Chapter 4. Video and Audio Compression

4.1. Lossless Compression Algorithms

Reference: Mark Nelson, "The Data Compression Book", 2nd ed., M&T Books, 1995. QA 76.9 D33 N46 1995

Reference: Khalid Sayood, "Introduction to Data Compression", Morgan Kaufmann, 1996. TK 5102 92 S39 1996

4.1.1 Basics of Information Theory

According to Shannon, the entropy of an information source S is defined as:

$$H(S) = \eta = \sum_i p_i \log_2 \frac{1}{p_i}$$

where p_i is the probability that symbol S_i in S will occur.

- $\log_2 \frac{1}{p_i}$ indicates the amount of information contained in S_i, i.e., the number of bits needed to code S_i.
- For example, in an image with uniform distribution of gray-level intensity, i.e. $p_i = 1/256$, then the number of bits needed to code each gray level is 8 bits. The entropy of this image is 8.
- Q: How about an image in which half of the pixels is white (I = 220) and half is black (I = 10)?

The Shannon-Fano Algorithm

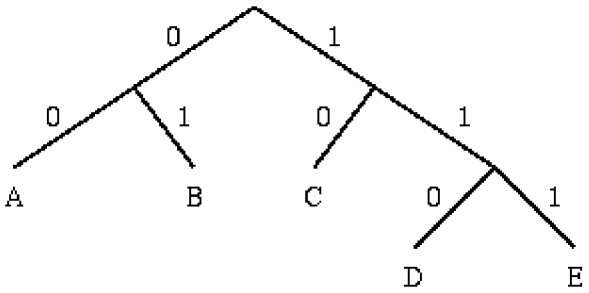
A simple example will be used to illustrate the algorithm:

Symbol	A	B	C	D	E
Count	15	7	6	6	5

Encoding for the Shannon-Fano Algorithm:

- A top-down approach

1. Sort symbols according to their frequencies/probabilities, e.g., ABCDE.
2. Recursively divide into two parts, each with approx. same number of counts.



Symbol	Count	$\log_2(1/p_i)$	Code	Subtotal (# of bits)
A	15	1.38	00	30
B	7	2.48	01	14
C	6	2.70	10	12
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 89

4.1.2 Huffman Coding

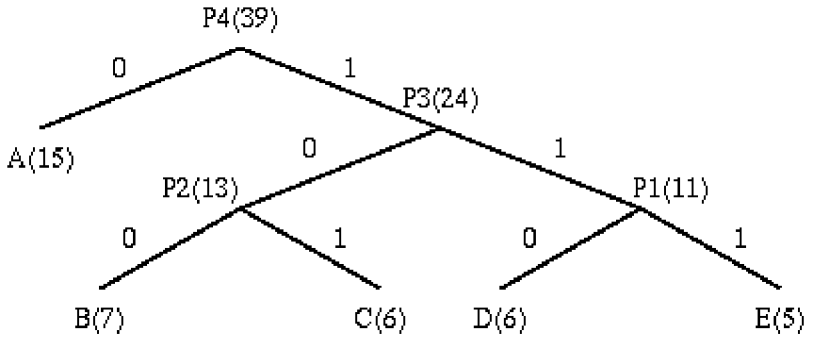
Encoding for Huffman Algorithm:

- A bottom-up approach

1. Initialization: Put all nodes in an OPEN list, keep it sorted at all times (e.g., ABCDE).

2. Repeat until the OPEN list has only one node left:

(a) From OPEN pick two nodes having the lowest frequencies/probabilities, create a parent node of them.
(b) Assign the sum of the children's frequencies/probabilities to the parent node and insert it into OPEN.
(c) Assign code 0, 1 to the two branches of the tree, and delete the children from OPEN.



Symbol	Count	$\log_2(1/p_i)$	Code	Subtotal (# of bits)
A	15	1.38	0	15
B	7	2.48	100	21
C	6	2.70	101	18
D	6	2.70	110	18
E	5	2.96	111	15

TOTAL (# of bits): 87

Discussions:

- Decoding for the above two algorithms is trivial as long as the coding table (the statistics) is sent before the data. (There is a bit overhead for sending this, negligible if the data file is big.)
- **Unique Prefix Property**: no code is a prefix to any other code (all symbols are at the leaf nodes) --> great for decoder, unambiguous.
- If prior statistics are available and accurate, then Huffman coding is very good.

```
In the above example:

entropy = (15 x 1.38 + 7 x 2.48 + 6 x 2.7 + 6 x 2.7 + 5 x 2.96) / 39
        = 85.26 / 39 = 2.19

Number of bits needed for Human Coding is: 87 / 39 = 2.23
```

4.1.3 Adaptive Huffman Coding

Motivations:

(a) The previous algorithms require the statistical knowledge which is often not available (e.g., live audio, video).
(b) Even when it is available, it could be a heavy overhead especially when many tables had to be sent when a non-order0 model is used, i.e. taking into account the impact of the previous symbol to the probability of the current symbol (e.g., "qu" often come together, ...).

The solution is to use adaptive algorithms. As an example, the Adaptive Huffman Coding is examined below. The idea is however applicable to other adaptive compression algorithms.

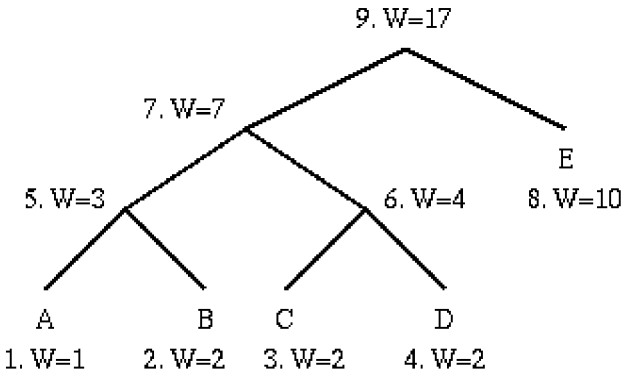
```
ENCODER                                  DECODER
-------                                  -------

Initialize_model();                      Initialize_model();
while ((c = getc (input)) != eof)        while ((c = decode (input)) != eof)
  {                                        {
    encode (c, output);                      putc (c, output);
    update_model (c);                        update_model (c);
  }                                        }
```

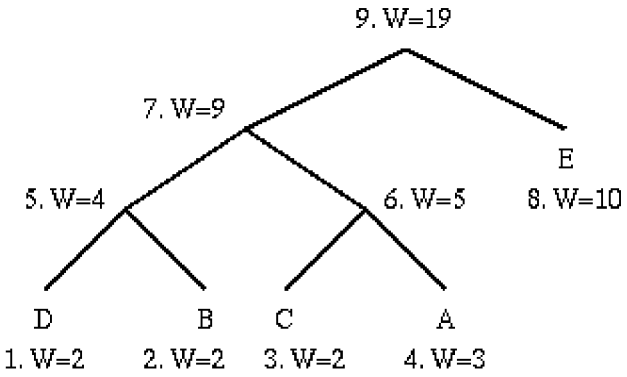
- The key is to have both encoder and decoder to use exactly the same *initialization* and *update_model* routines.
- *update_model* does two things: (a) increment the count, (b) update the Huffman tree.
    - During the updates, the Huffman tree will be maintained its *sibling property*, i.e. the nodes (internal and leaf) are arranged in order of increasing weights (see figure).

- When *swapping* is necessary, the farthest node with weight W is swapped with the node whose weight has just been increased to W+1.
  Note: If the node with weight W has a subtree beneath it, then the subtree will go with it.

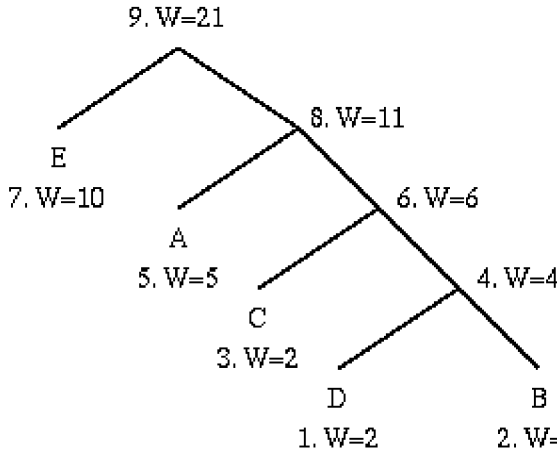
- The Huffman tree could look very different after node swapping, e.g., in the third tree, node A is again swapped and becomes the #5 node. It is now encoded using only 2 bits.



A Huffman Tree



After a node switch (A was incremented twice)



After A was incremented two more time

Note: Code for a particular symbol changes during the adaptive coding process.

4.1.4 Lempel-Ziv-Welch Algorithm

Motivation:

Suppose we want to encode the Webster's English dictionary which contains about 159,000 entries. Why not just transmit each word as an 18 bit number?

Problems: (a) Too many bits, (b) everyone needs a dictionary, (c) only works for English text.

- Solution: Find a way to build the dictionary adaptively.

- Original methods due to Ziv and Lempel in 1977 and 1978. Terry Welch improved the scheme in 1984 (called LZW compression). It is used in e.g., UNIX *compress*, GIF, V.42 bis.

Reference: Terry A. Welch, "A Technique for High Performance Data Compression", IEEE Computer, Vol. 17, No. 6, 1984, pp. 8-19.

LZW Compression Algorithm:

```
w = NIL;
while ( read a character k )
    {
      if wk exists in the dictionary
       w = wk;
      else
        add wk to the dictionary;
        output the code for w;
        w = k;
    }
```

- Original LZW used dictionary with 4K entries, first 256 (0-255) are ASCII codes.

Example: Input string is "^WED^WE^WEE^WEB^WET".

w	*k*	*Output*	*Index*	*Symbol*
NIL	^			
^	W	^	256	^W
W	E	W	257	WE
E	D	E	258	ED
D	^	D	259	D^
^	W			
^W	E	256	260	^WE
E	^	E	261	E^
^	W			
^W	E			
^WE	E	260	262	^WEE
E	^			
E^	W	261	263	E^W
W	E			
WE	B	257	264	WEB
B	^	B	265	B^
^	W			
^W	E			
^WE	T	260	266	^WET
T	EOF	T		

- A 19-symbol input has been reduced to 7-symbol plus 5-code output. Each code/symbol will need more than 8 bits, say 9 bits.

- Usually, compression doesn't start until a large number of bytes (e.g., > 100) are read in.

LZW Decompression Algorithm:

```
read a character k;
output k;
w = k;
while ( read a character k )    /* k could be a character or a code. */
    {
      entry = dictionary entry for k;
      output entry;
      add w + entry[0] to dictionary;
      w = entry;
    }
```

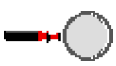
Example (continued): Input string is "^WED<256>E<260><261><257>B<260>T".

w	*k*	*Output*	*Index*	*Symbol*
	^	^		
^	W	W	256	^W
W	E	E	257	WE
E	D	D	258	ED
D	<256>	^W	259	D^
<256>	E	E	260	^WE
E	<260>	^WE	261	E^
<260>	<261>	E^	262	^WEE
<261>	<257>	WE	263	E^W
<257>	B	B	264	WEB
B	<260>	^WE	265	B^
<260>	T	T	266	^WET

- Problem: What if we run out of dictionary space?
    - Solution 1: Keep track of unused entries and use LRU (Least Recently Used)
    - Solution 2: Monitor compression performance and flush dictionary when performance is poor.
- Implementation Note: LZW can be made *really* fast; it grabs a fixed number of bits from input stream, so bit parsing is very easy. Table lookup is automatic.

Summary

- Huffman maps fixed length symbols to variable length codes. Optimal only when symbol probabilities are powers of 2.
- Lempel-Ziv-Welch is a dictionary-based compression method. It maps a variable number of symbols to a fixed length code.
- Adaptive algorithms do not need a priori estimation of probabilities, they are more useful in real applications.

Further Exploration 

The Squeeze Page

4.2. Image Compression -- JPEG

Reference: W.B. Pennebaker, J.L. Mitchell, "The JPEG Still Image Data Compression Standard", Van Nostrand Reinhold, 1993.

4.2.1. Overview of JPEG

What is JPEG?

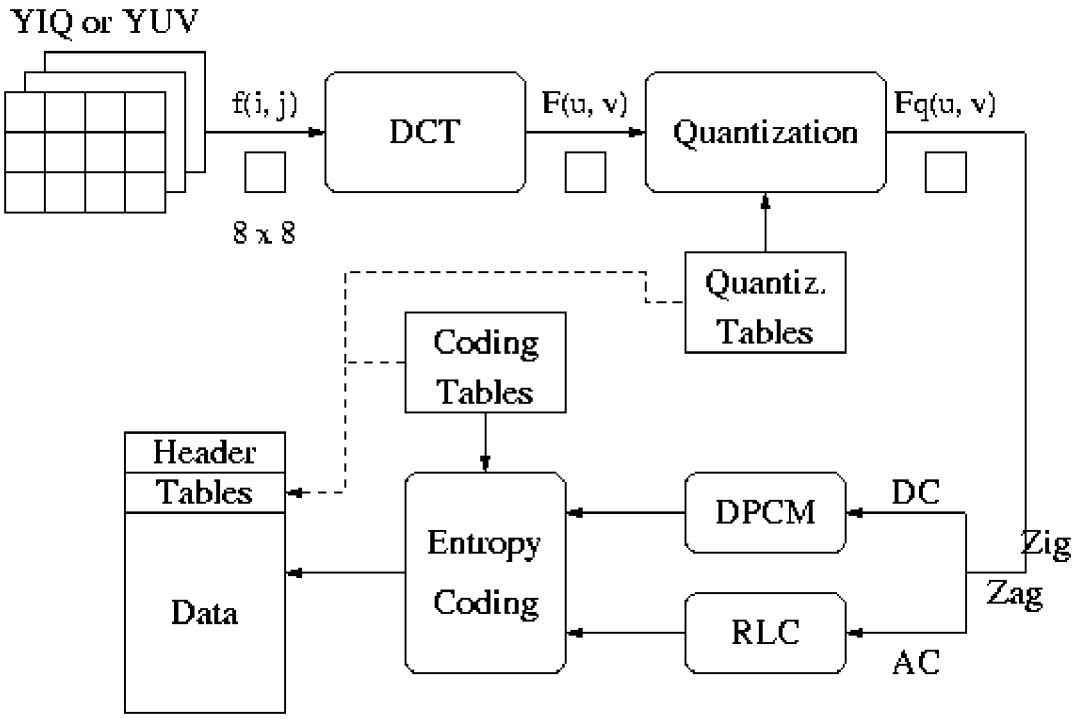
- "Joint Photographic Expert Group". Voted as international standard in 1992.
- Works with color and grayscale images, e.g., satellite, medical, ...

Motivation

- The *compression ratio* of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression, especially when the distribution of pixel values is relatively flat.

JPEG overview

- Encoding



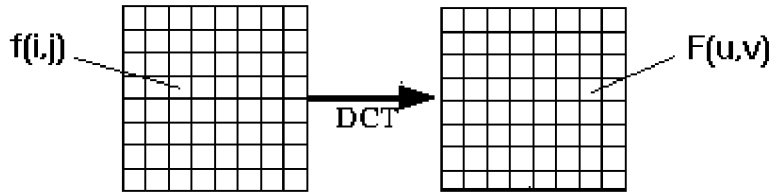
- Decoding -- Reverse the order

4.2.2. Major Steps

- DCT (Discrete Cosine Transformation)
- Quantization
- Zigzag Scan
- DPCM on DC component
- RLE on AC Components
- Entropy Coding

1. Discrete Cosine Transform (DCT)

- From spatial domain to frequency domain:



- **DEFINITIONS**

  Discrete Cosine Transform (DCT):

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

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

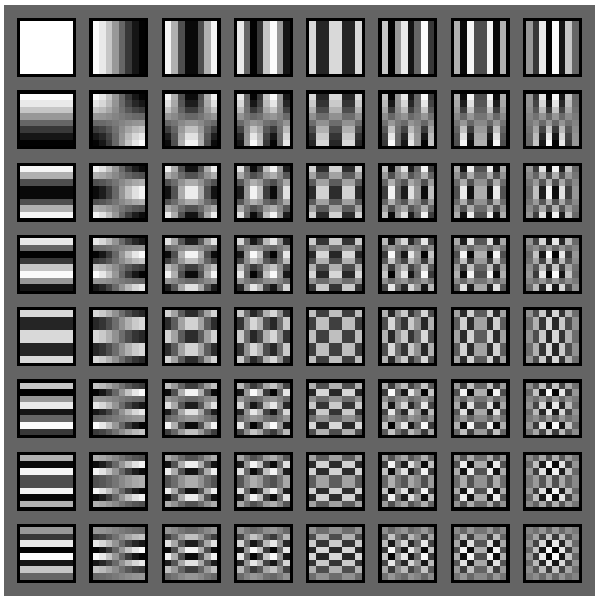
Inverse Discrete Cosine Transform (IDCT):

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

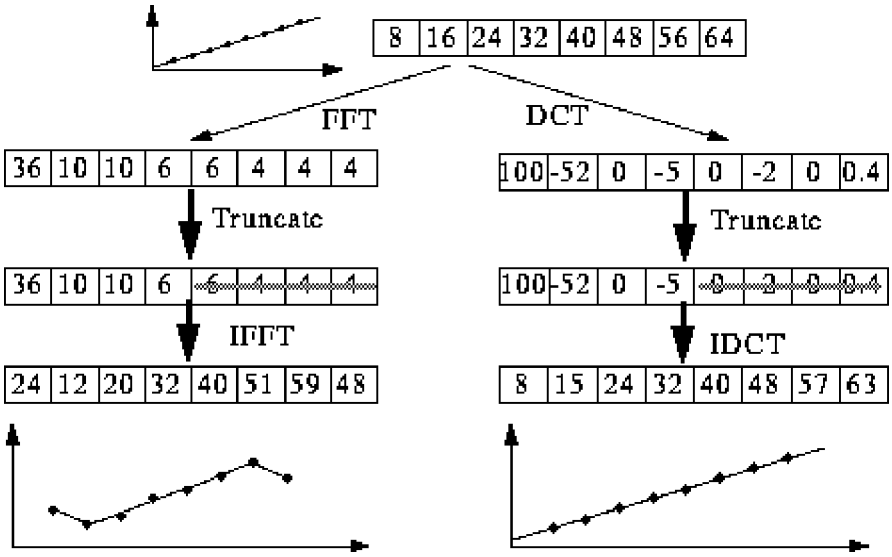
$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & for\ \xi = 0 \\ 1 & otherwise \end{cases}$$

Question: What are the DC and AC components, e.g., what is F[0,0]?

- The 64 (8 x 8) DCT basis functions:



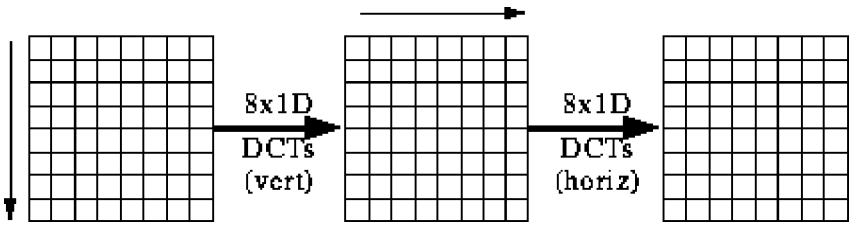
- Why DCT not FFT? -- DCT is like FFT, but can approximate linear signals well with few coefficients.



- Computing the DCT
  - Factoring reduces problem to a series of 1D DCTs:

$$F[u,v] = \frac{1}{2}\sum_i \Lambda(u)\cos\frac{(2i+1)u\pi}{16} G[i,v]$$

$$G[i,v] = \frac{1}{2}\sum_j \Lambda(v)\cos\frac{(2j+1)v\pi}{16} f[i,j]$$



  - Most software implementations use fixed point arithmetic. Some fast implementations approximate coefficients so all multiplies are shifts and adds.

2. Quantization

- F'[u, v] = round (F[u, v] / q[u, v]).

  Why? -- To reduce number of bits per sample

  Example: 101101 = 45 (6 bits).
  q[u, v] = 4 --> Truncate to 4 bits: 1011 = 11.

- Quantization error is the main source of the Lossy Compression.

Uniform Quantization

- Each F[u,v] is divided by the same constant *N*.

Non-uniform Quantization -- Quantization Tables

- Eye is most sensitive to low frequencies (upper left corner), less sensitive to high frequencies (lower right corner)

- The *Luminance Quantization Table* q(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
---------------------------------
```

The *Chrominance Quantization Table* q(u, v)

```
-------------------------------
17  18  24  47  99  99  99  99
18  21  26  66  99  99  99  99
24  26  56  99  99  99  99  99
47  66  99  99  99  99  99  99
99  99  99  99  99  99  99  99
99  99  99  99  99  99  99  99
99  99  99  99  99  99  99  99
99  99  99  99  99  99  99  99
-------------------------------
```

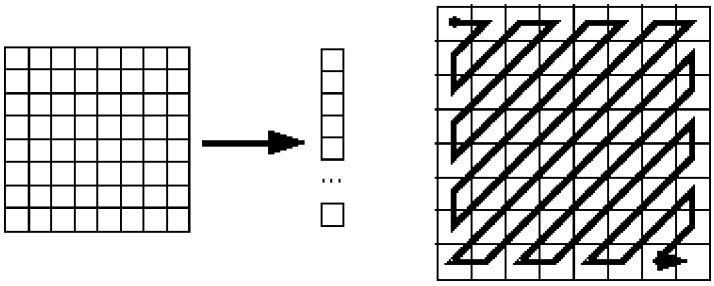
The numbers in the above quantization tables can be scaled up (or down) to adjust the so called *quality factor*.

Custom quantization tables can also be put in image/scan header.

3. Zig-zag Scan

- Why? -- to group low frequency coefficients in top of vector.

- Maps 8 x 8 to a 1 x 64 vector



4. Differential Pulse Code Modulation (DPCM) on DC component

- DC component is large and varied, but often close to previous value.

- Encode the difference from previous 8 x 8 blocks -- DPCM

5. Run Length Encode (RLE) on AC components

- 1 x 64 vector has lots of zeros in it

- Keeps *skip* and *value*, where *skip* is the number of zeros and *value* is the next non-zero component.

- Send (0,0) as end-of-block sentinel value.

6. Entropy Coding

- Categorize DC values into SIZE (number of bits needed to represent) and actual bits.

```
-----------------------------------
   SIZE               Value
-----------------------------------
     1              -1, 1
```

```
 2           -3, -2, 2, 3
 3           -7..-4, 4..7
  4          -15..-8, 8..15
  .                  .
  .                  .
  .                  .
 10      -1023..-512, 512..1023
-----------------------------------
```

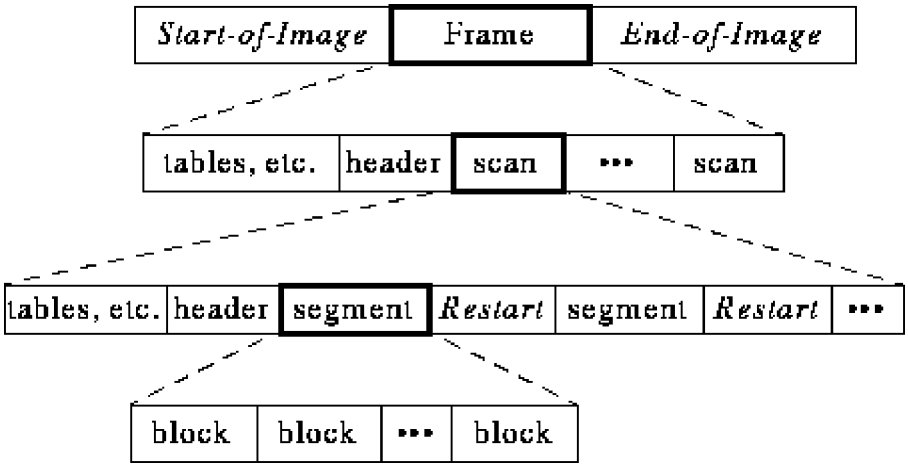
Example: if DC value is 4, 3 bits are needed.

Send off SIZE as Huffman symbol, followed by actual 3 bits.

- For AC components two symbols are used: Symbol_1: (*skip*, *SIZE*), Symbol_2: actual bits. Symbol_1 (*skip*, *SIZE*) is encoded using the Huffman coding, Symbol_2 is not encoded.

- Huffman Tables can be custom (sent in header) or default.

4.2.3. A Glance at the JPEG Bitstream



- A "Frame" is a picture, a "scan" is a pass through the pixels (e.g., the red component), a "segment" is a group of blocks, a "block" is an 8 x 8 group of pixels.

- Frame header:
  sample precision
  (width, height) of image
  number of components
  unique ID (for each component)
  horizontal/vertical sampling factors (for each component)
  quantization table to use (for each component)

- Scan header
  Number of components in scan
  component ID (for each component)
  Huffman table for each component (for each component)

- Misc. (can occur between headers)
  Quantization tables
  Huffman Tables
  Arithmetic Coding Tables
  Comments

Application Data

4.2.4. Four JPEG Modes

- Sequential Mode
- Lossless Mode
- Progressive Mode
- Hierarchical Mode

** In "Motion JPEG", Sequential JPEG is applied to each image in a video.

1. Sequential Mode

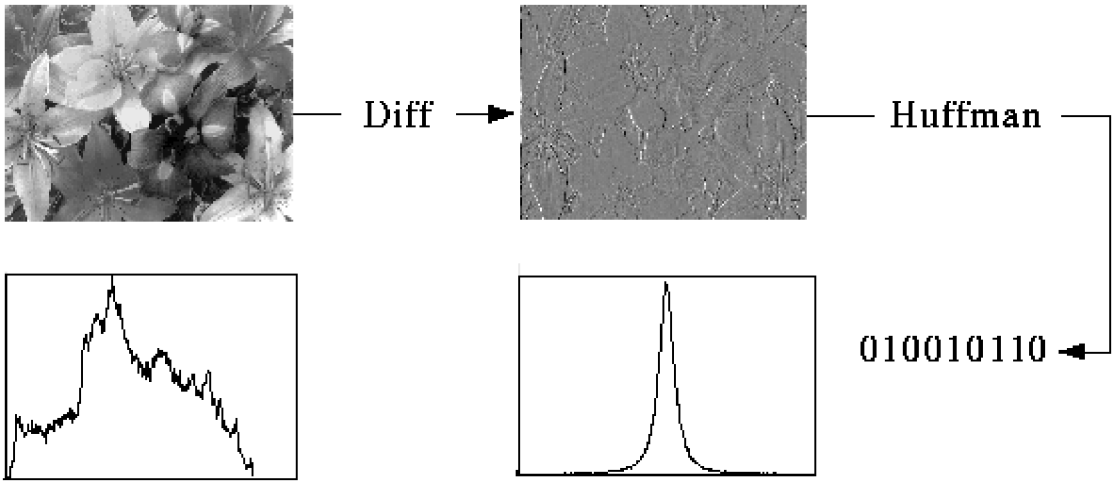
- Each image component is encoded in a single left-to-right, top-to-bottom scan.

  Baseline Sequential Mode, the one that we described above, is a simple case of the Sequential mode:
  - It supports only 8-bit images (not 12-bit images)
  - It uses only Huffman coding (not Arithmetic coding)

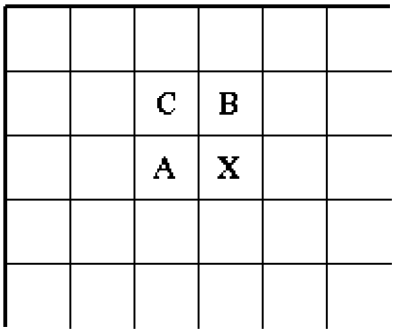
2. Lossless Mode

- A special case of the JPEG where indeed there is no loss.

  Its block diagram is as below:



- It does not use DCT-based method! Instead, it uses a *predictive* method:
  A predictor combines the values of up to three neighboring pixels (not blocks as in the Sequential mode) as the predicted value for the current pixel, indicated by "X" in the figure below. The encoder then compares this prediction with the actual pixel value at the position "X", and encodes the difference losslessly.

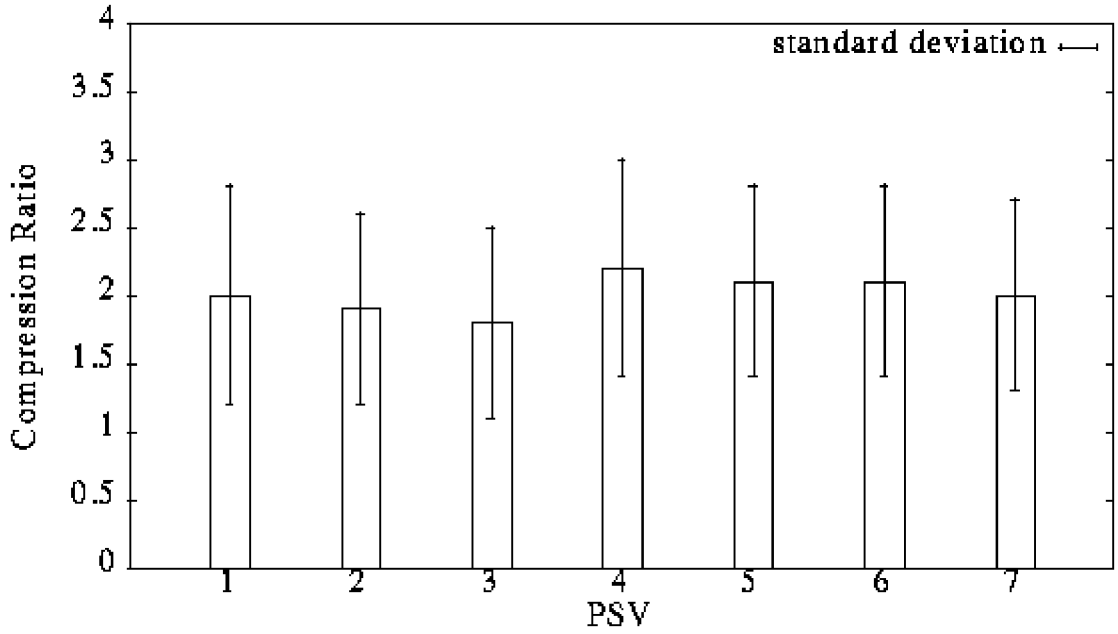


- It can use any one of the following seven predictors :

Predictor	Prediction
1	A
2	B
3	C
4	A + B - C
5	A + (B - C) / 2
6	B + (A - C) / 2
7	(A + B) / 2

  Since it uses only previously encoded neighbors, the very first pixel I(0, 0) will have to use itself. Other pixels at the first row always use P1, at the first column always use P2.

- Effect of Predictor (test result with 20 images):



Note: "2D" predictors (4-7) always do better than "1D" predictors.

Comparison with Other Lossless Compression Programs (compression ratio):

Compression Program	Compression Ratio			
	Lena	football	F-18	flowers
lossless JPEG	1.45	1.54	2.29	1.26
optimal lossless JPEG	1.49	1.67	2.71	1.33
compress (LZW)	0.86	1.24	2.21	0.87
gzip (Lempel-Ziv)	1.08	1.36	3.10	1.05
gzip -9 (optimal Lempel-Ziv)	1.08	1.36	3.13	1.05
pack (Huffman coding)	1.02	1.12	1.19	1.00

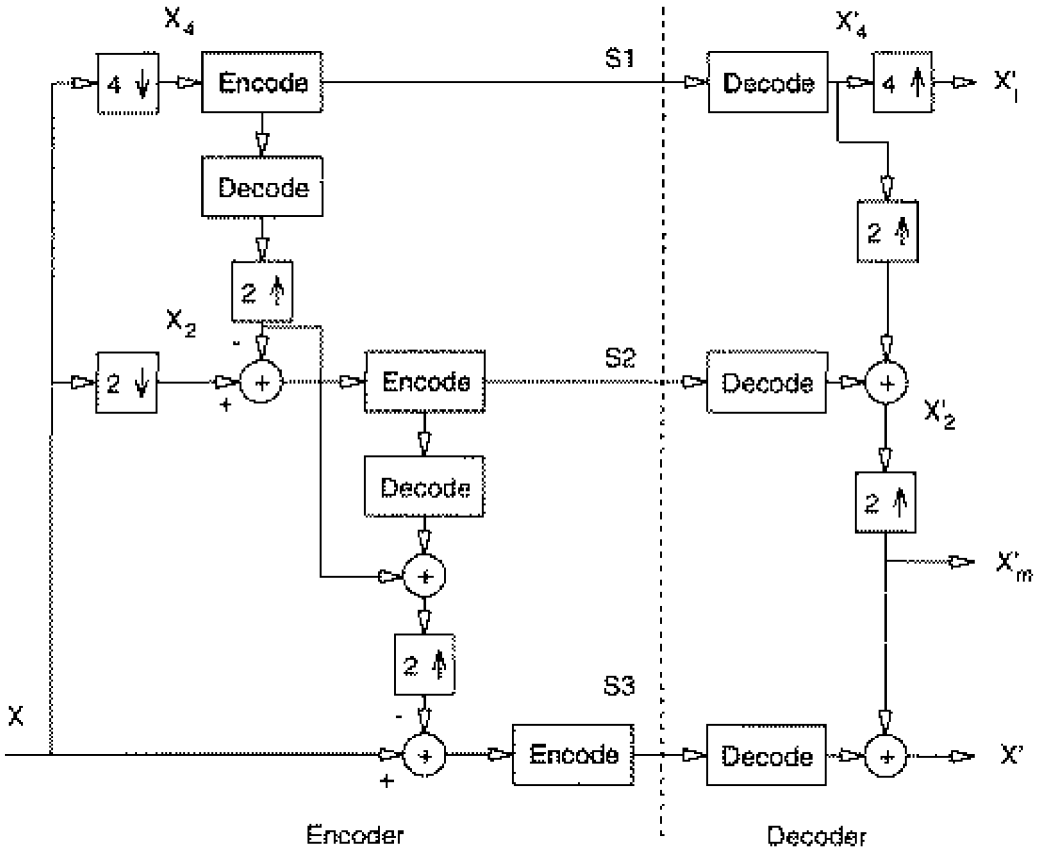
3. Progressive Mode

- Goal: display low quality image and successively improve.
- Two ways to successively improve image:
    1. *Spectral selection*: Send DC component and first few AC coefficients first, then gradually some more ACs.
    2. *Successive approximation*: send DCT coefficients MSB (most significant bit) to LSB (least significant bit).
    (Effectively, it is sending quantized DCT coefficients frist, and then the difference between the quantized and the non-quantized coefficients with finer quantization stepsize.)

4. Hierarchical Mode

A Three-level Hierarchical JPEG Encoder

(From *V. Bhaskaran and K. Konstantinides, "Image and Video Compression Standards: Algorithms and Architectures", 2nd ed., Kluwer Academic Publishers, 1997.*)



(a) Down-sample by factors of 2 in each dimension, e.g., reduce 640 x 480 to 320 x 240

(b) Code smaller image using another JPEG mode (Progressive, Sequential, or Lossless).

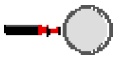
(c) Decode and up-sample encoded image

(d) Encode difference between the up-sampled and the original using Progressive, Sequential, or Lossless.

- Can be repeated multiple times.
- Good for viewing high resolution image on low resolution display.

4.2.5. JPEG 2000

- JPEG 2000 is the upcoming standard for Still Pictures (due Year 2000).
- Among many things it will address:
  - Low bit-rate compression performance,
  - Lossless and lossy compression in a single codestream,
  - Transmission in noisy environment where bit-error is high,
  - Application to both gray/color images and bi-level (text) imagery, natural imagery and computer generated imagery,
  - Interface with MPEG-4,
  - Content-based description.

Further Exploration 

Try the Interactive JPEG examples and the JPEG examples.

Information about JPEG 2000.

Last Updated: ***6/30/98***

Top | Chap 4 | CMPT 365 Home Page | CS

4.3. Video Compression

Reference: Chapter 6 of Steinmetz and Nahrstedt

- Uncompressed video data are huge. In HDTV, the bit-rate could exceed 1 Gbps. --> big problems for storage and network communications.
- We will discuss both Spatial and Temporal Redundancy Removal -- *Intra-frame and Inter-frame coding*.

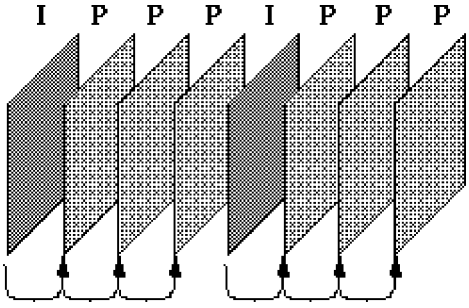
4.3.1. H. 261

- Developed by CCITT (Consultative Committee for International Telephone and Telegraph) in 1988-1990
- Designed for videoconferencing, videotelephone applications over ISDN telephone lines.

  Bit-rate is *p* x 64 Kb/sec, where *p* ranges from 1 to 30.

1. Overview of H. 261

- Frame Sequence

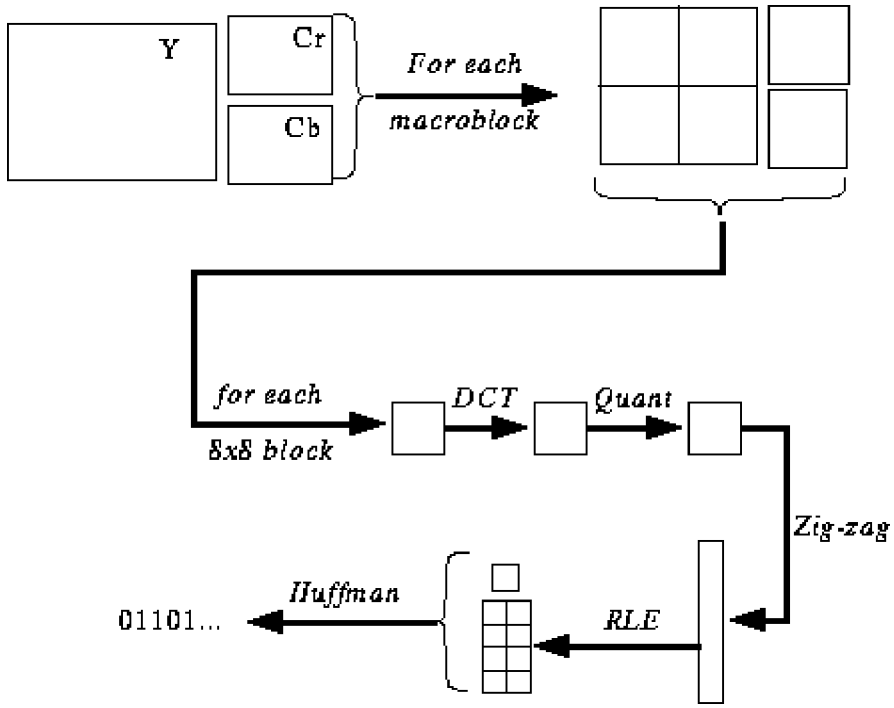


- Frame types are CCIR 601 CIF (352 x 288) and QCIF (176 x 144) images with 4:2:0 subsampling.
- Two frame types: Intra-frames (*I-frames*) and Inter-frames (*P-frames*):

  I-frame provides an accessing point, it uses basically JPEG.

  P-frames use "**pseudo-differences**" from previous frame ("predicted"), so frames depend on each other.

2. Intra-frame Coding



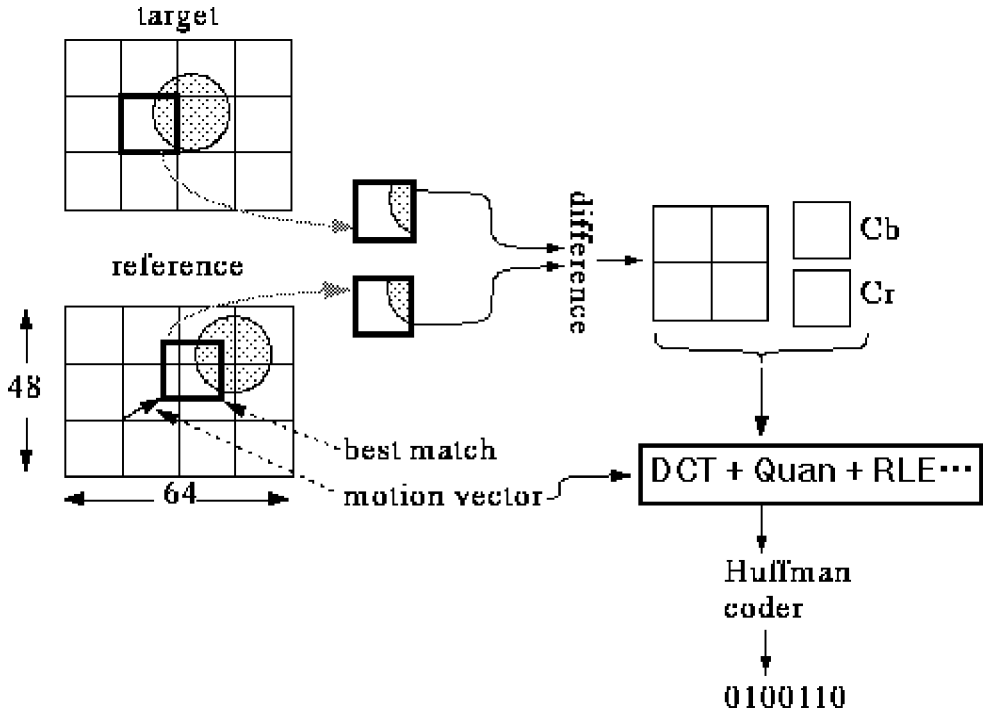
- Macroblocks are 16 x 16 pixel areas on Y plane of original image.

  A macroblock usually consists of 4 Y blocks, 1 Cr block, and 1 Cb block.

- Quantization is by constant value for all DCT coefficients (i.e., no quantization table as in JPEG).

3. Inter-frame (P-frame) Coding

- An Coding Example (P-frame)

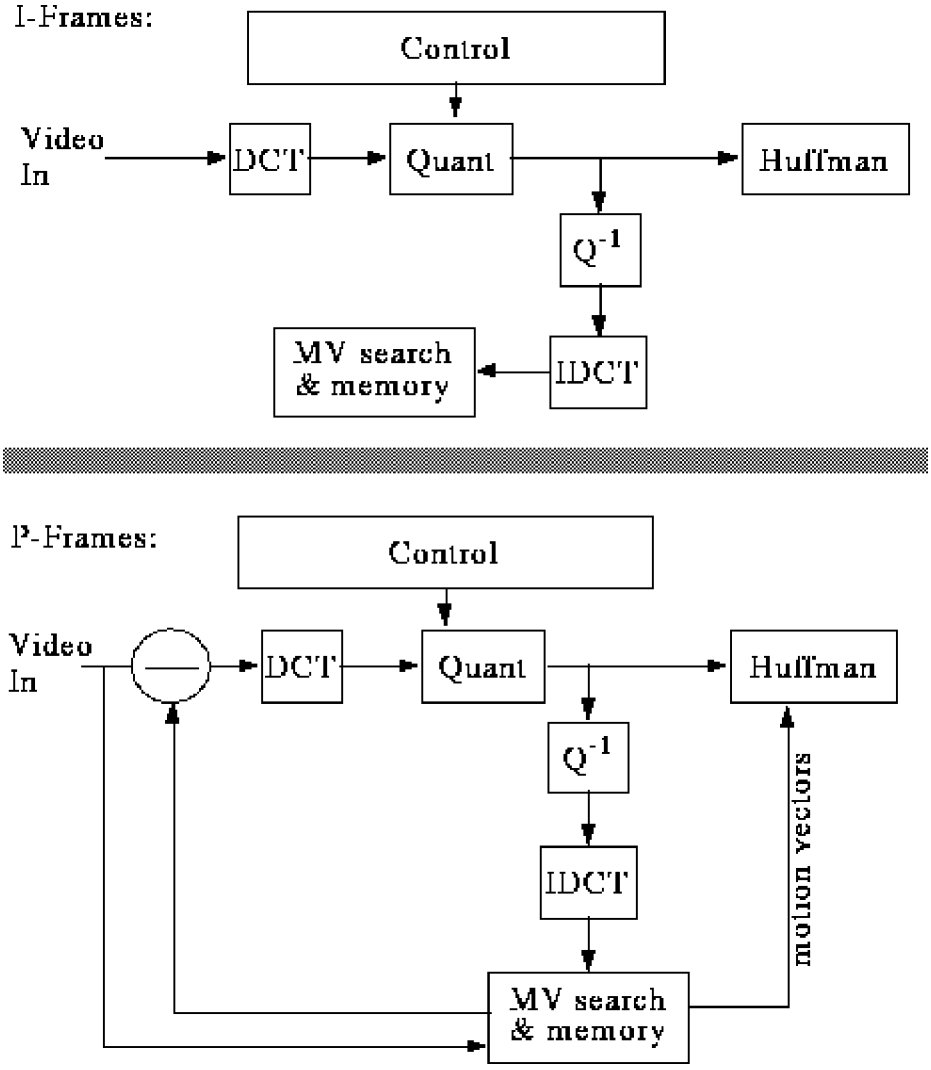


- Previous image is called *reference image*, the image to encode is called *target image*.

- **Points to emphasize:**

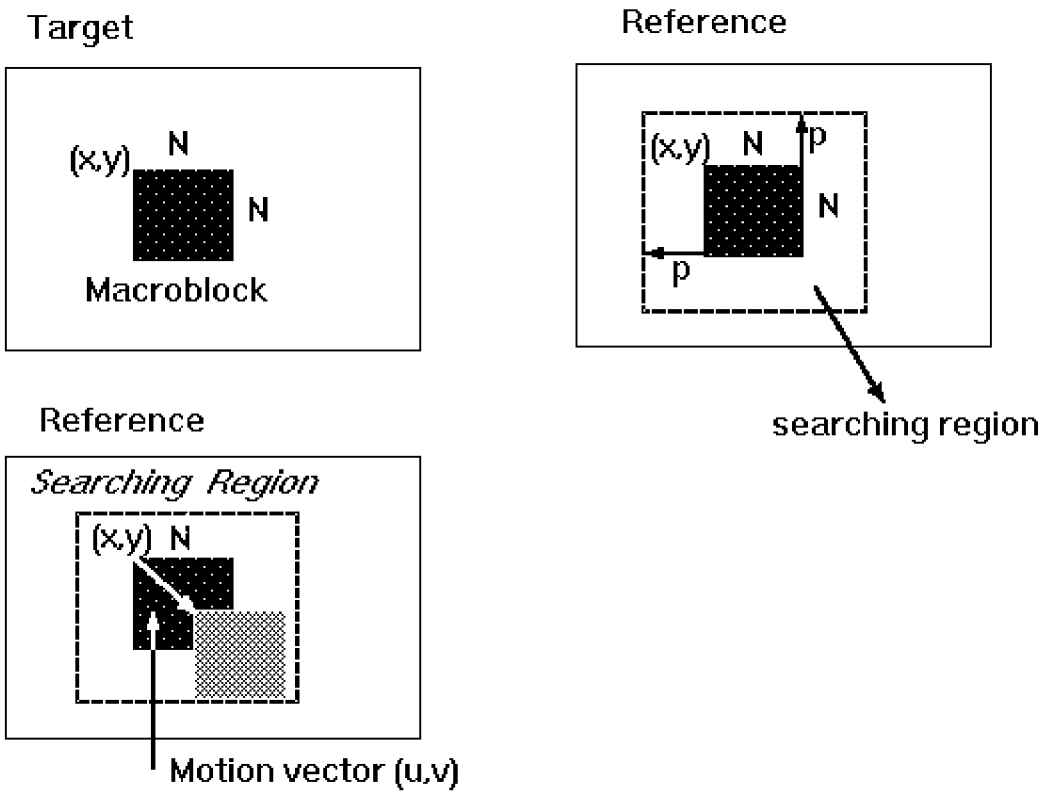
  1. The difference image (not the target image itself) is encoded.
  2. Need to use the decoded image as reference image, *not* the original.
  3. We're using "Mean Absolute Error" (MAE) to decide best block.
     Can also use "Mean Squared Error" (MSE) = sum(E*E)/N

4. H. 261 Encoder



- "Control" -- controlling the bit-rate. If the transmission buffer is too full, then bit-rate will be reduced by changing the quantization factors.
- "memory" -- used to store the reconstructed image (blocks) for the purpose of motion vector search for the next P-frame.

5. Methods for Motion Vector Searches



- C(x + k, y + l) -- pixels in the macro block with upper left corner (x, y) in the Target frame.

  R(x + i + k, y + j + l) -- pixels in the macro block with upper left corner (x + i, y + j) in the Reference frame.

  Cost function is:

$$MAE(i,j) = \frac{1}{N^2} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} |C(x+k, y+l) - R(x+i+k, y+j+l)|$$

  Where *MAE* stands for *Mean Absolute Error*.

- Goal is to find a vector (u, v) such that *MAE*(u, v) is minimum.

5.1 Full Search Method

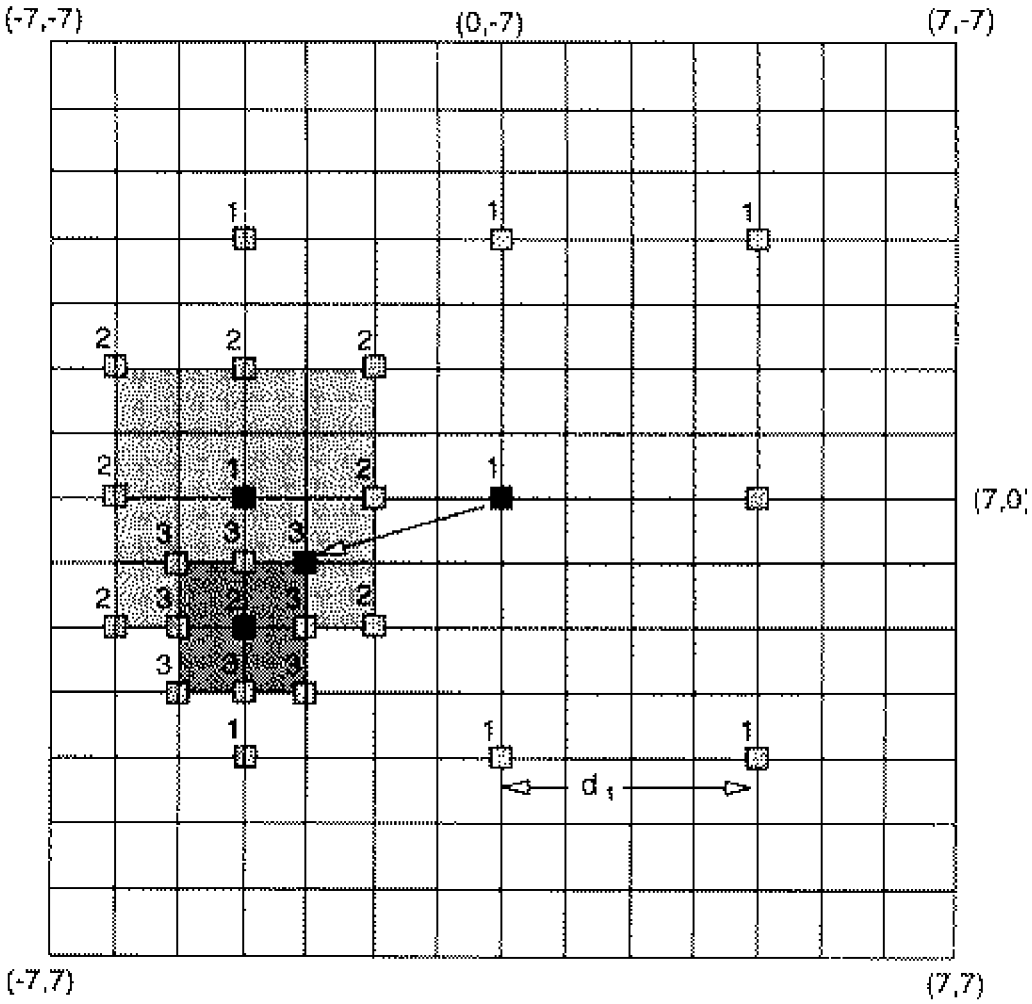
Sequentially search the whole [-p, p] region --> very slow

5.2 Two-Dimensional Logarithmic Search

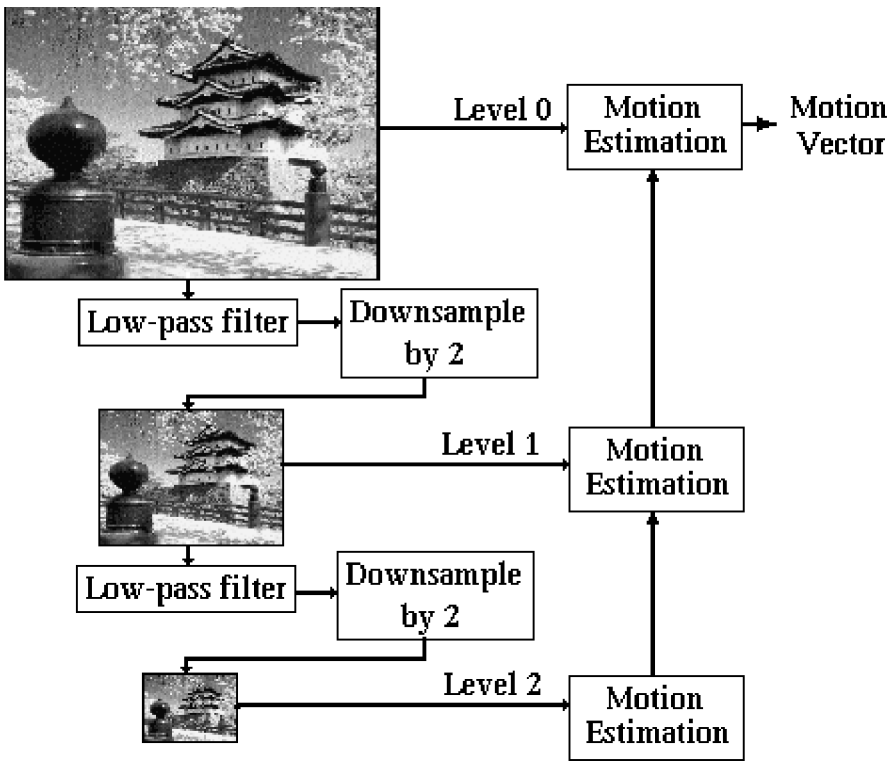
Similar to binary search. MAE function is initially computed within a window of [-p/2, p/2] at nine locations as shown in the figure.

Repeat until the size of the search region is one pixel wide:

1. Find one of the nine locations that yields the minimum MAE
2. Form a new searching region with half of the previous size and centered at the location found in step 1.



5.3 Hierarchical Motion Estimation



1. Form several low resolution version of the target and reference pictures
2. Find the best match motion vector in the lowerest resolution version.
3. Modify the motion vector level by level when going up

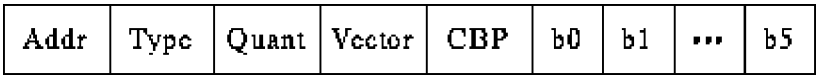
6. Some Important Issues

- Avoiding propagation of errors
  1. Send an I-frame every once in a while
  2. Make sure you use decoded frame for comparison
- Bit-rate control
  - Simple feedback loop based on "buffer fullness"

    If buffer is too full, increase the quantization scale factor to reduce the data.

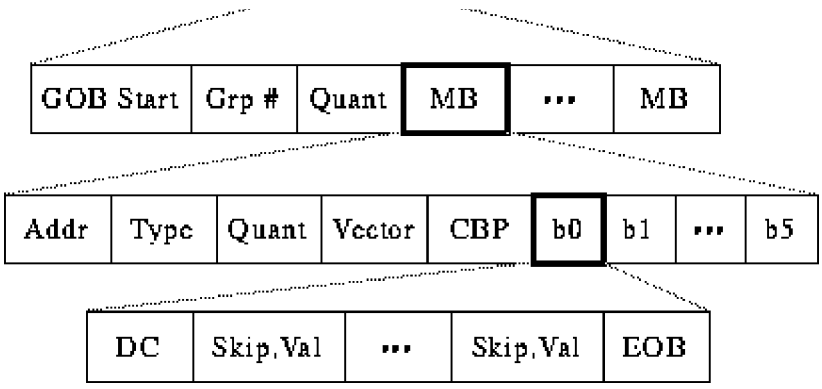
7. Details

7.1 How the Macroblock is Coded ?



- Many macroblocks will be exact matches (or close enough). So send address of each block in image --> *Addr*
- Sometimes no good match can be found, so send INTRA block --> *Type*
- Will want to vary the quantization to fine tune compression, so send quantization value --> *Quant*
- Motion vector --> *vector*
- Some blocks in macroblock will match well, others match poorly. So send bitmask indicating which blocks are present (Coded Block Pattern, or *CBP*).
- Send the blocks (4 Y, 1 Cr, 1 Cb) as in JPEG.

7.2. H. 261 Bitstream Structure



- Need to delineate boundaries between pictures, so send Picture Start Code --> *PSC*
- Need timestamp for picture (used later for audio synchronization), so send Temporal Reference --> *TR*
- Is this a P-frame or an I-frame? Send Picture Type --> *PType*
- Picture is divided into regions of 11 x 3 macroblocks called Groups of Blocks --> *GOB*
- Might want to skip whole groups, so send Group Number (*Grp #*)
- Might want to use one quantization value for whole group, so send Group Quantization Value --> *GQuant*
- Overall, bitstream is designed so we can skip data whenever possible while still unambiguous.

4.3.2. H. 263

- H. 263 is a new improved standard for low bit-rate video, adopted in March 1996.
  As H. 261, it uses the transform coding for intra-frames and predictive coding for inter-frames.
- Advanced Options:
  - Half-pixel precision in motion compensation
  - Unrestricted motion vectors
  - Syntax-based arithmetic coding
  - Advanced prediction and PB-frames
- In addition to CIF and QCIF, H. 263 could also support SQCIF, 4CIF, and 16CIF.
  The following is a summary of video formats supported by H. 261 and H. 263:

Video Formats Supported

Video format	Luminance Image Resolution	Chrominance Image Resolution	H.261 support	H.263 support	Bit-rate (Mbit/s) (if uncompressed, 30 fps)		Max bits allowed per picture (BPPmaxKb)
					B / W	Color	
SQCIF	128 x 96	64 x 48	Optional	Required	3.0	4.4	64
QCIF	176 x 144	88 x 72	Required	Required	6.1	9.1	64
CIF	352 x 288	176 x 144	Optional	Optional	24.3	36.5	256
4CIF	704 x 576	352 x 288	n/a	Optional	97.3	146.0	512
16CIF	1408 x 1152	704 x 576	n/a	Optional	389.3	583.9	1024

4.3.3. MPEG

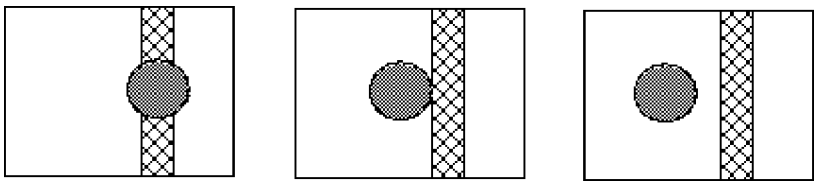
1. What is MPEG ?

- "Moving Picture Coding Experts Group", established in 1988 to create standard for delivery of video and audio.
- MPEG-1 Target: VHS quality on a CD-ROM (352 x 288 + CD audio @ 1.5 Mbits/sec)
- Standard had three parts: Video, Audio, and System (control interleaving of streams)

2. MPEG Video

- Problem: many macroblocks need information not in the reference frame.

  Example:

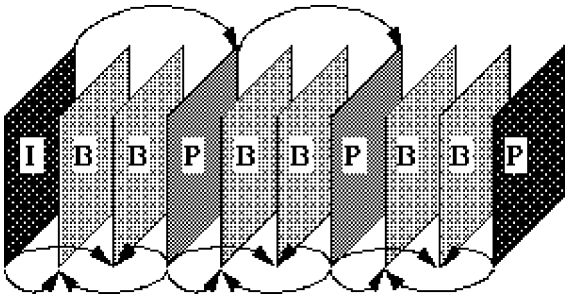


- MPEG solution: add third frame type: bidirectional frame, or *B-frame*

  B-frames search for macroblock in *past* and *future* frames.

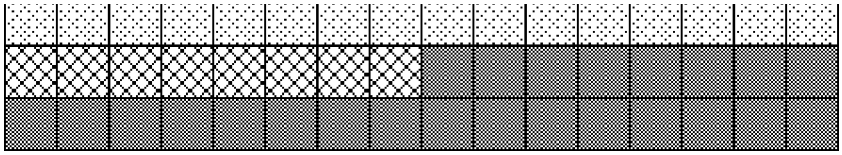
- Typical pattern is IBBPBBPBB IBBPBBPBB IBBPBBPBB

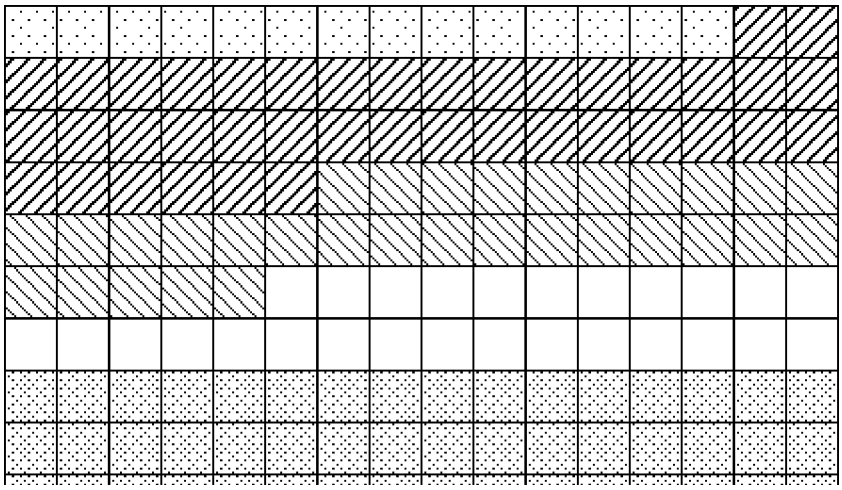
  Actual pattern is up to encoder, and need not be regular.



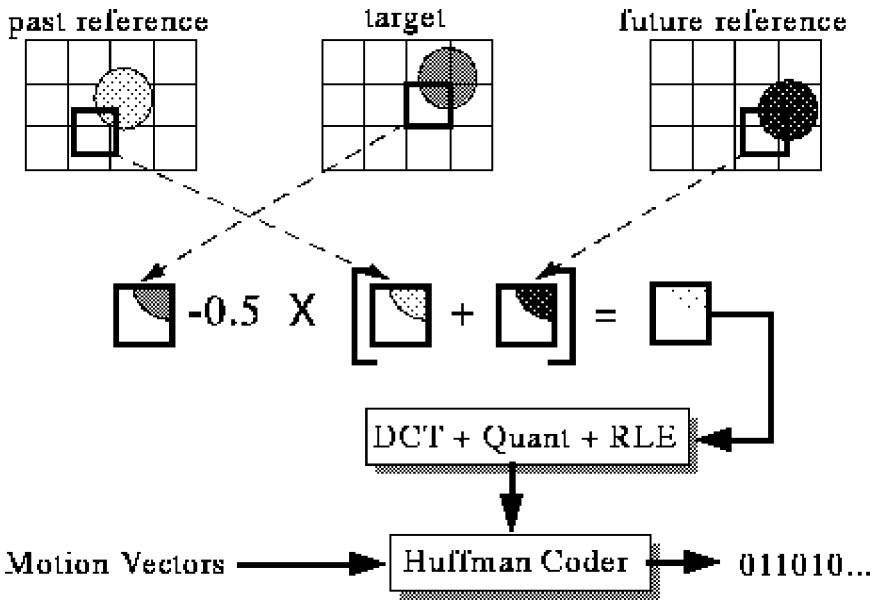
3. Differences from H. 261

- Larger gaps between I and P frames, so need to expand motion vector search range.
- To get better encoding, allow motion vectors to be specified to fraction of a pixel (1/2 pixel).
- Bitstream syntax must allow random access, forward/backward play, etc.
- Added notion of *slice* for synchronization after loss/corrupt data. Example: picture with 7 slices:





- B frame macroblocks can specify *two* motion vectors (one to past and one to future), indicating result is to be averaged.



- Compression performance of MPEG 1

Type	Size	Compression
I	18 KB	7:1
P	6 KB	20:1
B	2.5 KB	50:1
Avg	4.8 KB	27:1

4. MPEG Video Bitstream

- Click here for details

- Click here for details
- Public domain tool **mpeg_stat** and **mpeg_bits** will analyze a bitstream.

5. Decoding MPEG Video in Software

- Software Decoder goals: portable, multiple display types
- Breakdown of time

Function	% Time
Parsing Bitstream	17.4%
IDCT	14.2%
Reconstruction	31.5%
Dithering	24.5%
Misc. Arith.	9.9%
Other	2.7%

4.3.4. Newer MPEG Standards

---

1. MPEG-2

Unlike MPEG-1 which is basically a standard for storing and playing video on a single computer at low bit-rates, MPEG-2 is a standard for digital TV. It meets the requirements for HDTV and DVD (Digital Video/Versatile Disc).

MPEG-2 Level Table:

Level	size	Pixels/sec	bit-rate (Mbits)	Application
Low	352 x 288 x 30	3 M	4	consumer tape equiv.
Main	720 x 576 x 30	12 M	15	studio TV
High 1440	1440 x 1152 x 60	96 M	60	consumer HDTV
High	1920 x 1152 x 60	128 M	80	film production

- Other Differences from MPEG-1:
  1. Support both field prediction and frame prediction.
  2. Besides 4:2:0, also allow 4:2:2 and 4:4:4 chroma subsampling
  3. Scalable Coding Extensions: (so the same set of signals works for both HDTV and standard TV)
     - SNR (quality) Scalability -- similar to JPEG DCT-based Progressive mode, adjusting the quantization steps of the DCT coefficients.
     - Spatial Scalability -- similar to hierarchical JPEG, multiple spatial resolutions.
     - Temporal Scalability -- different frame rates.
  4. Frame sizes could be as large as 16383 x 16383

4. Frame sizes could be as large as 16383 x 16383
5. Non-linear macroblock quantization factor
6. Many minor fixes (see MPEG FAQ for more details)

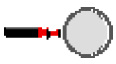
- MPEG-3: Originally planned for HDTV, got folded into MPEG-2

2. MPEG-4

- Drafted Nov. 1998, International Standard due Nov. 1999.
- Originally targeted at very low bit-rate communication (4.8 to 64 Kb/sec), it now aims at the following ranges of bit-rates:
  - video -- 5 Kb to 5 Mb per second
  - audio -- 2 Kb to 64 Kb per second
- It emphasizes the concept of *Visual Objects* --> Video Object Plane (VOP)
  - objects can be of arbitrary shape, VOPs can be non-overlapped or overlapped
  - supports content-based scalability
  - supports object-based interactivity
  - individual audio channels can be associated with objects
- Good for video composition, segmentation, and compression; networked VRML, audiovisual communication systems (e.g., text-to-speech interface, facial animation), etc.
- Standards being developed for shape coding, motion coding, texture coding, etc.

3. MPEG-7

- International Standard due by Nov. 2000.
- MPEG-7 is a standard for Multimedia Content Description Interface
  --> video and audio content-based retrieval
- For visual contents, the lower level descriptions will be color, texture, shape, size, etc., the higher level could include a semantic descripton such as "this is a scene with cars on the highway".

Further Exploration 

A good summary on MPEG, a good MPEG FAQ, MPEG Resources on the Web.

Last Updated: ***3/25/99***

Top | Chap 4 | CMPT 365 Home Page | CS

4.4. Audio Compression

Reference: Chapter 6 of Steinmetz and Nahrstedt

Reference: Davis Pan, "A Tutorial on MPEG/Audio Compression", IEEE Multimedia, Vol. 2, No. 2, pp. 60-74, 1995.

4.4.1 Simple Audio Compression Methods

- Traditional lossless compression methods (Huffman, LZW, etc.) usually don't work well on audio compression (the same reason as in image compression).

The following are some of the Lossy methods:

- Silence Compression - detect the "silence", similar to run-length coding
- Adaptive Differential Pulse Code Modulation (ADPCM)

  e.g., in CCITT G.721 -- 16 or 32 Kbits/sec.

  (a) Encodes the difference between two or more consecutive signals; the difference is then quantized --> hence the loss
  (b) Adapts at quantization so fewer bits are used when the value is smaller.

  - It is necessary to predict where the waveform is headed --> difficult
  - Apple has proprietary scheme called ACE/MACE. Lossy scheme that tries to predict where wave will go in next sample. About 2:1 compression.
- Linear Predictive Coding (LPC) fits signal to speech model and then transmits parameters of model --> sounds like a computer talking, 2.4 kbits/sec.
- Code Excited Linear Predictor (CELP) does LPC, but also transmits error term --> audio conferencing quality at 4.8 kbits/sec.

4.4.2 Psychoacoustics

Human hearing and voice

- Frequency range is about 20 Hz to 20 kHz, most sensitive at 2 to 4 KHz.

- Dynamic range (quietest to loudest) is about 96 dB
- Normal voice range is about 500 Hz to 2 kHz
    - Low frequencies are vowels and bass
    - High frequencies are consonants

Critical Bands

- Human auditory system has a limited, frequency-dependent resolution. The perceptually uniform measure of frequency can be expressed in terms of the width of the *Critical Bands.*
  It is less than 100 Hz at the lowest audible frequencies, and more than 4 kHz at the high end. Altogether, the audio frequency range can be partitioned into 25 critical bands.
- A new unit for frequency *bark* (after Barkhausen) is introduced:

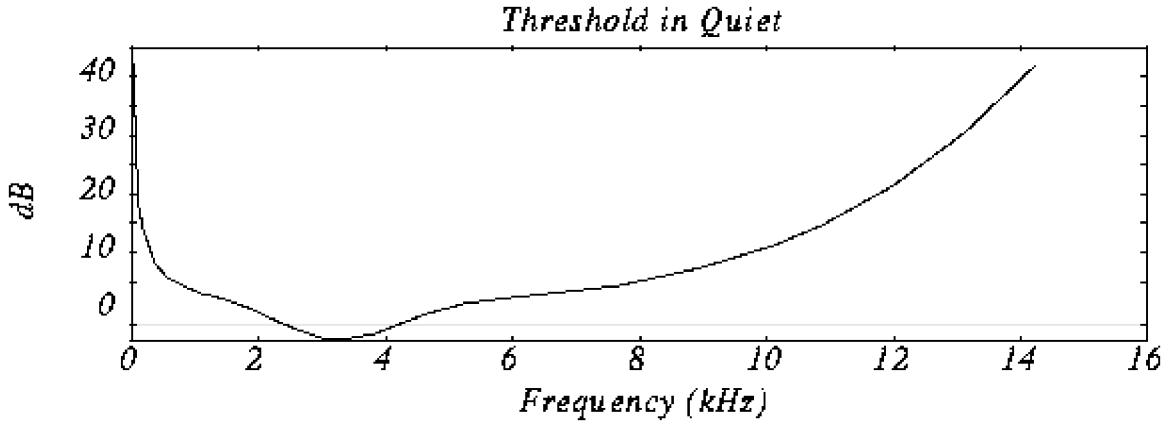
  1 Bark = width of one critical band

  For frequency < 500 Hz, it converts to $freq / 100$ Bark,

  For frequency > 500 Hz, it is $9 + 4 \cdot \log_2(freq/1000)$ Bark.

Sensitivity of human hearing in relation to frequency

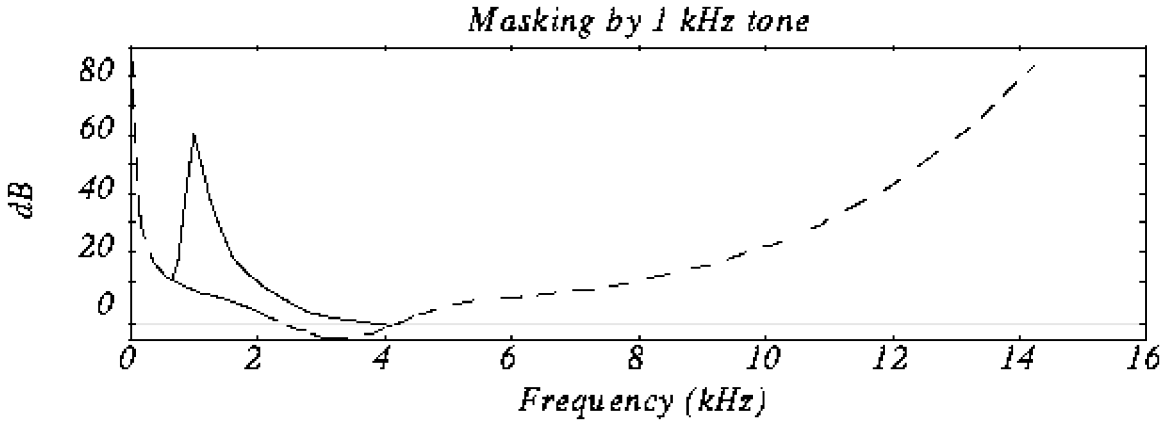
- Experiment: Put a person in a quiet room. Raise level of 1 kHz tone until just barely audible. Vary the frequency and plot



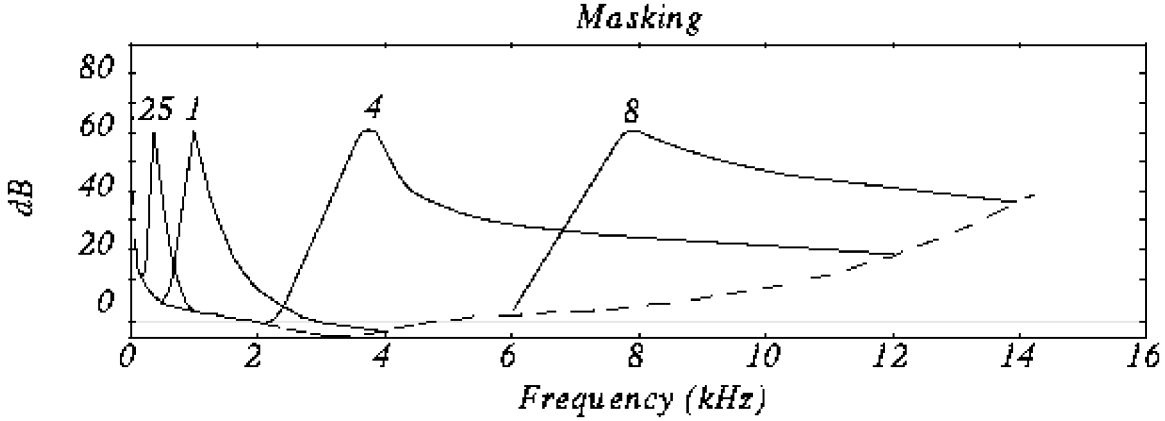
Frequency Masking

Question: Do receptors interfere with each other?

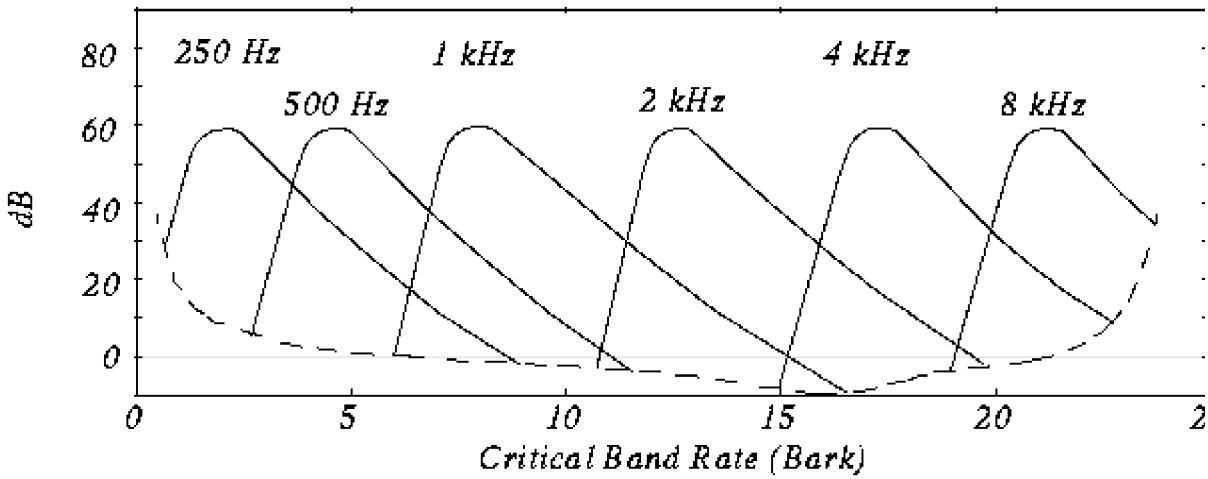
- Experiment: Play 1 kHz tone (*masking tone*) at fixed level (60 dB). Play *test tone* at a different level (e.g., 1.1 kHz), and raise level until just distinguishable.
- Vary the frequency of the test tone and plot the threshold when it becomes audible:



- Repeat for various frequencies of masking tones



- Frequency Masking on critical band scale:



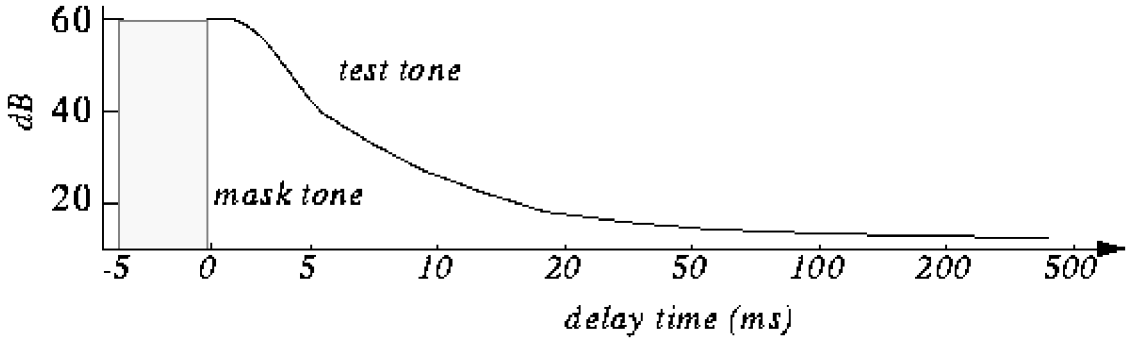
Temporal masking

- If we hear a loud sound, then it stops, it takes a little while until we can hear a soft tone nearby.
- Experiment: Play 1 kHz *masking tone* at 60 dB, plus a *test tone* at 1.1 kHz at 40 dB. Test tone can't be heard (it's masked).

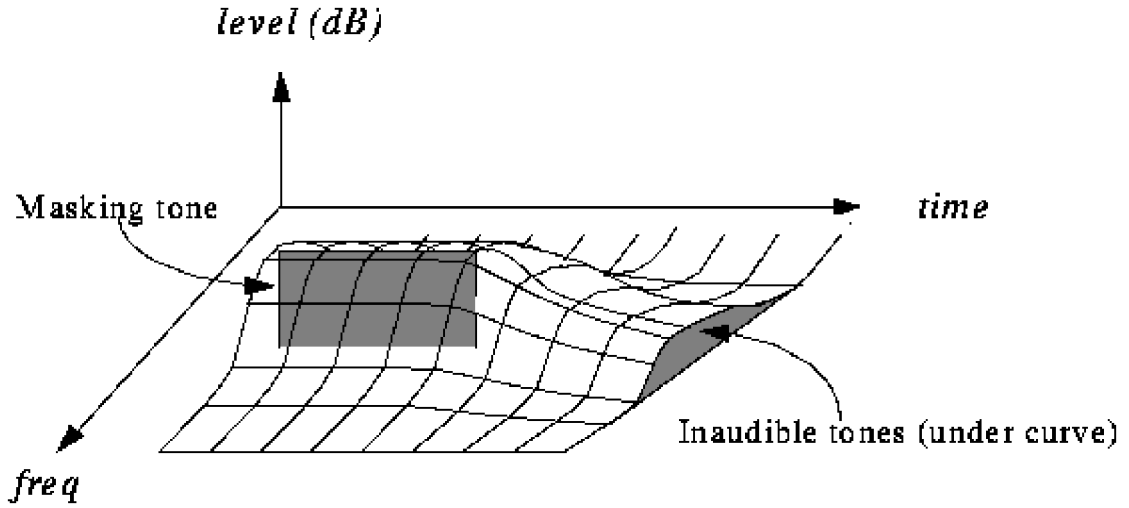
  Stop masking tone, then stop test tone after a short delay.

Adjust delay time to the shortest time when test tone can be heard (e.g., 5 ms).

Repeat with different level of the test tone and plot:



- Total effect of both frequency and temporal maskings:



4.4.3 MPEG Audio Compression

Some facts

- MPEG-1: 1.5 Mbits/sec for audio and video

  About 1.2 Mbits/sec for video, 0.3 Mbits/sec for audio

  (Uncompressed CD audio is 44,100 samples/sec * 16 bits/sample * 2 channels > 1.4 Mbits/sec)

- Compression factor ranging from 2.7 to 24.

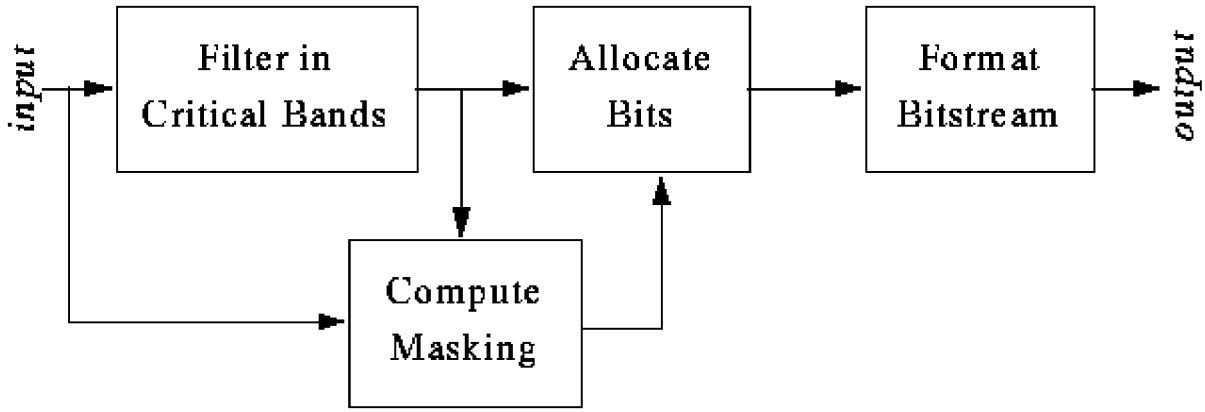
- With Compression rate 6:1 (16 bits stereo sampled at 48 KHz is reduced to 256 kbits/sec) and optimal listening conditions, expert listeners could not distinguish between coded and original audio clips.

- MPEG audio supports sampling frequencies of 32, 44.1 and 48 KHz.

- Supports one or two audio channels in one of the four modes:
    1. Monophonic -- single audio channel
    2. Dual-monophonic -- two independent channels, e.g., English and French
    3. Stereo -- for stereo channels that share bits, but not using Joint-stereo coding
    4. Joint-stereo -- takes advantage of the correlations between stereo channels

Steps in algorithm:

1. Use convolution filters to divide the audio signal (e.g., 48 kHz sound) into 32 frequency subbands --> *subband filtering*.
2. Determine amount of masking for each band caused by nearby band using the *psychoacoustic model* shown above.
3. If the power in a band is below the masking threshold, don't encode it.
4. Otherwise, determine number of bits needed to represent the coefficient such that noise introduced by quantization is below the masking effect (Recall that one fewer bit of quantization introduces about 6 dB of noise).
5. Format bitstream



Example:

- After analysis, the first levels of 16 of the 32 bands are these:

Band	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Level (db)	0	8	12	10	6	2	10	60	35	20	15	2	3	5	3	1

- If the level of the 8th band is 60dB,

  it gives a masking of 12 dB in the 7th band, 15dB in the 9th.

  Level in 7th band is 10 dB (< 12 dB), so ignore it.

  Level in 9th band is 35 dB (> 15 dB), so send it.
  [Only the amount above the masking level needs to be sent, so instead of using 6 bits to encode it,

we can use 4 bits -- a saving of 2 bits (= 12 dB).]

MPEG Layers

- MPEG defines 3 layers for audio. Basic model is same, but codec complexity increases with each layer.
- Divides data into frames, each of them contains 384 samples, 12 samples from each of the 32 filtered subbands as shown below.

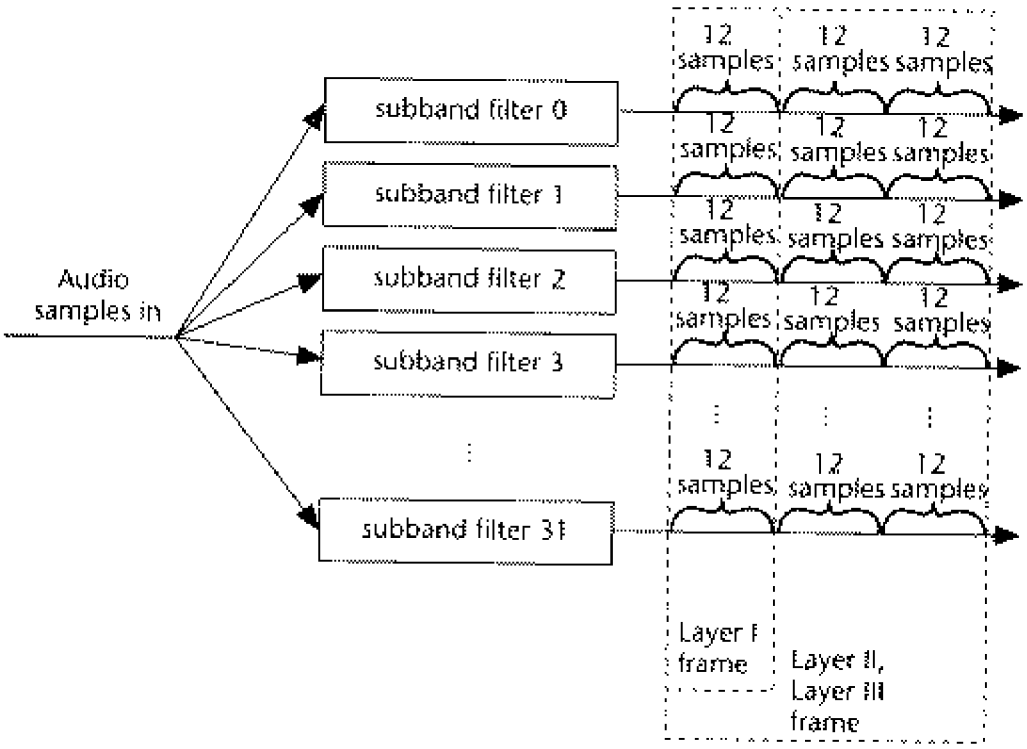


Figure: Grouping of Subband Samples for Layer 1, 2, and 3

- Layer 1: DCT type filter with one frame and equal frequency spread per band. Psychoacoustic model only uses frequency masking.
- Layer 2: Use three frames in filter (before, current, next, a total of 1152 samples). This models a little bit of the temporal masking.
- Layer 3: Better critical band filter is used (non-equal frequencies), psychoacoustic model includes temporal masking effects, takes into account stereo redundancy, and uses Huffman coder.

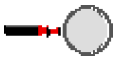
  Stereo Redundancy Coding:

  - Intensity stereo coding -- at upper-frequency subbands, encode summed signals instead of independent signals from left and right channels.
  - Middle/Side (MS) stereo coding -- encode middle (sum of left and right) and side (difference of left and right) channels.

Effectiveness of MPEG audio

Layer	Target Bit-rate	Ratio	Quality at 64 kb/s	Quality at 128 kb/s	Theoretical Min. Delay
Layer 1	192 kb/s	4:1	---	---	19 ms
Layer 2	128 kb/s	6:1	2.1 to 2.6	4+	35 ms
Layer 3	64 kb/s	12:1	3.6 to 3.8	4+	59 ms

- Quality factor: 5 - perfect, 4 - just noticeable, 3 - slightly annoying, 2 - annoying, 1 - very annoying
- Real delay is about 3 times of the theoretical delay

Further Exploration 

MPEG Resources on the Web.

Last Updated: ***7/8/98***

Top | Chap 4 | CMPT 365 Home Page | CS