Multimedia Systems
Lecture – 21

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

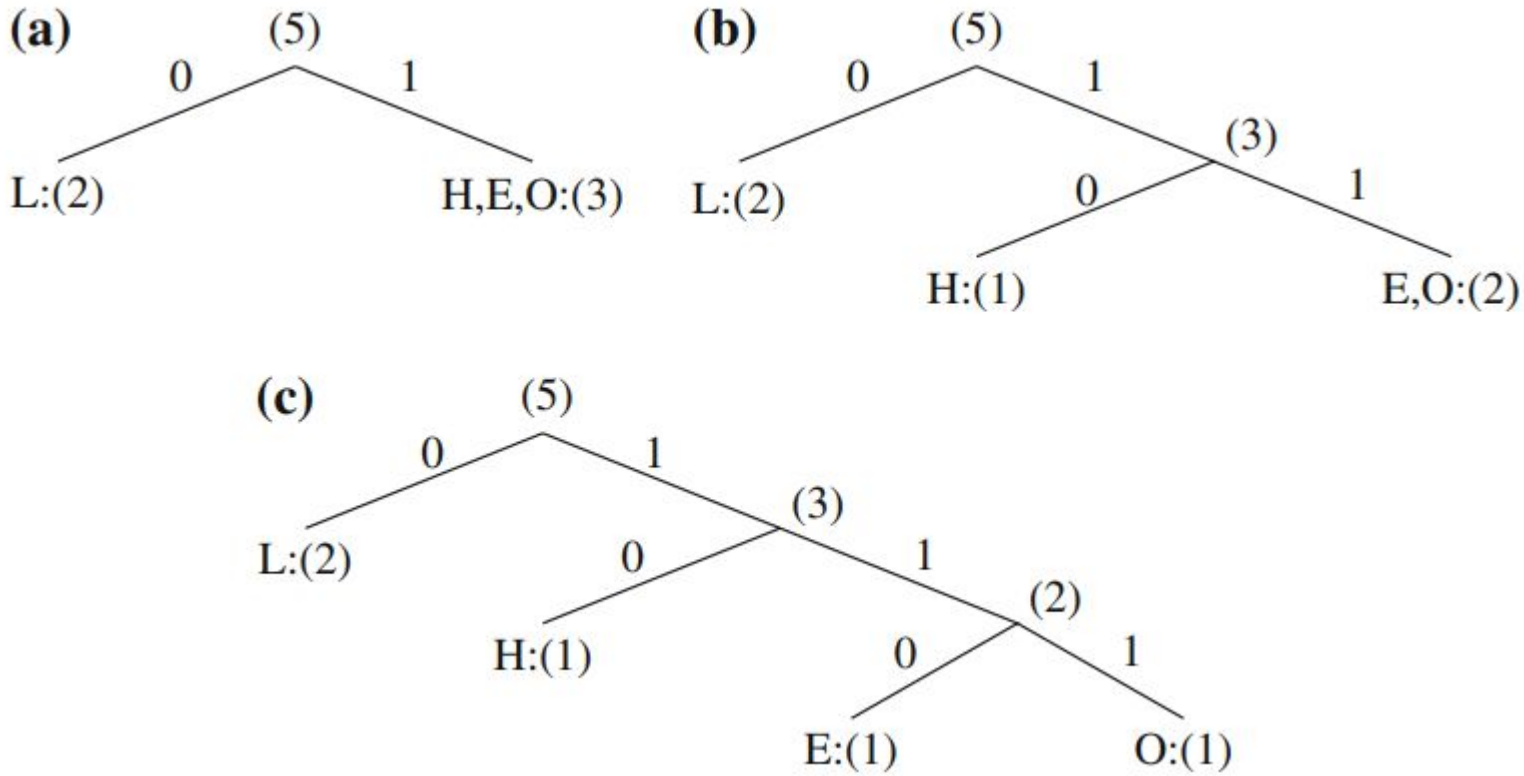
Variable-Length Coding

- 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

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

Huffman Coding

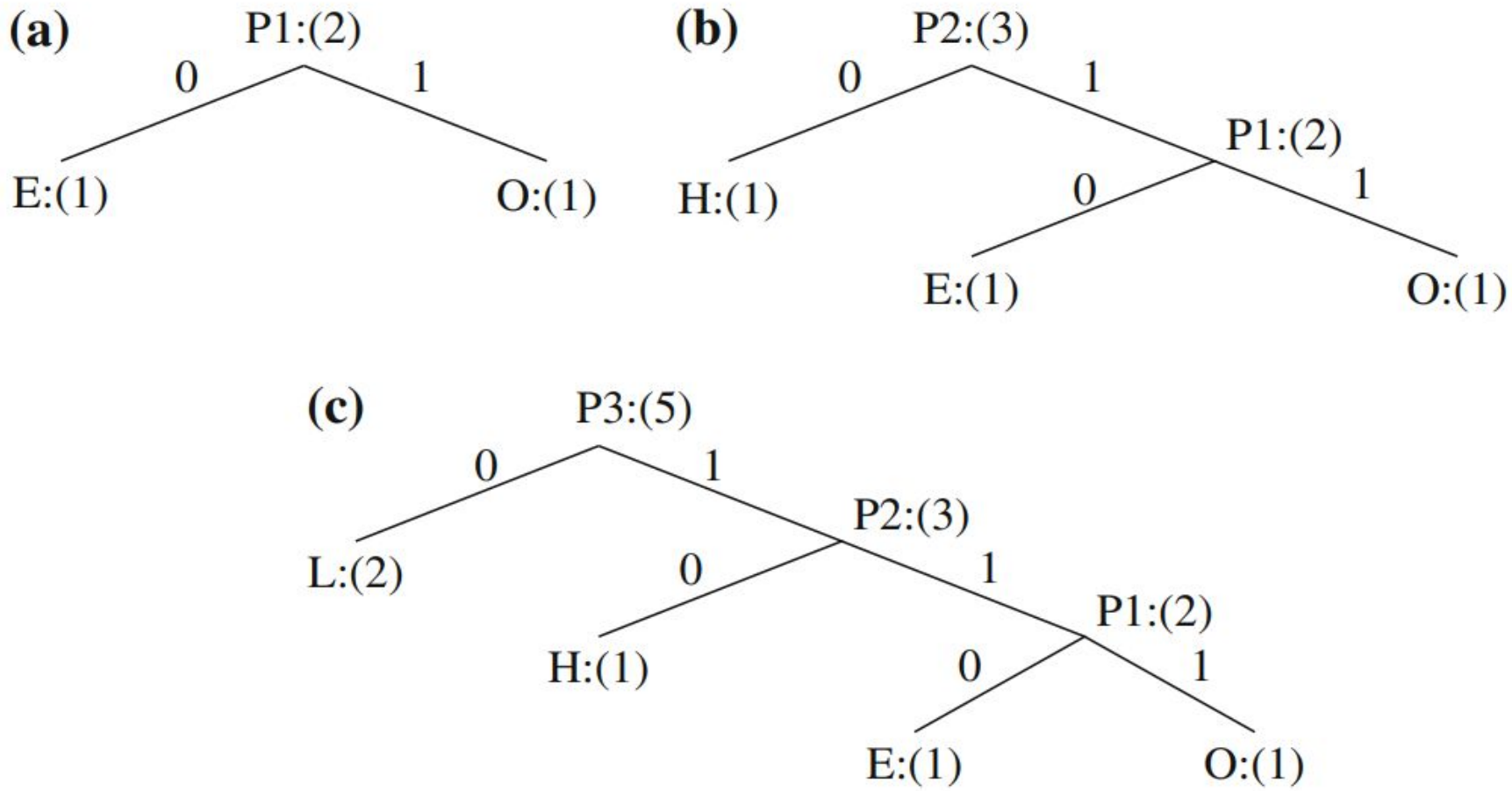
- First presented by Huffman in a 1952.
- This method attracted an overwhelming amount of research and has been adopted in many important and/or commercial applications, such as fax machines, JPEG, and MPEG.
- In contradistinction to Shannon–Fano, which is top-down, the encoding steps of the Huffman 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.

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 – 21,22

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

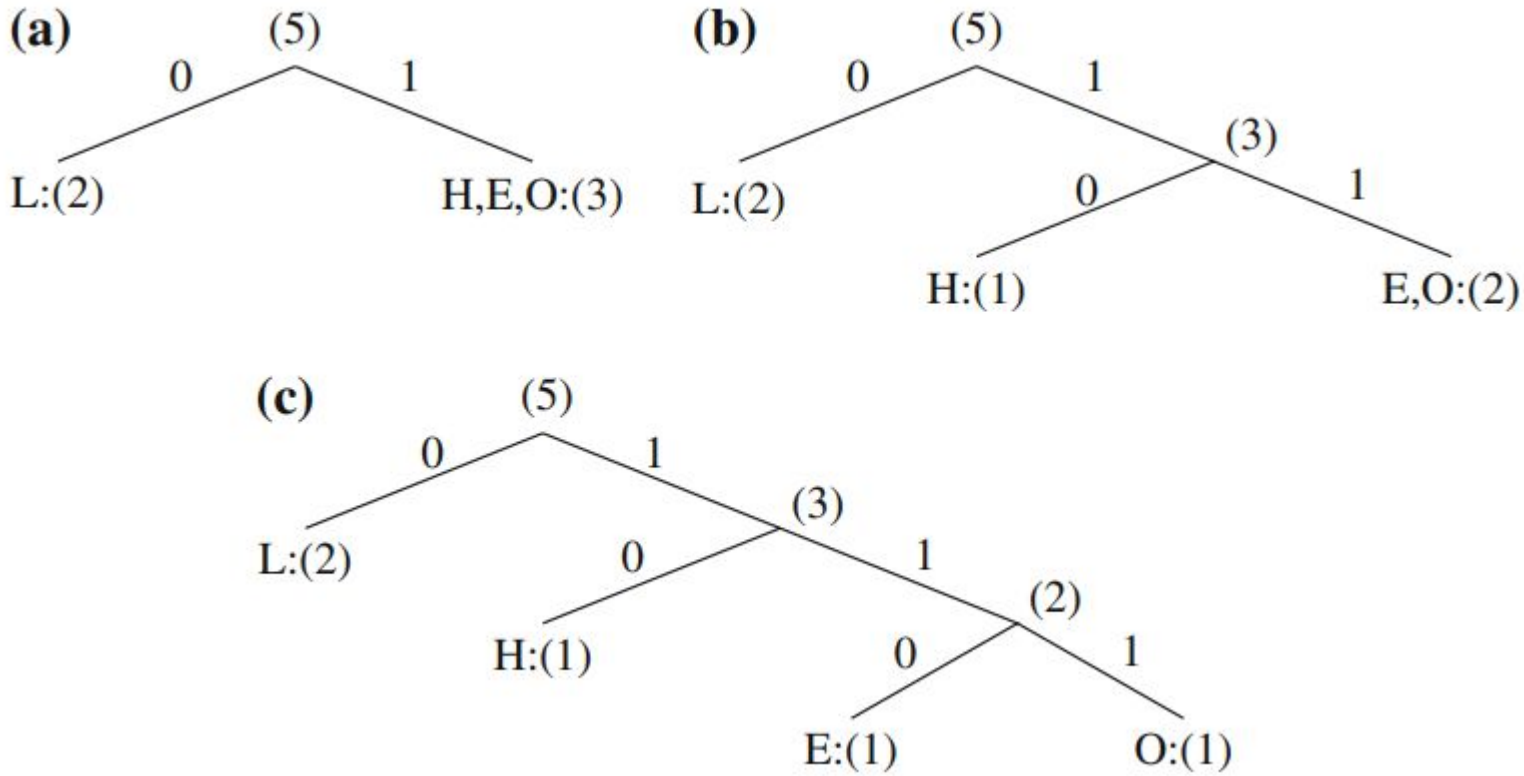
Variable-Length Coding

- 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

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

Huffman Coding

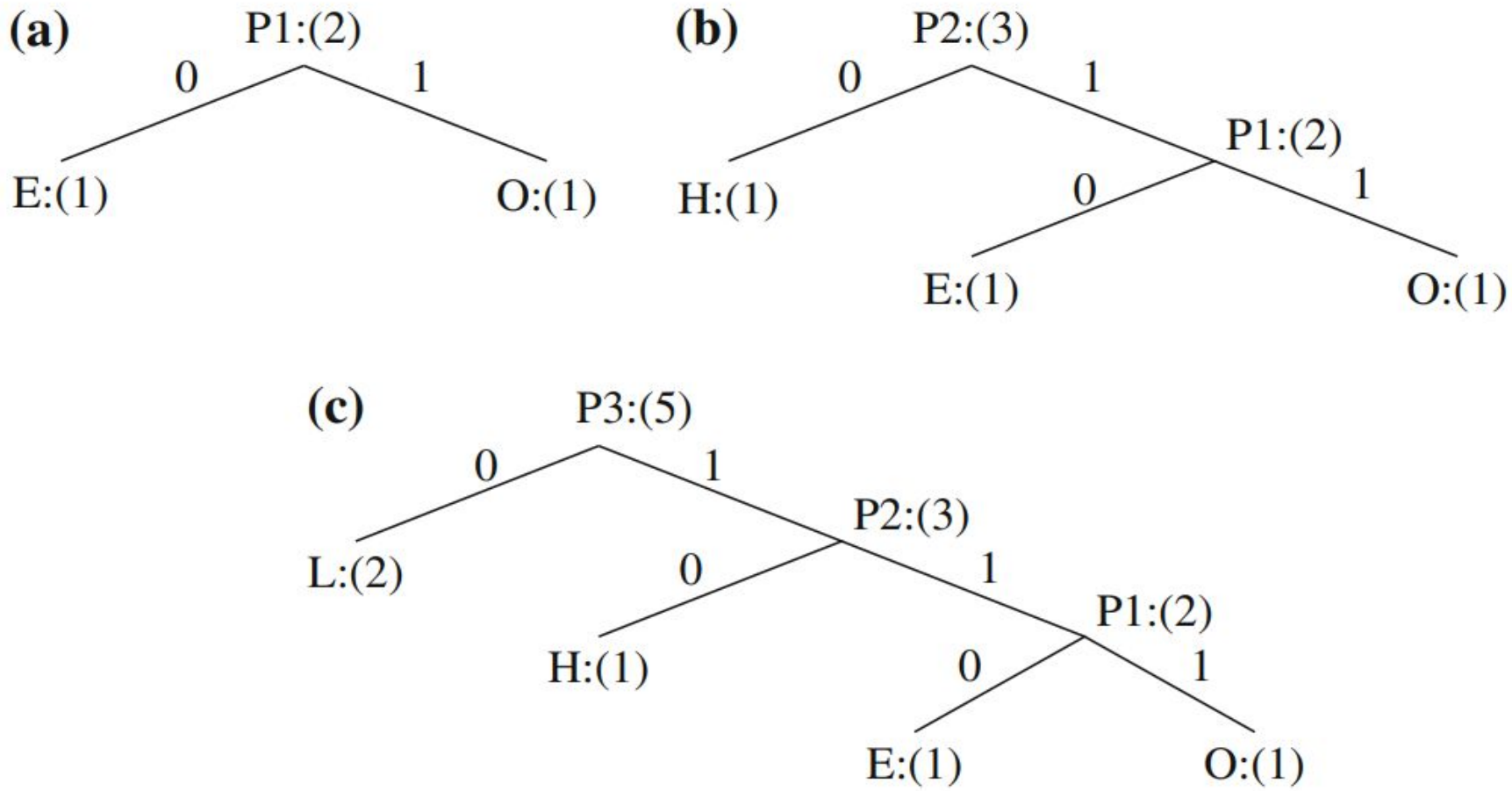
- First presented by Huffman in a 1952.
- This method attracted an overwhelming amount of research and has been adopted in many important and/or commercial applications, such as fax machines, JPEG, and MPEG.
- In contradistinction to Shannon–Fano, which is top-down, the encoding steps of the Huffman 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.

- If correct probabilities (“prior statistics”) are available and accurate, the Huffman coding method produces good compression results.

- Decoding for the Huffman coding is trivial as long as the statistics and/or coding tree are sent before the data to be compressed (in the file header, say). This overhead becomes negligible if the data file is sufficiently large.

Properties of Huffman Coding

- **Unique Prefix Property**:
  - No Human code is a prefix of any other Human code - precludes any ambiguity in decoding.
- **Optimality**:
  - *minimum redundancy code* - proved *optimal* for a given data model (i.e., a given, accurate, probability distribution) under certain conditions.
  - The two least frequent symbols will have the same length for their Human codes, differing only at the last bit.
  - Symbols that occur more frequently will have shorter Huffman codes than symbols that occur less frequently.
- Average Huffman code length for an information source *S* is strictly less than *entropy*+ 1
- So, we have $\eta \leq l < \eta + 1$.

Example

- Source alphabet A = {a, b, c, d, e}
- Probability distribution: {0.2, 0.4, 0.2, 0.1, 0.1}
- Code: {01, 1, 000, 0010, 0011}

- Entropy:

  $H(S) = -(0.2*\log_2(0.2)*2 + 0.4*\log_2(0.4)+0.1*\log_2(0.1)*2)$

  $= 2.122$ bits / symbol

- Average Huffman codeword length:

  $L = 0.2*2+0.4*1+0.2*3+0.1*4+0.1*4 = 2.2$ bits / symbol

- In general: $H(S) \leq L < H(S) + 1$

Limitations of Huffman Code

- Need a probability distribution
  - Usually estimated from a training set
  - But the practical data could be quite different
- Hard to adapt to changing statistics
  - Must design new codes on the fly
  - Context-adaptive method still need predefined table
- Minimum codeword length is 1 bit
  - Serious penalty for high-probability symbols
  - Example: Binary source, P(0)=0.9
    - Entropy: -0.9*log2(0.9)-0.1*log2(0.1) = 0.469 bit
    - Huffman code: 0, 1 □ Avg. code length: 1 bit
    - More than 100% redundancy !!!
    - Joint coding is not practical for large alphabet.

Extended Huffman Code

- Code multiple symbols jointly
  - Composite symbol: (X1, X2, …, Xk)
  - Alphabet increased exponentioally: N^k

- Code symbols of different meanings jointly
- If the entropy of S is η, then the average number of bits needed for each symbol in S is now

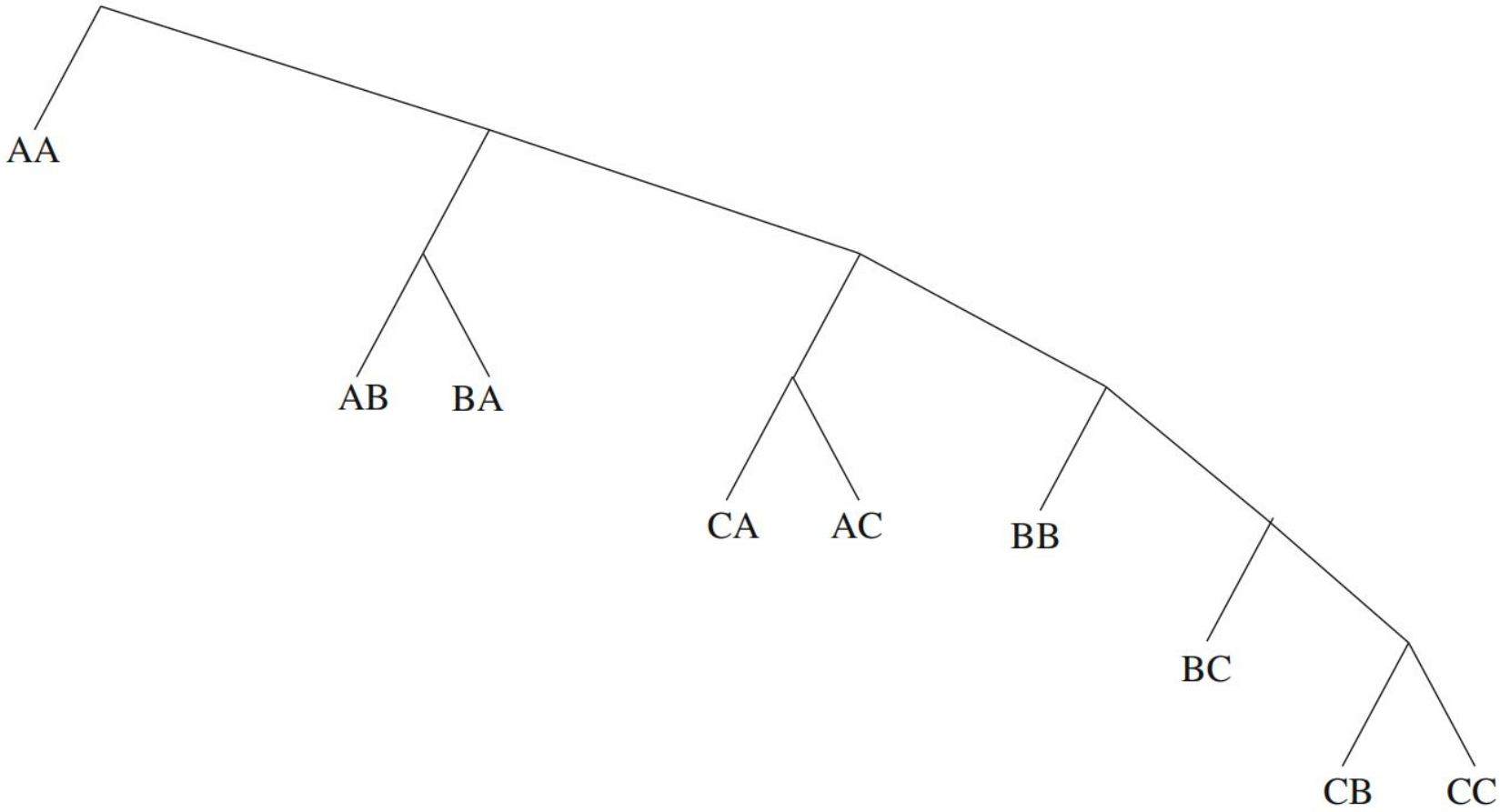
$$\eta \leq \bar{l} < \eta + \frac{1}{k}$$

Example

- Source alphabet S = {A, B, C,}
- Probability distribution: {0.6, 0.3, 0.1}
- Find out the average number of bits required and entropy ?

- Now let us extend this code by grouping symbols into 2-character groups—i.e., we use blocks of k = 2 characters.

The Extended Huffman Tree is



- Codeword bitlengths are then as follows:

Symbol group	Probability	Codeword	Bitlength
AA	0.36	0	1
AB	0.18	100	3
BA	0.18	101	3
CA	0.06	1100	4
AC	0.06	1101	4
BB	0.09	1110	4
BC	0.03	11110	5
CB	0.03	111110	6
CC	0.01	111111	6

Multimedia Systems
Lecture – 23

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.
- For the non-extended version of Huffman coding, order-0 model—that is, symbols/characters were treated singly.
- 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.

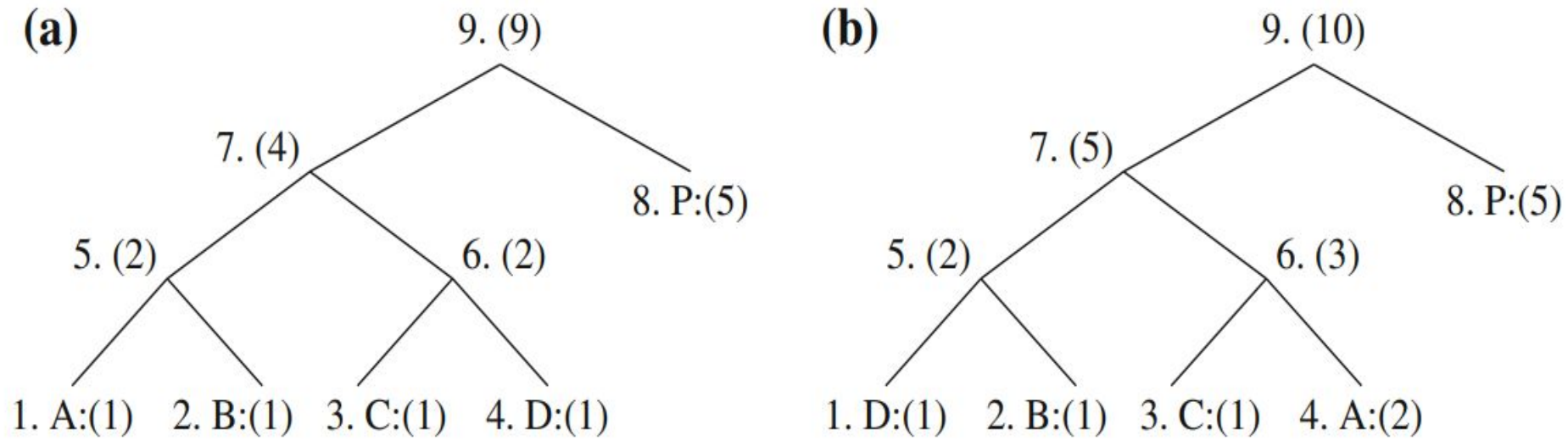
Procedure for Adaptive Huffman Coding

```
ENCODER
-------

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

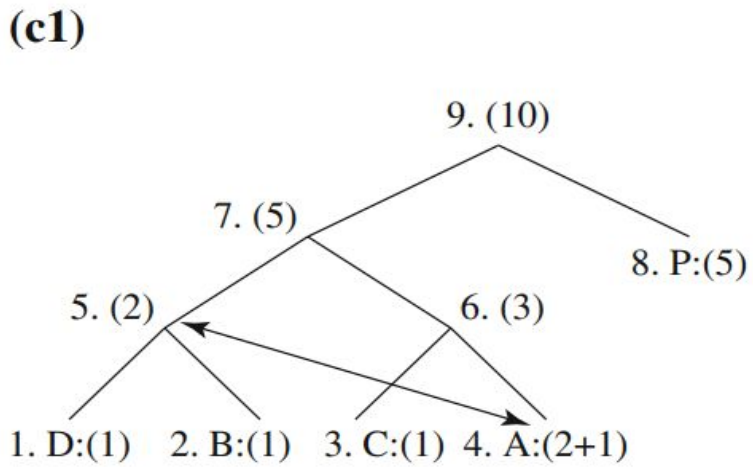
```
DECODER
-------

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

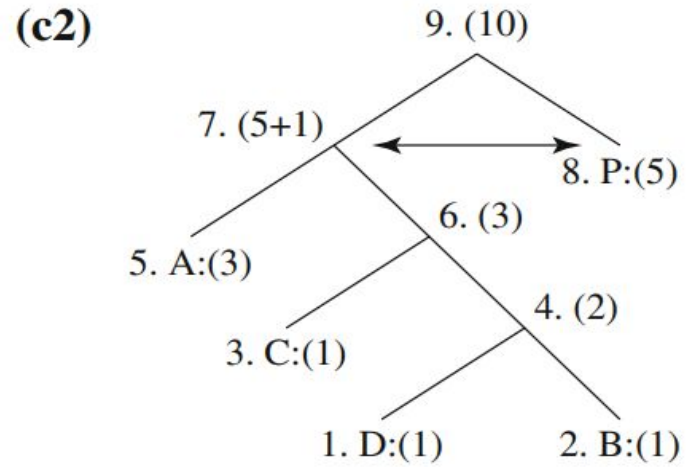
- **Initial_code** assigns symbols with some initially ***agreed-upon codes***, without any prior knowledge of the frequency counts for them.
- Example: ASCII
- **update_tree** is a procedure for constructing an adaptive Huffman tree. It basically does two things: it increments the frequency counts for 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. Note that 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.

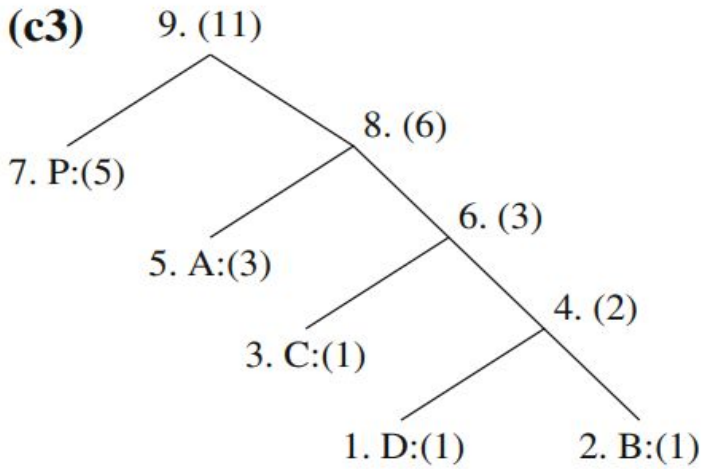
Node swapping for updating an adaptive Huffman tree: a a Huffman tree; b receiving 2nd ‘A’ triggered a swap;



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



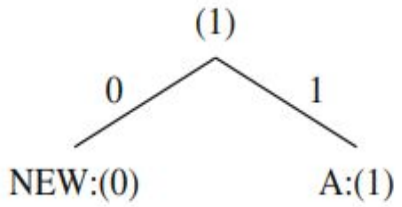




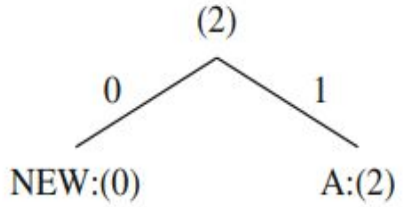
- Example: Adaptive Huffman Coding for Symbol String AADCCDD.
- Let us assume that the initial code assignment for both the encoder and decoder simply follows the ASCII order for the 26 symbols in an alphabet, A through Z.
- To improve the implementation of the algorithm, we adopt an additional rule: 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
⋮	⋮

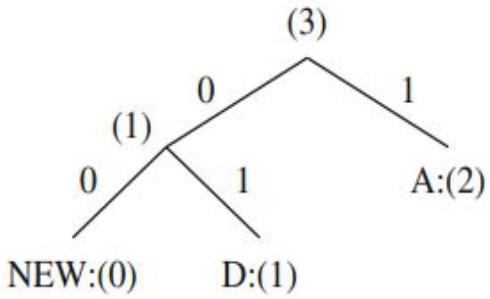
Adaptive Huffman tree for AADCCDD



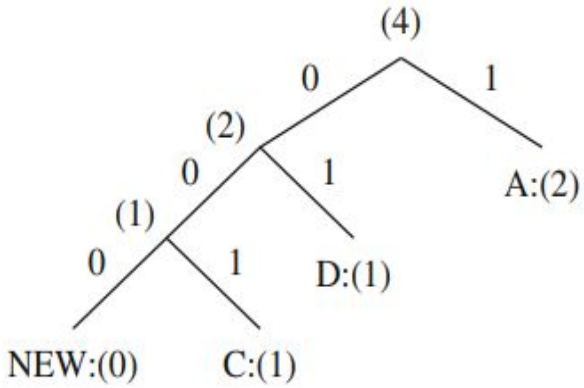
"A"



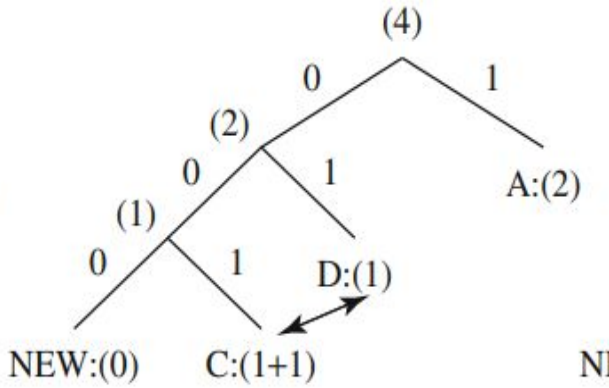
"AA"



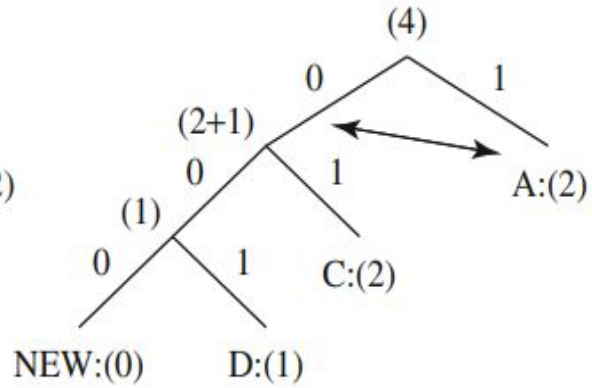
"AAD"



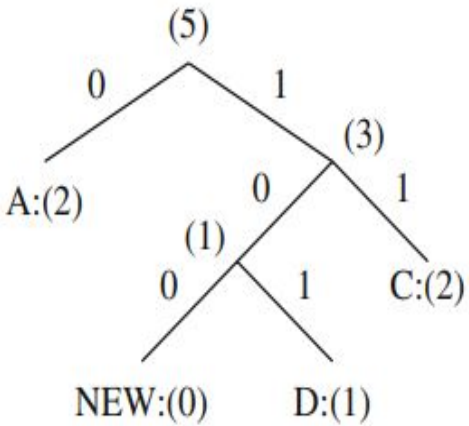
"AADC"



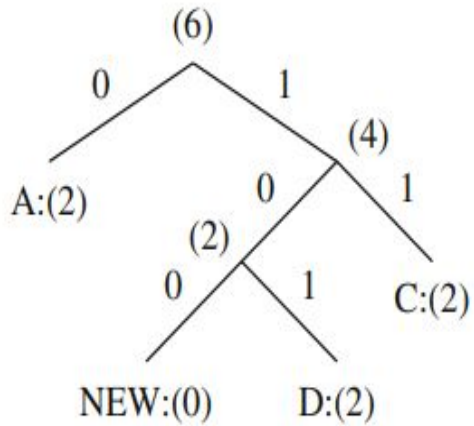
"AADCC" step 1



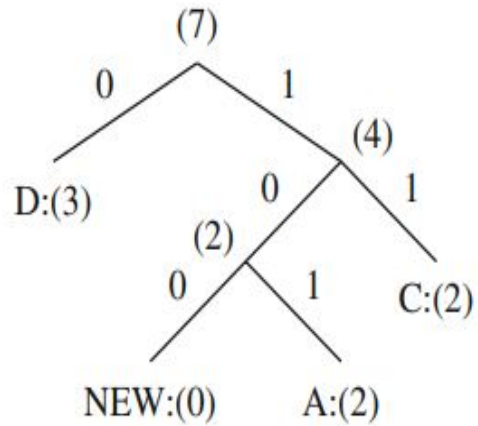
"AADCC" step 2



"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 – 25

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Arithmetic Coding

- Unlike the previous methods, arithmetic coding overcomes the requirement of every symbol to be coded independently.
- 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.
- It 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***.

- The algorithm can be outlined as follows:
  - *Divide the interval [0,1) into n segments corresponding to the n symbols; the segment of each symbol has a length proportional to its probability.*
  - *Each segment i has an upper bound U and lower bound L corresponding to the start of the segment and the end of the segment (U L Pi).*
  - *Choose the segment that corresponds to the first symbol in the message string. This is the new current interval with its computed new upper and lower bounds.*
  - *Divide the new current interval again into n new segments with length proportional to the symbols probabilities and compute the new intervals accordingly.*
  - *From these new segments, choose the one corresponding to the next symbol in the message.*
  - *Continue Steps 3 and 4 until the whole message is coded.*
  - *Represent the segment's value by a binary fraction in the final interval.*

- 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

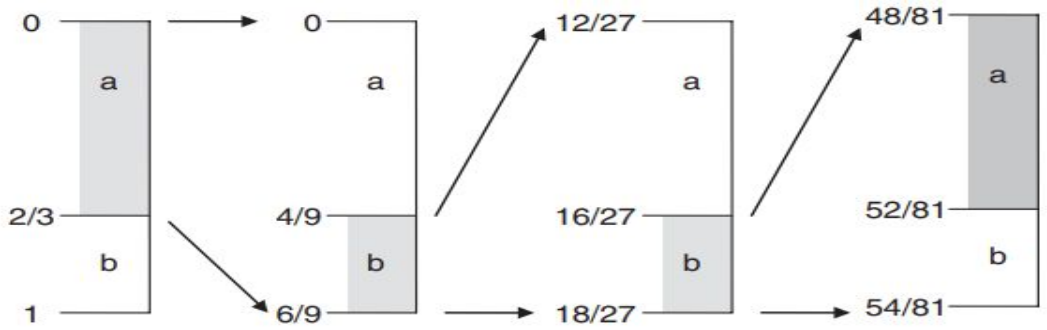
Encode ‘B’: low=0+0.4*1=0.4 high=0+0.8*1=0.8

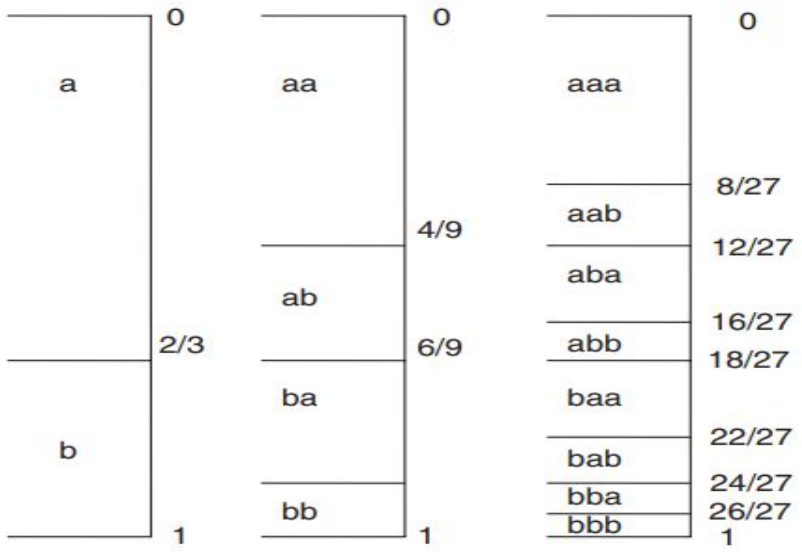
Encode ‘A’: low=0.4+(0)*(0.4)=0.4 high=0.4+(0.4)(0.4)=0.56

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

- 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”





- This process repeats for the third symbol *b* and fourth symbol *a* till we get the final interval [48/81, 52/81].
- This interval uniquely represents the message "*abba*" and thus any number in the interval can be chosen as a binary representation of the message.
- The number that is typically chosen is one that has the least number of bits in that interval. For the interval [48/81, 52/81], the binary representation of this range (up to eight binary decimals) is [0.10010111, 0.1010010].
- This interval can be represented by 0.101, which corresponds to the decimal 0.625.
- Thus, 101 should suffice as a code for "*abba*". This information is enough for the decoder to reconstruct the sequence "*abba*" having knowledge of the probability distributions.

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 – 26

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***.

- LZW encoder and decoder builds up the same dictionary dynamically while receiving the data.
- 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			

- A more careful examination of the above simple version of the LZW decompression algorithm will reveal a potential problem.
- In adaptively updating the dictionaries, the encoder is sometimes ahead of the decoder.
- For example, after the sequence ABABB, the encoder will output code 4 and create a dictionary entry with code 6 for the new string ABB.
- On the decoder side, after receiving the code 4, the output will be AB, and the dictionary is updated with code 5 for a new string, BA.
- Welch points out that the simple version of the LZW decompression algorithm will break down when the following scenario occurs.

- Assume that the input string is ABABBABCABBABBAX....
- The LZW encoder:

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	B			
ABB	A	6	10	ABBA
A	B			
AB	B			
ABB	A			
ABBA	X	10	11	ABBAX
		.		
		.		

- The sequence of output codes from the encoder (and hence the input codes for the decoder) is 1 2 4 5 2 3 6 10 ...
- The simple LZW decoder:

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	6	ABB	9	CA
ABB	10	???		

LZW Decompression (Modified)

```
BEGIN
    s = NIL;
    while not EOF
        {
          k = next input code;
          entry = dictionary entry for k;

          /* exception handler */
          if (entry == NULL)
             entry = s + s[0];

          output entry;
          if (s != NIL)
             add string s + entry[0] to dictionary
             with a new code;
          s = entry;
        }
END
```

Multimedia Systems
Lecture – 27

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Lossy Compression

- Entropy-coding or lossless compression has theoretical limits on the amount of compression that can be achieved.
- In certain situations, it might be necessary and appropriate to sacrifice some amount of information and thereby increase the compression obtained.
- For instance, human visual experiments on image perception have shown distortion introduced by image compression is still perceptually acceptable.
- In such cases, the original signal cannot be recovered in its entirety and there is a loss or distortion introduced.
- Most lossy compression schemes introduce a distortion because of quantizing the symbol code representations.
- Quantization is inherently lossy.

Transform Coding

- Transform coding techniques work by performing a mathematical transformation on the input signal that results in a different signal.
- These transformation changes the signal representation to a domain, which can result in reducing the signal entropy and, hence, the number of bits required to compress the signal.
- Transform coding techniques by themselves are not lossy; however, they frequently employ quantization after a transform.
- Transform techniques can be grouped as follows:
  - Frequency transforms—Discrete Fourier transforms, Hadamard transforms, Lapped Orthogonal transforms, Discrete Cosine transforms
  - Statistical transforms—Karhunen-Loeve transforms
  - Wavelet transforms—While similar to frequency transforms, these transforms work more efficiently because the input is transformed to a multiresolution frequency representation

Transform Coding

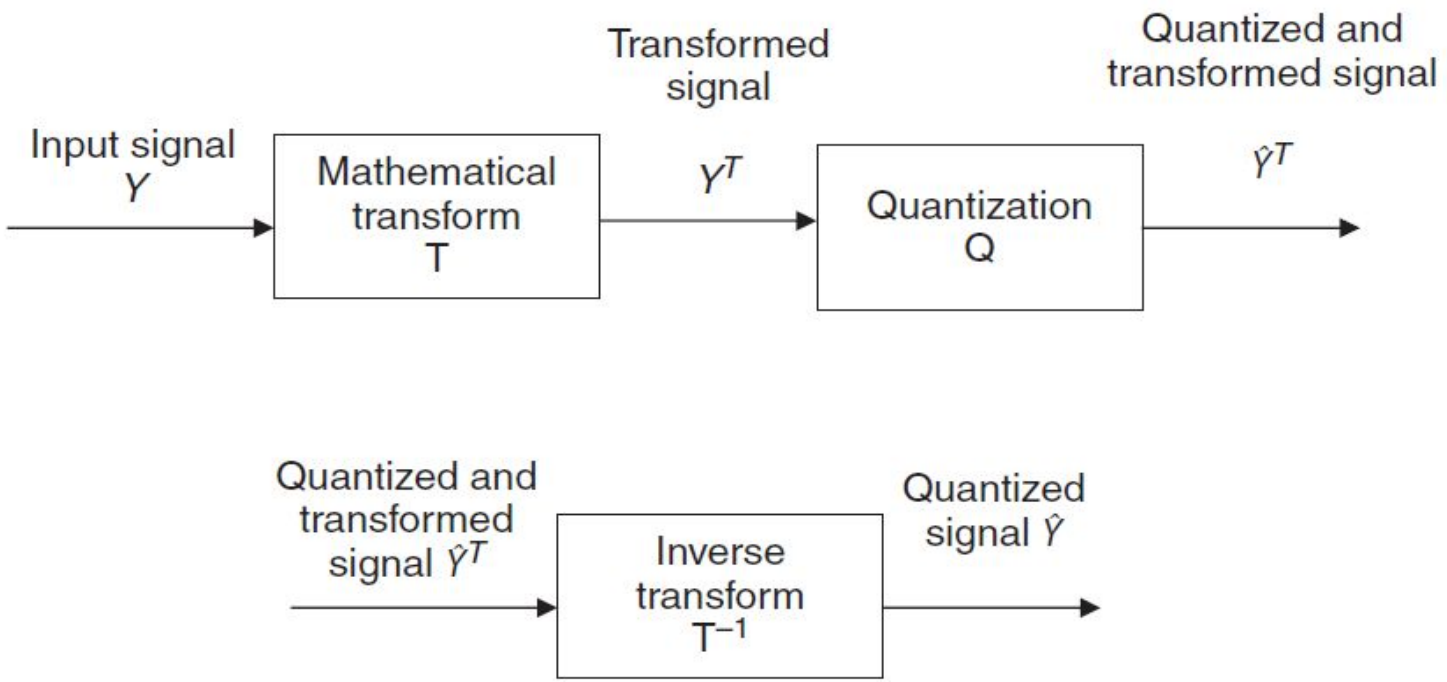


Image Compression Techniques

- An explosion in the availability of digital images.
- The need to efficiently process and store images in digital form.
- Image compression techniques can be purely lossless or lossy.
- Good image compression techniques often work as hybrid schemes.
- Based two important aspects:
  - Irrelevancy reduction: information associated with some pixels might be irrelevant and can, therefore, be removed. visual irrelevancy and application-specific irrelevancy.
  - Redundancy reduction: statistical redundancy because pixel values are not random but highly correlated, either in local areas or globally

- Local pixel correlation. Images are not a random collection of pixels, but exhibit a similar structure in local neighborhoods.
- Three magnified local areas are shown on the right

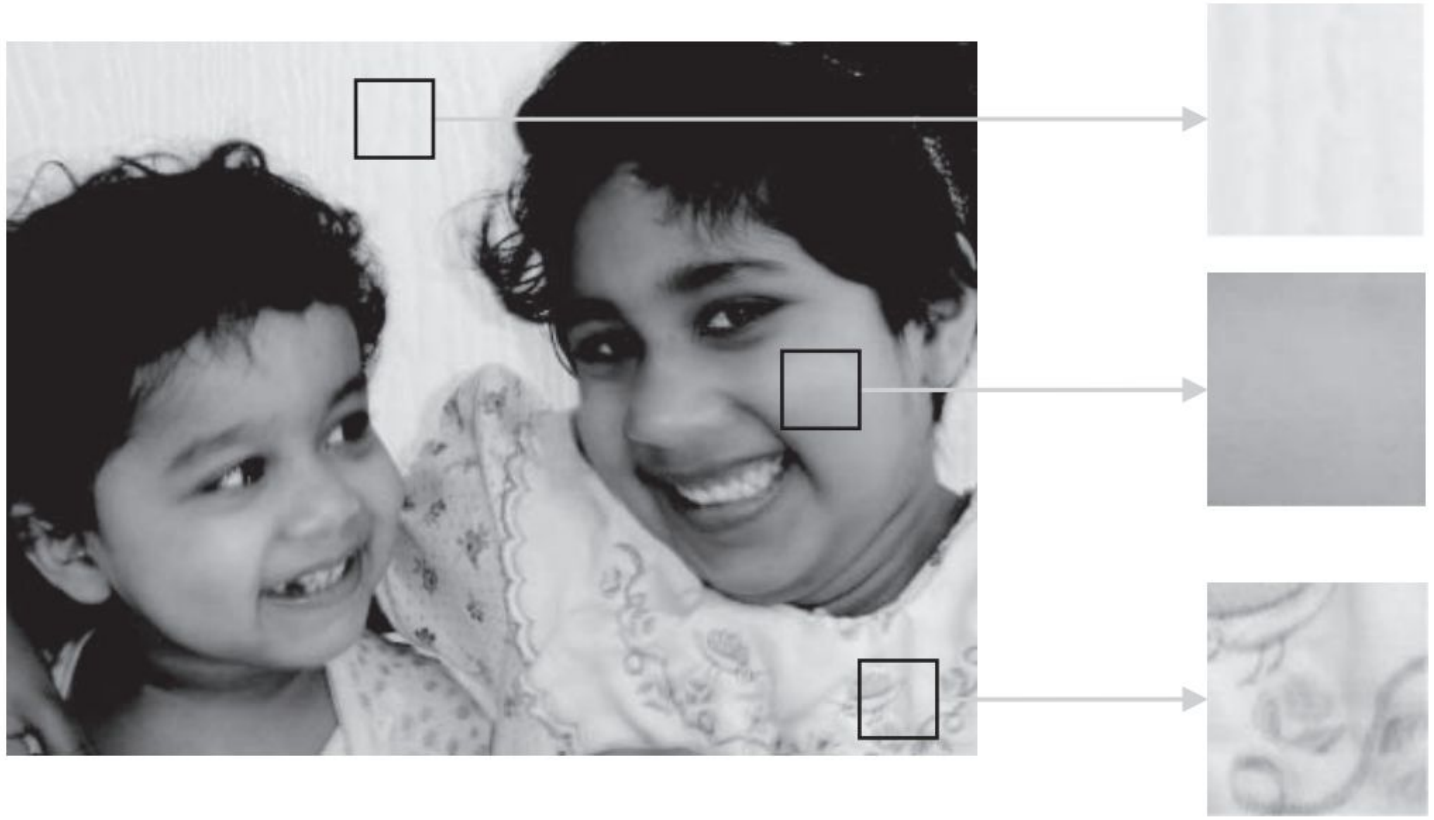
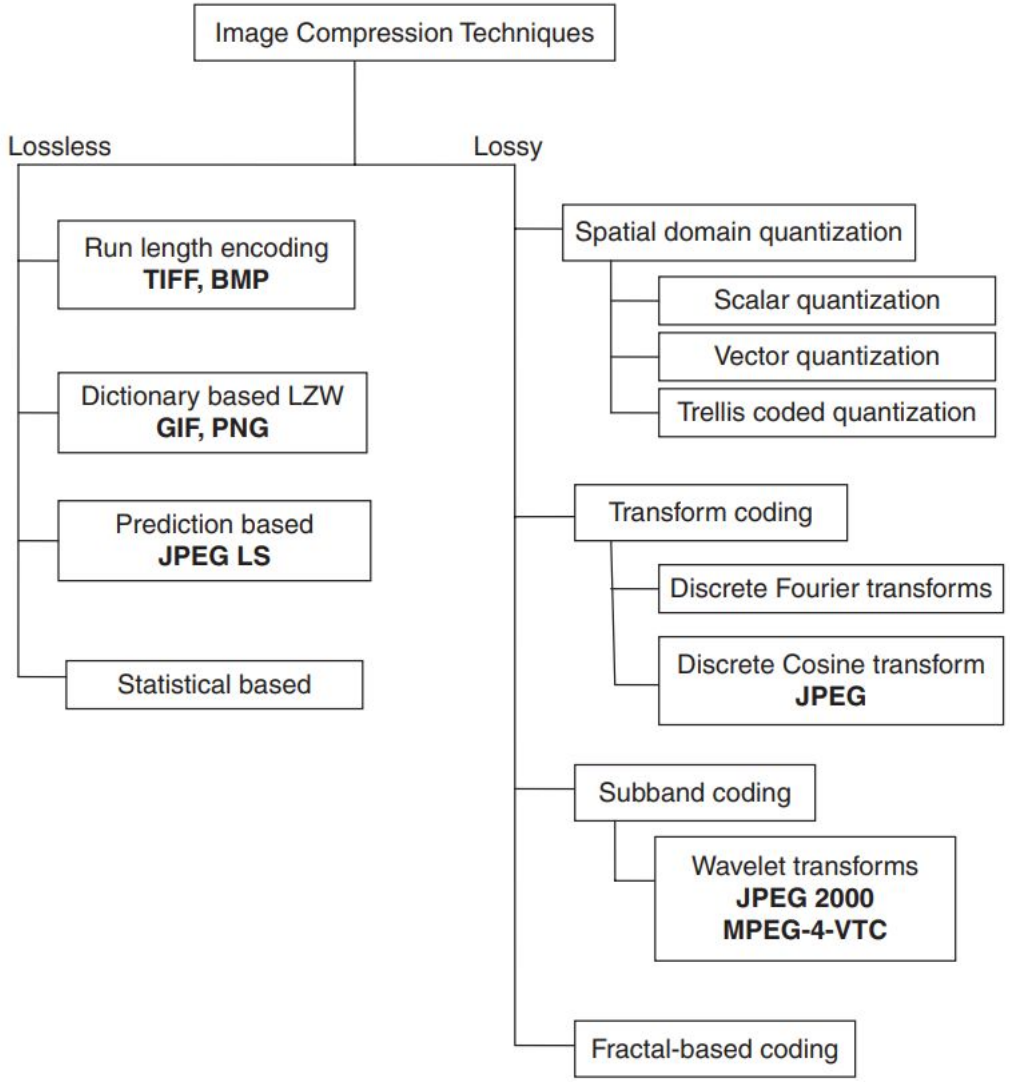


Image Compression Taxonomy



DCT Image Coding and the JPEG Standard

- The JPEG standard is based on a transform image coding technique that utilizes the *Discrete Cosine transform (DCT).*

- The DCT was chosen by the JPEG community because of its good ***frequency domain energy distribution*** for natural images, as well as its ease of adoption for efficient ***hardware-based computations***.

- To understand how compression occurs in the JPEG pipeline, it is imperative that the DCT behavior be well understood.

Discrete Cosine Transform (DCT)

- The Discrete Cosine Transform (DCT), a widely used transform coding technique, is able to perform decorrelation of the input signal in a data-independent manner.

Definition of DCT

- Given a function f (i, j) over two integer variables i and j (a piece of an image), the 2D DCT transforms it into a new function F(u, v), with integer u and v running over the same range as i and j.
- The general definition of the transform is

$$F(u,v) = \frac{2\,C(u)\,C(v)}{\sqrt{MN}} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \cos\frac{(2i+1)u\pi}{2M} \cos\frac{(2j+1)v\pi}{2N} f(i,j)$$

- where i, u = 0, 1,..., M − 1, j, v = 0, 1,..., N − 1, and the constants C(u) and C(v) are determined by

$$C(\xi) = \begin{cases} \frac{\sqrt{2}}{2} & \text{if } \xi = 0, \\ 1 & \text{otherwise.} \end{cases}$$

- In the JPEG image compression standard, an image block is defined to have dimension M = N = 8. Therefore, the definitions for the 2D DCT and its inverse (IDCT) in this case are as follows:
- 2D Discrete Cosine Transform (2D DCT)

$$F(u,v) = \frac{C(u)\,C(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}f(i,j),$$

where i, j, u, v = 0, 1,..., 7, and the constants C(u) and C(v) are determined.

- 2D Inverse Discrete Cosine Transform (2D IDCT)
- The inverse function is almost the same, with the roles of f (i, j) and F(u, v) reversed, except that now C(u)C(v) must stand inside the sums:

$$\tilde{f}(i,j) = \sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)\,C(v)}{4}\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}F(u,v)$$

- where i, j, u, v = 0, 1,..., 7.

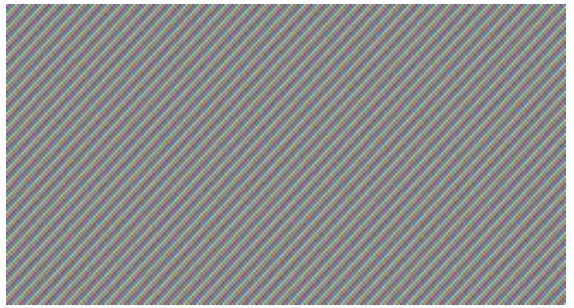
Multimedia Systems
Lecture – 28

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

- JPEG is a lossy image compression method.
- The effectiveness of the DCT transform coding method in JPEG relies on three major observations:
- Observation-1:
  - Useful image contents change relatively slowly across the image— that is, it is unusual for intensity values to vary widely several times in a small area—for example, in an 8 × 8 image block.
  - ***Spatial frequency*** indicates how many times pixel values change across an image block.
  - The DCT formalizes this notion with a measure of how much the image contents change in relation to the number of cycles of a cosine wave per block.

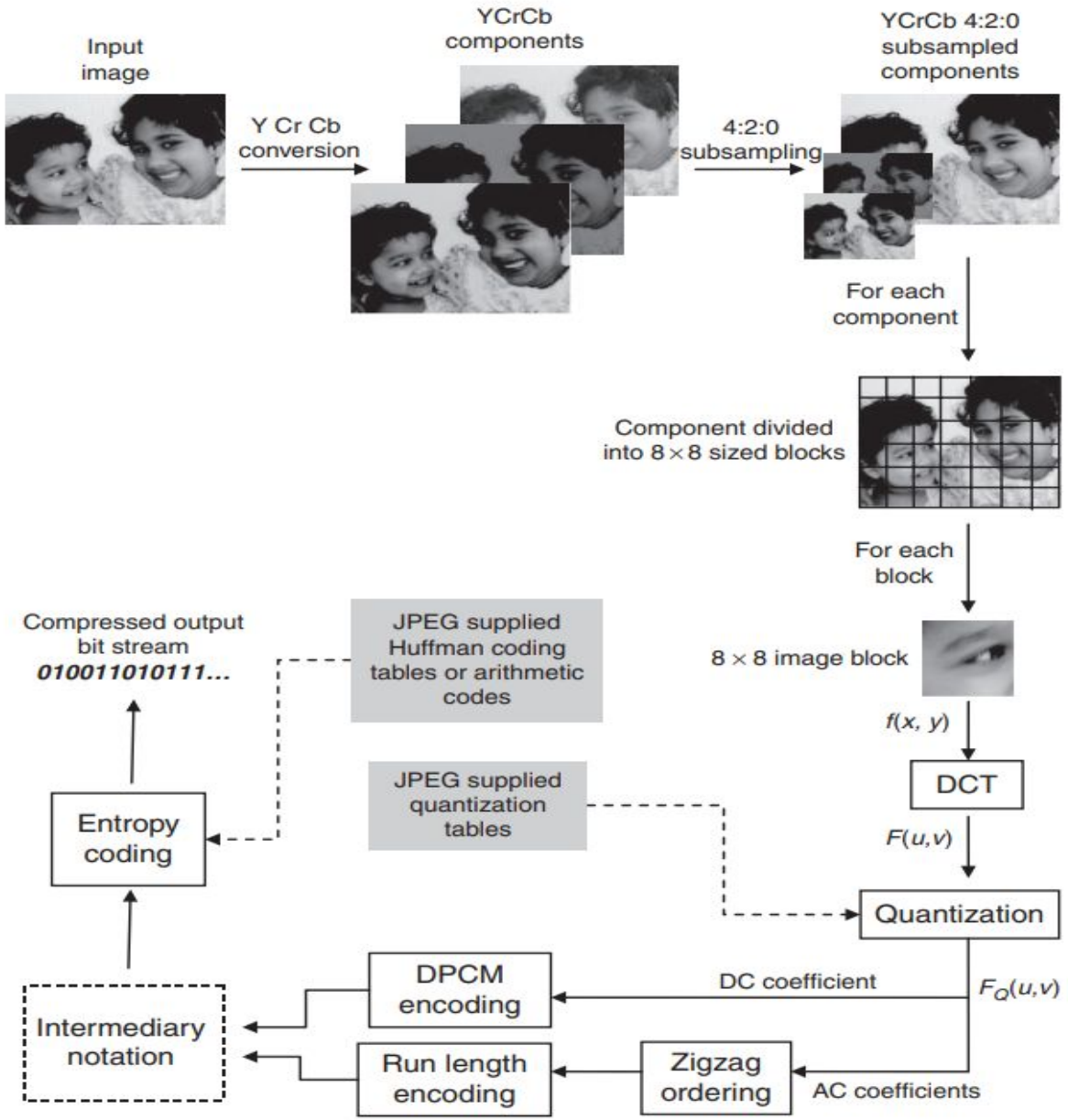
- Observation-2:
  - Psychophysical experiments suggest that humans are much less likely to notice the loss of very high-spatial frequency components than lower frequency components.
- Observation-3:
  - Visual acuity (accuracy in distinguishing closely spaced lines) is much greater for gray (“black and white”) than for color.

Low spatial frequency



High spatial frequency

JPEG Compression pipeline



- The steps followed in JPEG compression are as follows.
- ***Step 1:*** The image is first converted to the ***YCrCb*** format to decouple the image chrominance from the luminance.
- ***Step 2:*** The YCrCb representation undergoes a ***4:2:0*** subsampling, where the chrominance channels Cr and Cb are subsampled to one-fourth the original size.
- ***Step 3:*** Each channel (Y, Cr, and Cb) is processed independently. Each channel is divided into ***8×8*** blocks.
  - On the average, the 8×8 size seems to be the most optimal area for spatial and spectral correlation that the DCT quantization can exploit.
  - Smaller sizes ***increase the number of blocks*** in an image, and larger sizes ***reduce the correlation seen among pixels***.
  - If the image width (or height) is not a multiple of 8×8, the boundary blocks get ***padded with zeros*** to attain the required size.
  - The blocks are processed independently.

- ***Step 4:*** Each 8×8 block (for all the channels) undergoes a ***DCT*** transformation, which takes the image samples f(x,y) and computes frequency coefficients F(u,v).
  - The first coefficient F(0, 0), is normally the highest, and is called the ***DC coefficient***.
  - Most of the energy in natural photographs is concentrated among the lowest frequencies.
  - The remaining coefficients are called ***AC coefficients***.
- ***Step 5:*** The DCT coefficients F(u,v) are quantized using a quantization table supplied by JPEG.
  - The quantization table values are not random.
  - They are based on experimental evaluations with human subjects, which have shown that low frequencies are dominant in images, and the human visual system is more sensitive to loss in the low-frequency range.
  - The numbers in the ***low-frequency area are smaller*** and ***increase as you move toward the high-frequency coefficients*** in the other three corners.

178	187	183	175	178	177	150	183
191	174	171	182	176	171	170	188
199	153	128	177	171	167	173	183
195	178	158	167	167	165	166	177
190	186	158	155	159	164	158	178
194	184	137	148	157	158	150	173
200	194	148	151	161	155	148	167
200	195	172	159	159	152	156	154

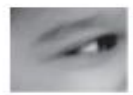
Pixel values $f(x, y)$

1359	46	61	26	38	−21	−5	−18
31	−35	−25	−11	13	10	12	−3
13	20	−17	−14	−11	−7	6	5
−5	5	2	−8	−11	−26	8	−4
10	15	−10	−16	−21	−7	8	7
−6	1	0	7	5	−7	−1	−3
−13	−8	1	10	8	4	−3	−4
−5	−5	−2	5	5	0	0	−3

DCT values $F(u,v)$

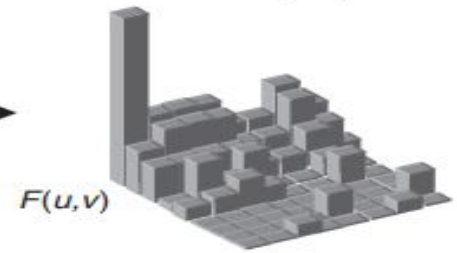
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization table



$f(x, y)$

DCT



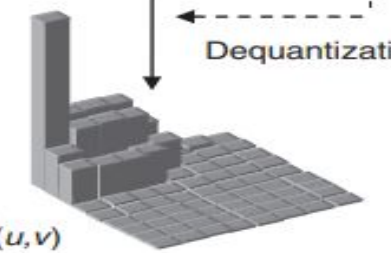
$F(u,v)$

Quantization

85	4	6	2	2	−1	0	0
3	−3	−2	−1	1	0	0	0
1	2	−1	−1	0	0	0	0
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$F_Q(u,v)$

Dequantization



$\hat{F}(u,v)$

Inverse DCT

$\hat{f}(x, y)$

192	185	178	152	193	162	155	190
181	172	162	154	187	164	159	192
183	168	150	158	181	167	162	190
198	177	150	155	177	169	161	182
202	180	148	148	171	167	159	175
193	176	145	141	162	162	156	170
195	184	155	145	159	156	150	164
209	200	170	154	160	153	144	156

1360	44	60	32	48	−40	0	0
36	−36	−28	−19	26	0	0	0
14	26	−16	−24	0	0	0	0
0	0	0	0	0	0	0	0
18	22	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

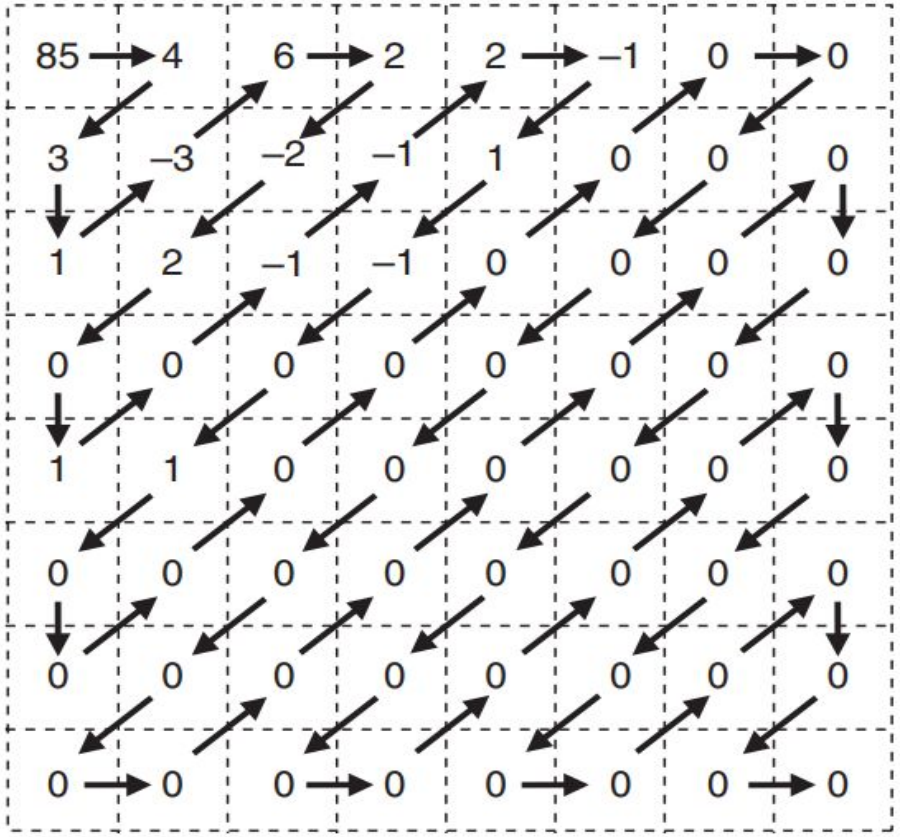
- Using this table, $F_Q(u,v)$ is computed as $$F_Q(u,v) = \left\lceil \frac{F(u,v)}{Q(u,v)} \right\rceil$$

where Q(u, v) is the value in the quantization table.

- After quantization, almost all the high-frequency $F_Q(u,v)$ are zero, while a few low frequency values remain.

- ***Step 6:*** The quantized coefficients $F_Q(u, v)$ are then encoded into an intermediary pattern.
  - DC $F_Q(0,0)$, which normally corresponds to the ***highest energy***, is treated differently when compared with the other higher-frequency coefficients, called AC coefficients.
  - The DC coefficients of the blocks are encoded using ***differential pulse code modulation (DPCM)***.
  - The AC coefficients are first scanned in a ***zigzag order***

Zigzag ordering of quantized AC coefficients. The resulting sequence has a much longer run of zeros and, hence, can be more efficiently entropy coded.



DC coefficient = 85

AC coefficient stream

4 3 1 -3 6 2 -2 2 0 1 0 -1 -1 2
-1 1 -1 0 1 0 0 0 0 0 0 0 0 0 ...

- ***Step 7:*** The next step is to entropy code the run of DC and AC coefficients. Prior to entropy coding, these coefficients are converted into intermediary representations.
  - For the representation of the DC coefficient, it is first DPCM coded with the previous block's DC value and the difference is represented as a ***2-tuple***, which shows the ***size in bits*** used to encode the DPCM difference and the ***amplitude of the DPCM difference***.
  - Example: Here, the quantized DC value of the block is 85. Assuming that the DC value of the previous block was 82, the DPCM difference is 3, which needs two bits to encode. Therefore, the intermediary notation of the DC coefficient is <2><3>.

- For AC coefficients, the intermediary notation is used only for the ***nonzero AC coefficients***.
- Each nonzero AC coefficient is again represented by two symbols.
  - The first symbol here represents two pieces of information—***runlength*** and ***size***.
  - The runlength is the number of zeros in the run that precede the non-zero coefficient in the zigzag sequence and the size is the number of bits used to encode the amplitude of the non-zero AC coefficient.
  - The second symbol encodes the amplitude of the nonzero AC coefficient.
  - In our example, the first nonzero AC coefficient has a value of 4 and there are no zeros preceding it, so the run length is 0. The amplitude of 4 needs 3 bits to encode. Therefore, the intermediary representation is <0,3><4>.

DC coefficient representation:

symbol -1	symbol -2
<SIZE>	<AMPLITUDE>

AC coefficient representation:

symbol -1	symbol -2
<RUNLENGTH, SIZE>	<AMPLITUDE>

Intermediary stream

<2><3> <0,3><4> <0,2><3> <0,1><1> <0,2><-3> <0,3><6>
<0,2><2> <0,2><-2> <0,2><2> <1,1><1> <1,1><-1> <0,1><-1>
<0,2><2> <0,1><-1> <0,1><1> <0,1><-1> <1,1><1> EOB

- ***Step 8:*** The intermediary representations are then entropy coded using codes supplied by the JPEG organization.
- The first symbol in the intermediary representation for both DC and AC coefficients is encoded using ***Huffman coding***.
- The second symbol, which is the amplitude, is encoded using ***variable length integer code***.
- To save bits, RUNLENGTH and SIZE are allocated only ***4 bits*** each and squeezed into a single byte.
- The 4-bit RUNLENGTH can represent only zero-runs of length ***0 to 15***. Occasionally, the zero run-length exceeds 15; then a special extension code, (15, 0), is used for Symbol 1.

Baseline entropy coding details—size category

Size	Amplitude
1	−1, 1
2	3, −2, 2, 3
3	−7 .. −4, 4 .. 7
4	−15 .. −8, 8 .. 15
.	.
.	.
.	.
10	−1023 .. −512, 512 .. 1023

Binary Stream:

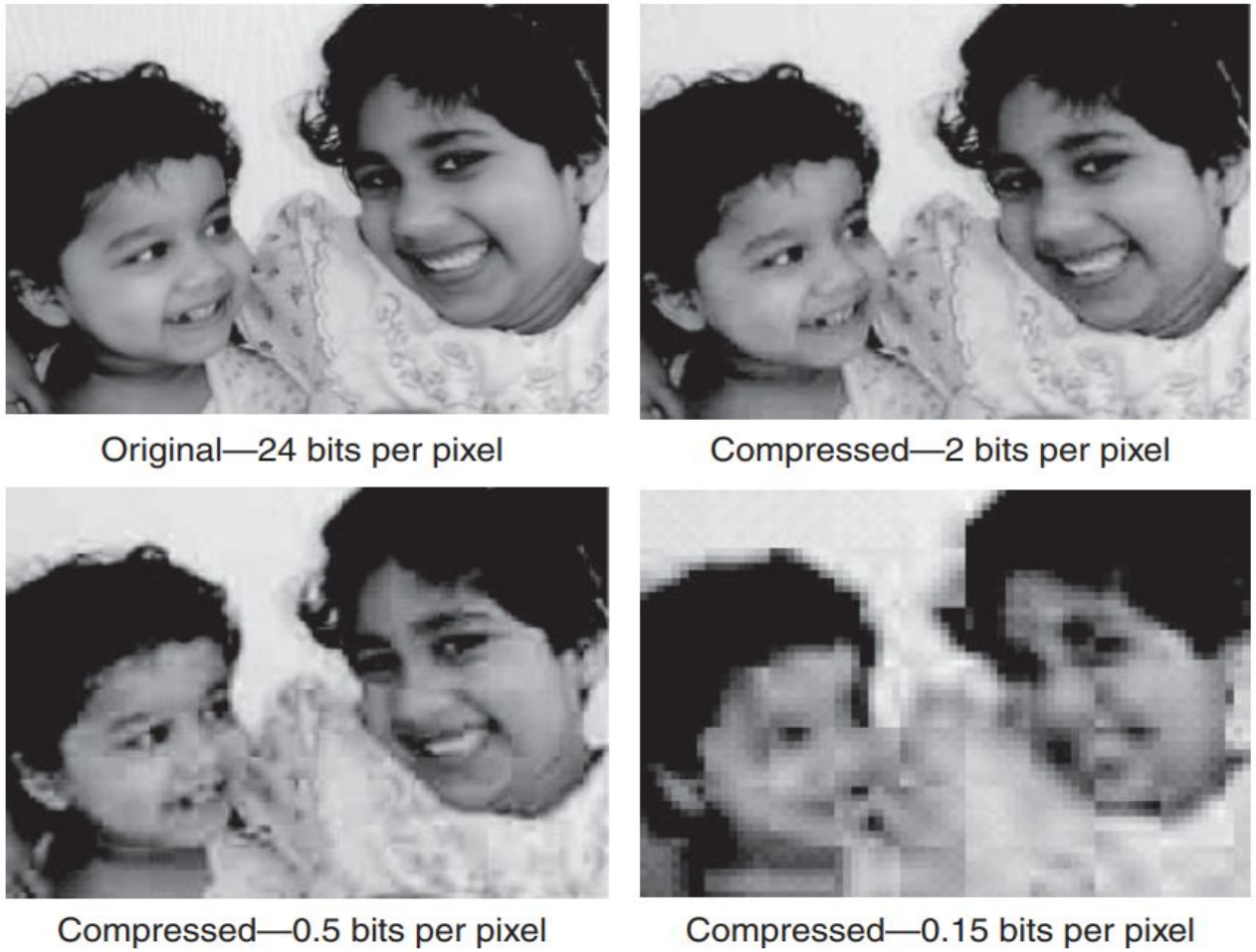
01111100100011100101001100110010101101111100000011000001000111010

Intermediary stream

<2><3> <0,3><4> <0,2><3> <0,1><1> <0,2><-3> <0,3><6>
<0,2><2> <0,2><-2> <0,2><2> <1,1><1> <1,1><-1> <0,1><-1>
<0,2><2> <0,1><-1> <0,1><1> <0,1><-1> <1,1><1> EOB

Intermediary symbol	Binary representation of first symbol (prefixed Huffman Codes)	Binary representation of second symbol (non-prefixed variable integer codes)
<2> <3>	011	11
<0,3> <4>	100	100
<0,2> <3>	01	11
<0,1> <1>	00	1
<0,2> <−3>	01	00
<0,3> <6>	100	110
<0,2> <2>	01	10
<0,2> <−2>	01	01
<0,2> <2>	01	10
<1,1> <1>	11	1
<1,1> <−1>	11	0
<0,1> <−1>	00	0
<0,2> <2>	01	10
<0,1> <−1>	00	0
<0,1> <1>	00	1
<0,1> <−1>	00	0
<1,1> <1>	11	1
EOB	1010	

JPEG compression results. The upper-left image shows the original at 24 bits per pixel. The remaining three images show the reconstructed outputs after JPEG compression at different compression ratios. The blocky artifacts increase at lower bit rates.



Original—24 bits per pixel

Compressed—2 bits per pixel

Compressed—0.5 bits per pixel

Compressed—0.15 bits per pixel

Multimedia Systems
Lecture – 29,30

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Drawbacks of JPEG

- JPEG produced ***decent quality*** for the storage and transmission needs. Also, it has an ***acceptable computational complexity***, and the DCT computation can easily be ***implemented in hardware*** *for devices that involve image capture*.

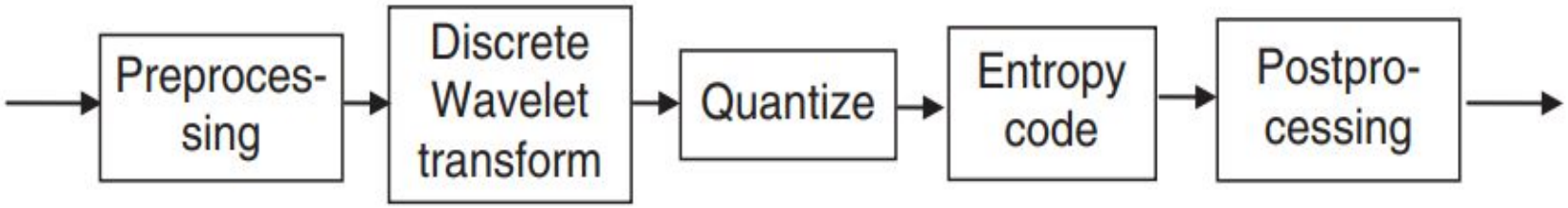
- However, today's needs are far greater than the provided functionality of JPEG, such as ***better compression requirements*** for ***larger imagery*** in a variety of applications, processing needs on compressed image bit streams, and so on.

- ***Poor low bit-rate compression***—JPEG offers excellent rate-distortion performance in the mid and high bit rates, but at ***low bit rates, the perceived distortion becomes unacceptable***.
- ***Lossy and lossless compression***—There is currently no standard that can provide superior ***lossless and lossy compression in a single coded stream.***
- ***Random access of the bit stream***—JPEG does ***not allow random access*** because each of the 8×8 blocks is interdependent. This prevents certain application scenarios where the end user might want to decode and see only a certain part of the image (or region of interest).
- ***Large image handling***—JPEG does not allow for the compression of images larger than ***64K by 64K*** without tiling.

- ***Single compression architecture***—The current JPEG standard has about 44 modes. Many of these are application specific and not used by the majority of decoders.
- ***Transmission in noisy environments***—JPEG was created before wireless communications became an everyday reality; therefore, it does not acceptably handle such an ***error-prone channel***.
- **Computer-generated images and documents**—JPEG was optimized for natural images and ***does not perform well on computer-generated images and document imagery***.
- **Compound documents –** does not support metadata mechanisms for incorporating additional nonimage data as part of the file.

JPEG 2000

- The JPEG 2000 compression pipeline makes use of the ***Discrete Wavelet transform (DWT)*** to compress images.
- DWT is known to work better than the Discrete Cosine transform in the way it ***distributes energy among the frequency coefficients***.



1. THE PREPROCESSING STEP

It is responsible for ***tiling***, ***conversion into the YCrCb formats***, and ***level offsetting***.

- **Tiling:**
  - Tiling process partitions the image into **rectangular**, but **equal-sized** and **nonoverlapping blocks**.
  - Each tile is then independently processed for DWT analysis, quantization, entropy coding, and so on.
  - The tiling process is purely optional and is done only to reduce memory requirements, as needed to deal with very large images.
- **YCrCb conversion process**: It is similar to the JPEG pipeline.
- **Level offsetting process:** It refers to shifting the DC levels by subtracting a constant value from each pixel value in the image (or tiles).

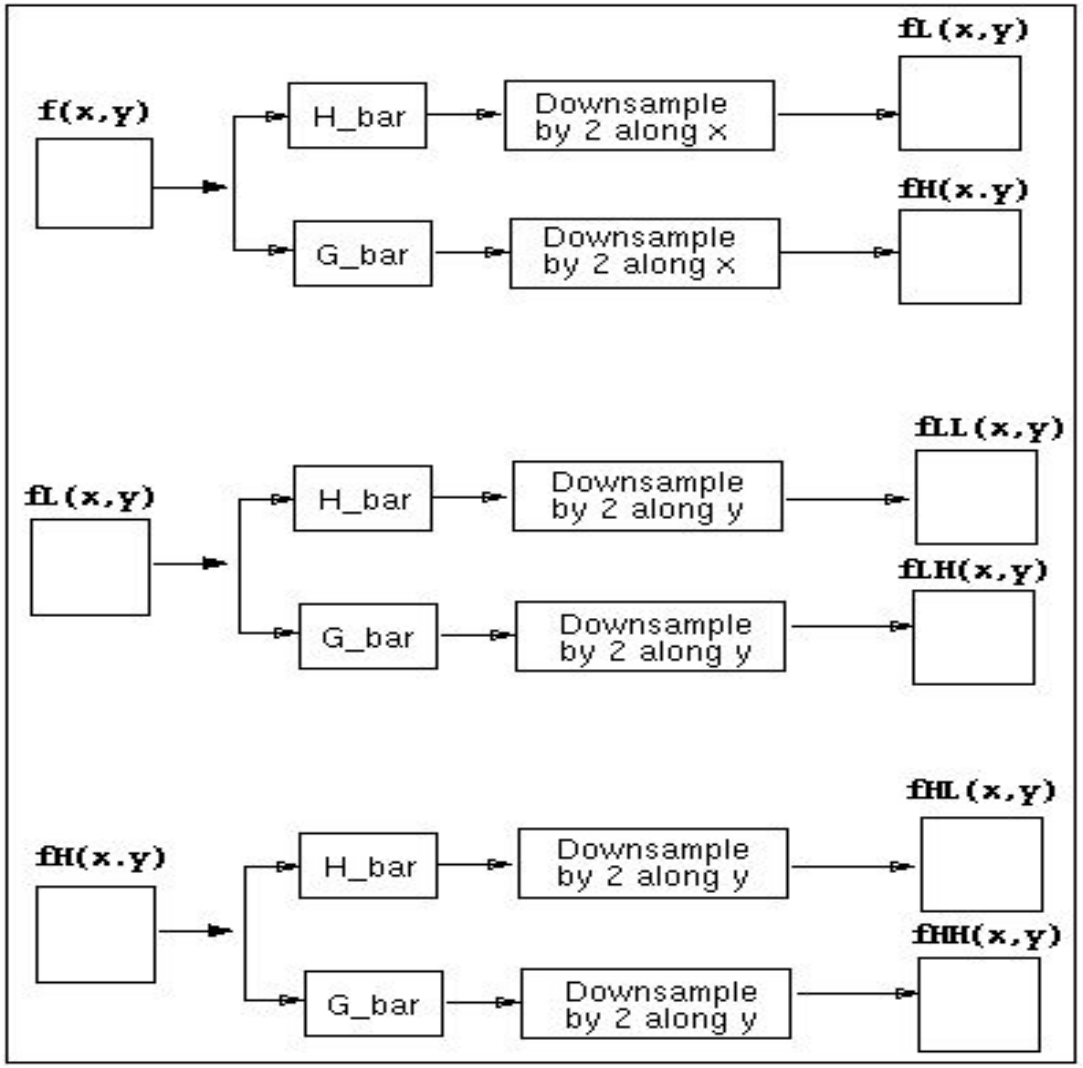
2. The Discrete Wavelet Transform

- It seeks to represent a signal with good resolution in ***both time and frequency***, by using a set of basis functions called **wavelets**.
- Wavelet transform coders process ***high and low frequency parts of image independently***
  - DCT methods have difficulties with high-frequency information
- Wavelet method ***transforms image as a whole*** (not subdivided into pixel blocks)
  - No blocking artifacts occur
  - Wavelet coders degrade gracefully

- Image is first filtered along the x dimension, resulting in low- pass and high-pass image

- Since bandwidth of both low pass and high pass image is now half that of the original image, both filtered images can be down-sampled by factor 2 without loss of information

- Then both filtered images are again filtered and down-sampled along the y dimension resulting in four sub-images



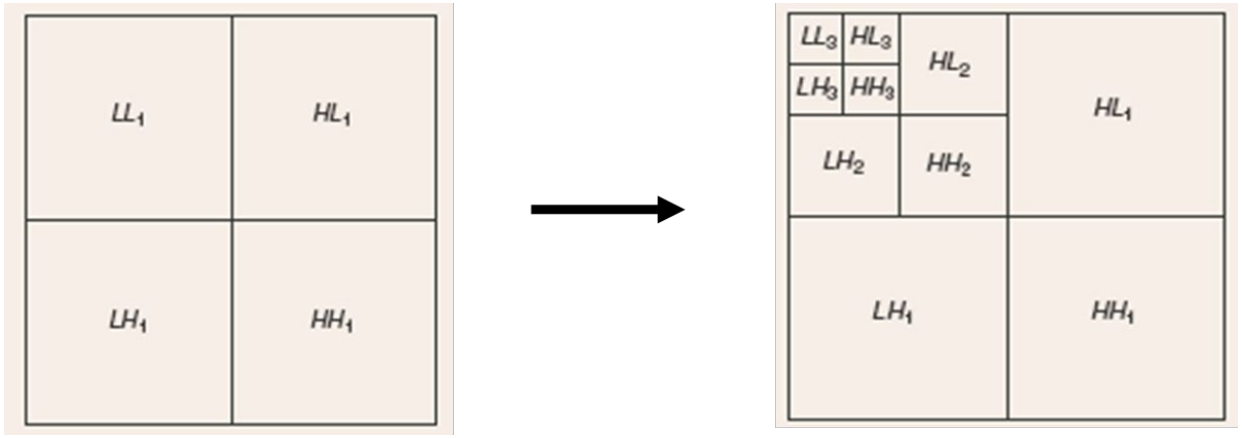
Wavelet Transform

LL—Low subbands of the filtering in both dimensions, rows and columns

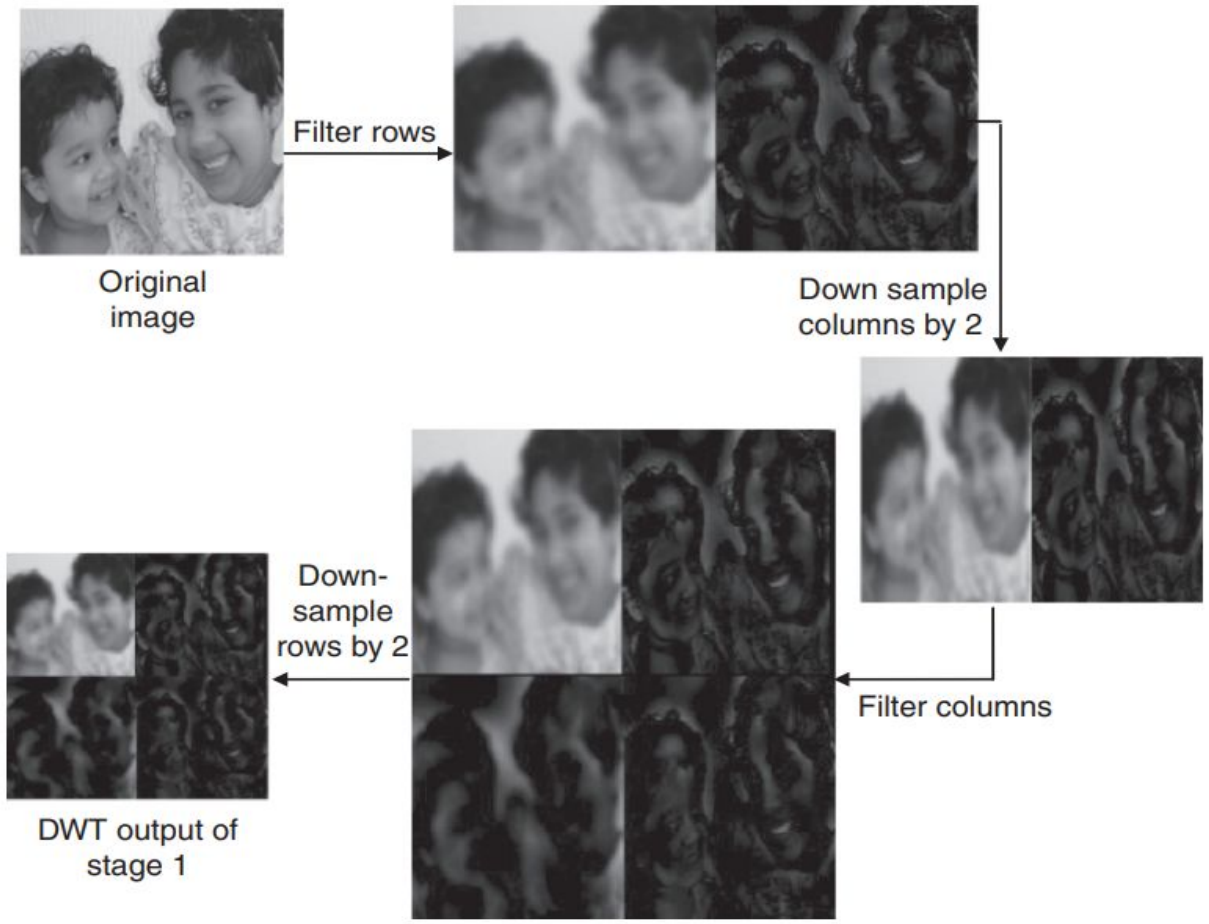
HL—High subbands after row filtering and low subbands after column filtering

LH—Low subbands after row filtering and high subbands after column filtering

HH—High subbands after both row and column filtering

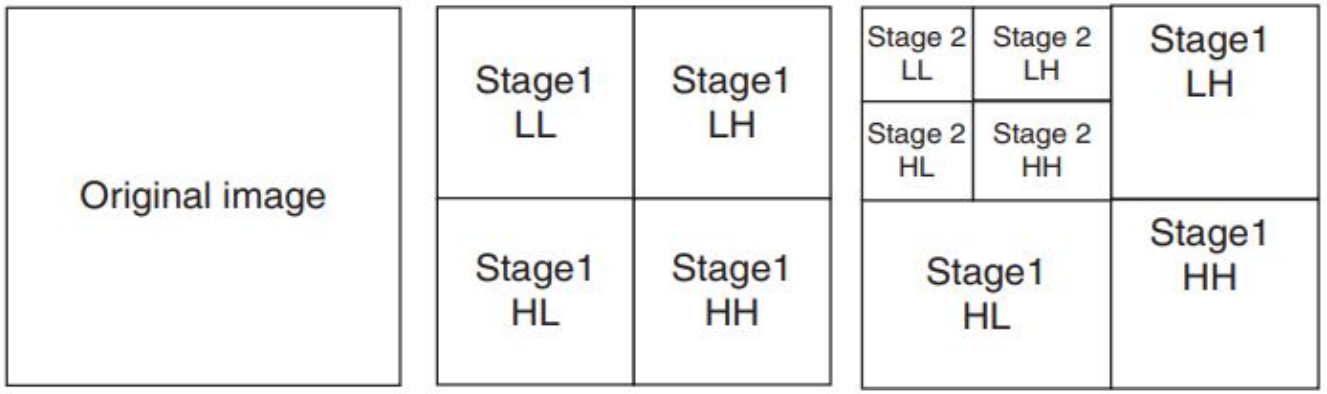


DWT process for the Y component of the sample image used



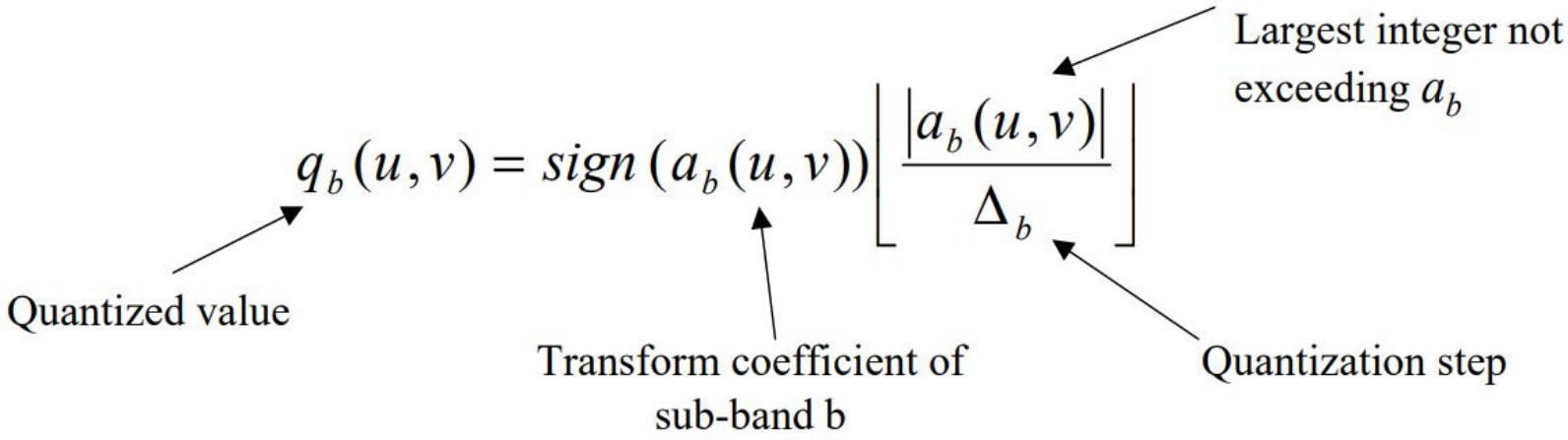
- In JPEG 2000, every original input image signal goes through ***multiple stages of the DWT***.
- The number of stages performed depends on the implementation.
- The second stage repeats the same process with the ***LL subband*** output of the previous stage.
- The higher subbands (HL, LH, and HH) hardly contain any significant samples and, therefore, only the LL subband is further transformed.
- Although JPEG 2000 supports from 0 to 32 stages, usually 4 to 8 stages are used for natural images.

Original input and the output of discrete wavelet processing at the first two levels. The top row shows the imaged outputs. The bottom row shows what each block at a level contains.



3. Qunatization

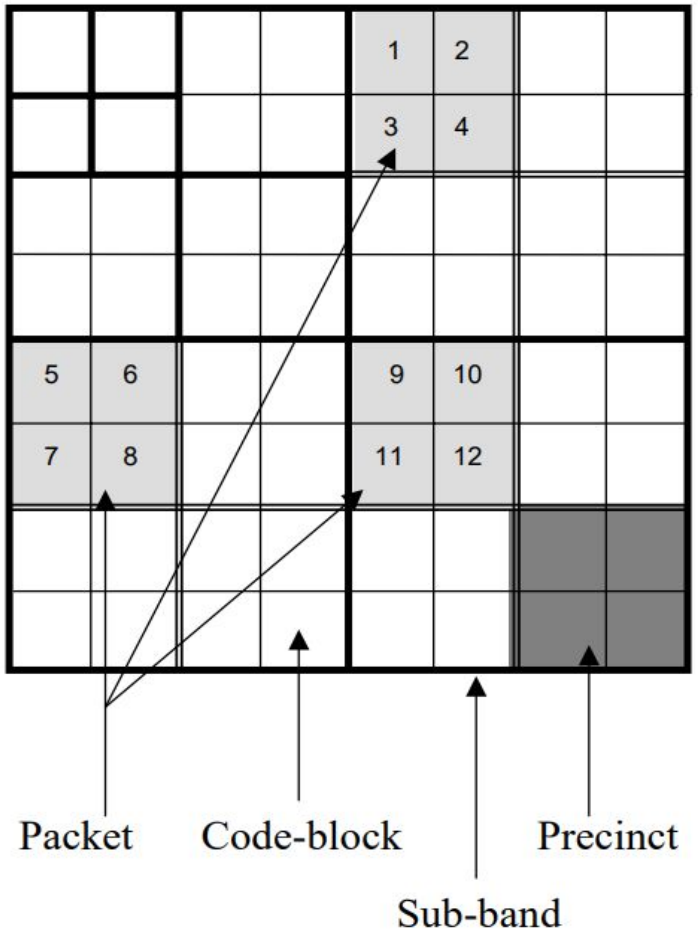
- After transformation, all coefficients are quantized using scalar quatization.
- Quantization reduces coefficients in precision. The operation is lossy unless the quatization step is 1 and the coefficients integers
- The process follows the formula:



4. Encoding

- It is based on block-based encoding scheme known as Embedded Block Coding with Optimized Truncations (***EBCOT***).
- Each subband is partitioned in blocks that are encoded independently via the EBCOT algorithm.
- typically 64 × 64, or other size no less than 32 × 32.

- Code-blocks, precincts and packets



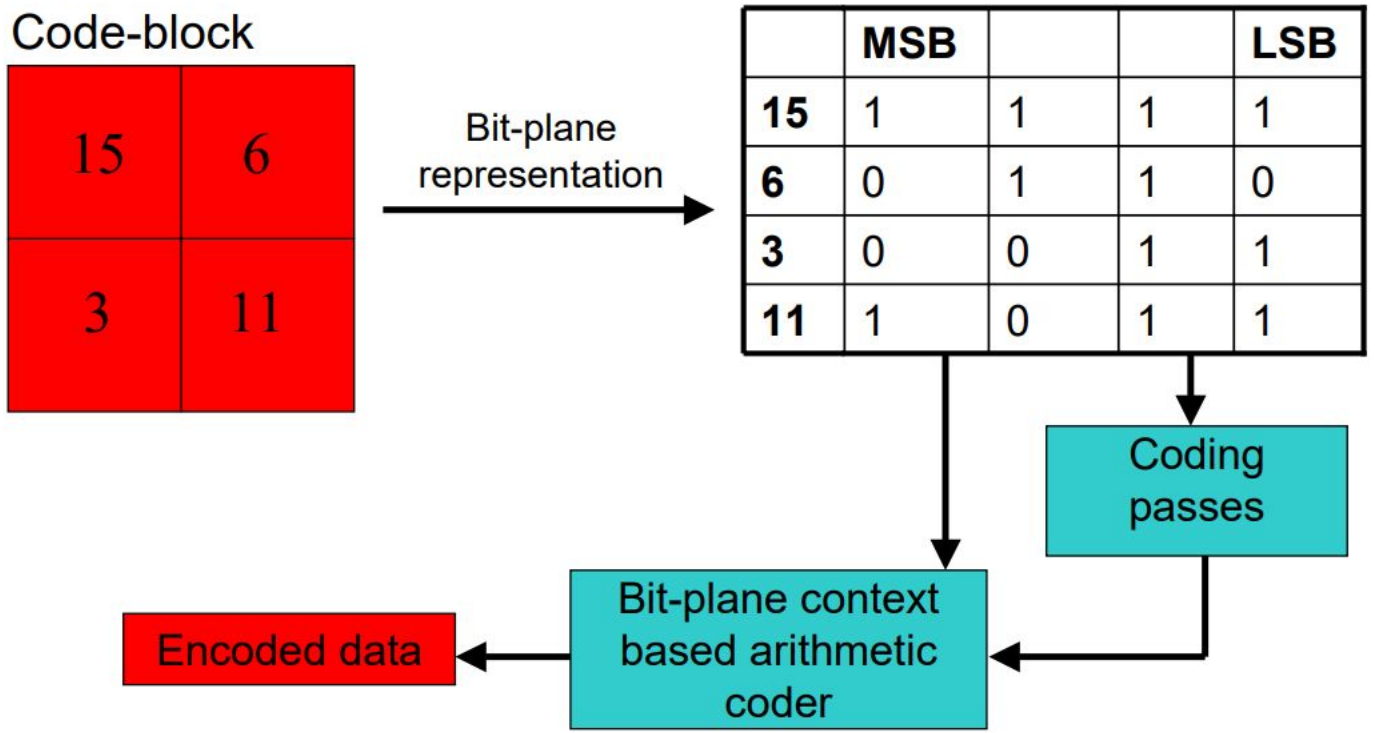
Precinct: each sub-band is divided into rectangular blocks called precincts.
Packets: three spatially consistent rectangles comprise a packet.
Code-block: each precinct is further divided into non-overlapping rectangles called code-blocks.

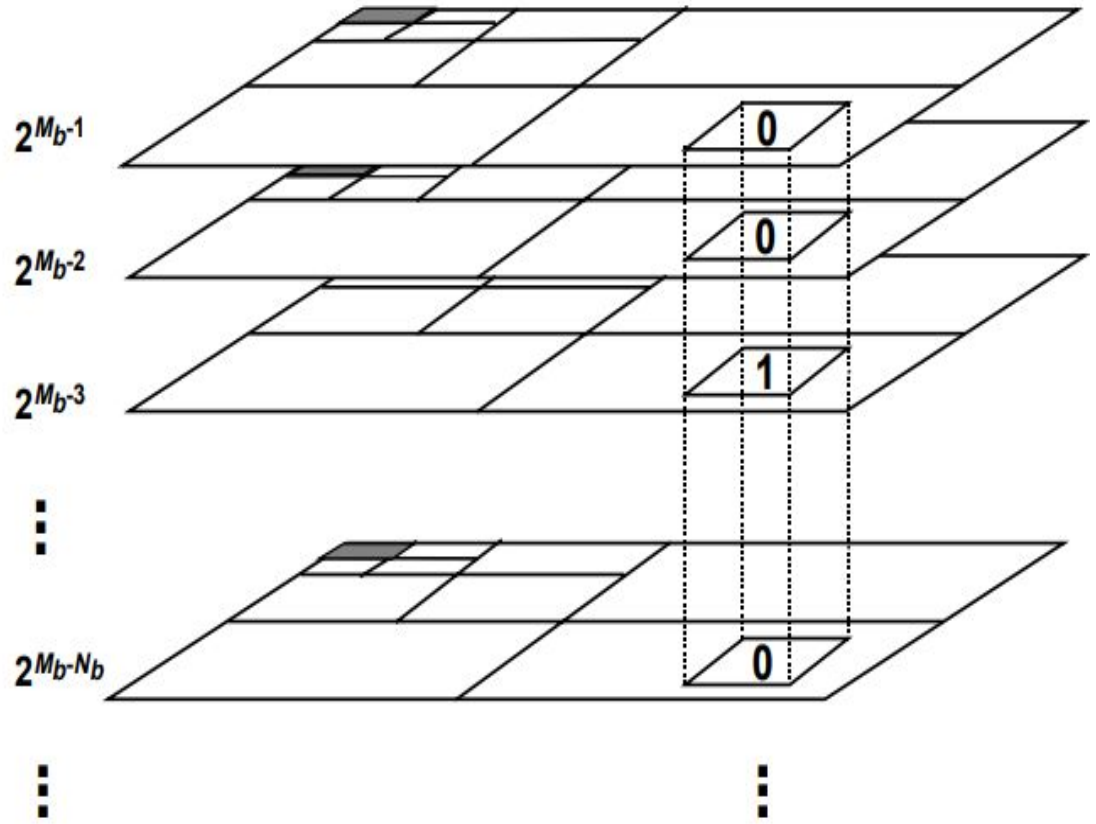
Each code-block forms the input to the entropy encoder and is encoded independently.

Within a packet, code-blocks are visited in raster order.

Entropy Coding: Bit-planes

- The coefficients in a code block are separated into bit-planes. The individual bit-planes are coded in 1-3 coding passes.





- During this progressive bitplane encoding, a quantized wavelet coefficient is called ***insignificant*** if the quantizer index is still zero
- Once the first nonzero bit is encoded, the coefficient becomes ***significant***.
- Once a coefficient becomes significant, all subsequent bits are referred to as ***refinement*** bits.

Since the DWT packs most of the energy in the low frequency subbands, the majority of the wavelet coefficients will have low amplitudes. Consequently, many quantized indices will be insignificant in the earlier bitplanes, leading to a very low information content for those bitplanes.

Entropy Coding: Coding Passes

- Each of these coding passes collects contextual information about the bitplane data. The contextual information along with the bit-planes are used by the arithmetic encoder to generate the compressed bit-stream.
- The coding passes are:
  - ***Significance propagation pass:*** *T*he insignificant coefficients that have the highest probability of becoming significant, as determined by their immediate eight neighbors, are encoded.
  - ***Magnitude refinement pass:*** The magnitude bit of a coefficient that has already become significant in a previous bitplane is arithmetic encoded.
  - ***Clean-up pass:*** *A*ll the remaining coefficients in the bitplane are encoded as they have the lowest probability of becoming significant..

Entropy Coding: MQ Coder

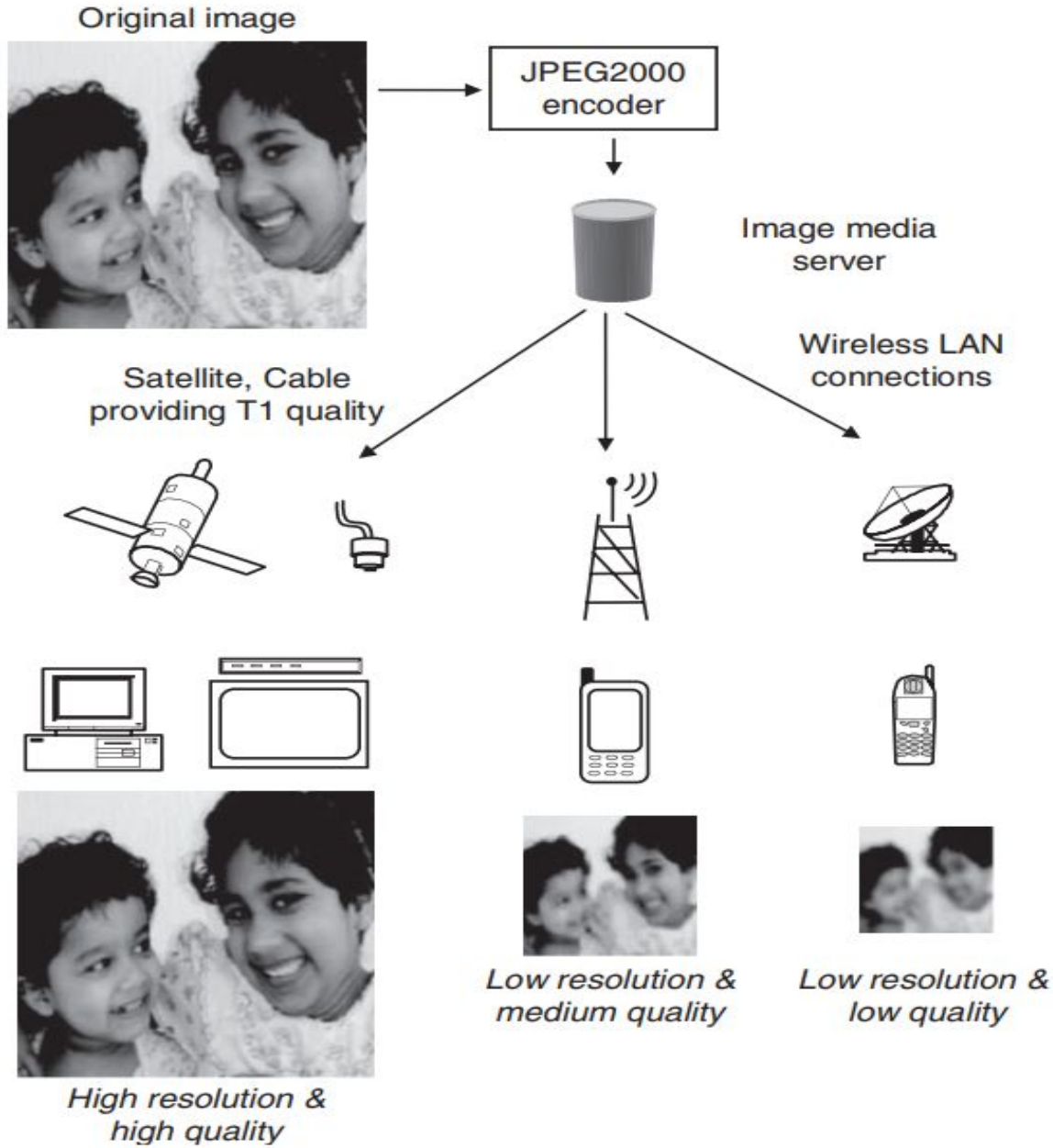
- JPEG2000 uses an efficient coding method for exploiting the redundancy of the bit-planes known as context-based adaptive binary arithmetic coding (***MQ Coder***).
- An adaptive binary arithmetic coder can be viewed as an encoding device that accepts the binary symbols in a source sequence, along with their corresponding probability estimates, and produces a codestream with a length at most two bits greater than the combined ideal codelengths of the input symbols.
- Adaptivity is provided by updating the probability estimate of a symbol based upon its present value and history.

- In JPEG2000, the probability of a binary symbol is estimated from a ***context*** formed from its current ***significance state*** as well as the significance states of its immediate eight neighbors (256 different contexts, only 9 considered) as determined from the previous bitplane and the current bitplane, based on coded information up to that point.
- if a coefficient is found to be significant, its ***sign needs to be encoded***. The sign value is also arithmetic encoded using five contexts that are determined from the significance and the sign of the coefficient's four horizontal and vertical neighbors.
- A special mode, referred to as the ***run mode***, is used to aggregate the coefficients that have the highest probability of remaining insignificant. More specifically, a run mode is entered if all the four samples in a vertical column of the stripe have insignificant neighbors.
- An encoded value of zero implies insignificance for all four samples, while an encoded value of one implies that at least one of the four samples becomes significant in the current bitplane.
- Each codeblock employs its own MQ-coder to generate a single arithmetic codeword for the entire codeblock.

- The arithmetic coding of the bitplane data is referred to as ***tier-1 (T1) coding***.
- JPEG2000 organizes the compressed data from the codeblocks into units known as packets and layers during the ***tier-2 coding*** step.
- For each precinct, the compressed data for the codeblocks is first organized into one or more packets.
- A packet is the fundamental building block in a JPEG2000 codestream. Each packet starts with a packet header.
  - The packet header contains information about the number of coding passes for each codeblock in the packet.
  - Also contains the length of the compressed data for each codeblock.
- Packets from each precinct at all resolution levels in a tile are then combined to form layers.

JPEG 2000 Versus JPEG

- Encode once—platform-dependent decoding:
  - In JPEG 2000, the encoder decides the maximum resolution and image quality to be produced—from the highly compressed to completely lossless.
  - Because of the hierarchical organization of each band's coefficients, any image quality can be decompressed from the resulting bit stream.
  - Additionally, the organization of the bit stream also makes it possible to perform random access by decompressing a desired part of the image or even a specific image component.
  - Unlike JPEG, it is not necessary to decode and decompress the entire bit stream to display section(s) of an image or the image itself at various resolutions.


Original image
JPEG2000 encoder
Image media server
Satellite, Cable providing T1 quality
Wireless LAN connections
High resolution & high quality
Low resolution & medium quality
Low resolution & low quality

- Region-of-interest encoding—
  - Region-of-interest (ROI) coding pertains to coding a specific region with higher quality compared with the rest of the image.
  - This process can be predefined, or can change dynamically.
  - Predefined ROIs are normally set at encoding time, whereas dynamic ROIs are used in applications where specific regions of the image are chosen for decoding at finer resolutions compared with other parts of the image.
  - JPEG 2000 elegantly allows for ROI control by specifying a parameter known as an ***ROI mask***. The ROI mask informs the encoder (if static selection) or the decoder (if dynamic selection) about the range of wavelet coefficients that contribute to a region's reconstruction. These coefficients can be decoded first, before all the background is decoded.

- Working with compressed images—
  - Normally, imaging operations such as simple geometrical transformations (cropping, flipping, rotating, and so on) and filtering (using the frequency domain) are performed on the pixel domain representation of the image.
  - If the image is compressed, it is decompressed, and recompressed after the required operation.
  - However, with JPEG 2000, such operations can be applied to the compressed representation of the image.
- High compression (especially at low bitrates) with better quality
- Error resilience (robustness to bit errors when communication or storage devices are unreliable)
- Support for tiling

Multimedia Systems
Lecture – 31

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Need for Video Compression

- HDTV broadcast has resolution of 1920×1080 at 30fps using 8bits to represent primary colors.
  - This leads to a total of ***1.5 Gbps*** data rate

- But channel b.w. is only 6 MHz that supports around 19.2Mbps data rate only.
  - The channel has also to support audio and other data
  - Effective data rate reduced to only 18Mbps
- Therefore compression required is 1.5Gbps/ 18Mbps = 83
- Compression ratio required is 83:1

Basics of Video Compression

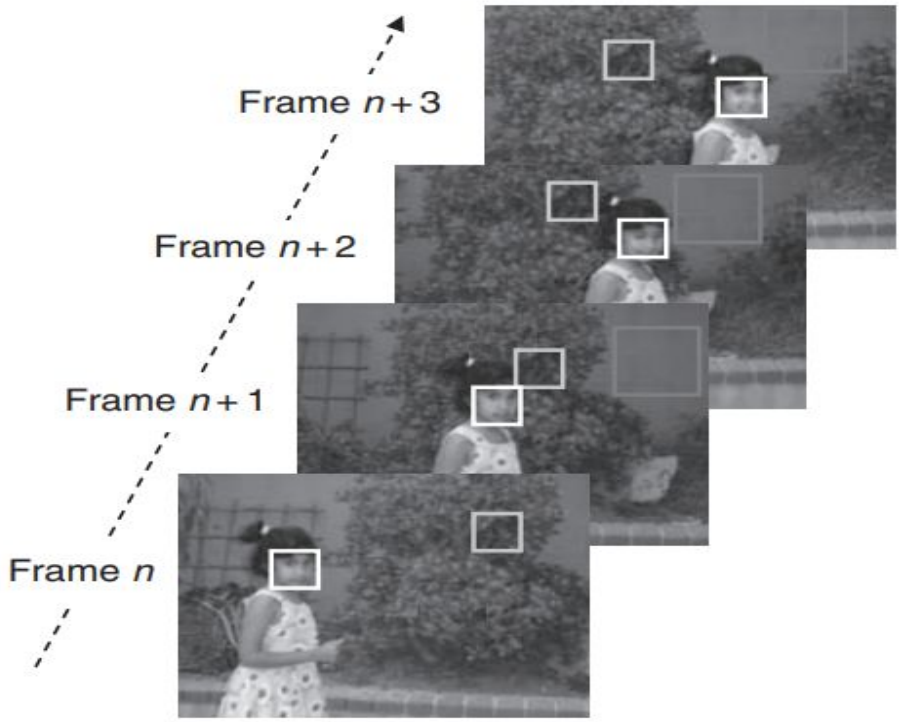
- Video consist of time-ordered sequence of frames, i.e. images.
- Images are having highly redundant data i.e. adjacent pixels are highly correlated.
- Image compression techniques mostly exploit both spatial and spectral redundancy.
- Image compression techniques like JPEG can be used to compress frames individually ?

- However, significantly higher compression rates can be achieved by exploiting another kind of redundancy in video—***temporal redundancy***.

- Just as pixel values are locally similar within a frame, they are also correlated across frames.

- For example, when you see a video, most pixels that constitute the backdrop or background do not vary much from frame to frame. Areas may move either when objects in the video move, or when the camera moves, but this movement is continuous and, hence, predictable.

Temporal Redundancy



Although each frame is different on a pixel-by-pixel basis, there is a great deal of correlation from frame to frame. The girl in the foreground moves, so the pixels on her face/dress have different locations in each frame, but the color values of these pixels do not change much.

- For video, it makes more sense to code only the changed information from frame to frame rather than coding the whole frame—***predictive DPCM techniques***.
- The temporal correlation should result in the difference frame having lower information content than the frame itself.
- It would be more efficient to encode the video by encoding the difference frames.
- The coherency from frame to frame is better exploited by observing that groups of contiguous pixels, rather than individual pixels.

Frame difference

Frame n | Frame $n+1$ | Frame $(n+1)-$Frame n

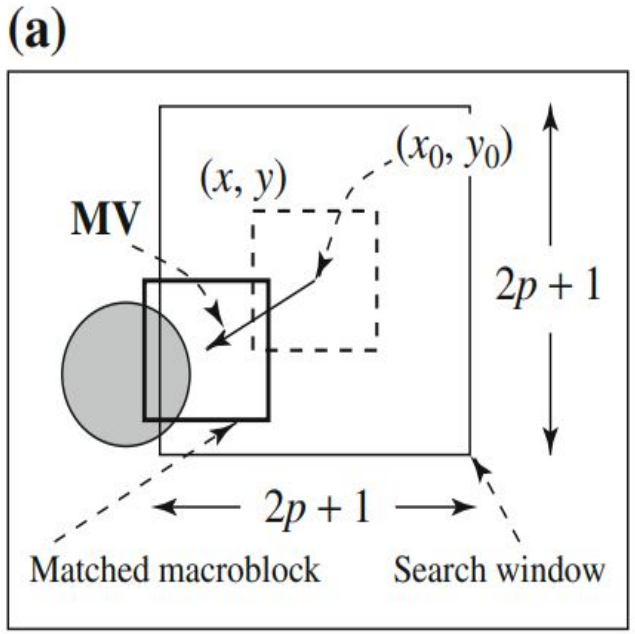
Key Idea

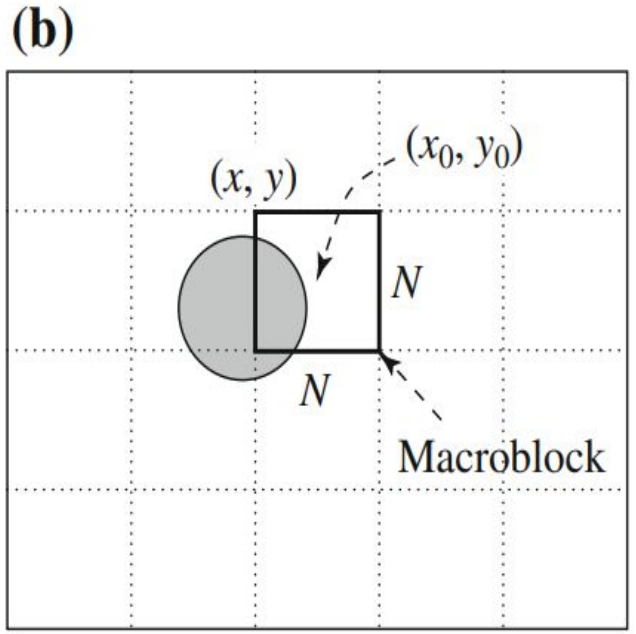
- Predict a new frame from the previous frame and specify the prediction error.
- Prediction error can be coded using image coding methods.
- Prediction from past frames is known as *forward prediction*.
- The current image frame, which needs to be compressed, is known as the *target frame*. The target frame is predicted on a block-by-block basis from a previous frame, which is known as the *reference frame*.
- Video compression makes use of *motion compensation* to predict a frame from the previous and/or next frame.

Key Terms

- ***Macro Blocks:*** Compression method works on a block of 16×16 pixels called macro blocks.
- ***I (intra) frames***
  - Independent frames
  - Coded without reference to other frames
  - I frames are interspersed periodically in the encoding process, and provide access points to the decoder to decode the succeeding predicted frames.
- ***Inter or P (Predictive) frame***
  - Not independent
  - These frames are predicted from a past frame (I or P)
  - Coded by forward predictive coding
    - Current macro block is predicted from similar macro block in the previous I or P frame and the difference between the macro blocks is coded.

Macroblocks and motion vector in video compression





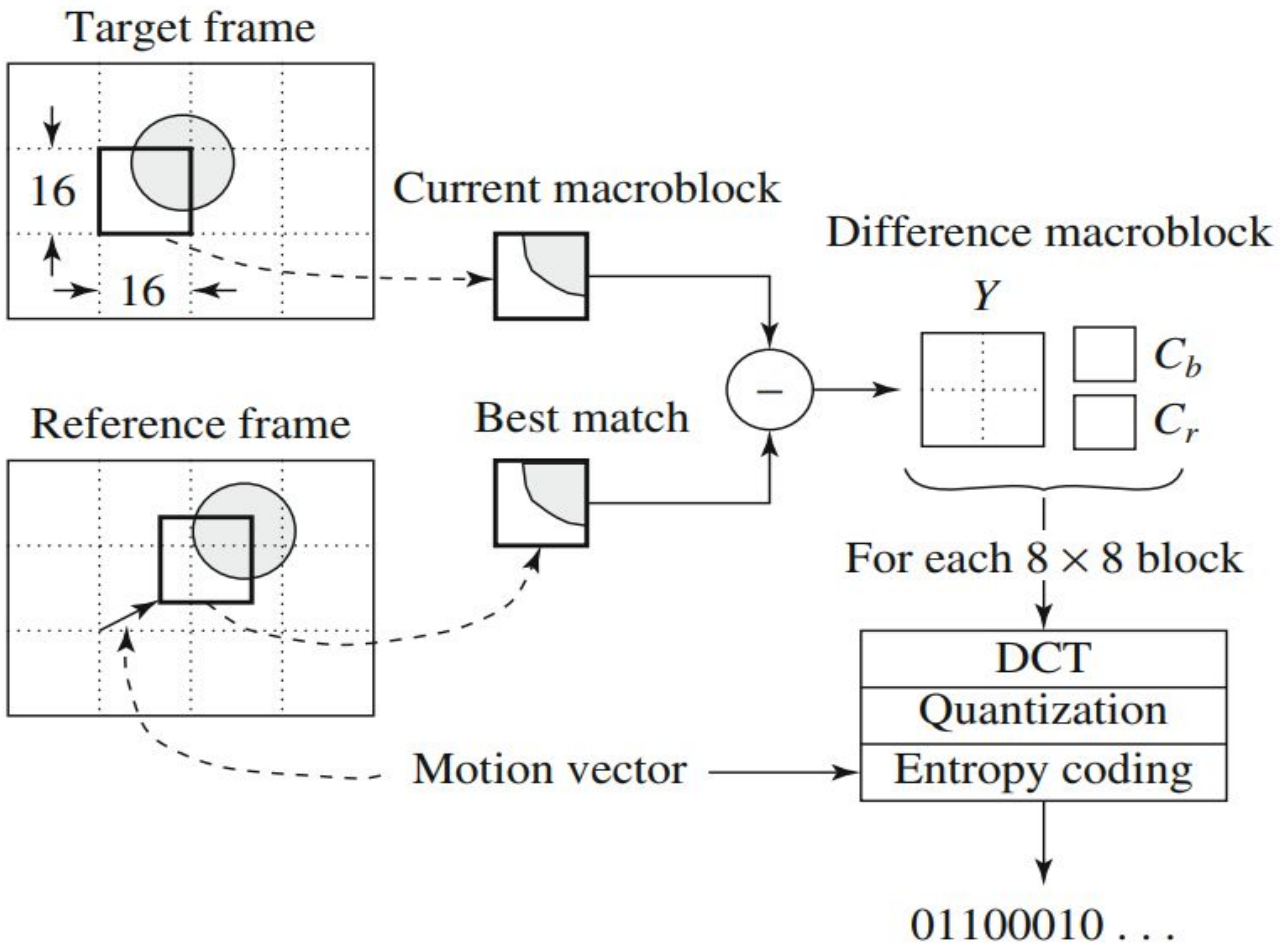
Reference Frame

Target Frame

Video Compression based on Motion Compensation

- The displacement of the reference macroblock to the target macroblock is called a *motion vector* **MV**.
- The process of compressing a frame (after the first frame) in this mode involves the following:
  - Finding the best motion vector for each macroblock
  - Creating a predicted or motion-compensated frame image
  - Deriving the prediction error image
  - Compressing error image using the JPEG pipeline (lossy) and the motion vectors using entropy coding (lossless)

P-frame coding based on motion compensation

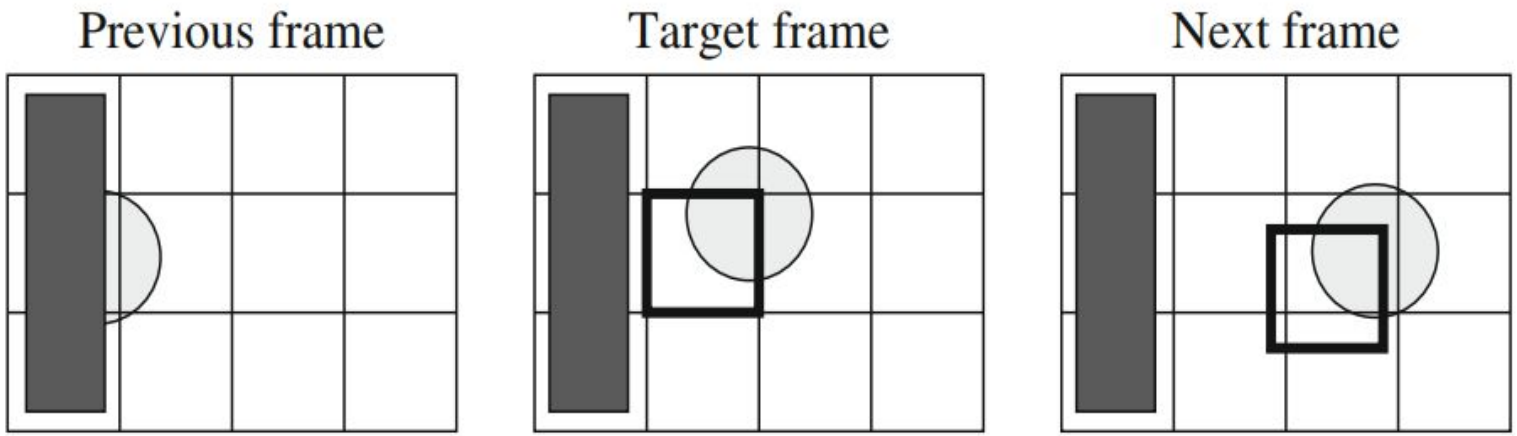


MPEG-1 Evolution

- Approved by the ISO/IEC MPEG group in November 1991 for Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s.
- Common digital storage media include ***compact discs (CDs)*** and ***video compact discs (VCDs).***
- MPEG-1 supports only ***noninterlaced video***.
- It uses ***4:2:0*** chroma subsampling.
- Audio-coding standard consists of three layers of audio-coding schemes with increasing complexity, resulting in successively better compression.
- These are better known as MPEG-1 Layer I, MPEG-1 Layer II, and MPEG-1 Layer III, popularly known as ***MP3***.

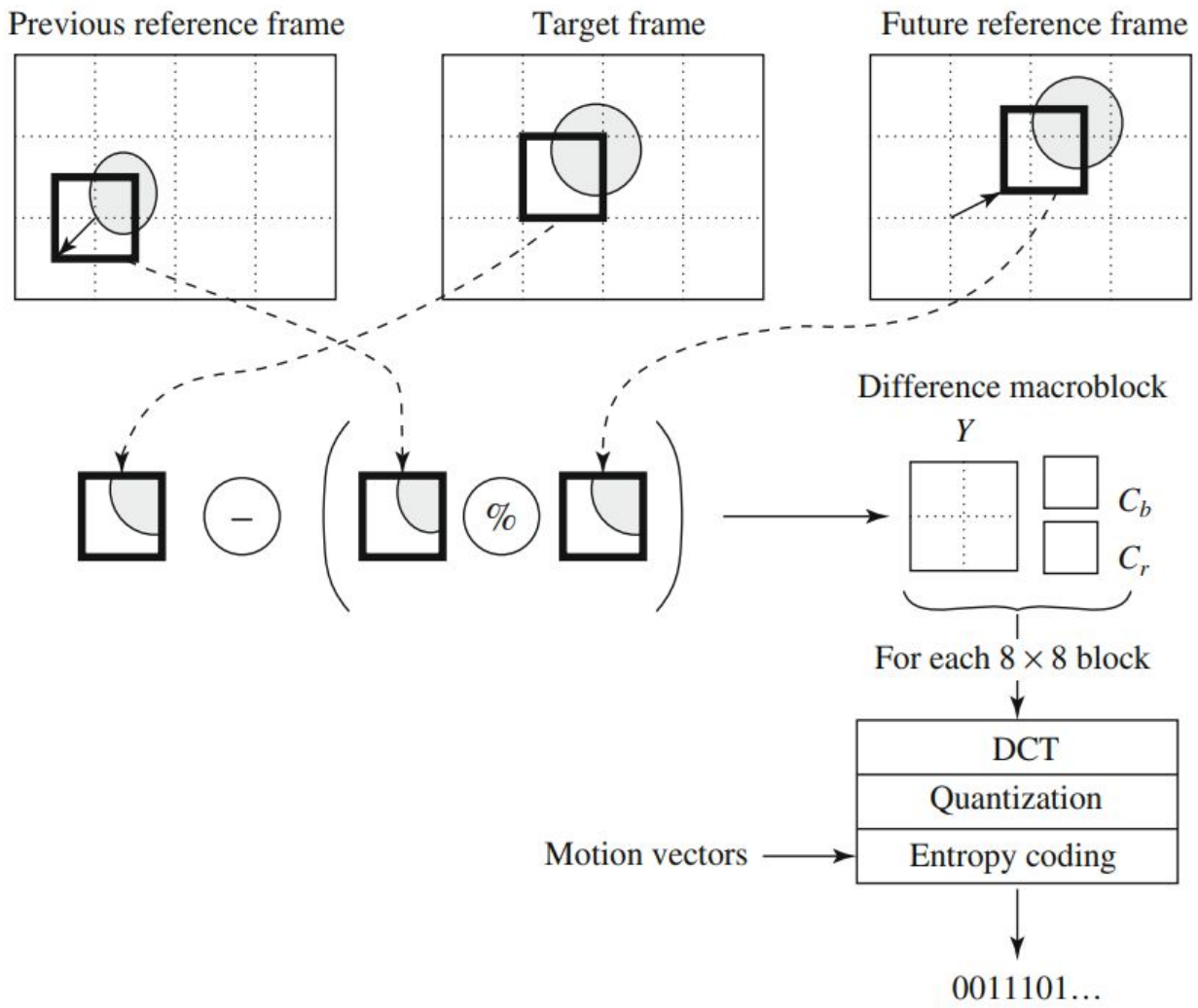
- ***MPEG*** introduces a new ***B-frame***.
- ***B-frame (Bidirectional predictive coded)***
  - Due to unexpected movement in all real scenes, the target macro block may not have a good matching in the previous frame.
  - Therefore, B-frame is coded with reference to both previous and future reference frames (either I or P).
  - When prediction is from a previous frame, it is called forward prediction.
  - When the matching macroblock is obtained from a future I- or P-frame in the video sequence, then it is known as ***backward prediction***.

The need for bidirectional search



The macroblock containing part of a ball in the target frame cannot find a good matching macroblock in the previous frame, because half of the ball was occluded by another object. However, a match can readily be obtained from the next frame.

Motion Compensation in MPEG-1



- If matching in both directions is successful, two motion vectors will be sent, and the two corresponding matching macroblocks are averaged (indicated by “%” in the figure) before comparing to the target macroblock for generating the prediction error.

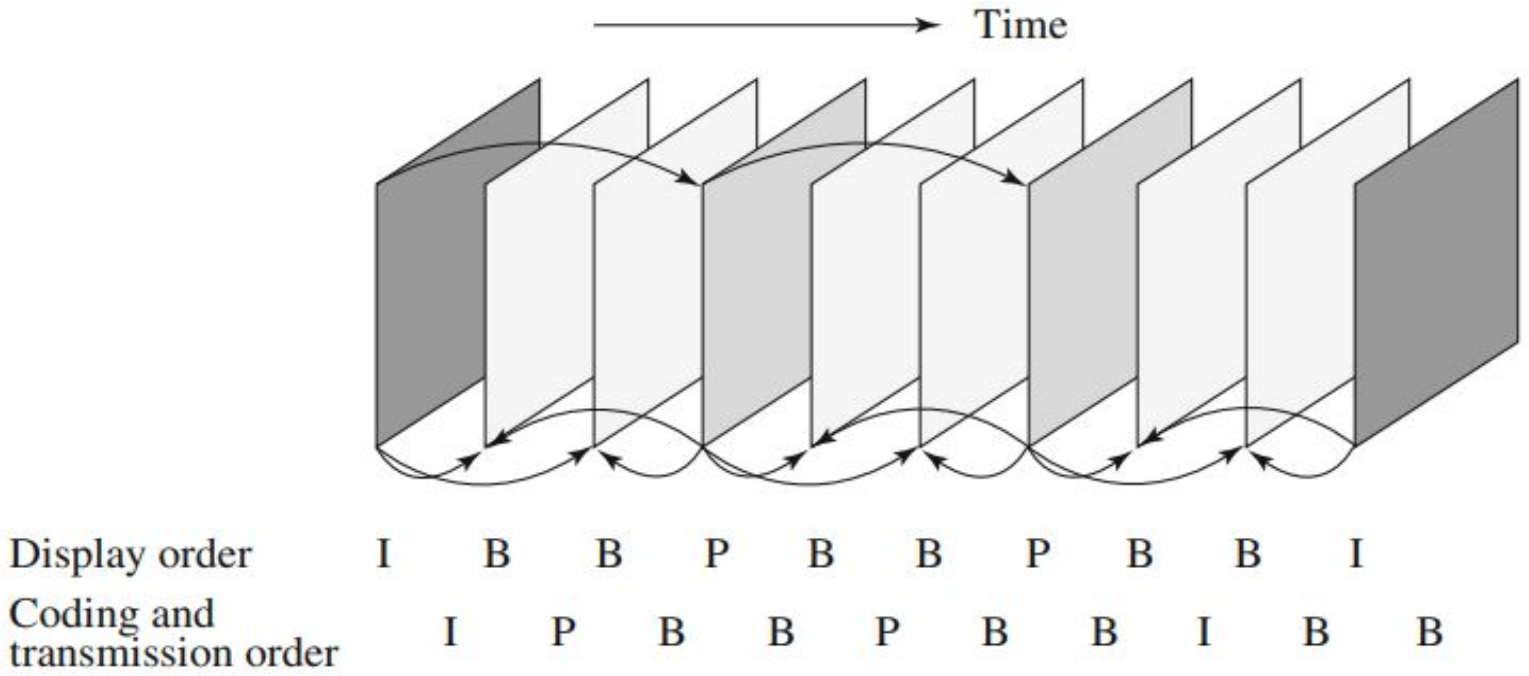
- If an acceptable match can be found in only one of the reference frames, only one motion vector and its corresponding macroblock will be used from either the forward or backward prediction.

- B frames also necessitate reordering frames during transmission, which causes delays. The order in which frames arrive at the encoder is known as the ***display order***.

Multimedia Systems
Lecture – 32

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

MPEG Frame sequence



The decoder has to have both the past and the future reference frames ready before it can decode the current B frame. Consequently, the encoder has to encode and send to the decoder both the future and past reference frames before coding and transmitting the current B frame. This enforces the encoder to make the coding and transmission order different from the display order.

- All potential B frames need to be ***buffered*** while the encoder codes the future reference frame, causing a ***delay*** during transmission.
- The actual frame pattern is determined at encoding time and is specified in the video's header.
- MPEG uses M to indicate the interval between a P-frame and its preceding I- or P-frame, and N to indicate the interval between two consecutive I-frames. Here, M = 3, N = 9.
- A special case is M = 1, when no B-frame is used.
- Issues with MPEG-1
- The inevitable delay and need for buffering become an important issue in real-time network transmission, especially in streaming MPEG video.
- Stores and plays video on the CD of a single computer at a low bitrate (1.5 Mbps)

MPEG-2

- Project started in 1990 and finalized in 1994.
- Provided high quality video with bit rate around 4Mbps.
- It was developed as a standard for digital TV broadcast.
- It gained wide popularity in terrestrial, satellite, or cable network.
- Adopted in DVDs

Salient features of MPEG-2 video coding

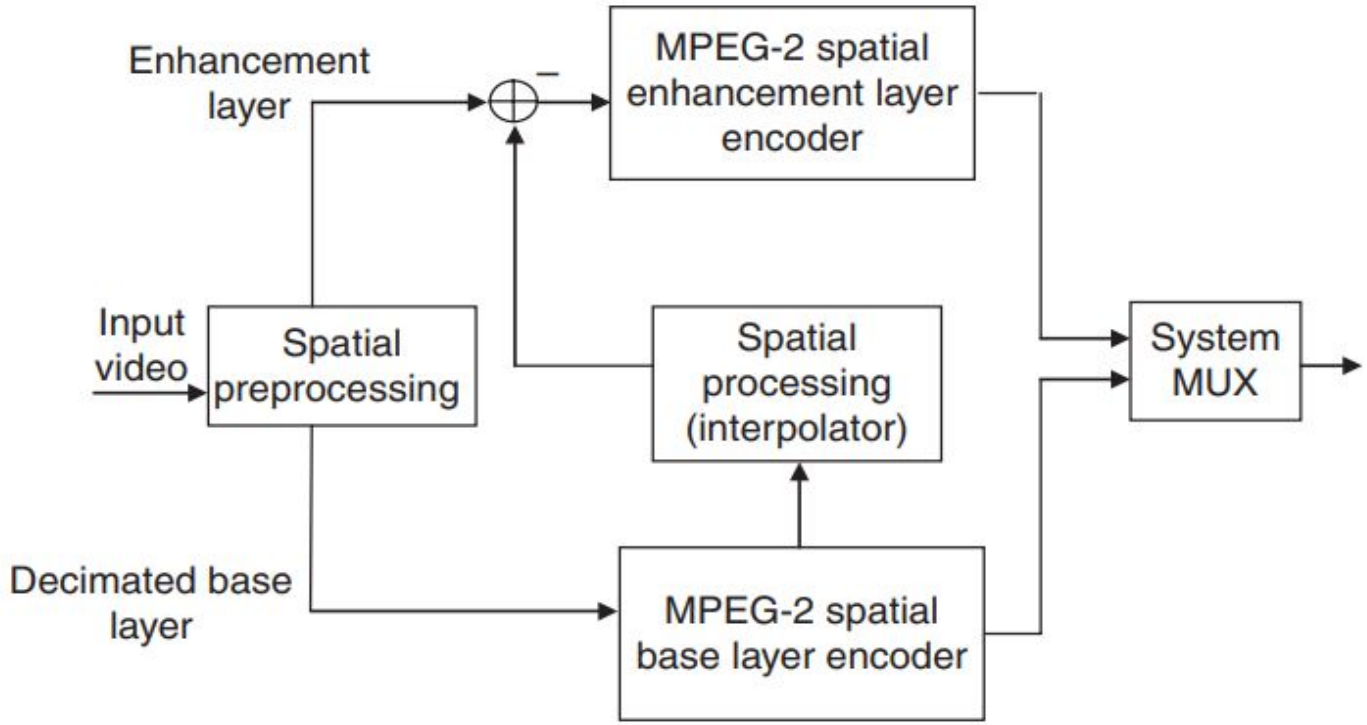
- MPEG-2 encodes video using I, P, and B frames similar to MPEG-1 with ***halfpixel approximation*** for motion vectors.
- MPEG-2 has support for ***interlaced video*** that consists of alternating fields.
  - During intermode compression, fields need to be predicted from fields. For this, MPEG-2 defines different modes of prediction for predicting ***frames from frames***, ***fields from fields***, and ***fields from frames.***
- In addition to defining video and audio, MPEG-2 was also designed as a transmission standard providing support for a variety of ***packet formats*** with error-correction capability across noisy channels.

- One new aspect introduced in MPEG-2 video was ***scalable encoding***, which generates coded bit streams in such a way that decoders of varying complexities can decode different resolutions and, thus, different quality from the same bit stream.

- This allows for encoders to encode once and support different bandwidths, having end clients with decoders of different complexity.

- **Support of 4:2:2 and 4:4:4 chroma subsampling**

Scalable MPEG-2 video encoding



The input video is broken into two layers, a base layer containing low frequency and an enhancement layer containing high frequency. Both layers are coded independently and then multiplexed into the coded bit stream.

MPEG-4

- MPEG-4 is one of the latest encoding standards from the MPEG community and has a much broader scope than just encoding video or audio.
- Finalized and ratified in 1999; however, the later versions such as MPEG-4 version 10 were completed later in 2003.
- Version 10 of the visual part is also known as ***MPEG-4 AVC*** (Advanced Visual Coding)
- MPEG-4 arose from a need to have a scalable standard that supports a wide bandwidth range from low bandwidth to high bandwidth.

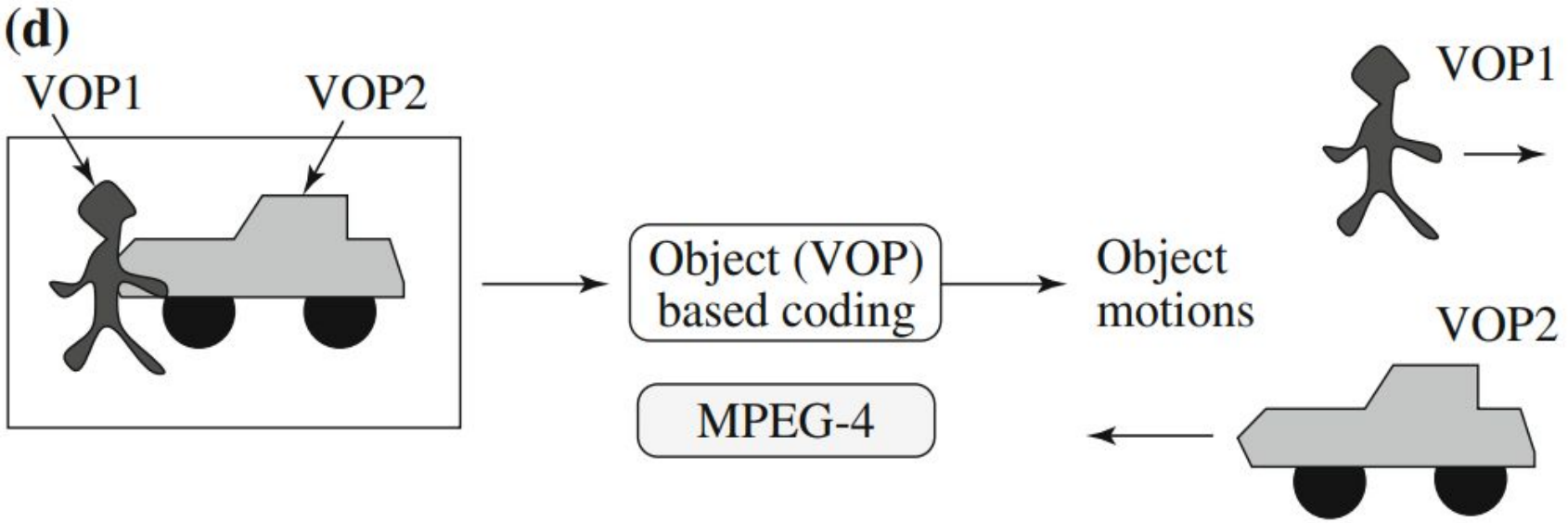
Salient features of MPEG-4

- It supports both ***progressive and interlaced*** video encoding.
- The video compression obtained by MPEG-4 ***ASP (Advanced Simple Profile)*** is better than MPEG-2 by ***25%*** for the same video quality. This is because MPEG-4 does ***quarter-pixel accuracy*** during motion prediction.
- The standard is ***object based*** and supports multiple video streams that can be organized (along with other audio, graphics, image streams).
- MPEG-4 supports coding video into ***video object planes***. A video can consist of different video object planes (***VOP***s), and the image frames in each VOP can have ***arbitrary shapes***, not necessarily rectangular. This is useful for sending two video feeds to the client, who can then composite them for a MPEG-4 player.

- For example, one video could be of a reporter in a studio chroma-keyed in front of a blue screen, while the other could be that of an outdoor setting. The composition of both occurs at the client side. This increases the flexibility in creating and distributing content.

- MPEG-4 compression provides ***temporal scalability***, which utilizes advances in object recognition. By separating the video into different layers, the foreground object, such as an actor or speaker, can be encoded with lower compression while background objects, such as trees and scenery, can be encoded with higher compression.

Object-based coding in MPEG-4



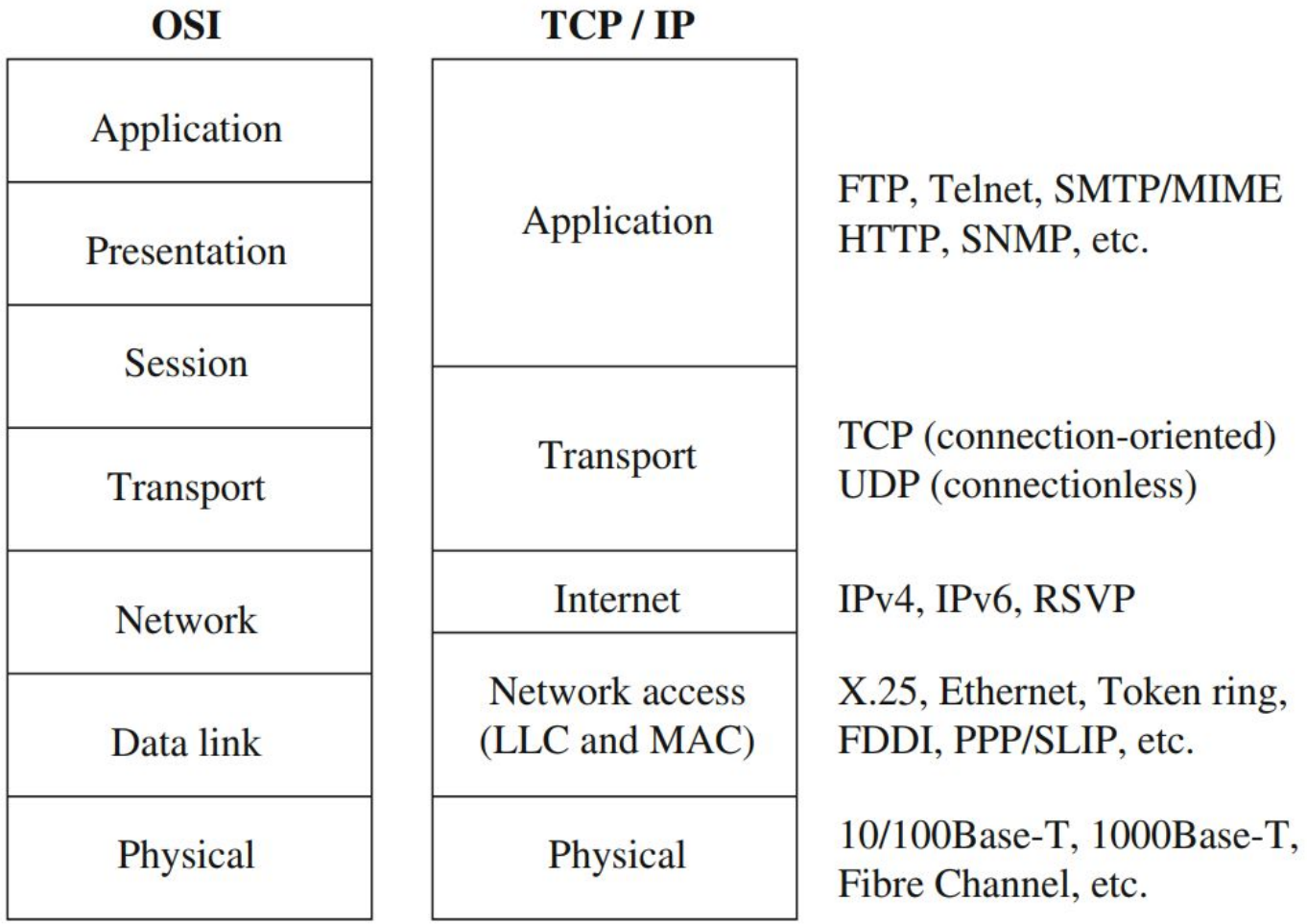
Multimedia Systems
Lecture – 33

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Multimedia Communications and Networking

- Computer communication networks are essential to the modern computing environment.
- Multimedia communications and networking share all major issues and technologies of computer communication network.
- The evolution of the Internet, particularly in the past two decades, has been largely driven by the ever-growing demands from numerous conventional and new generation multimedia applications.

Protocol Layers of Computer Communication Networks



Quality-of-Service for Multimedia Communications

- Fundamentally, multimedia network communication and traditional computer network communication are similar, since they both deal with data communications.

- However, challenges in multimedia network communications arise due to a series of distinct characteristics of audio/video data:

- ***Voluminous and Continuous***
  - They demand high data rates, and often have a lower bound to ensure continuous playback.
  - A user expects to start playing back audio/video objects before they are fully downloaded. Commonly referred to as ***continuous media*** or ***streaming media***.
- ***Real-Time and Interactive***
  - They demand low startup delay and synchronization between audio and video for “lip sync”.
  - Interactive applications such as video conferencing and multi-party online gaming require two-way traffic, both of the same high demands.
- ***Rate fluctuation***
  - The multimedia data rates fluctuate drastically and sometimes bursty.
  - In VoD or VoIP , no traffic most of the time but burst to high volume.
  - In a variable bit rate (VBR) video, the ***average rate*** and the ***peak rate*** can differ significantly, depending on the scene complexity

Quality of Service

- *Quality of Service, also termed QoS, indicates how well a network performs with multimedia applications, regardless of whether the network is conforming to the required traffic for the application (*the capability of a network to provide a level of service to deliver network packets from a sender to a receiver*).*
- QoS for multimedia data transmission depends on many parameters.
- ***Bandwidth:***
  - A measure of transmission speed over digital links or networks, often in kilobits per second (kbps) or megabits per second (Mbps).
  - The data rate of a multimedia stream can vary dramatically, and both the average and the peak rates should be considered when planning for bandwidth for transmission.
- ***Latency (maximum frame/packet delay)***
  - The maximum time needed from transmission to reception, often measured in milliseconds (msec, or ms).

Requirement on network bandwidth/bitrate

Application	Speed requirement
Telephone	16 kbps
Audio conferencing	32 kbps
CD-quality audio	128–192 kbps
Digital music (QoS)	64–640 kbps
H. 261	64–2 Mbps
H. 263	<64 kbps
H. 264	1–12 Mbps
MPEG-1 video	1.2–1.5 Mbps
MPEG-2 video	4–60 Mbps
MPEG-4 video	1–20 Mbps
HDTV (compressed)	>20 Mbps
HDTV (uncompressed)	>1 Gbps
MPEG-4 video-on-demand (QoS)	250–750 kbps
Videoconferencing (QoS)	384 kbps–2 Mbps

- ***Packet loss or error rate***
  - The packets can get lost due to network congestion or garbled during transmission over the physical links. They may also be delivered late or in the wrong order.
  - Error rate measures the total number of bits (packets) that were corrupted or incorrectly received compared with the total number of transmitted bits (packets).
  - For real-time multimedia, retransmission is often undesirable, and therefore alternative solutions like forward error correction (FEC), interleaving, or error-resilient coding are to be used.
- ***Jitter (or delay jitter):***
  - A measure of smoothness (along time axis) of the audio/video playback. Technically, jitter is related to the variance of frame/packet delays.
  - A large buffer (jitter buffer) can be used to hold enough frames to allow the frame with the longest delay to arrive, so as to reduce playback jitter. However, this increases the latency and may not be desirable in real-time and interactive applications.
- ***Sync skew:***
  - A measure of multimedia data synchronization, often measured in milliseconds (msec). For a good lip synchronization, the limit of sync skew is ±80 msec between audio and video. In general, ±200 msec is still acceptable.

Tolerance of latency and jitter in digital audio and video

Application	**Average latency tolerance** (msec)	**Average jitter tolerance** (msec)
Low-end videoconference (64 kbps)	300	130
Compressed voice (16 kbps)	30	130
MPEG NTSC video (1.5 Mbps)	5	7
MPEG audio (256 kbps)	7	9
HDTV video (20 Mbps)	0.8	1

Different applications have varied QoS requirements. Some sample requirements include the following:

- ***Streaming media*** might impose QoS requirements on a guaranteed bandwidth and low latency but a more tolerable jitter.
- ***Videoconferencing or IP telephony***, which are real-time applications, might require stricter limits on jitter and maximum delay, but could tolerate errors.
- ***Online games*** where multiple players play together across a network need more real-time QoS constraints with bounded latency and jitter.
- A ***remote surgery***, which is a health-critical application, might need a guaranteed level of connection availability compared with other requirements.

- A better categorization of QoS levels are ***best-effort service***, ***differentiated service***, and ***guaranteed service***.
- ***best-effort service (lack of QoS):*** The best-effort service provides basic connectivity with no guarantees or priorities to packets in transit across the network.
- ***differentiated service (soft QoS):*** Packets are marked with a priority, which is used to give partial preference while routing of packets at intermediary nodes.
- ***guaranteed service (hard QoS):*** In this service, packet delivery is guaranteed at any cost. There is an absolute reservation of network resources that are dedicated to packet delivery.

Multimedia Communication Protocols

General Protocols: This subsection first discusses general protocols used in communication that do not necessarily take care of requirements imposed by real-time media.

- ***Internet Protocol (IP):***
  - The Internet Protocol (IP) is a Network layer protocol that represents the main methodology of communication over the Internet.
  - To provide fragmentation of large data into packets and its consequent reassembly.
  - To provide a connectionless method of data packet delivery.
  - The IP addressing mechanism is indispensable to how routing occurs in this protocol.
  - Because of the connectionless nature of the forwarding, packets can arrive via different routes and, consequently, out of order.

- ***Transmission Control Protocol (TCP):***
  - It is a connection-oriented transport layer protocol, which provides reliable data transfer.
  - TCP requires a connection to be established prior to sending data, and the connection must be terminated upon completion of transmission. To establish a connection, TCP uses a three-step agreement to open a TCP connection from a client to a server, the server acknowledges, and, finally, the client also acknowledges.
  - Every packet has both the source and destination addresses.
  - In-order packet reception
  - Error-free data transfer and Discarding duplicate packets
  - Although TCP ensures reliable transmission, the overhead of retransmission is normally a hindrance to maintaining real-time synchronization of multimedia data.

- ***User Datagram Protocol (UDP)***
  - The User Datagram Protocol (also termed as the Universal Datagram Protocol) is a connectionless protocol that sends each packet during a session along different routes.
  - UDP does not provide the reliability and in-order data transmission as TCP does because data packets might arrive out of order or be dropped by the network and no acknowledgements are sent by the receiver.
  - However, as a result, UDP is much faster compared with TCP and efficient for the transmission of time-sensitive data, such as in the case of synchronized multimedia data.

- ***TCP/IP Family:***
  - TCP itself is a reliable connection-oriented protocol to transfer data on a network; however, to deliver data on the connectionless Internet, it has to ride on IP and provide reliability, which includes in-order delivery and errorfree delivery, with the IP protocol
  - Thus, this family of protocols has the commonality of using IP as a Network layer to communicate data while providing support from the higher layers. The TCP/IP families of protocols include general data transfer protocols, such as ***FTP, HTTP, IMAP, SNMP, SMTP, TELNET***, and so on.

Media-Related Protocols

Apart from the general protocol requirements such as those imposed by UDP or TCP, you also have media related requirements imposed by needs such as ***real time delivery***, ***resource reservation requirements*** for guaranteeing a certain QoS level, ***identifying the media type*** that is being communicated and so on.

Hypertext Transfer Protocol (HTTP):

- The Hypertext Transfer Protocol is an application layer protocol. It has been in use by the WWW.
- HTTP works in a ***request-response*** manner between a client and a server. The originating client, typically a Web browser running on a computer, cell phone, PDA, or other end terminal, requests a document residing at a server location.
- The destination server, typically called a Web server, responds by locating the document and sending the information back to the client. The information might be individual files of any kind or a collection of text, images, and other media organized in the form of an HTML document.

Real-Time Transport Protocol (RTP) with RTCP

- The original IP protocols used on the Internet provided delivery of data packets on a "best-effort" basis. Although this might suffice for non-real-time applications such as e-mail and FTP, it certainly is not suited to real-time multimedia applications.
- The RealTime Transport Protocol (RTP) along with a monitoring ***Real-Time Transport Control Protocol (RTCP)*** were designed to solve the real-time needs for multimedia traffic, such as video and audio streams in applications, video-on-demand, videoconferencing, and other live streaming media.
- ***RTP runs on top of UDP***, which provides efficient delivery over connectionless networks. It is preferably run over UDP rather than TCP because TCP's reliability achieved by in-order delivery of packets does not address the real-time needs of multimedia traffic if packets are lost or held up along the route.
- RTP has its own time stamping on each packet and sequencing mechanisms to help the receiver synchronize and render the information in sequence.

Real-Time Streaming Protocol (RTSP)

- It is useful in streaming media, as opposed to downloading the entire media content.
- RTSP is designed for streaming communication between a client and a stored media server and achieves real-time communication by issuing simple commands like play, pause, stop, and so on and, therefore, allows time-based access to files on a server.
- ***RTSP normally rides on top of RTP*** as the transport protocol for actual audio/video data.
- ***RTSP requests are sent over HTTP.***

Resource Reservation Setup Protocol (RSVP)

- The Resource Reservation Setup Protocol is a setup protocol where a ***receiver node reserves network resources during a communication session*** so as to efficiently monitor and guarantee the desired QoS for unicast and multicast multimedia communication over an integrated services Internet**.**
- RSVP works by applying two messages—a Path message and a Resv message.
  - A ***Path*** message is initiated by the sender to the receiver(s) and contains information of the path and nodes along the path to the receiver.
  - This is followed by one or more ***Resv*** messages from the receivers to reserve the intermediary node resources
- In itself, ***RSVP is not a routing protocol***, but works with the routing protocols to transmit data along allocated resources.

Session Initiation Protocol (SIP)

- The Session Initiation Protocol is an Application-layer control protocol that has been set up to ***establish, modify, and terminate sessions*** in Internet telephony.
- Although designed for ***Voice over IP*** applications, it is not limited to it, and is also used for multimedia-related communications, such as ***videoconferences***, ***multimedia distribution***, and so forth.
- SIP transparently supports ***name mapping*** and ***redirection***, which increases personal mobility, so users can maintain an external visible identifier regardless of their network location.
- SIP is a client/server protocol where the sender client node connects to another receiver client node in a multimedia application by initiating a request to a server(s).

Session Description Protocol (SDP)

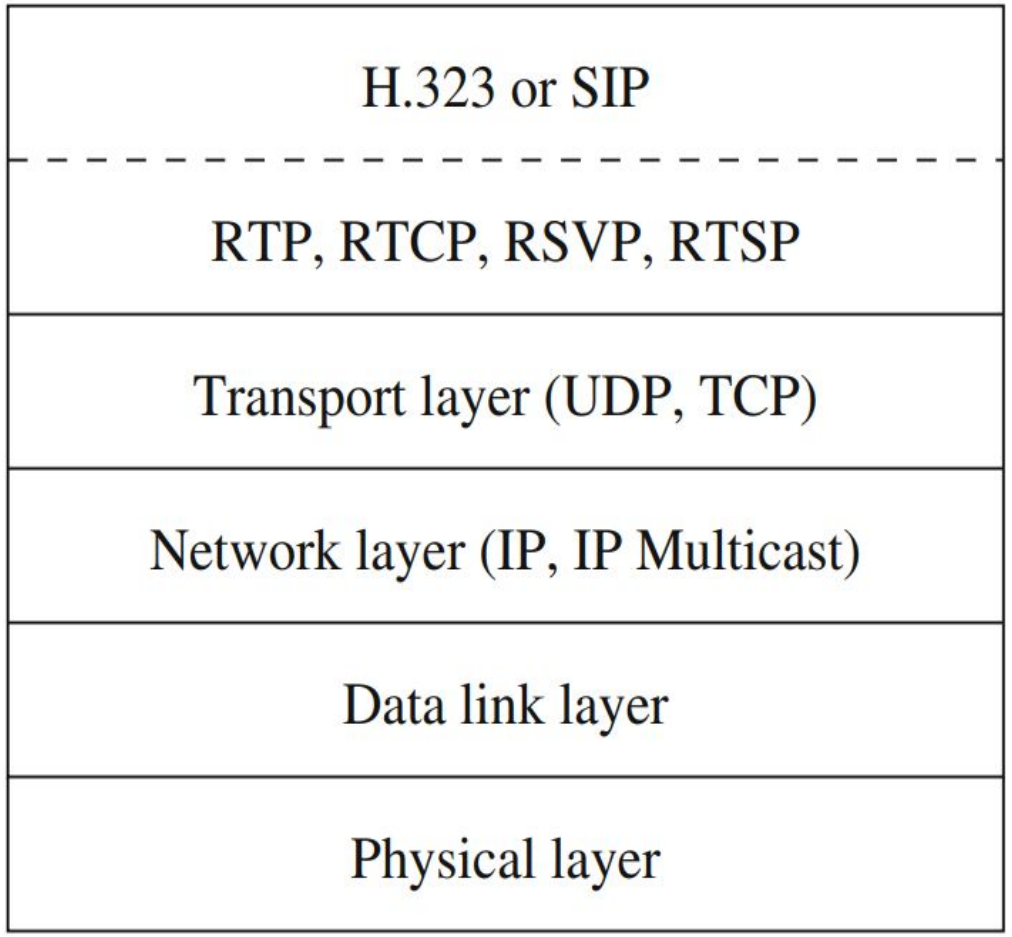
- The Session Description Protocol describes a multimedia session that needs to be in session between ***a caller client*** and ***a callee client***.
- A caller client must include this information when inviting the callee client during the initiation of communication, such as in the SIP Invite command.
- The description information includes the ***media stream types*** used in the session (audio, video, and so on) and ***its capabilities***, ***destination address*** (unicast or multicast) for each stream, ***sending and receiving port numbers, stream type indicators***, and other things necessary to communicate media information.

A Case Study: Internet Telephony or Voice over IP (VoIP)

- With ever-increasing network bandwidth and the ever-improving quality of multimedia data compression, Internet telephony has become a reality.
- The main advantages of Internet telephony over the plain old telephone services are
  - It provides great flexibility and extensibility in accommodating such new services as voicemail, video conversations, live text messages, and so on.
  - It uses packet switching, not circuit switching; hence, network usage is much more efficient.
  - With the technologies of multicast or multipoint communication, multiparty calls are not much more difficult than two-party calls.
  - With advanced multimedia data-compression techniques, various degrees of QoS can be supported and dynamically adjusted according to the network traffic.
  - Richer graphical user interfaces can be developed to show available features and services, monitor call status and progress, and so on.

Network protocol structure for internet telephony

- The transport of real-time audio (and video) in Internet telephony is supported by RTP (with its control protocol, RTCP).
- Streaming media is handled by RTSP and Internet resource reservation, if available, is taken care of by RSVP.



Multimedia Systems
Lecture – 34

By
Dr. Priyambada Subudhi
Assistant Professor
IIIT Sri City

Multimedia Conferencing

- It is the live connection between two or more remote parties over the internet through audio and/or visual medium.
- ***Video conferencing*** is a type of online meeting where two or more people engage in a live audio-visual call. With a strong internet connection, the participants can see, hear, and talk to each other in real time, no matter where in the world they are.
- Video conferencing brings people working from different places together in a virtual meeting room. To make that possible, we'll need:
  - A stable internet connection
  - A video display device (laptop, desktop monitor, or a television screen)
  - A computer or conference phone
  - Other peripherals (webcam, microphone, headset, speaker, etc.)
  - Video conferencing software

Types of video conferencing

1. Point-to-point conferencing:

In point-to-point video conferencing, there are only two participants communicating from different locations in real time.

e.g: One-on-one customer support, Job Interviews

2. Multipoint conferencing

Multipoint video conferencing involves three or more participants; that’s why it’s also called “group video conferencing”.

e.g: Team meetings, Webinars

How does video conferencing work?

- Video conferencing is powered by VoIP, the technology that makes voice communications over the internet possible. In order to transfer audio and video signals between two locations, VoIP relies on special algorithms called ***codecs (coder-decoder)***.

Data compression (coding)

- Imagine an ongoing video meeting. The camera captures analog video signals and, when someone speaks, the microphone captures their audio signals. VoIP turns these signals into data packets for the internet to understand and for the transfer to begin.

Data transfer and decompression (decoding)

- The packets of data travel over the internet. When they reach their destination, they change back into analog video and audio signals for the attendees on the other side to see and hear.

- Conferencing solutions typically come with echo cancellation to eliminate sound delays. This way, the audio and video remain in sync.
- Depending on the provider you choose, you'll find that there are numerous other features that make video conferencing such a powerful business tool. Some features are
  - **Screen sharing**
  - **Chat box**
  - **File sharing**
  - **Video call recording**
  - **Noise cancellation**

Video on Demand (VOD)

- **Video on demand** (**VOD**) is a media distribution system that allows users to access videos without a traditional video playback device and the constraints of a typical static broadcasting schedule.
- VOD, or video-on-demand, is any content distribution platform that **gives viewers the ability to choose when, where, and how they view media.**
- Because VOD is streamed via the internet, it doesn't rely on cable or satellite connections like traditional broadcast television. If you have enough bandwidth, you can watch!
- This allows users to find and watch pre-recorded streaming content using any internet-enabled device.

- Video-on-demand has 3 core advantages. It allows viewers to:
  - **Watch at any time.** Users can play content whenever they want. Unlike “linear” TV programming, which only broadcasts in real-time, VOD doesn’t rely on a set schedule.
  - **Control what they watch:** Users have more opportunities to pick and choose what they watch, compared to traditional TV scheduling.
  - **Use media controls:** Users can play, pause, rewind, fast forward, and completely control how they watch content. This way, they’ll never miss anything important.

How VOD Works?

- To deliver a video on demand, the video is first converted into a digital format and stored on a video server. It is then compressed and transmitted to the viewer via broadband or cable.
- After reaching its destination, the video is decoded and decompressed via a ***set-top box*** and stored on a video server in the viewer's device.
- The viewer can then watch the video instantly with the ability to control its speed and other (play, stop) features.

How Does VOD Makes Money?

Providers of VOD usually create impactful video on demand content to make money.

Different Types Of VOD Models To Choose

1.TVOD Or Transactional VOD:

- The transactional video on demand is based on a pay-per-view monetization model and best suited for the most popular videos with small video libraries. Here, the viewers purchase or go for digital renting for the specific content they want to watch.
- For example, buying full seasons or individual episodes of your favorite tv shows through Amazon Prime

SVOD Or Subscription VOD

- The subscription video on demand is the most popular monetization model for varied content creators in order to broadcast with large video libraries. Here, the viewers subscribe to a video service for a certain period of time (weekly, monthly, yearly) to access its content.
- E.g: Netflix, Amazon Prime

AVOD (Advertising Video-On-Demand)

- AVOD, or Advertising Video-On-Demand, is essentially “free” for viewers because there’s no up-front cost to watch.
- AVOD revenue comes from businesses paying to advertise with short commercials throughout your videos.