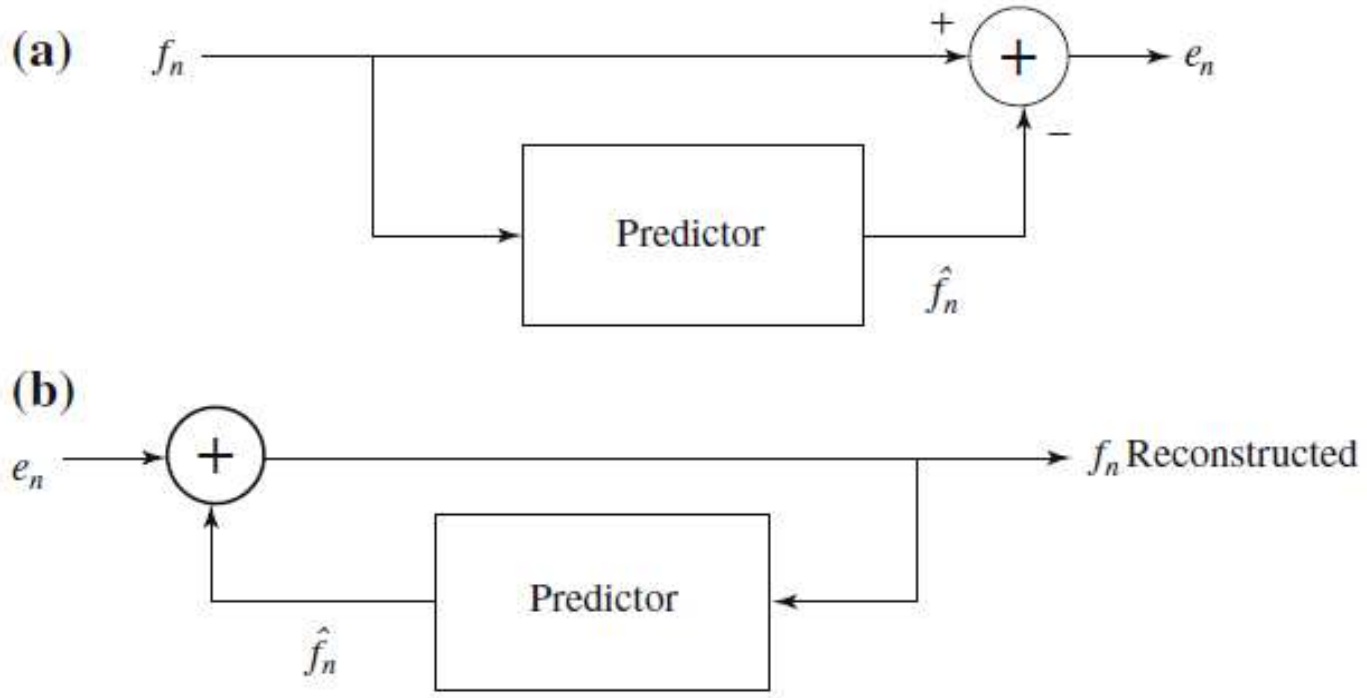
$$\hat{f}_2 = 21, \quad e_2 = 22 - 21 = 1$$

$$\hat{f}_3 = \lfloor \frac{1}{2}(f_2 + f_1) \rfloor = \lfloor \frac{1}{2}(22 + 21) \rfloor = 21$$
$$e_3 = 27 - 21 = 6$$

$$\hat{f}_4 = \lfloor \frac{1}{2}(f_3 + f_2) \rfloor = \lfloor \frac{1}{2}(27 + 22) \rfloor = 24$$
$$e_4 = 25 - 24 = 1$$

$$\hat{f}_5 = \lfloor \frac{1}{2}(f_4 + f_3) \rfloor = \lfloor \frac{1}{2}(25 + 27) \rfloor = 26$$
$$e_5 = 22 - 26 = -4$$

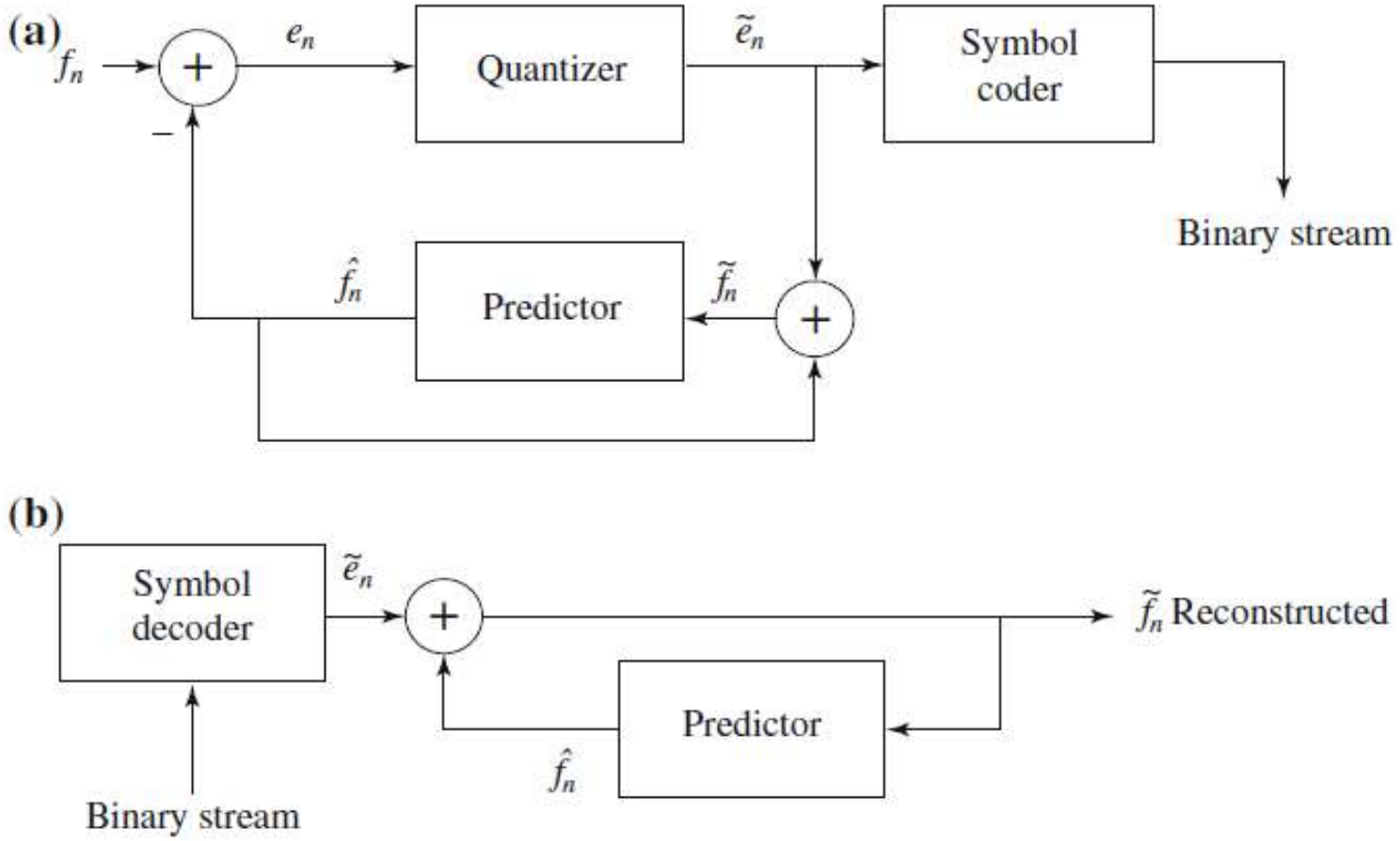
Schematic diagram for Predictive Coding: **a** encoder; **b** decoder



- ***Differential Pulse Code Modulation (DPCM)*** is exactly the same as Predictive Coding, Predictive coding except that it incorporates a ***quantizer step***.
- We shall call the original signal f_n, the predicted signal $\hat{f}_n$, and the quantized, reconstructed signal $\tilde{f}_n$. How DPCM operates is to *form the prediction*, *form an error e_n by subtracting the prediction from the actual signal*, then *quantize the error to a quantized version*, $\tilde{e}_n$.
- The equations that describe DPCM are as follows

$$\hat{f}_n = \text{function\_of } (\tilde{f}_{n-1}, \tilde{f}_{n-2}, \tilde{f}_{n-3}, \ldots)$$
$$e_n = f_n - \hat{f}_n$$
$$\tilde{e}_n = Q[e_n]$$
$$\text{transmit codeword}(\tilde{e}_n)$$
$$\text{reconstruct: } \tilde{f}_n = \hat{f}_n + \tilde{e}_n$$

Schematic diagram for DPCM: **a** encoder; **b** decoder



- Codewords for quantized error values $\tilde{e}_n$ are produced using ***entropy coding***, such as Huffman coding.
- Notice that the predictor is always based on the reconstructed, quantized version of the signal $\tilde{f}_n$: the reason for this is that then the encoder side is not using any information not available to the decoder side.
- The main effect of the coder–decoder process is to produce $\tilde{f}_n$ reconstructed, quantized signal values $\tilde{f}_n = \hat{f}_n + \tilde{e}_n$
- The distortion is the average squared error $[\sum_{n=1}^{N}(\tilde{f}_n - f_n)^2]/N$
- The predictor makes use of the reconstructed, quantized signal values not actual signal values f_n—that is, the encoder simulates the decoder in the predictor path. The quantizer can be uniform or nonuniform.

- The prediction value $\hat{f}_n$ is based on however much history the prediction scheme requires: we need to buffer previous values of $\tilde{f}_n$ to form the prediction.
- Notice that the quantization noise, $f_n - \tilde{f}_n$ is equal to the quantization effect on the error term, $e_n - \tilde{e}_n$
- Example: Suppose we adopt a particular predictor as

$$\hat{f}_n = \text{trunc}\left[\left(\tilde{f}_{n-1} + \tilde{f}_{n-2}\right)/2\right]$$

so that $e_n = f_n - \hat{f}_n$ is an integer.

- the particular quantization scheme

$$\tilde{e}_n = Q[e_n] = 16 * \text{trunc}\left[(255 + e_n)/16\right] - 256 + 8$$
$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n$$

- Suppose we wish to code the sequence $f1, f2, f3, f4, f5 = 130, 150, 140, 200, 230$.

- Here, the error is in the range −255 .. 255—that is, 511 levels are possible for the error term.
- The quantizer takes the simple course of dividing the error range into 32 patches of about 16 levels each.

e_n in range	Quantized to value
−255 .. −240	−248
−239 .. −224	−232
⋮	⋮
−31 .. −16	−24
−15 .. 0	−8
1 .. 16	8
17 .. 32	24
⋮	⋮
225 .. 240	232
241 .. 255	248

- We prepend extra values $f_0 = 130$ in the datastream that replicate the first value, f_1, and initialize with quantized error $\tilde{e}_1 \equiv 0$, so that we ensure the first reconstructed value is exact: $\tilde{f}_1 = 130$.
- Then subsequent values calculated are as follows

$$
\begin{aligned}
\hat{f} &= \boxed{130},\ 130,\ 142,\ 144,\ 167 \\
e &= \quad \boxed{0},\ 20,\ -2,\ 56,\ 63 \\
\tilde{e} &= \quad \boxed{0},\ 24,\ -8,\ 56,\ 56 \\
\tilde{f} &= \boxed{130},\ 154,\ 134,\ 200,\ 223
\end{aligned}
$$

Delta Modulation (DM)

- It is a much-simplified version of DPCM often used as a quick analog-to-digital converter.

Uniform-Delta DM

- The idea in DM is to use only a *single* quantized error value, either positive or negative. Such a 1-bit coder thus produces coded output that follows the original signal in a staircase fashion.
- The relevant set of equations is as follows:

$$\hat{f}_n = \tilde{f}_{n-1}$$

$$e_n = f_n - \hat{f}_n = f_n - \tilde{f}_{n-1}$$

$$\tilde{e}_n = \begin{cases} +k \text{ if } e_n > 0, \text{ where } k \text{ is a constant} \\ -k \text{ otherwise,} \end{cases}$$

$$\tilde{f}_n = \hat{f}_n + \tilde{e}_n$$

- Note that the prediction simply involves a delay.
- Example: Suppose signal values are as follows

f_1	f_2	f_3	f_4
10	11	13	15

- We also define an exact reconstructed value $\tilde{f}_1 = f_1 = 10$.
- Suppose we use a step value $k = 4$. Then we arrive at the following values:

$$\hat{f}_2 = 10,\ e_2 = 11 - 10 = 1, \quad \tilde{e}_2 = 4, \quad \tilde{f}_2 = 10 + 4 = 14$$
$$\hat{f}_3 = 14,\ e_3 = 13 - 14 = -1,\ \tilde{e}_3 = -4,\ \tilde{f}_3 = 14 - 4 = 10$$
$$\hat{f}_4 = 10,\ e_4 = 15 - 10 = 5, \quad \tilde{e}_4 = 4, \quad \tilde{f}_4 = 10 + 4 = 14$$

- We see that the reconstructed set of values 10, 14, 10, 14 never strays far from the correct set 10, 11, 13, 15.
- It is not difficult to discover that DM copes well with more or less constant signals, but not as well with rapidly changing signals.

Adaptive DM

- However, if the slope of the actual signal curve is high, the staircase approximation cannot keep up.
- A straightforward approach to dealing with a steep curve is to simply change the step size *k adaptively*—that is, in response to the signal's current properties.

Commonly Used Audio Formats

- Digital audio formats emerged with the use and distribution of CD audio discs. These were uncompressed pulse code modulated digital signals in mono and in stereo (*Mono signals are recorded and played back using a single audio channel, while stereo sounds are recorded and played back using two audio channels*).
- However, a number of formats have now become mainstream with the need for streaming, mobile, and surround sound technologies (**Surround sound** is a technique for enriching the fidelity and depth of sound reproduction by using multiple audio channels from speakers that surround the listener (surround channels). Its first application was in movie theaters).

File suffix or logo	Filename	File type	Features
.wav	WAV	Uncompressed PCM coded	Default standard for audio on PCs. WAV files are coded in PCM format.
.au	G.711 μ-law, or ITU μ-law	Uncompressed audio	Universal support for telephone. Packs each 16-bit sample into 8 bits, by using logarithmic table to encode with a 13-bit dynamic range. Encoding and decoding is very fast.
GSM 06.10	Global System for Mobile Communication	Lossy Compressed mobile audio	International standard for cellular telephone technology. Uses linear predictive coding to substantially compress the data. Compression/decompression is slow. Freely available and, thus, widely used
.mp3	MPEG1 Layer3	Compressed audio file format	Uses psychoacoustics for compression Very good bandwidth savings and, hence, used for streaming and Internet downloads.

.ra	Real Audio	Compressed format	Proprietary to Real Audio. Capable of streaming and downloading. Comparable quality to mp3 at high data rates but not so at low data rates
AAC	Advanced Audio Codec MPEG4	Compressed format	Superior quality to .mp3.
.mid	MIDI—Musical Instrument Digital Interface	Descriptive format	MIDI is a language of communication among musical instruments. Description achieved by frequencies, decays, transients, and event lists. Sound has to be synthesized by the instrument.

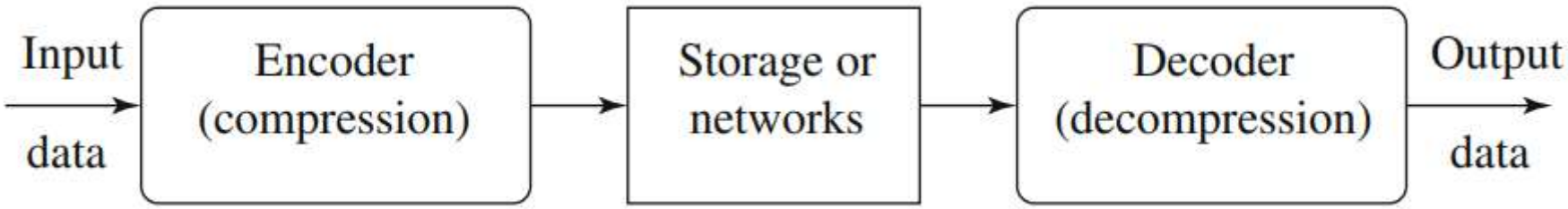
Multimedia Systems
Lecture – 20

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Multimedia Data Compression

- The amount of digital media data that is produced in the form of text, video, audio, 3D graphics, and combinations of these media types is extraordinarily large, and the rate of creation increases every day.
- This growing mass of data needs to be stored, accessed, and delivered to a multitude of clients over digital networks, which have varying bandwidths.
- *The existence of voluminous data, from creation to storage and delivery, motivates the need for compression.*
- The role played in multimedia by data compression, perhaps the most important enabling technology that makes modern multimedia systems possible.

- In a general data compression scheme, in which compression is performed by an encoder and decompression is performed by a decoder. general data compression scheme.



- We call the output of the encoder *codes* or *codewords*.
- The intermediate medium could either be data storage or a communication /computer network.
- If the compression and decompression processes induce no information loss, the compression scheme is *lossless*; otherwise, it is *lossy*.

- Compression Ratio: If the total number of bits required to represent the data before compression is B0 and the total number of bits required to represent the data after compression is B1, then we define the compression ratio as

$$\text{compression ratio} = \frac{B_0}{B_1}.$$

- In general, we would desire any codec (encoder/decoder scheme) to have a *compression ratio much larger than 1.0*.
- The *higher the compression ratio, the better the lossless compression* scheme, as long as it is computationally feasible.

Basics of Information Theory

- When transmitting information from a source to a destination, information theory concerns itself with the efficiency and reliability of the transmission.
- Information theory allows us to describe the process to compress data without losing its information content.
- The data could represent virtually anything— simple text, documents, images, graphics, and even binary executables.
- Here we will consider all these varied data types as generic data represented digitally in a *binary form by a sequence of 1s and 0s*.

- To understand compression, it is necessary to understand the information content of a message.
- Information relates to the organization in the data as a sequence of symbols. If the sequence changes, so does the information.
- *For example, you might have a binary data stream that consists of 80,000 bits. These bits might not need to be treated individually, but might instead be grouped into symbols. For instance, if it is known that bits represent a gray-intensity image with each pixel represented by 8 bits, then we have 10,000 pixels or symbols. Furthermore, if width and height of the image are both 100, the bit stream might be interpreted as a gray image of size 100 × 100. Here, each symbol is represented by 8 bits, and there can be $2^8 = 256$ different possible gray levels or symbols.*
- Also, the arrangement of the symbols is important.

- Alphabet and Symbol: Information can be thought of as an organization of individual elements called *symbols*.
- An *alphabet* is defined as a distinct and nonempty set of symbols.
- The number or length of symbols in the set is known as the *vocabulary*.
- We can define an alphabet of symbols as S = {s1, s2, s3, s4 . . . sn}. Though n can be very large, in practice, an alphabet is limited and finite and, hence, has a well-defined vocabulary.
- In the previous example involving a gray image, each pixel is represented by 8 bits and can have one of 2^8 or 256 unique values. Here, vocabulary consists of 256 symbols, where each symbol is represented or coded by 8 bits.
- Sequence: A series of symbols of a given alphabet form a sequence. For the alphabet {s1, s2, s3, s4}, a sample sequence is s1 s2 s1 s2 s2 s2 s1 s2 s3 s4 s1 s2 s3.
- A sequence of symbols is also termed a message produced by the source using the alphabet, and represents information.
- When looking at a sequence, some symbols might occur more commonly than other symbols. The frequency of occurrence of a symbol is an important factor when coding information represented by the symbols. The frequency is also known as probability.

- Symbol Probability: The probability of occurrence of a symbol is defined by the ratio of the number of occurrences of that symbol over the length of the entire message.

$$P_i = \frac{m_i}{N},$$

- where m_i is the number of times symbol s_i occurs in the message of length N.
- Most coding algorithms make extensive use of symbol probabilities to obtain optimal codes for compression.

- Entropy: Shannon's information theory borrows the definition of entropy from physics to quantify the amount of information contained in a message of symbols given their probabilities of occurrence.
- For source-producing symbols, where each symbol i has a probability distribution P_i , the entropy is defined as

$$H = \sum P_i \log_2\left(\frac{1}{P_i}\right) = -\sum P_i \log_2 P_i.$$

- For the symbol s_i having a probability P_i , Shannon defined the notion of *self-information of the symbol given by* $\log_2(1/P_i)$.
- The self-information represents number of bits of information contained in the symbol and, hence, the *number of bits used to send that message*.
- Entropy, then, becomes the weighted average of the information carried by each symbol and, hence, the *average symbol length*.

- Example: If the probability of having the character n in a manuscript is 1/32, the amount of information associated with receiving this character is 5 bits.
- In other words, a character string nnn will require 15 bits to code.
- The definition of entropy is aimed at *identifying often-occurring symbols in the datastream as good candidates for short codewords in the compressed bitstream*.
- We use a *variable-length coding* scheme for entropy coding—frequently occurring symbols are given codes that are quickly transmitted, while infrequently occurring ones are given longer codes.

- if the information source S is a gray-level digital image, each si is a gray-level intensity ranging from 0 to (2k − 1), where k is the number of bits used to represent each pixel in an uncompressed image. The range is often [0, 255], since 8 bits are typically used.
- Fig. a shows the histogram of an image with uniform distribution of gray-level intensities—that is, $\forall i\ p_i = 1/256$. Hence, the entropy of this image is

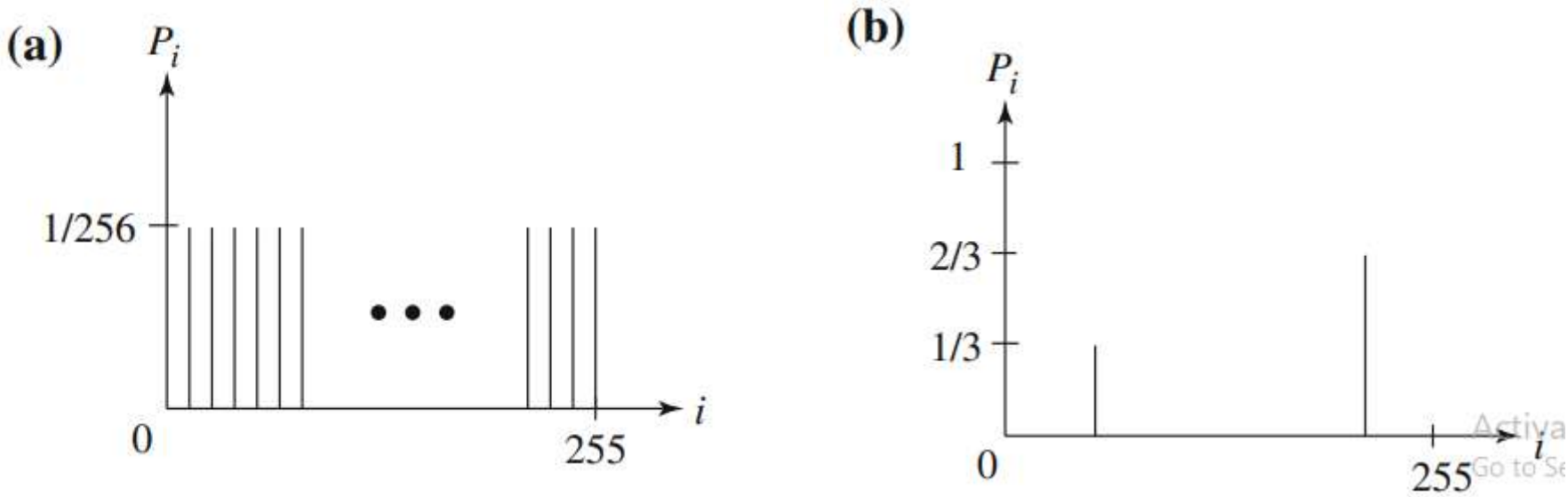
$$\eta = \sum_{i=0}^{255} \frac{1}{256} \cdot \log_2 256 = 256 \cdot \frac{1}{256} \cdot \log_2 256 = 8$$

- Fig. b shows the histogram of another image, in which 1/3 of the pixels are rather dark and 2/3 of them are rather bright. The entropy of this image is

$$\begin{aligned} \eta &= \frac{1}{3} \cdot \log_2 3 + \frac{2}{3} \cdot \log_2 \frac{3}{2} \\ &= 0.33 \times 1.59 + 0.67 \times 0.59 = 0.52 + 0.40 = 0.92 \end{aligned}$$

- The entropy is greater when the probability distribution is flat and smaller when it is more peaked

Histograms for two gray-level images. a Uniform distribution; b A sample binary image



Multimedia Systems
Lecture – 21

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Variable-Length Coding

- Since the entropy indicates the information content in an information source S, it leads to a family of coding methods commonly known as *entropy coding methods*.
- *Variable-length coding* (VLC) is one of the best-known such methods.
- We will study the
  - Shannon–Fano algorithm
  - Huffman coding

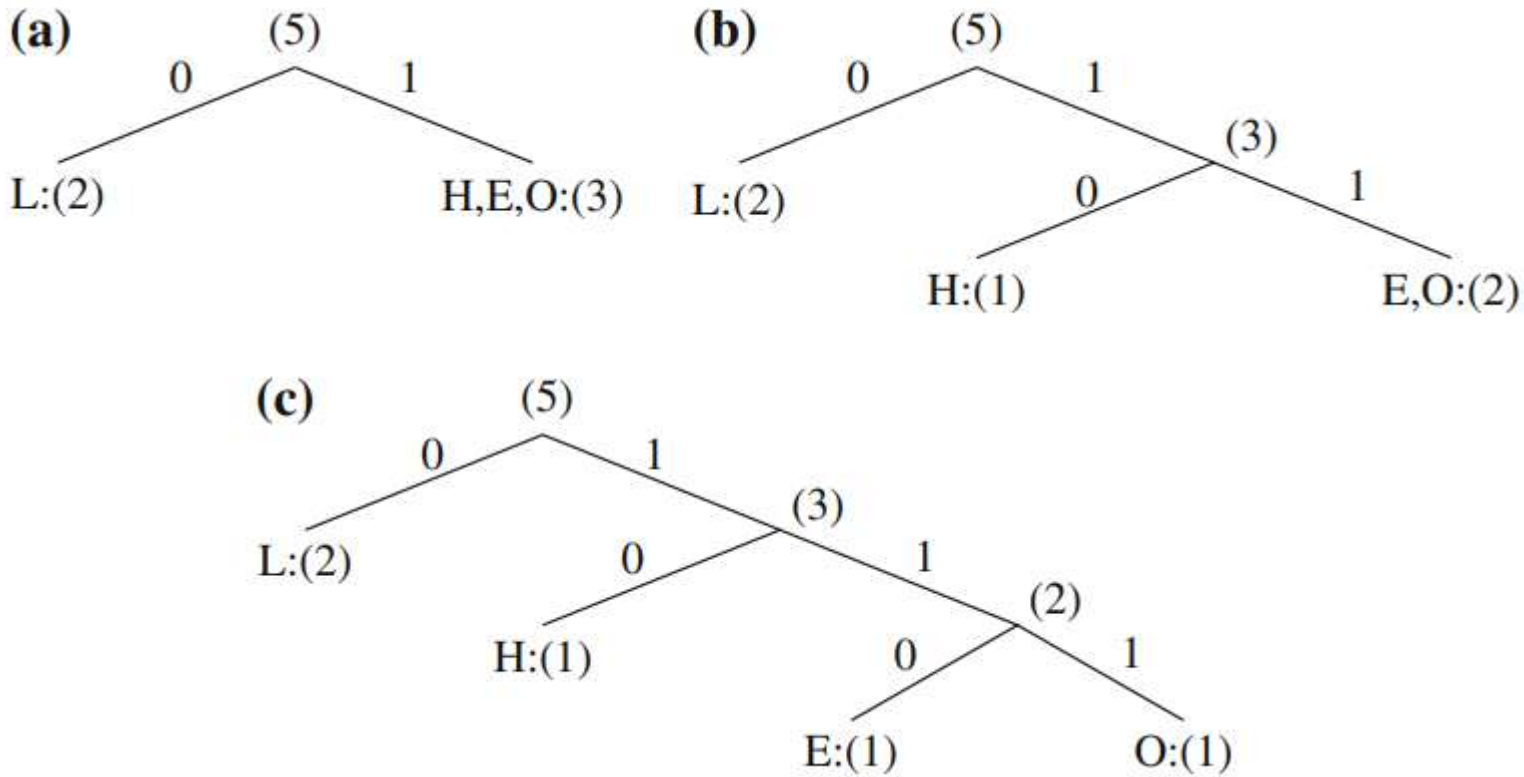
Shannon–Fano Algorithm

- Let us suppose the symbols to be coded are the characters in the word HELLO. The frequency count of the symbols is

Symbol	H	E	L	O
Count	1	1	2	1

- The encoding steps of the Shannon–Fano algorithm can be presented in the following *top-down manner*:
  - Sort the symbols according to the frequency count of their occurrences.
  - Recursively divide the symbols into two parts, each with approximately the same number of counts, until all parts contain only one symbol.

- Initially, the symbols are sorted as LHEO



One result of performing the Shannon–Fano algorithm on HELLO

Symbol	Count	$\log_2 \frac{1}{p_i}$	Code	Number of bits used
L	2	1.32	0	2
H	1	2.32	10	2
E	1	2.32	110	3
O	1	2.32	111	3
TOTAL number of bits:				10

- Entropy is given by

$$\eta = p_L \cdot \log_2 \frac{1}{p_L} + p_H \cdot \log_2 \frac{1}{p_H} + p_E \cdot \log_2 \frac{1}{p_E} + p_O \cdot \log_2 \frac{1}{p_O}$$
$$= 0.4 \times 1.32 + 0.2 \times 2.32 + 0.2 \times 2.32 + 0.2 \times 2.32 = 1.92$$

Huffman Coding

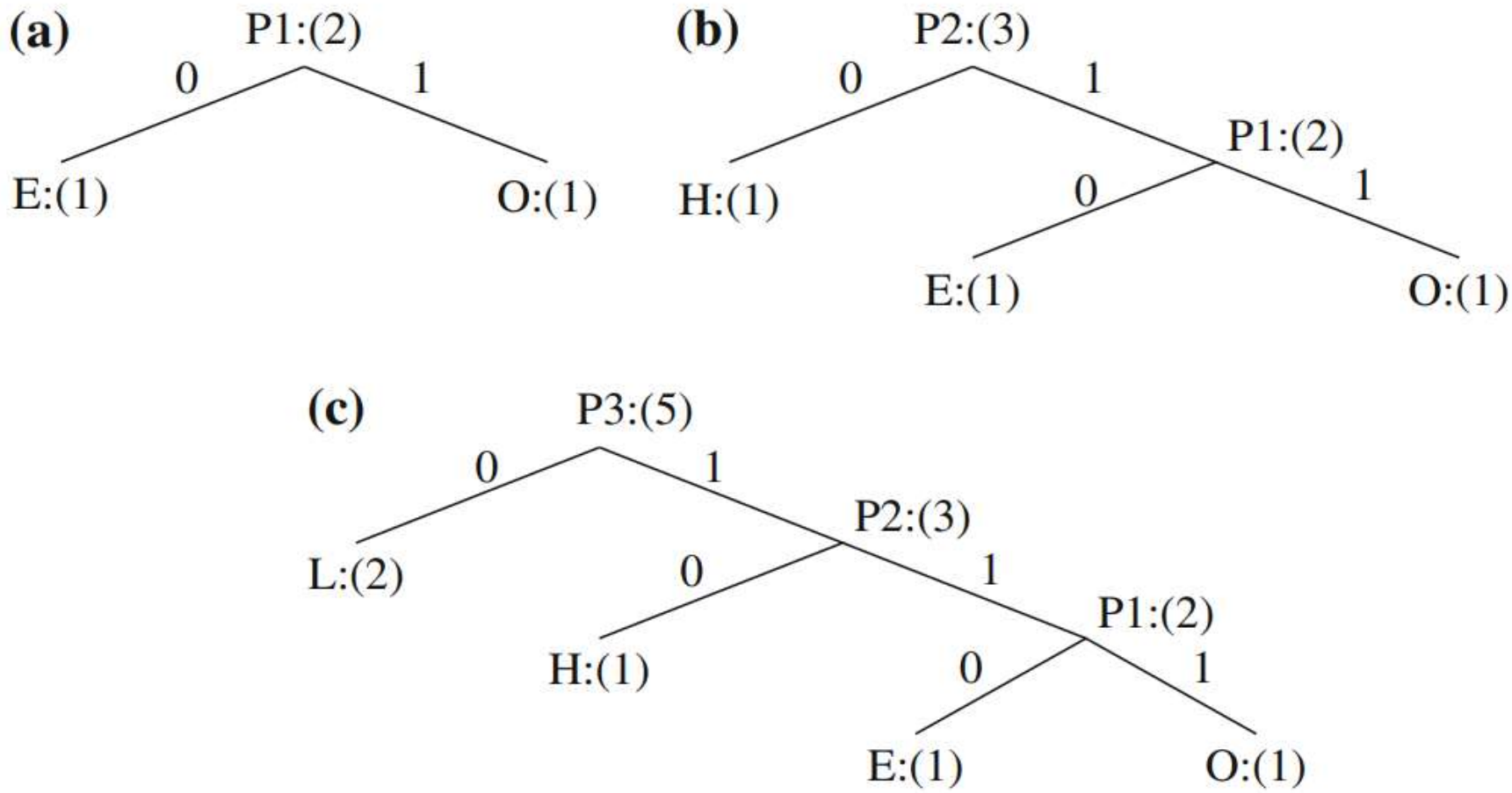
- First presented by Huffman in a 1952.
- This method attracted an overwhelming amount of research and has been adopted in many important and/or commercial applications, such as fax machines, JPEG, and MPEG.
- In contradistinction to Shannon–Fano, which is top-down, the encoding steps of the Huffman algorithm is a *bottom-up approach*.
- Let us use the same example word, **HELLO**. A similar binary coding tree will be used as above, in which the left branches are coded 0 and right branches 1. A simple list data structure is also used.

Huffman Coding Algorithm

Algorithm 7.1 (Huffman Coding).

1. Initialization: put all symbols on the list sorted according to their frequency counts.
2. Repeat until the list has only one symbol left.
   (a) From the list, pick two symbols with the lowest frequency counts. Form a Huffman subtree that has these two symbols as child nodes and create a parent node for them.
   (b) Assign the sum of the children's frequency counts to the parent and insert it into the list, such that the order is maintained.
   (c) Delete the children from the list.
3. Assign a codeword for each leaf based on the path from the root.

Coding tree for HELLO using the Huffman algorithm. a First iteration; b Second iteration; c Third iteration



- symbols P1, P2, P3 are created to refer to the parent nodes in the Huffman coding tree.
- The contents in the list are illustrated below:
  - After initialization: L H E O
  - After iteration (a): L P1 H
  - After iteration (b): L P2
  - After iteration (c): P3
- The average number of bits used to code each character is also 2, (i.e., (1 + 1 + 2 + 3 + 3)/5 = 2).
- Example-2: Consider a text string containing a set of characters and their frequency counts as follows: A:(15), B:(7), C:(6), D:(6) and E:(5). Find out the number of bits required in case of Shanon-Fano and Huffman Coding.

The average code length for an information source S is strictly less than $\eta + 1$, i.e.

$$\eta \leq \bar{l} < \eta + 1$$

Run Length Encoding

- Instead of assuming a memoryless source, run-length coding (RLC) exploits memory present in the information source.
- It is one of the simplest forms of data compression.
- The basic idea is that if the information source we wish to compress has the property that symbols tend to form continuous groups, *instead of coding each symbol in the group individually*, *we can code one such symbol and the length of the group*.
- A sample string of symbols is as follows:

  BBBBEEEEEEEECCCCDAAAAA.
- It can be represented as

  4B8E4C1D5A.

- Run length encoding is used in a variety of tools involving *text, audio, images, and video*.

  Consider a bilevel image (one with only 1-bit black and white pixels) with monotone regions. This information source can be efficiently coded using run-length coding. In fact, since there are only two symbols, we do not even need to code any symbol at the start of each run. Instead, we can assume that the starting run is always of a particular color (either black or white) and simply code the length of each run.

Multimedia Systems
Lecture – 22

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Adaptive Huffman Coding

- The Huffman algorithm *requires prior statistical knowledge* which is often not available. *Example in live (or streaming) audio and video.*
- Even when the statistics are available, *the transmission of the symbol table could represent heavy overhead*.
- To include contextual information, examine k preceding (or succeeding) symbols each time; this is known as an order-k model.
- This implies much more statistical data to be stored and sent.

- The solution is to use **adaptive compression algorithms.**
- Here, *statistics are gathered and updated dynamically* as the datastream arrives.
- The probabilities are no longer based on prior knowledge but on the actual data received so far.
- The new coding methods are “adaptive” because, *as the probability distribution of the received symbols changes, symbols will be given new (longer or shorter) codes*.
- This is especially desirable for multimedia data, when the content and hence the statistics can change rapidly.

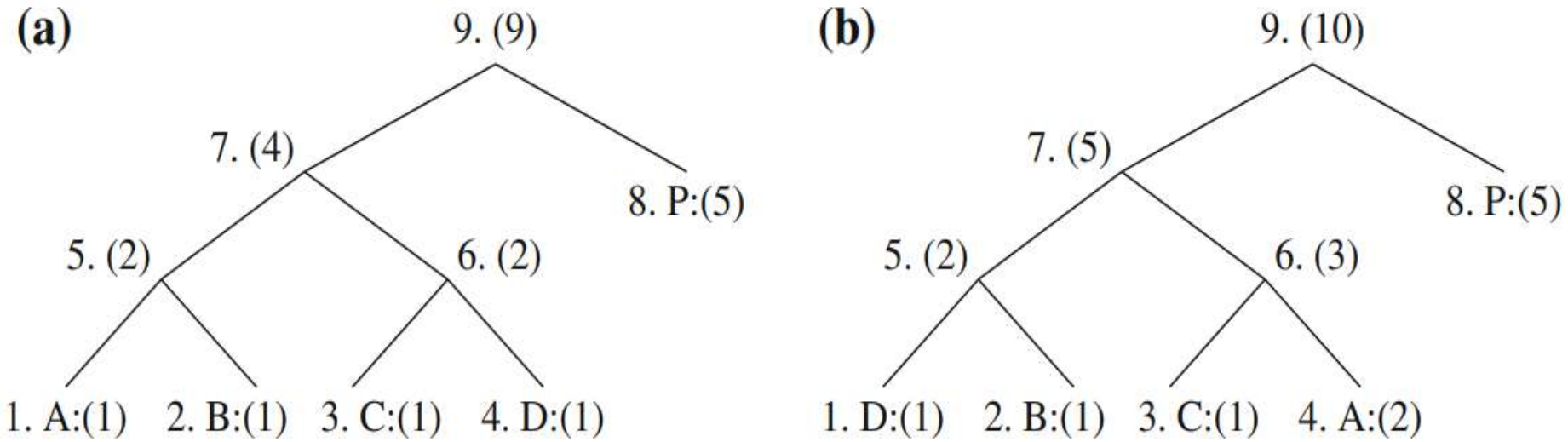
Procedure for Adaptive Huffman Coding

```
ENCODER
-------

Initial_code();
while not EOF
 {
    get(c);
    encode(c);
    update_tree(c);
 }
```

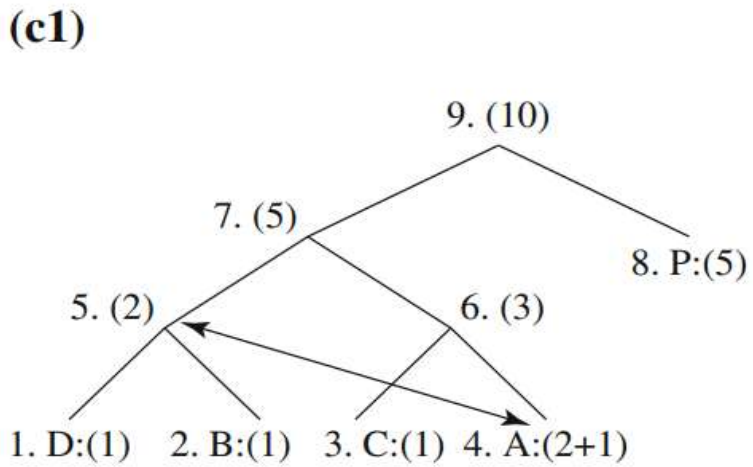
```
DECODER
-------

Initial_code();
while not EOF
   {
      decode(c);
      output(c);
      update_tree(c);
    }
```

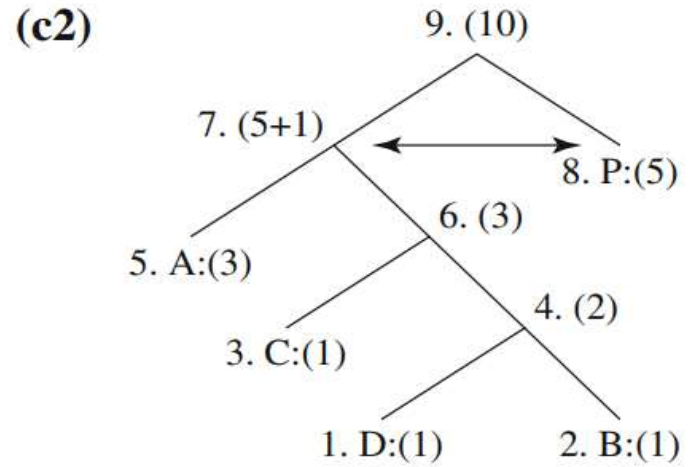
- **Initial_code** assigns symbols with some initially ***agreed-upon codes***, without any prior knowledge of the frequency counts for them.
- Example: ***ASCII***
- **update_tree** is a procedure for constructing an adaptive Huffman tree. It increments the frequency counts for the symbols (including any new ones), and updates the configuration of the tree
  - The Huffman tree must always maintain its ***sibling property***—that is, all nodes (internal and leaf) are arranged in the order of increasing counts. Nodes are numbered ***in order from left to right, bottom to top***.
  - If the sibling property is about to be violated, a ***swap procedure*** is invoked to update the tree by rearranging the nodes.
  - When a swap is necessary, the farthest node with count N is swapped with the node whose count has just been increased to N + 1. If the node with count N is not a leaf-node—it is the root of a subtree—the entire subtree will go with it during the swap
- The encoder and decoder must use exactly the same Initial_code and update_tree routines.

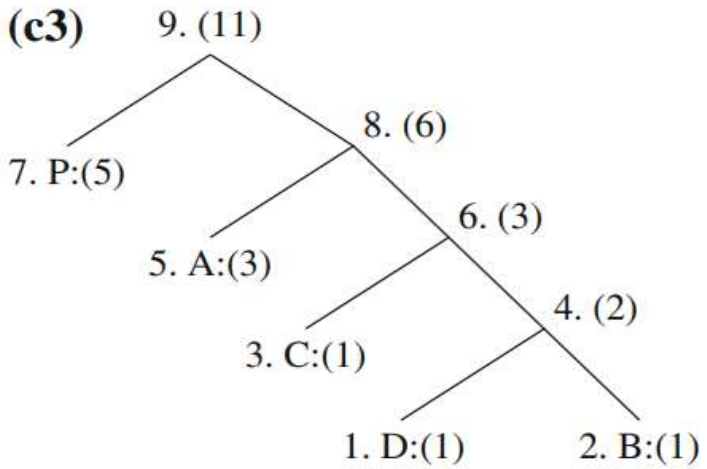
Node swapping for updating an adaptive Huffman tree: a a Huffman tree; b receiving 2nd ‘A’ triggered a swap;



c1 a swap is needed after receiving 3rd ‘A’; c2 another swap is needed; c3 the Huffman tree after receiving 3rd ‘A’





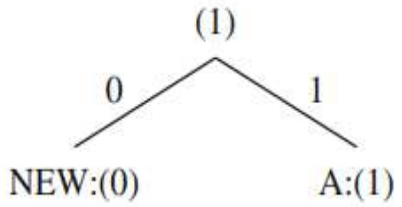


Example: Adaptive Huffman Coding for Symbol String AADCCDD.

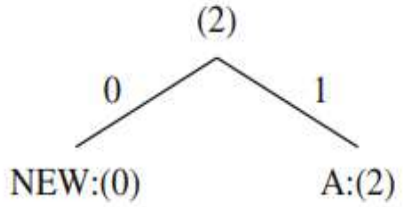
- Let us assume that the initial code assignment for both the encoder and decoder simply follows the ASCII order for the 26 symbols in an alphabet, A through Z.
- To improve the implementation of the algorithm, we adopt an additional rule: *if any character/symbol is to be sent the first time, it must be preceded by a special symbol,* ***NEW****. The initial code for NEW is 0. The count for NEW is always kept as 0*.

Symbol	Initial code
NEW	0
A	00001
B	00010
C	00011
D	00100
⋮	⋮

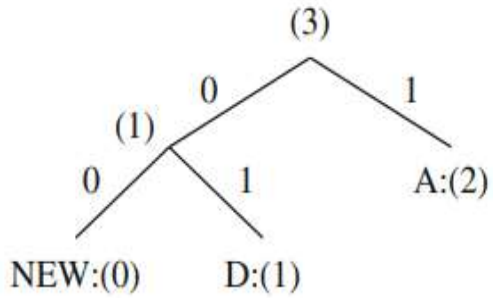
Adaptive Huffman tree for AADCCDD



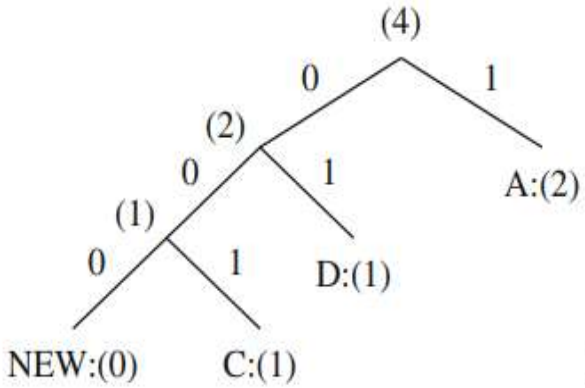
"A"



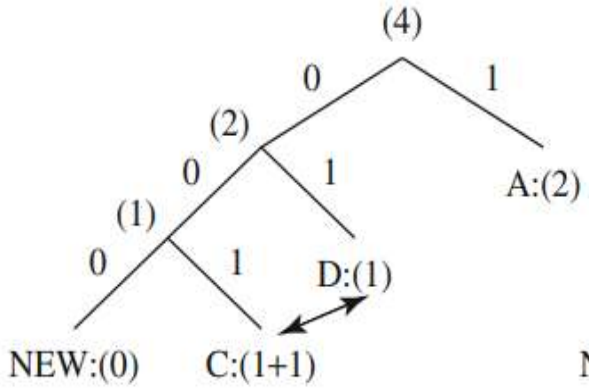
"AA"



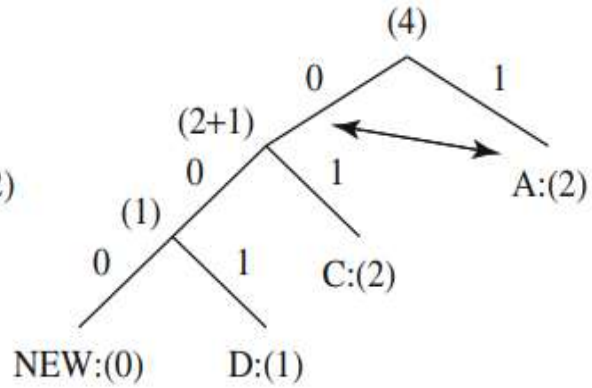
"AAD"



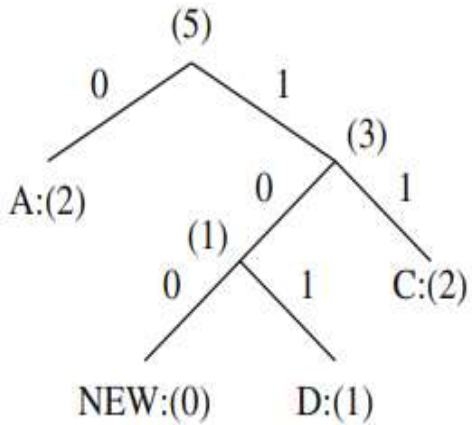
"AADC"



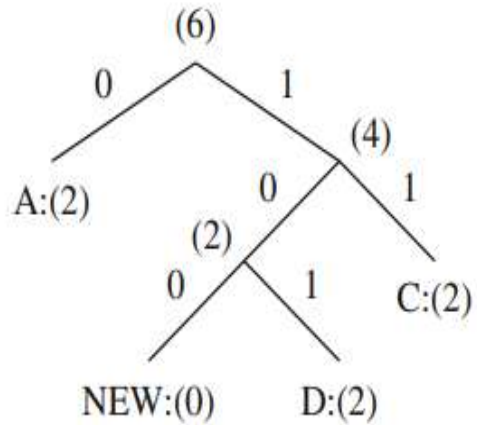
"AADCC" step 1



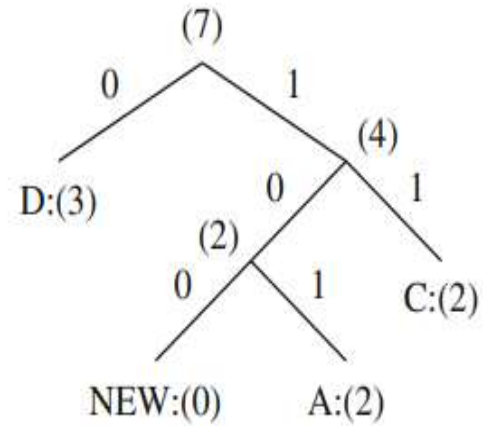
"AADCC" step 2



"AADCC" step 3



"AADCCD"



"AADCCDD"

Sequence of symbols and codes sent to the decoder

Symbol	NEW	A	A	NEW	D	NEW	C	C	D	D
Code	0	00001	1	0	00100	00	00011	001	101	101

Multimedia Systems
Lecture – 23

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Arithmetic Coding

- Arithmetic coding overcomes the requirement of every symbol to be coded independently.
- In other words, coding every symbol was represented individually by a code, or a group was represented. Thus, a whole number of bits were required to encode a symbol (or symbol group).
- Arithmetic coding overcomes this constraint by *mapping an entire message to a real number between zero and one. This real number representing the entire message is coded as a binary number.*
- Arithmetic coding, thus, encodes a message entirely without assigning a fixed binary code to each symbol and, thereby, *tends to produce better compression ratios.*

- The coding process uses a one-dimensional table of probabilities instead of a tree structure.

- Given an alphabet of n symbols, there are an infinite number of messages that are possible. *Each message is mapped to a unique real number in the interval [0,1).*

- The interval contains an infinite amount of real numbers, so it must be possible to code any message uniquely to one number in the interval.

- The interval is first set at [0,1) for the first symbol, and then partitioned according to the symbol probabilities.

Arithmetic Coding Encoder

```
BEGIN
    low = 0.0;    high = 1.0;    range = 1.0;
    initialize symbol;           // so symbol != terminator

    while (symbol != terminator)
        {
          get (symbol);
          low =  low + range * Range_low(symbol);
          high = low + range * Range_high(symbol);
          range = high - low;
        }

    output a code so that low <= code < high;
END
```

- Example -1

Symbol statistics

Symbol	Probability	Cumulative Distribution	low	high
A	0.4	0.4	0	0.4
B	0.4	0.8	0.4	0.8
L	0.2	1.0	0.8	1.0

Encoding Sequence BALL

Encode ‘B’: low=0+0.4*1=0.4 high=0+0.8*1=0.8

Encode ‘A’: low=0.4+(0)*(0.4)=0.4 high=0.4+(0.4)(0.4)=0.56

Encode ‘L’: low=0.4+(0.8)*(0.16)=0.528 high=0.4+(1.0)(0.16)=0.56

Encode ‘L’: low=0.528+(0.8)*(0.032)=0.5536 high=0.528+(1.0)(0.032)=0.56

Generating Codeword for Encoder

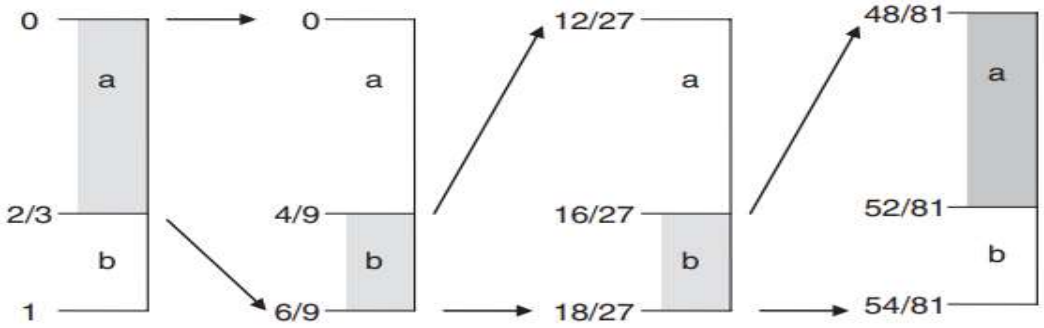
- By then low and high are 0.5536 and 0.56, respectively, It is apparent that finally we have, range = $P_B \times P_A \times P_L \times P_L = 0.4 \times 0.4 \times 0.2 \times 0.2 = 0.0064$
- Following algorithm ensures that the shortest binary codeword is found if low and high are the two ends

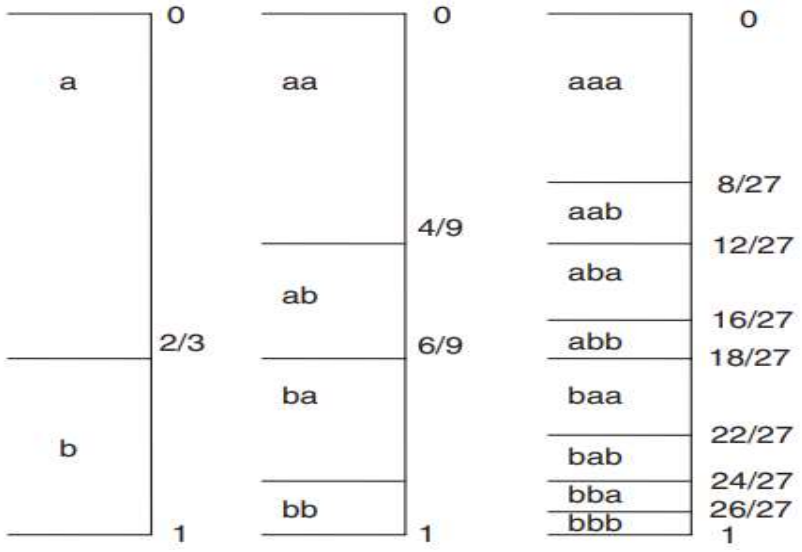
```
BEGIN
    code = 0;
    k = 1;
    while (value(code) < low)
        {
          assign 1 to the kth binary fraction bit;
          if (value(code) > high)
             replace the kth bit by 0;
          k = k + 1;
        }
END
```

- In the worst case, the shortest codeword in arithmetic coding will require k bits to encode a sequence of syr

$$k = \lceil \log_2 \frac{1}{\text{range}} \rceil = \lceil \log_2 \frac{1}{\prod_i P_i} \rceil$$

- Example -2 Two symbols a, b having probabilities as follows P(a) 2/3, P(b) 1/3. To encode the message of length 4 “abba”





Adaptive Arithmetic Coding

- Like adaptive Huffman coding, adaptive arithmetic coding is also possible.
- The difference between the two is that there is no need to keep a tree for the codewords. ***The only information that needs to be synchronized is the frequency of occurrence of the symbols.***
- Unlike the previous example, the ***statistics table will be updated as symbols are encoded***. The symbols are also updated when symbols are decoded.

- Initial table

symbol	frequency	probability	low	high
A	4	0.4	0	0.4
B	4	0.4	0.4	0.8
L	2	0.2	0.8	1

Encode ‘B’: low=0+0.4*1=0.4 high=0+0.8*1=0.8

Table after ‘B’ is encoded

symbol	frequency	probability	low	high
A	5	4/11	0	4/11
B	4	5/11	4/11	9/11
L	2	2/11	9/11	1

Encode ‘A’: low=0.4+(0)*(0.4)=0.4 high=0.4+(0.4)(4/11)=6/11

Table after ‘A’ is encoded

symbol	frequency	probability	low	high
A	5	5/12	0	5/12
B	5	5/12	5/12	10/12
L	2	2/12	10/12	1

Encode ‘L’: low=0.4+(8/55)*(10/12)=86/165
high=0.4+(1.0)(8/55)=0.56

Table after ‘L’ is encoded

symbol	frequency	probability	low	high
A	5	5/13	0	5/13
B	5	5/13	5/13	10/13
L	2	3/13	10/13	1

Encode ‘L’: low=(86/165)+(10/13)*(44/1815)=0.53986
high=(86/165)+(1)(44/1815)=0.54545

Assignment

- A,B, and C, P(A) = 0.5, P(B) = 0.25 and P(C) =0.25, Find out the arithmetic code for BACA.
- Ans: [0.59375, 0.609375)

Multimedia Systems
Lecture – 24

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Dictionary-Based Coding

- Dictionary-based coding techniques replace input strings with a code to an entry in a dictionary.
- The most well known dictionary-based techniques are *Lempel-Ziv* Algorithms and their variants.
- The *Lempel-Ziv-Welch (LZW)* algorithm employs an adaptive, dictionary-based compression technique.
- Unlike variable-length coding, in which the lengths of the codewords are different, LZW uses fixed-length codewords to represent variable length strings of symbols/characters that commonly occur together, such as words in English text.

- As in the other adaptive compression techniques, the *LZW encoder and decoder builds up the same dictionary dynamically* while receiving the data—the encoder and the decoder both develop the same dictionary.
- Since a single code can now represent more than one symbol /character, data compression is realized.
- LZW proceeds by placing longer and longer repeated entries into a dictionary, then emitting the code for an element rather than the string itself, if the element has already been placed in the dictionary.
- The predecessors of LZW are LZ77 and LZ78.
- LZW is used in many applications, such as *UNIX compress, GIF for images, WinZip*, and others.

LZW Compression

```
BEGIN
   s = next input character;
   while not EOF
      {
        c = next input character;

        if s + c exists in the dictionary
           s = s + c;
        else
          {
            output the code for s;
            add string s + c to the dictionary with a new code;
            s = c;
          }
      }
   output the code for s;
END
```

Example (LZW Compression for String ABABBABCABABBA)

- Let us start with a very simple dictionary (also referred to as a string table), initially containing only three characters, with codes as follows:

code	string
1	A
2	B
3	C

- Now if the input string is ABABBABCABABBA, the LZW compression algorithm works as follows:

s	c	output	code	string
			1	A
			2	B
			3	C
A	B	1	4	AB
B	A	2	5	BA
A	B			
AB	B	4	6	ABB
B	A			
BA	B	5	7	BAB
B	C	2	8	BC
C	A	3	9	CA
A	B			
AB	A	4	10	ABA
A	B			
AB	B			
ABB	A	6	11	ABBA
A	EOF	1		

- The output codes are 1 2 4 5 2 3 4 6 1. Instead of 14 characters, only 9 codes need to be sent.
- If we assume each character or code is transmitted as a byte, that is quite a saving (*the compression ratio would be 14/9 = 1.56*).
- LZW is an adaptive algorithm, in which the encoder and decoder independently build their own string tables. Hence, there is no overhead involving transmitting the string table.
- The above example is replete with a great deal of redundancy in the input string, which is why it achieves compression so quickly.
- In general, savings for LZW would not come until the text is more than a few hundred bytes long.

LZW Decompression

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;
          output entry;
          if (s != NIL)
             add string s + entry[0] to dictionary
             with a new code;
          s = entry;
        }
END
```

- Example (LZW decompression for string ABABBABCABABBA).
- Input codes to the decoder are 1 2 4 5 2 3 4 6 1. The initial string table is identical to what is used by the encoder. The LZW decompression algorithm then works as follows:

s	k	entry/output	code	string
			1	A
			2	B
			3	C
NIL	1	A		
A	2	B	4	AB
B	4	AB	5	BA
AB	5	BA	6	ABB
BA	2	B	7	BAB
B	3	C	8	BC
C	4	AB	9	CA
AB	6	ABB	10	ABA
ABB	1	A	11	ABBA
A	EOF			