Video Compression Basics

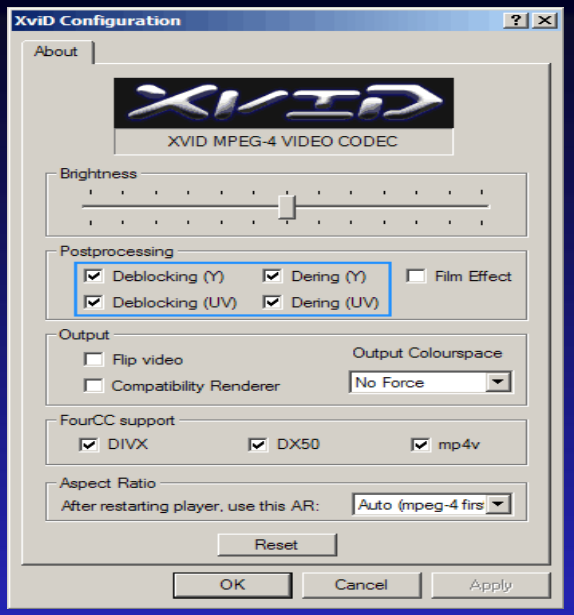
Sanjiv Malik

Agenda

- Introduction to Image/Video Compression
- Still Picture compression
  - RGB to YCrCb
  - DCT
  - Entropy Coding
- Video Compression
  - Motion Compensation
- Compression Standards : JPEG, MPEG-1, MPEG-II, H.264, MPEG4, WMV

VOB/MPEG-2

MP4, MOV, H.264



MPEG2/MPEG4

Movies

MPEG2/MPEG4/H.264

PSP : Supports MP4/H.264

Why Compress/Encode ?

- Digital color image: 352x288 pixel
- RGB representation: 24 bpp (8bits for red,green,blue)
- Total amount of bytes: > 300K
- JPEG: common image compression standard, < 20K, similar quality

Why Compress/Encode ?

Video Source	Output Data Rate**[Kbits/sec]**
Quarter VGA @20 frames/sec	36 864.00
CIF camera @30 frames/sec	72 990.72
VGA @30 frames/sec	221 184.00

Transmission Medium	**Data Rate [Kbits/sec]**
Wireline modem	56
GPRS (estimated average rate)	30
3G/WCDMA (theoretical maximum)	384

Image Compression

Steps in Image Compression

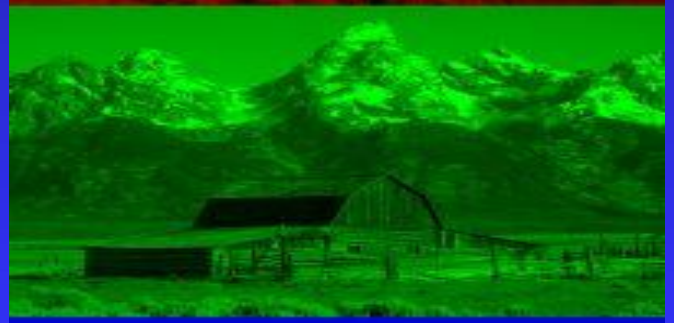
1. If the color is represented in RGB mode, translate it to YCrCb.
2. Divide the file into 8 X 8 blocks.
3. Transform the pixel information from the spatial domain to the frequency domain with the Discrete Cosine Transform.
4. Quantize the resulting values by dividing each coefficient by an integer value and rounding off to the nearest integer.
5. Look at the resulting coefficients in a zigzag order. Do a run-length encoding of the coefficients ordered in this manner. Follow by Huffman coding.

RGB and YCrCb Color space

Original

R

G



B

Original

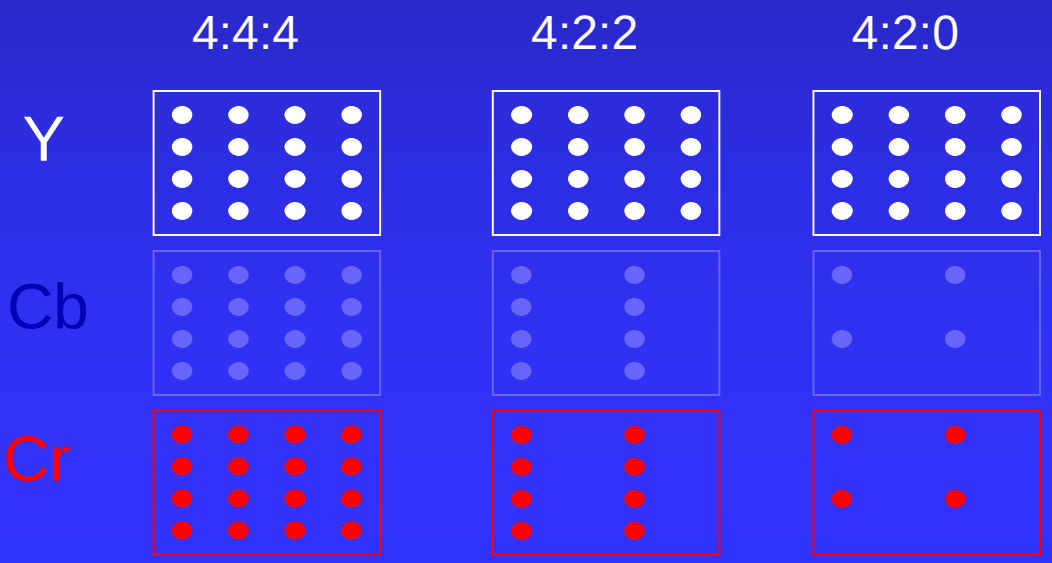
Y

Cr

Cb

Step 1: Converting RGB to YCrCb-I

- YCrCb color mode stores color in terms of its luminance (brightness) and chrominance (hue).
- The human eye is less sensitive to chrominance than luminance. Compression can be achieved by storing more luma details then chroma details. This is called chroma-sub sampling.
- Chroma sub sampling formats : 4:4:4, 4:2:1, 4:2:2, 4:1:1, 4:2:0



Step 1: Converting RGB to YUV-II

- It is simple arithmetic to convert RGB to YUV. One example is:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$U = -0.1687 * R - 0.3313 * G + 0.5 * B + 128$$

$$V = 0.5 * R - 0.4187 * G - 0.813 * B + 128$$

- When conversion is done from RGB to YCrCb, it makes 4:4:4 format, but 2 color components are discarded. Thus bandwidth is reduced by 50%

Step 2: Divide into 8 X 8 blocks

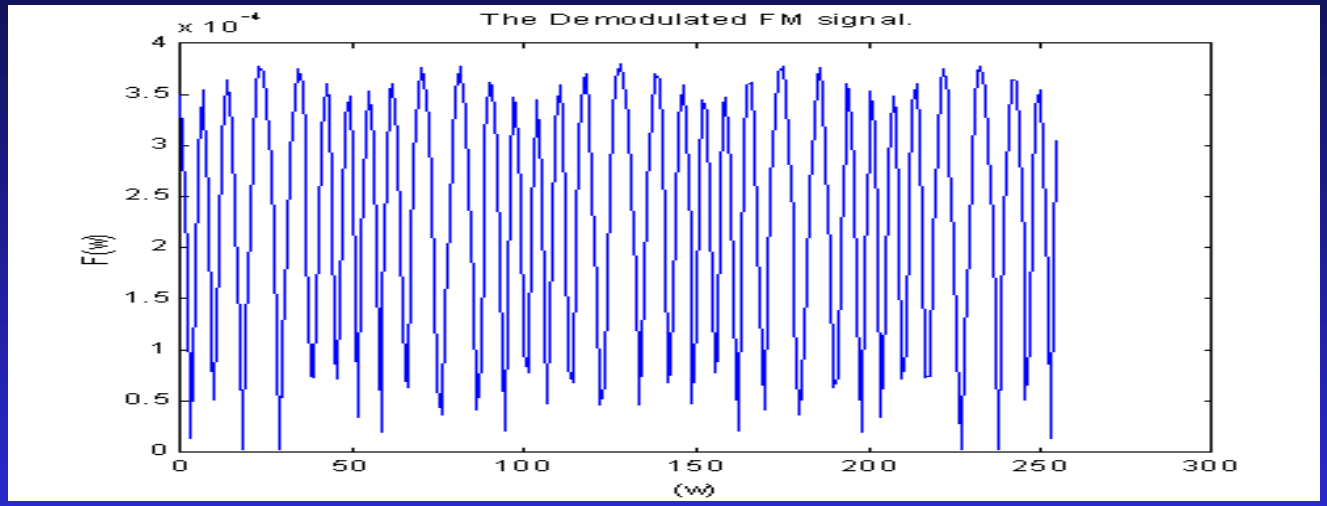
- With YCrCb color, we have 16 pixels of information in each block for the Y component (though only 8 in each direction for the Cr and Cb components).
- If the file doesn't divide evenly into 8 X 8 blocks, extra pixels are added to the end and discarded after the compression.

Step 3 : Transform to Frequency Domain : Discrete Cosine Transform

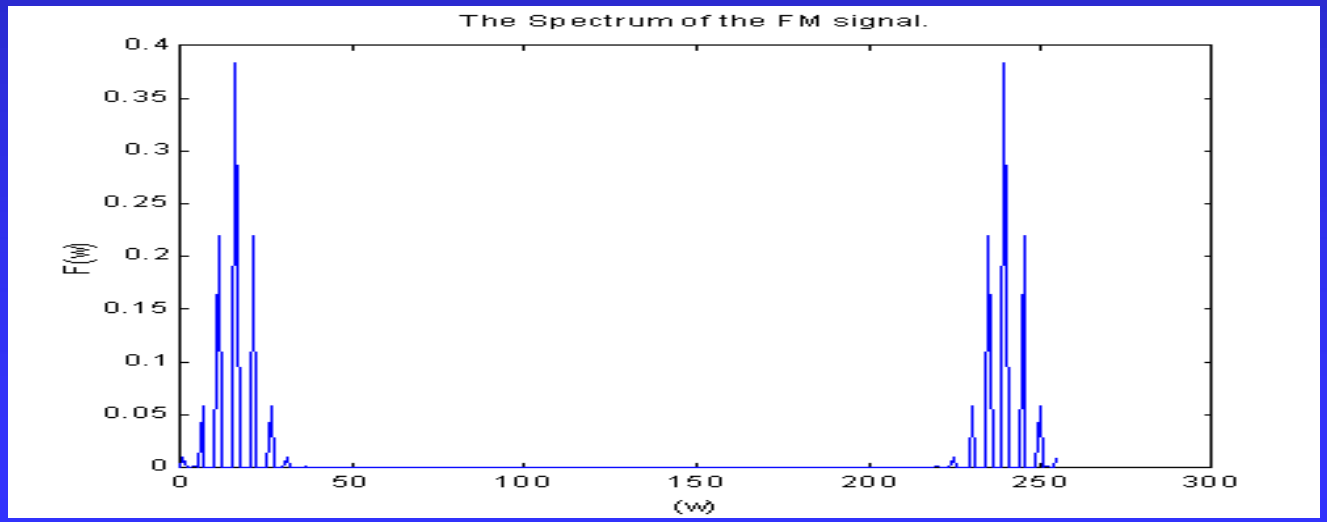
- The DCT transforms the data from the spatial domain to the frequency domain.
- The spatial domain shows the amplitude of the color as you move through space
- The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.
- For the 8 X 8 matrix of color data, we'll getting an 8 X 8 matrix of coefficients for the frequency components.

Step 3 : Time(Space) Domain and Frequency domain

Time Domain



Frequency Domain



Step 3: Discrete Cosine Transform

- **Discrete cosine transform** (DCT) is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers.
- The most common variant of discrete cosine transform is the type-II DCT, which is often called simply "the DCT"; its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT".
- The frequency domain is a better representation for the data because it makes it possible for you to separate out – and throw away – information that isn’t very important to human perception.
- The human eye is not very sensitive to high frequency changes – especially in photographic images, so the high frequency data can, to some extent, be discarded.

Step 3: Preparation for DCT

- An Example 8x8 block of pixel data is shown here >
- The next step is to transform the 8x8 matrix from a positive range to one centered around zero.
- Since 8 bits are used for each value, each value is subtracted by 128.
- The resultant block is shown here ->

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

$x \longrightarrow$

$$\begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

$y \downarrow$

Step 3: DCT Calculation

For an N X N pixel image $[p_{uv}, 0 \le u < N, 0 \le v < N]$ the DCT is an array of coefficients $[DCT_{uv}, 0 \le u < N, 0 \le v < N]$ where

$$DCT_{uv} = \frac{1}{\sqrt{2N}} C_u C_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p_{xy} \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

where

$$C_u C_v = \frac{1}{\sqrt{2}} \quad for \quad u, v = 0$$

$$C_u C_v = 1 \quad otherwise$$

Step 3: Result of DCT

- Original 8x8 Pixel block

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

- Corresponding DCT coefficient block

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

$u \longrightarrow$ (horizontal)

$v \downarrow$ (vertical)

Step 3: Result of DCT

- The DCT is lossless in that the reverse DCT will give you back exactly your initial information (ignoring the rounding error that results from using floating point numbers.)
- The values from the DCT are initially floating-point.
- They are changed to integers by quantization.
- Because of the heavy calculations involved in DCT process, embedded systems implement it in hardware.

Step 4: Quantization

- Quantization involves dividing each coefficient by an integer between 1 and 255 and rounding off.
- The quantization table is chosen to reduce the precision of each coefficient to no more than necessary.
- The quantization table is carried along with the compressed file.

$$B_{j,k} = \text{round}\left(\frac{A_{j,k}}{Q_{j,k}}\right) \quad \text{for } j = 0, 1, 2, \cdots, N_1 - 1; k = 0, 1, 2, \cdots, N_2 - 1$$

Step 4: Quantization

- Quantization Table

16	11	10	16	24	40	51	61
12	12	14	16	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

- Resultant 8x8 DCT coefficients after each coefficient is divided by corresponding Quantization factor

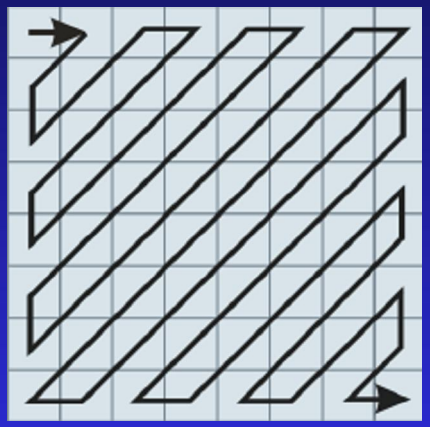
-26	-3	-6	2	2	-1	0	0
0	-2	-4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Step 5: Entropy Coding

- This is done so that the coefficients are in order of increasing frequency.
- The higher frequency coefficients are more likely to be 0 after quantization.
- This improves the compression of run-length encoding.
- Do run-length encoding and Huffman coding.

Step 5a: Arrange in “zigzag” order

- Zig Zag scan Pattern



- Resultant data

-26,
-3, 0,
-3, -2, -6,
2, -4, 1, -3,
1, 1, 5, 1, 2,
-1, 1, -1, 2, 0, 0,
0, 0, 0, 0, -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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Step 5b: Run Length Encoding

- The Output of the Zigzag scan is further encoded using Run Length Encoding procedure.
- In Run Length Encoding, consecutive pixels with the same value are encoded using a run-length and value pair. E.g.
  - 0x000x000x000x000x000x000x000x000x000x000 ->
    0x0A 0x00

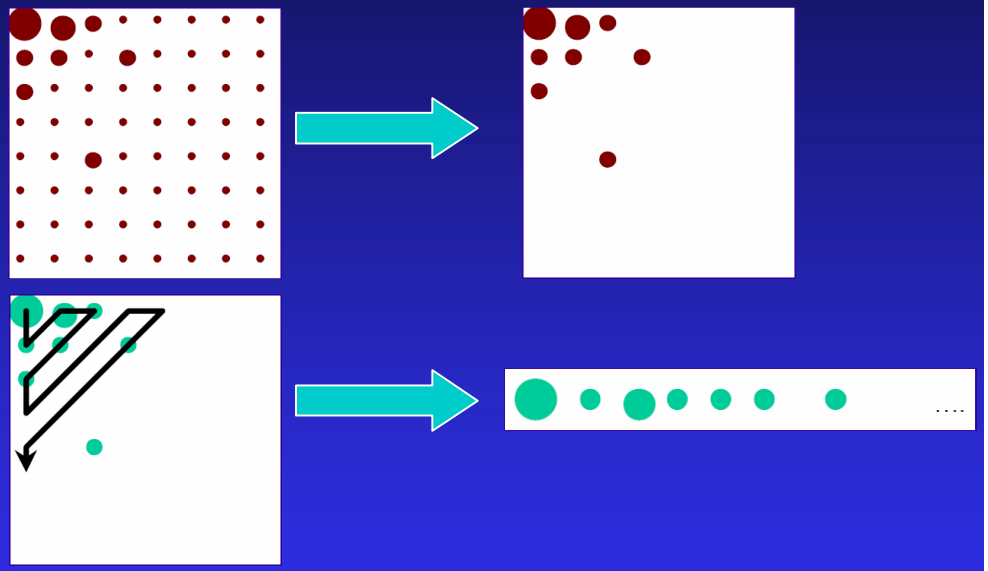
Step 5c: Huffman Encoding

- The Output of the RLE data is further encoded using Huffman encoding
- Huffman coding refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.
- An EOB (End Of Block) marker is put at the end and the data is written to a file

Char	Freq	Code
space	7	111
a	4	010
e	4	000
f	3	1101
h	2	1010
i	2	1000
m	2	0111
n	2	0010
s	2	1011
t	2	0110
l	1	11001
o	1	00110
p	1	10011
r	1	11000
u	1	00111
x	1	10010

Summary of Entropy Coding

- Weighted scalar quantization: loss of precision, few non-zero coefficients are left
- Zig-zag scan: non-zero coefficients tend to be grouped together
- Run-Level encoding: encode each coefficient value as a (run,level) pair
  - run: number of zeros preceding values
  - lenght: non-zero value



Video Compression

Video Compression Means

- Reduce *redundancy* and *irrelevancy*
- Sources of redundancy:
  - spatial: nearby pixels often correlated (as in still images)
  - temporal: adjacent frames highly correlated
- Irrelevancy:
  - Perceptually unimportant information

Commonly used terms - 1

- To encode a frame each operation is performed at *macroblock* (*MB*) level (nxn block of pixel. n=16 in MPEG2)
- Intra coded frame (I): every MB of the frame is coded using spatial redundancy
- Inter coded frame (P): most of the MBs of the frame are coded exploiting temporal redundancy (in the past)
- Bi-predictive frame (B): most of the MBs of the frame are coded exploiting temporal redundancy in the past and in the future.
- Group of Picture (GOP): sequence of pictures between two I-frames.

Commonly used terms - 2

- Main idea: predict current frame using previously coded one (*reference frame*)
- For each MB motion information is extracted from current and reference frame ⇒ Motion Estimation (ME)
- Temporal predicted frame is obtained from reference frame using motion information estimated ⇒ Motion Compensation (MC)
- The residual (original-MC) and motion information are coded using transform coding like for intra frame
- The prediction can be improved using frames in the future (bi-directional prediction)

Motion Estimation/Compensation

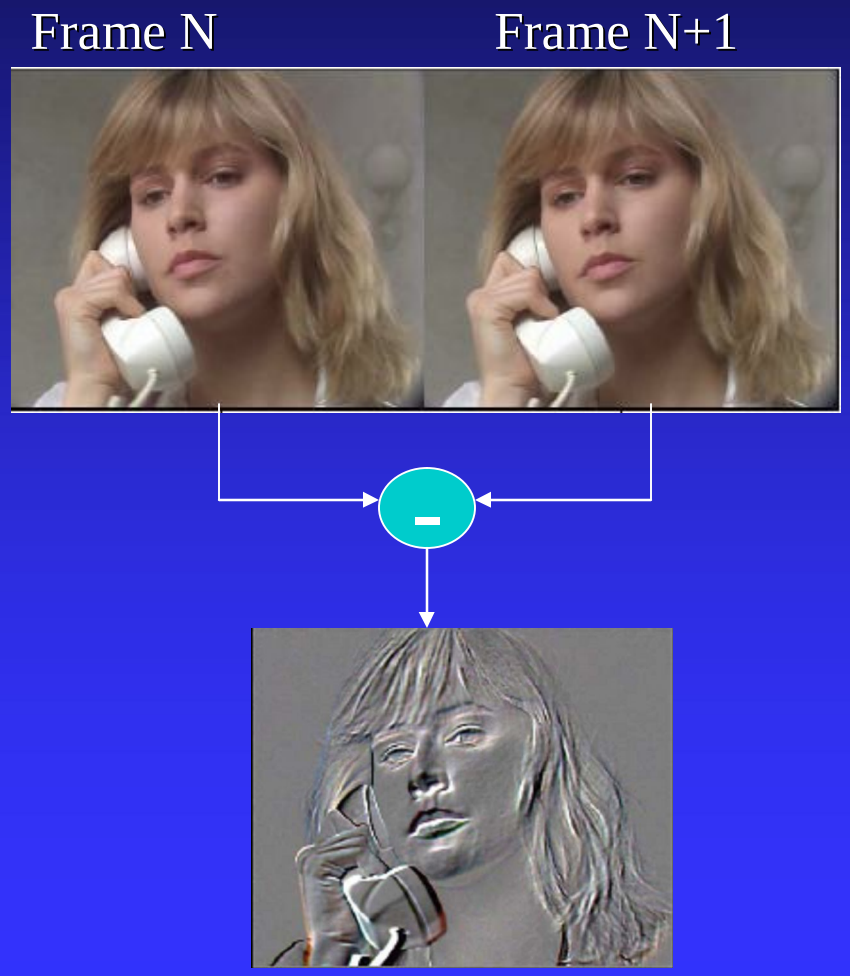
- The frames are partitioned in blocks of pixels (e.g. A block of 16x16 as in MPEG)
- Each block is predicted from a block of equal size in the reference frame.
- The blocks are not transformed in any way apart from being shifted to the position of the predicted block. This shift is represented by a *motion vector*.
- The motion vectors of two neibouring MBs are not independent. These are encoded differentially means only the *difference* in motion vectors is encoded
- One main disadvantage of block motion compensation is that it introduces discontinuities at the block borders (blocking artifacts).

Types of Motion Compensation

- Frame Based Motion Compensation
- Fixed Size Block Motion Compensation
- Variable Size Block Motion Compensation
- Object based Motion Compensation
  - Fixed Size
  - Variable Size

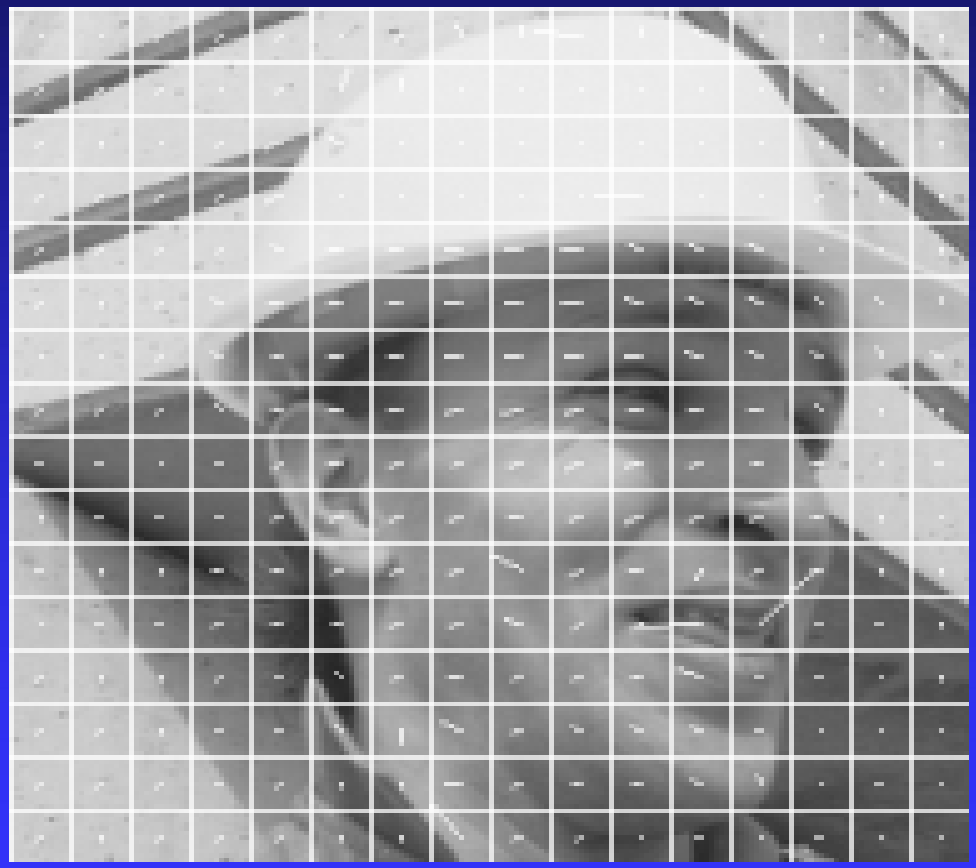
Frame Based Motion Estimation

- This approach is applicable in camera motions such as dolly (forward, backwards), track (left, right), boom (up, down), pan (left, right), tilt (up, down) and roll (along the view axis). It works best for still scenes without moving objects.



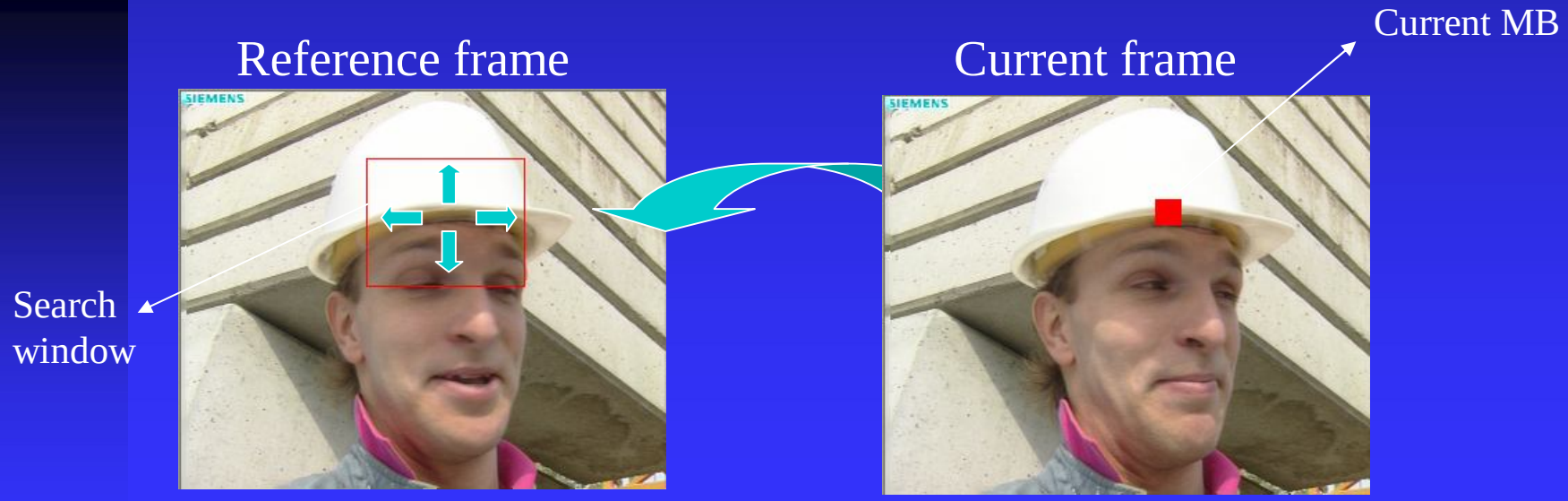
Fixed Size Block based Motion Estimation-1

- Most common approach in use today
- The current frame is divided in blocks of fixed size (e.g. 16x16 in MPEG)
- the maximum displacement might be +-64 pixels from a block's original position
- Various search strategies like full search, grid search etc. possible



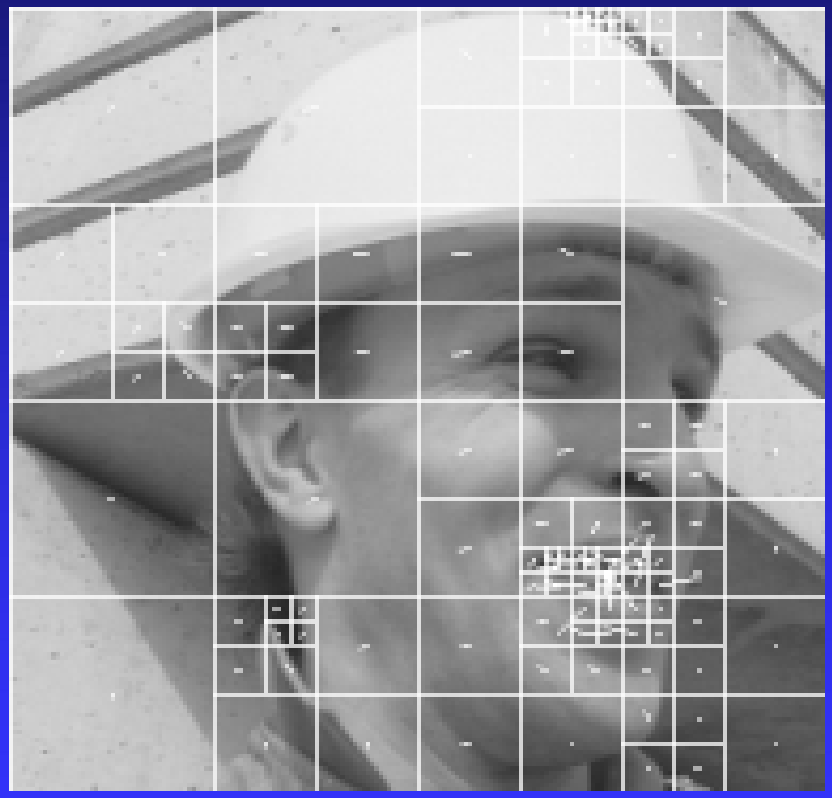
Fixed Size Block based Motion Estimation-2

- Find the MB in the reference frame which is the most similar to the current MB
- To find optimal motion vectors, one basically has to calculate the block prediction error for each motion vector within a certain search range and pick the one that has the best compromise between the amount of error and the number of bits needed for motion vector data.



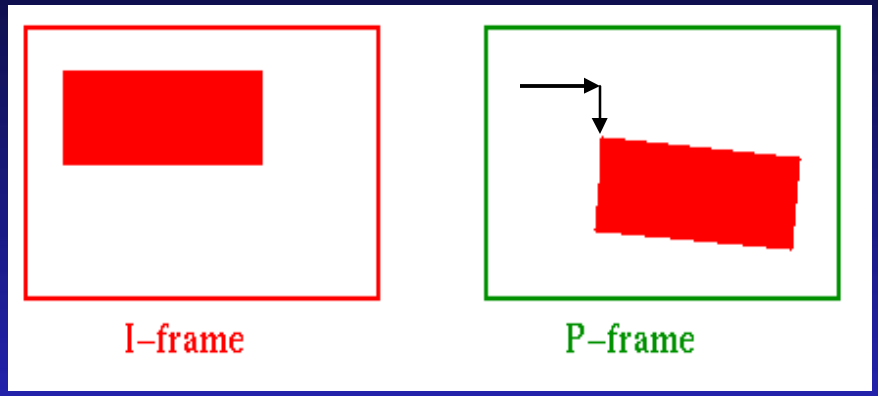
Variable Size Block Based Motion Compensation

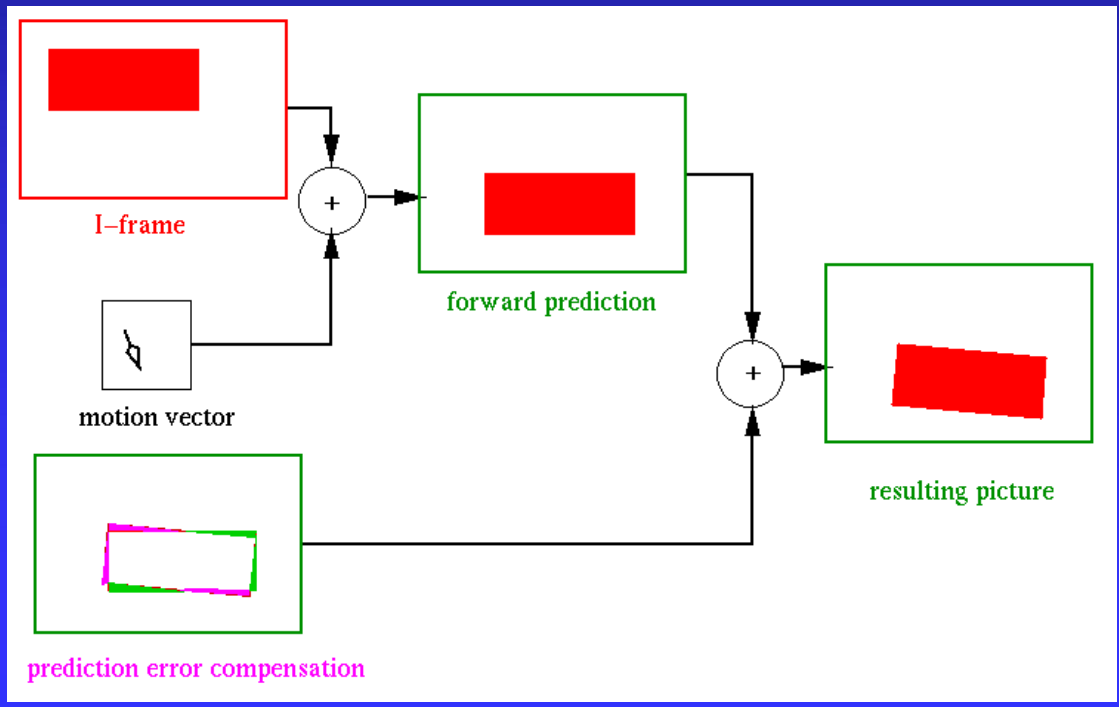
- In this approach, the estimation is first started with large size block which are then repeatedly divided into smaller blocks.
- More efficient, but very complex to implement



Motion Vector

- The Motion Vector is having two components Horizontal and Vertical
- There can be maximum of 2 motion vector in standards like MPEG, but can be 16 MVs in H.264

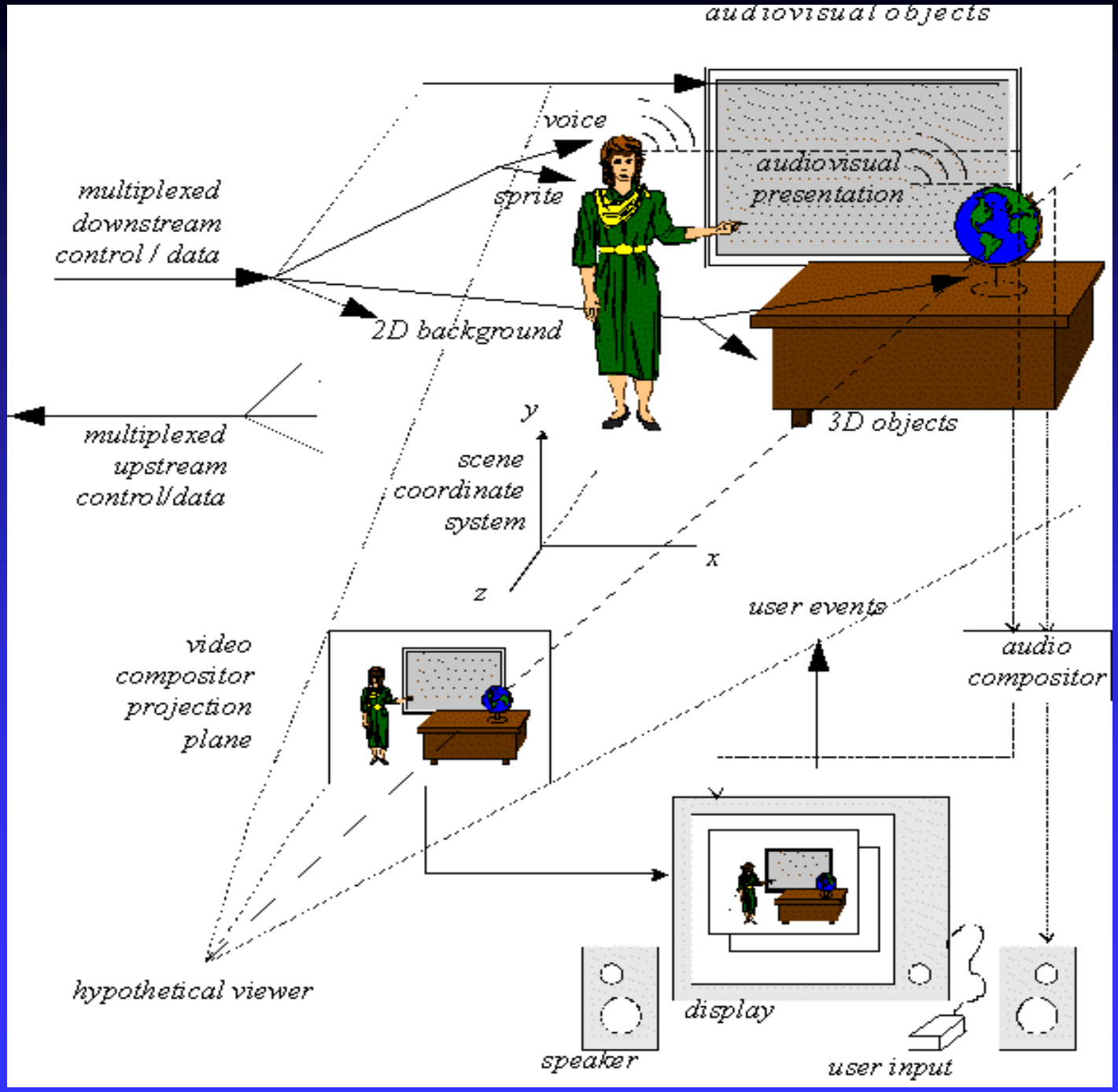




Object Based Motion Compensation

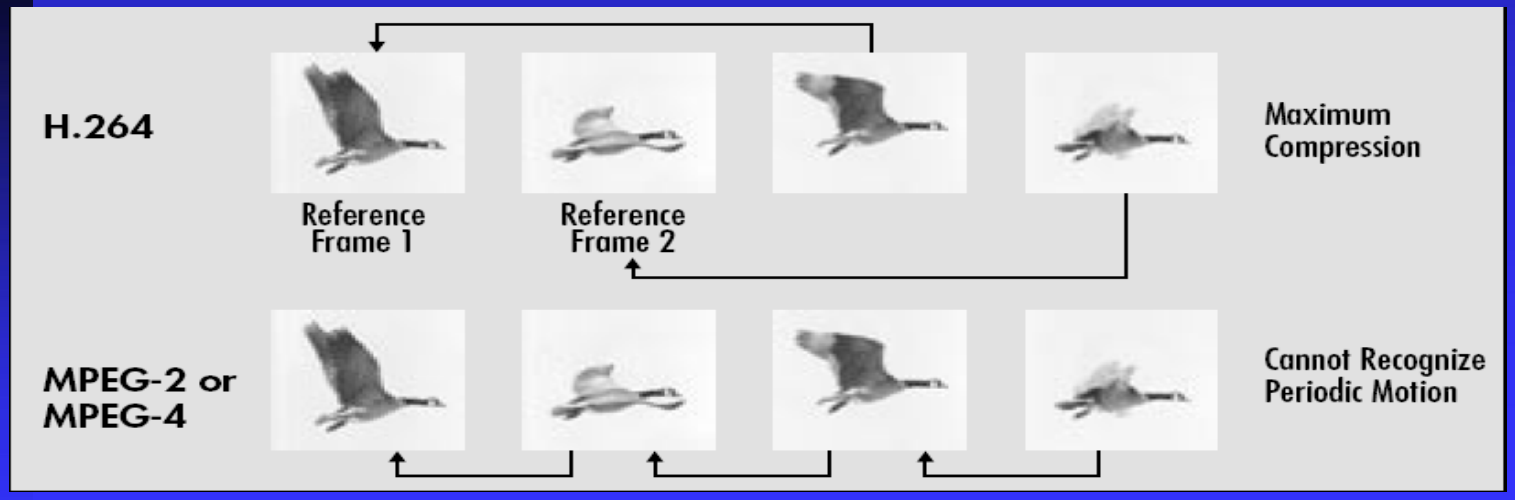
- The latest standards like MPEG4 are based on media objects. Media Object can be:
  - Still images (e.g. as a fixed background);
  - Video objects (e.g. a talking person - without the background;
  - Audio objects (e.g. the voice associated with that person, background music)

Example of Media Objects as explained in MPEG4 standard



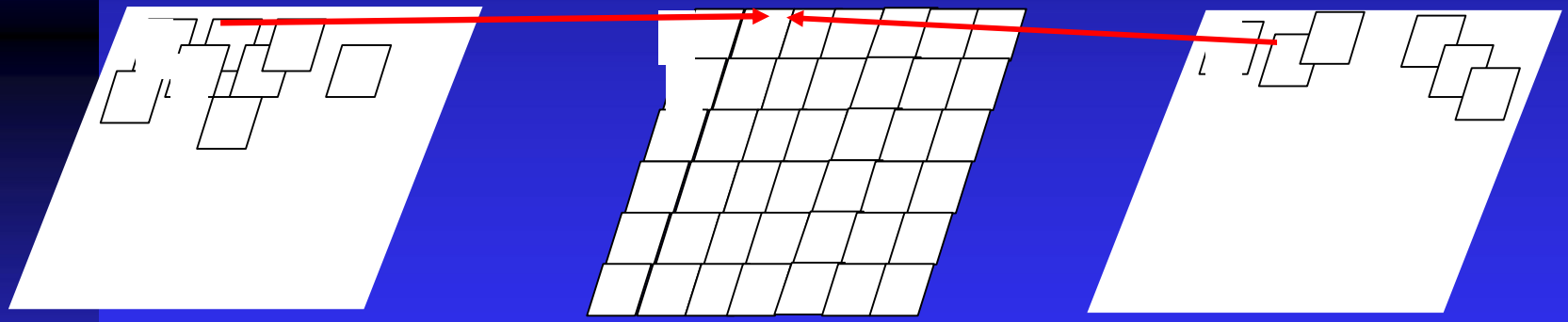
Multiple Reference Frames

- The encoder can select for motion compensation among a larger number of pictures previosly decoded and stored
- Useful when dealing with motion that is periodic or in presence of camera switching between 2 scenes



Bidirectional Prediction

- it is possible to temporally predict MBs from both previous and future coded frames.
- The size of a B-Picture may be reduced to 10% of the actual picture size

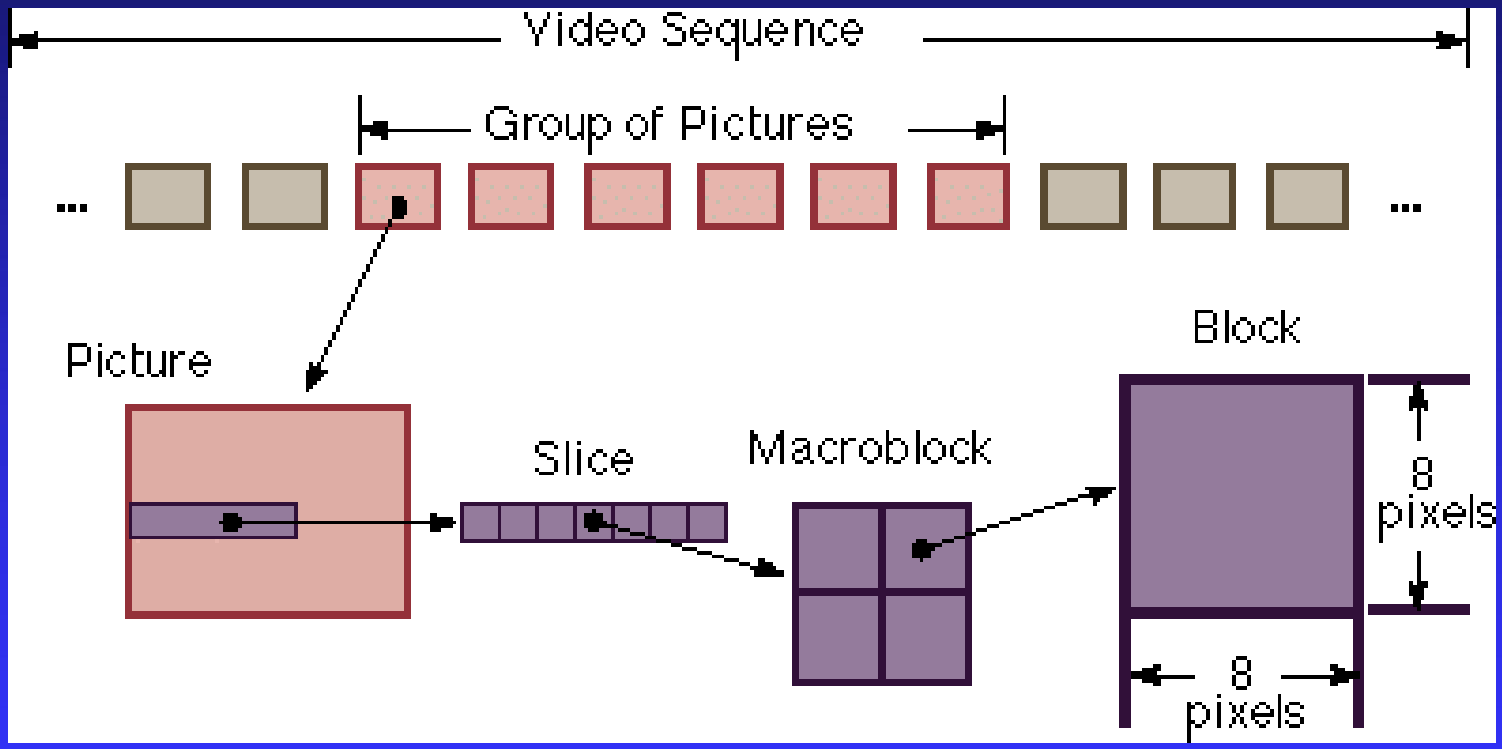


Previous frame

Current frame (B- frame)

Future frame

Video Stream Format Example : MPEG1/2



MPEG1/2/4 Complete Decoder

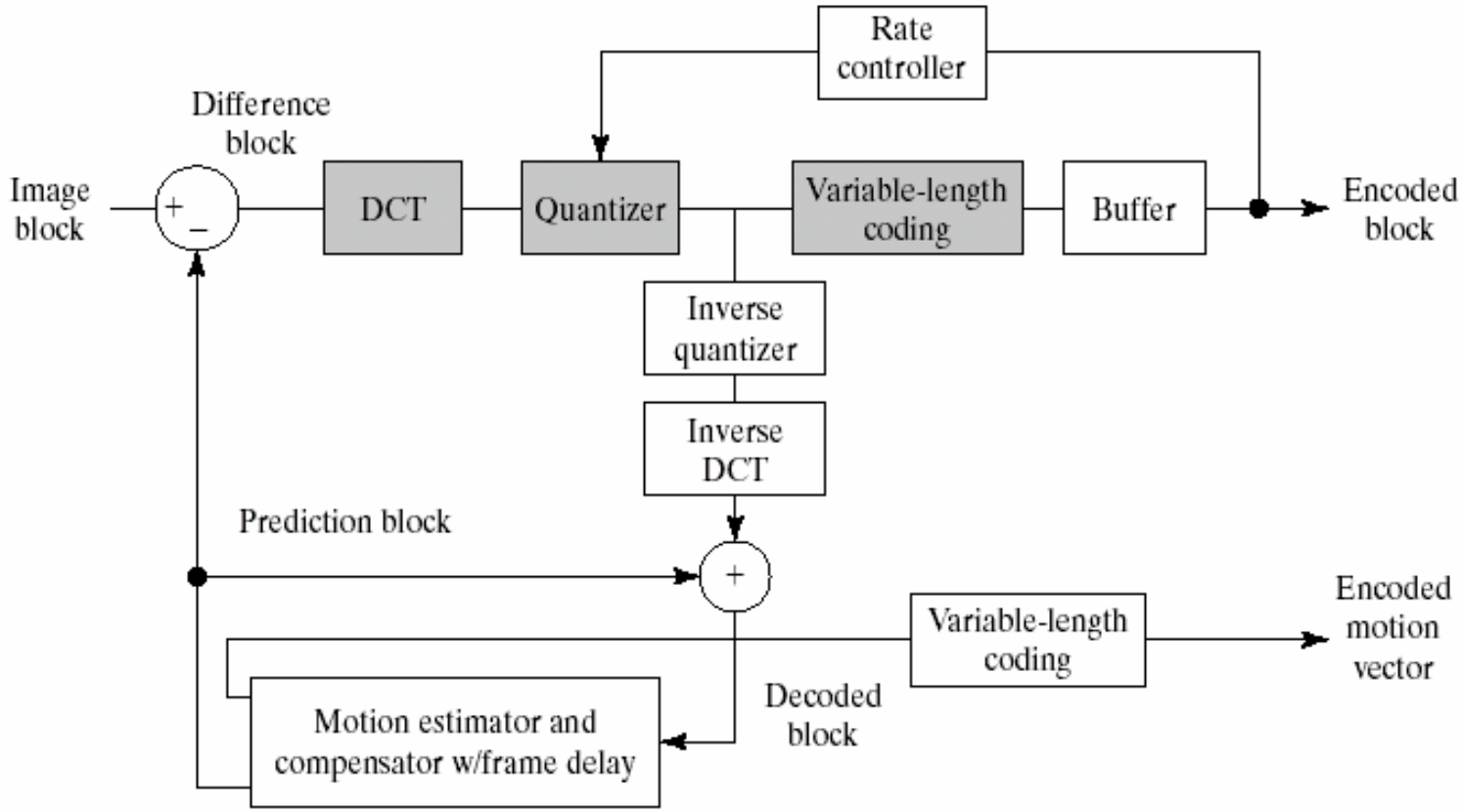


FIGURE 8.47 A basic DPCM/DCT encoder for motion compensated video compression.

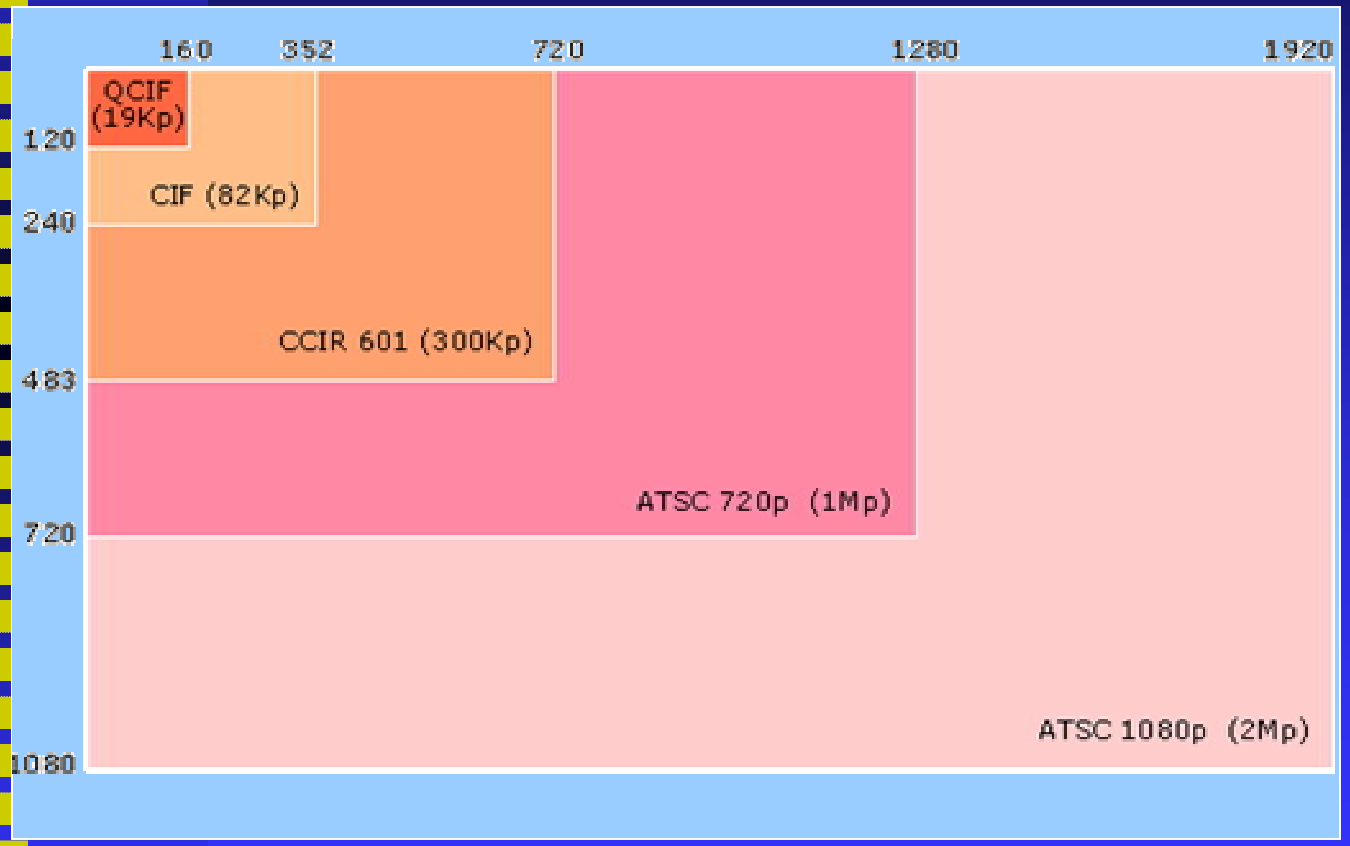
Deblocking filter

- Block artifact is one of the main disavantageof block-based coding
- Latest Video Codecs like MPEG4/H.264 deploy an adaptive deblocking filter to smoothen block edges
- The strength of the filter is controlled by the value of several syntax element

No deblock

Deblock

Video Frame Sizes



Video Standards

MPEG-4 (MP4) - 1

- MPEG4 is the latest standard from MPEG group. In 2002, Apple adopted it in QuickTime and then in iPod Video, making it highly popular video standard.
- MPEG4 absorbs most good features from MPEG2 and other related standards and adds these new features like:
  - VRML(Virtual Reality Modeling Language) based 3D rendering support
  - Video Objects based Motion Compensation
  - Support for DRM
  - Support for various type of interactivity.
- All most popular video codecs available today like DivX, Xvid, Nero Digital and 3ivx and by Quicktime 6 are based on MPEG4 part 2
- Other most recent ones are Nero Digital AVC, Quicktime 7 and next-gen DVD formats like HD DVD and Blu-ray Disc use MPEG4 part 10.
- All high-quality multimedia on wireless devices like mobile phones, use 3GPP and 3GPP2, are based on MPEG-4

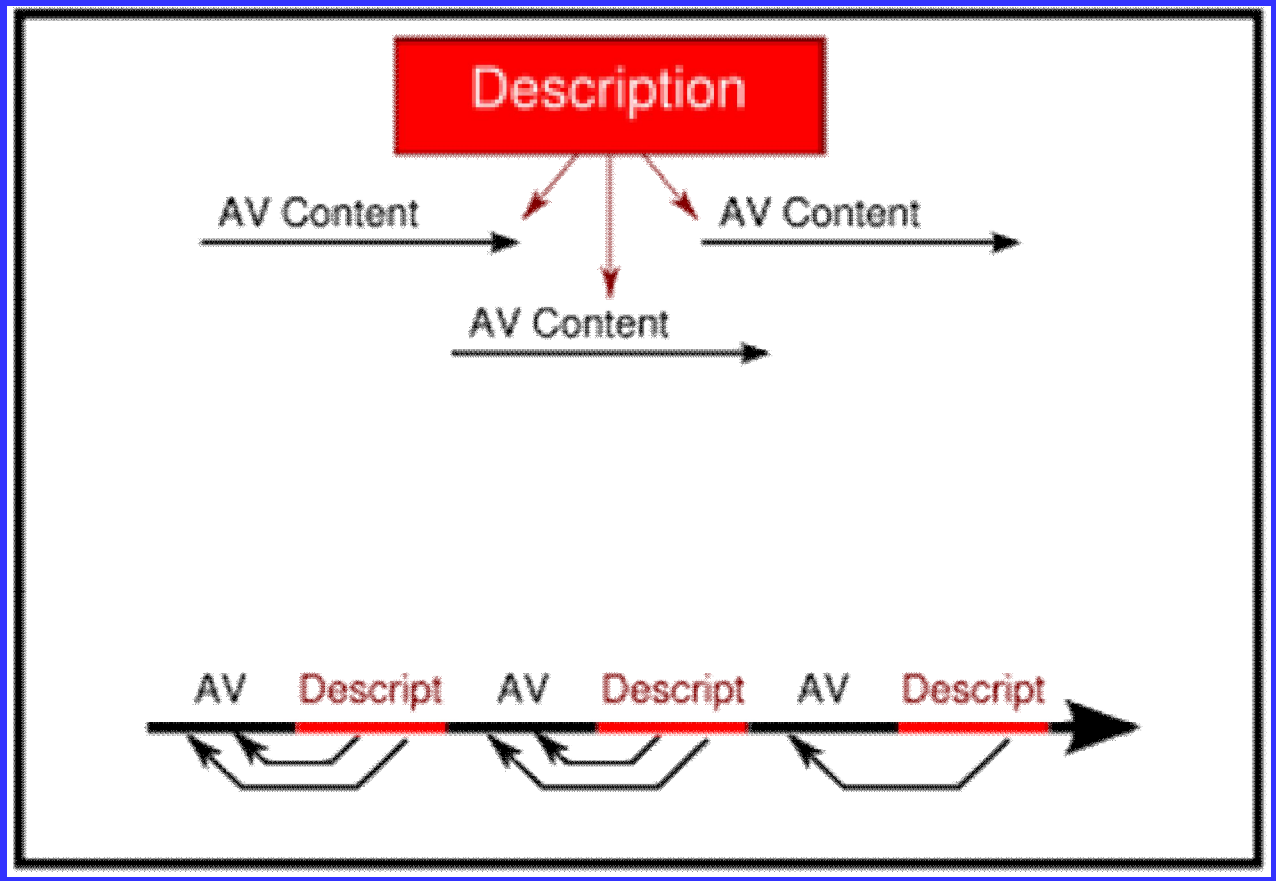
MPEG-4 (MP4) - 2

- Most of the features in MPEG4 are optional, means these may or may not be followed by the developers. This means that there are probably no complete implementations of the entire MPEG-4 set of standards.
- To deal with this, the standard includes the concept of "profiles" and "levels", allowing a specific set of capabilities to be defined in a manner appropriate for a subset of applications.
- Profiles describe somehow the content of an MPEG-4 object. E.g. the phone call would use the Speech Audio Profile, while the orchestra would require the High Quality Audio Profile.
- MPEG-4 is patented proprietary technology. This means that, although the software to create and play back MPEG-4 videos may be readily available, a license is needed to use it legally in countries that acknowledge software patents.
- Patents covering MPEG-4 are claimed by over two dozen companies.

MPEG-7

- Media tagging format for doing searches on arbitrary media formats via feature extraction algorithms
- Visual descriptors such as:
  - Basic Structures
  - Color
  - Texture
  - Shape
  - Localization of spatio-temporal objects
  - Motion
  - Face Recognition
- Audio descriptors such as :
  - Sound effects description
  - Musical Instrument Timbre Description
  - Spoken Content Description
  - Melodic Descriptors (search by tune)
  - Uniform Silence Segment
- Example application: Play a few notes on a keyboard and have matched song retrieved.

MPEG-7


Description
AV Content
AV Content
AV Content
AV
Descript
AV
Descript
AV
Descript

Video Container Formats

- A **container format** is a file format that can contain various types of data, compressed by means of standardized audio/video codecs.
- The container file is used to identify and interleave (multiplex) the different data types.
- Advanced container formats can support multiple audio and video streams, subtitles, chapter-information, and meta-data (tags) - along with the synchronization information needed to play back the various streams together.
- Example :
  - AVI (Audio Video Interleaved) from Microsoft. Very popular in movies
  - MP4 (standard audio and video container for the MPEG-4 multimedia portfolio) - used by iPod and most mobile phones
  - 3gp (used by iPod and most mobile phones)
  - MOV (standard QuickTime video container from Apple Inc.)
  - VOB : DVD Video OBject is a container based on DVD media. It internally contains MPEG-2 codec.
  - DAT : a container format for popular with VCDs.

Conclusion

- Media compression is indispensable even as storage and streaming capacities increase
- Future goals oriented towards increasing ease of access to media information (similar to google for text based information)


Thank You