Fundamentals of Media Compression

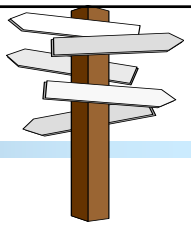
Prof. Ebroul Izquierdo

School of Electronic Engineering and Computer Science
Queen Mary, University of London

Seminario sobre TV Digital

Universidad del Valle
Cali – Colombia, October 28th 2008

Road Map



- **JPEG**
- **Video**
- **Motion Compensation**
- **MPEG**

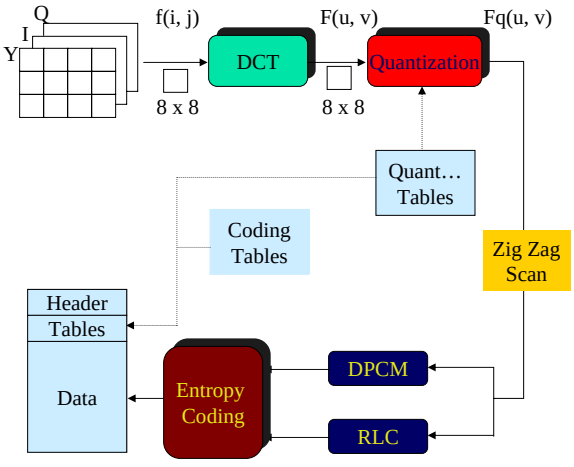
JPEG Image Compression

Why JPEG? The *compression ratio* of lossless methods (e.g., Huffman, Arithmetic) is not high enough for image and video compression.

JPEG uses *transform coding*, it is based on the following observations:

- A large majority of useful image contents change relatively slowly across images, i.e., it is unusual for intensity values to alter up and down several times in a small area, for example, within an 8 x 8 image block. A translation of this fact into the spatial frequency domain, implies, generally, lower spatial frequency components contain **more information** than the high frequency components which often correspond to less useful details and noises.
- Humans are more immune to loss of higher spatial frequency components than loss of lower frequency components.

JPEG Overview

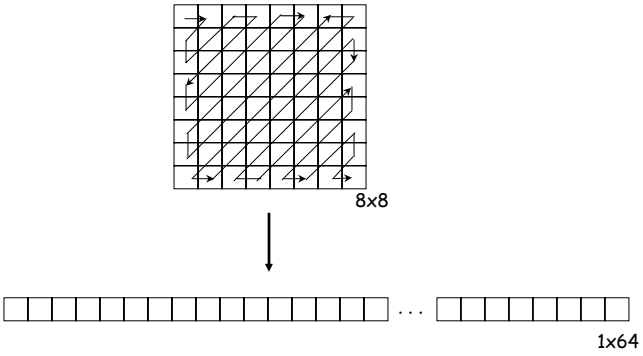


- Discrete Cosine Transform of each 8x8 pixel array f(x,y) $\rightarrow_T$ F(u,v)
- Quantization using a table or using a constant
- Zig-Zag scan to exploit redundancy
- Differential Pulse Code Modulation(DPCM) on the DC component and Run length Coding of the AC components
- Entropy coding (Huffman) of the final output

Zig-Zag Scan

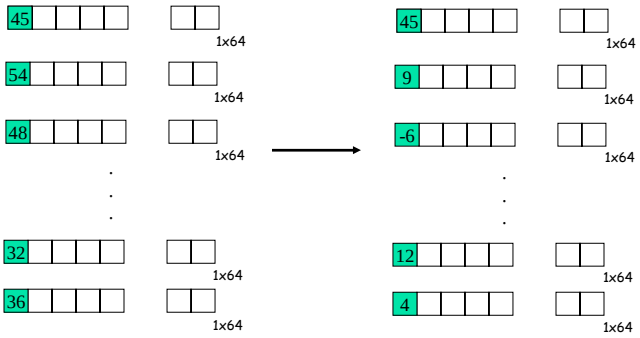
To group low frequency coefficients in top of vector and high frequency

Maps 8 x 8 to a 1 x 64 vector



DPCM on DC Components

- The DC component value in each 8x8 block is large and varies across blocks, but is often close to that in the previous block.
- Differential Pulse Code Modulation: Encode the difference between the current and previous 8x8 block.



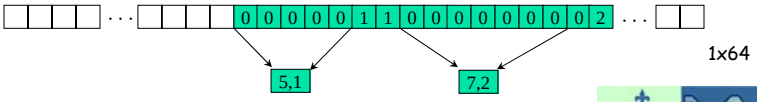
RLE on AC Components

The 1x64 vectors have a lot of zeros in them, more so towards the end of the vector.

- Higher up entries in the vector capture higher frequency (DCT) components which tend to be capture less of the content.
- Could have been as a result of using a quantization table

Encode a series of 0s as a (*skip*,*value*) pair, where *skip* is the number of zeros and *value* is the next non-zero component.

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



DC Entropy Coding (I)

DC components are differentially coded as (**SIZE**,**Value**)

- The code for a **Value** is derived from the following table

SIZE	Value	Code
0	**0**	**---**
1	**-1,1**	**0,1**
2	**-3, -2, 2,3**	**00,01,10,11**
3	**-7,…, -4, 4,…, 7**	**000,…, 011, 100,…111**
4	**-15,…, -8, 8,…, 15**	**0000,…, 0111, 1000,…, 1111**
.		**.**
.		**.**
11	**-2047,…, -1024, 1024,… 2047**	**…**

DC Entropy Coding

SIZE	Code Length	Code
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

DC components are differentially coded as (**SIZE**,**Value**). The code for a **SIZE** is derived from this table

Example: If a DC component is 40 and the previous DC component is 48. The difference is -8. Therefore it is coded as:

1010111

0111: The value for representing –8

(table in previous slide)

101: The size from the same table reads 4. The corresponding code from the table at left is 101.

DC Entropy Coding (II)

Example: If a DC component is 83 and the previous DC component is 80.

How is this component coded?

The difference is 3.

11: The value for representing 3

The size for 3 (from the same table) reads 2. The corresponding code from the next table is 011

Therefore it is coded as:

01111

JPEG Modes (I)

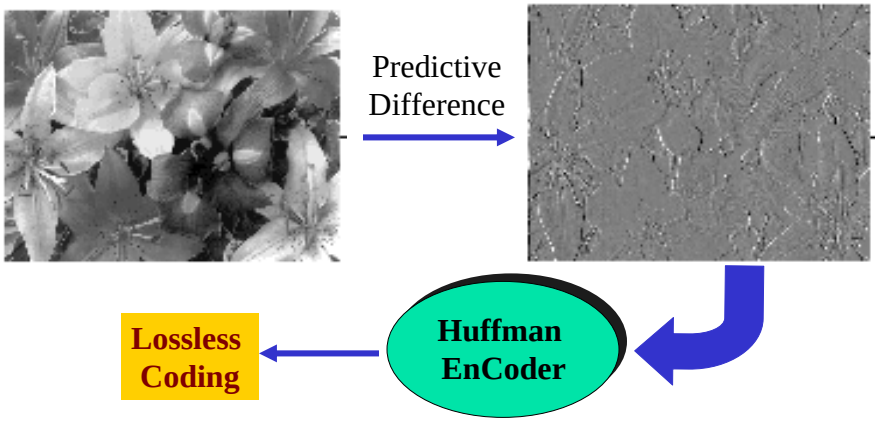
Sequential Mode:

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

- The technique we have been discussing so far is an example of such a mode, also referred to as the **Baseline Sequential Mode.**
- It supports only 8-bit images as opposed to 12-bit images as described before.

JPEG Modes (II)

Lossless Mode: It is a predictive coding mechanism as opposed to the baseline mechanism which is based on DCT and quantization(the source of the loss).



Lossless Mode

Predictive Difference:

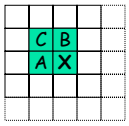
- For each pixel a predictor (one of 7 possible) is used that *best* predicts the value contained in the pixel as a combination of up to 3 neighboring pixels.
- The difference between the predicted value and the actual value (**X**)contained in the pixel is used as the *predictive difference* to represent the pixel.
- The predictor along with the predictive difference are encoded as the pixel's content.

JPEG Lossless Compression (I)

Predictive Difference:

- For each pixel a predictor (one of 7 possible) is used that *best* predicts the value contained in the pixel as a combination of up to 3 neighboring pixels.
- The difference between the predicted value and the actual value (**X**)contained in the pixel is used as the *predictive difference* to represent the pixel.
- The predictor along with the predictive difference are encoded as the pixel's content.
- The series of pixel values are encoded using huffman coding

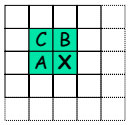
JPEG Lossless Compression (II)



Predictor	Prediction
P1	A
P2	B
P3	C
P4	A+B-C
P5	A + (B-C)/2
P6	B + (A-C)/2
P7	(A+B)/2

- The very first pixel in location (0, 0) will always use itself.
- Pixels at the first row always use P1,
- Pixels at the first column always use P2.
- The best (of the 7) predictions is always chosen for any pixel.

Lossless Mode



Predictor	Prediction
P1	A
P2	B
P3	C
P4	A+B-C
P5	A + (B-C)/2
P6	B + (A-C)/2
P7	(A+B)/2

- The very first pixel in location (0, 0) will always use itself.
- Pixels at the first row always use P1,
- Pixels at the first column always use P2.
- The best (of the 7) predictions is always chosen for any pixel.

Lossless Mode

Predictor	Prediction
P1	A
P2	B
P3	C
P4	A+B-C
P5	A + (B-C)/2
P6	B + (A-C)/2
P7	(A+B)/2

64	**67**
65	**61**

Predictor	Prediction
P1	65
P2	67
P3	64
P4	68
P5	66.5
P6	68.5
P7	66

Prediction is P3 = 64

JPEG Compression

Original LENA

450 K

JPEG Compression

Encoded Version

80% Quality

85 K

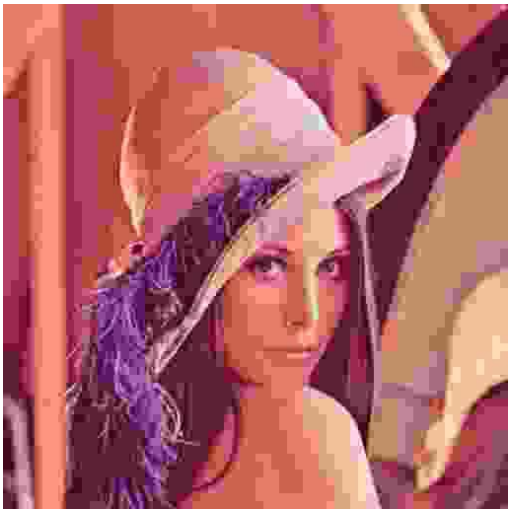
JPEG Compression

Strong compressed Version

10% Quality

15 K

JPEG Compression



Strong compressed Version

5% Quality

5 K

JPEG Compression 1% Compression

80 % Compression

99 % Compression

JPEG 2000

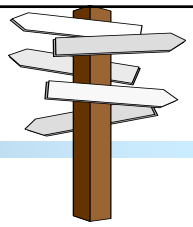
Some application requirements

- Strip processing
- Information embedding
- Repetitive encoding/decoding
- ROI encoding/decoding (static and dynamic)
- Fast/Random data access

Embedded block coding with optimized truncation

- Subbands partitioned into equal blocks
- Blocks encoded independently
- Post process to determine how each block's bitstream should be truncated
- Final bitstream composed of a collection of layers

Road Map



- **JPEG**
- **Video**
- **Motion Compensation**
- **MPEG**

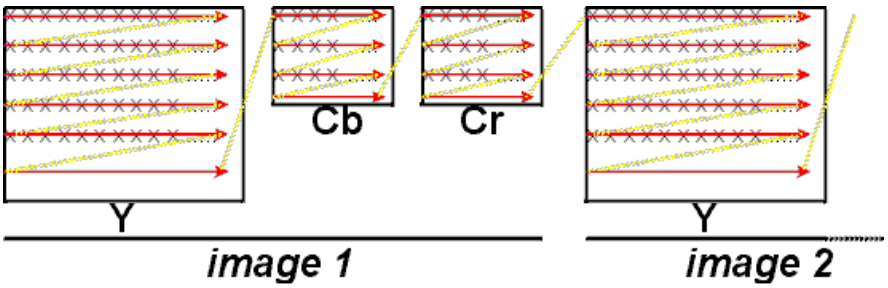
Video Formats (I)

RAW Video Format: The RAW video format is the simplest type of video format.

- The sequence in a RAW file consists of the luminance component (Y) followed by the chrominance components (CB, Cr)
- The frame data consists of the pixels from the Y, Cb and Cr frames in standard raster order. Thus the file looks like:

Y DATA FRAME 0
Cb DATA FRAME 0
Cr DATA FRAME 0
Y DATA FRAME 1
Cb DATA FRAME 1
Cr DATA FRAME 1
…

RAW Video Format



Grayscale movies can be specified by setting the Cb and Cr frame dimensions to 0.

Y DATA FRAME 0
Y DATA FRAME 1 …

Digital Video (I)

Advantages over analog:

- Direct random access. Good for nonlinear video editing
- No problem for repeated recording
- No need for blanking and sync pulse

- Almost all digital video uses *component video*
- The human eye responds more precisely to brightness information than it does to color, *chroma subsampling* (*decimating*) takes advantage of this.

Digital Video (II)

4:4:4 scheme: each 8×8 matrix of RGB pixels converts to three Y, Cr, Cb 8×8 matrices: one for luminance (Y) and one for each of the two chrominance bands Cr and Cb (or U, V)

YUV to RGB conversion:

B = (1.164 * (Y - 16) + 2.018 * (V - 128))

G = (1.164 * (Y - 16) - 0.813 * (U – 128) –
0.391 * (V - 128))

R = (1.164 * (Y - 16) + 1.596 * (U - 128))

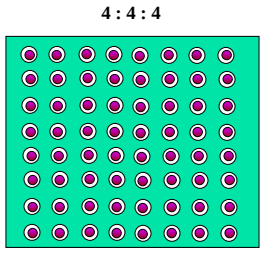
Digital Video (III)

4:2:2 scheme also creates one 8×8 luminance matrix but decimates every two horizontal pixels to create each chrominance-matrix entry

- Thus reducing the amount of data to 2/3^{rds} of a 4:4:4 scheme.

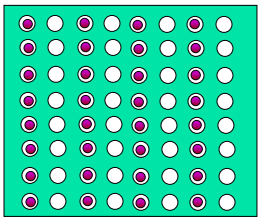
4:2:0 scheme decimate chrominance both horizontally and vertically, resulting in four Y, one Cr, and one Cb 8×8 matrix for every four 8×8 pixel-matrix sources. This conversion creates half the data required in a 4:4:4 chroma ratio.

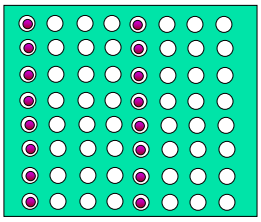
Chroma Subsampling

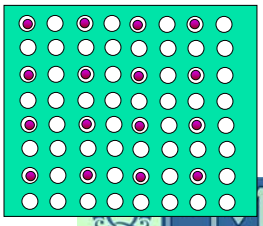


4:1:1 and 4:2:0 are used in JPEG and MPEG

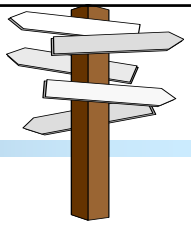
256-level gray-scale JPEG images aren't usually much smaller than their 24-bit color counterparts, because most JPEG implementations aggressively subsample the color information. Color data therefore represents a small percentage of the total file size.





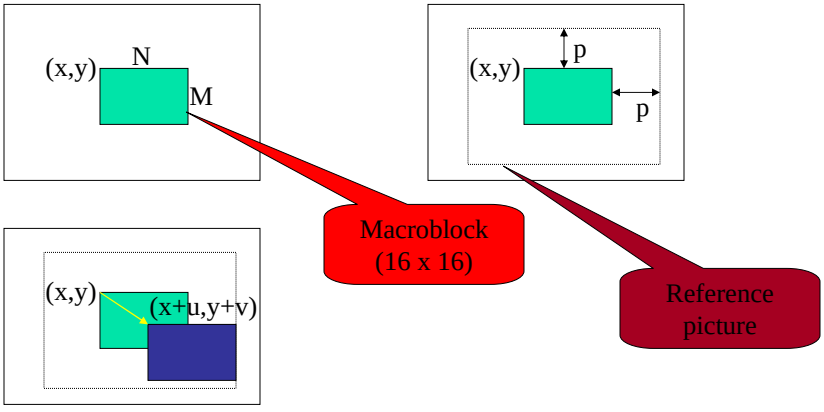


Road Map



- JPEG
- Video
- **Motion Compensation**
- MPEG

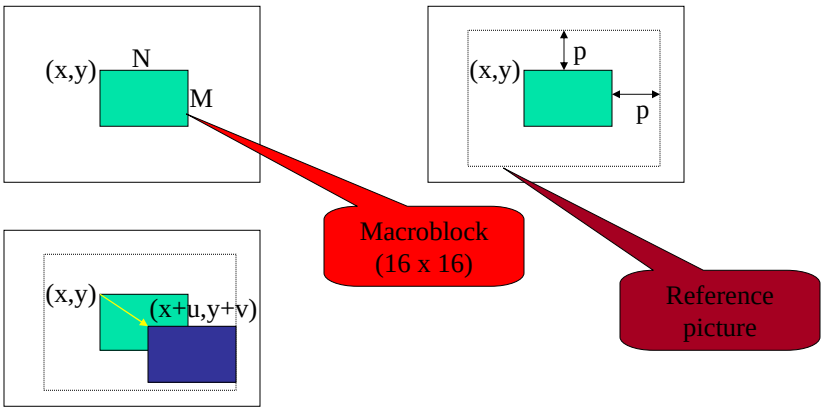
Motion Compensation



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

Minimize MAE

Motion Compensation



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

Minimize MAE

Motion Estimation (I)

Algorithm 1: full search

Algorithm 2: 2D-logarithmic search

- Partition the [-p,p] rectangle into a [-p/2,p/2] rectangle and the rest
- Compute the MAE function at the center and 8 perimeter points of the [-p/2,p/2] rectangle. Let the points be d_1 pixels apart
- Find the point with the minimum MAE
- Start with this location and repeat the above steps, but reduce the distance to $d_1/2$
- Repeat until the *k*-th search when the distance between the points is 1 pixel

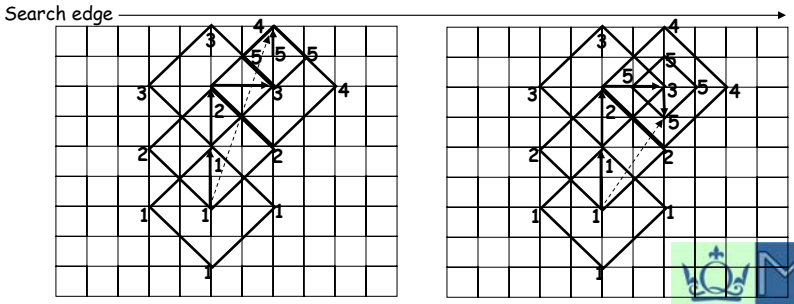
It does not always lead to the same result

2-D Logarithmic Search

Iterative comparison of the error measure (MAE) at five neighboring points

Logarithmic refinement (divide by 2) of the search pattern if

- Best match is in the center of the 5-point pattern (right figure)
- OR, center of search pattern touches the border of the search range (left figure).



Motion Estimation (II)

Other faster but less accurate algorithms:

Algorithm 3: Diamond search

Algorithm 4: Hexagonal search

Algorithm 5: Cross-search

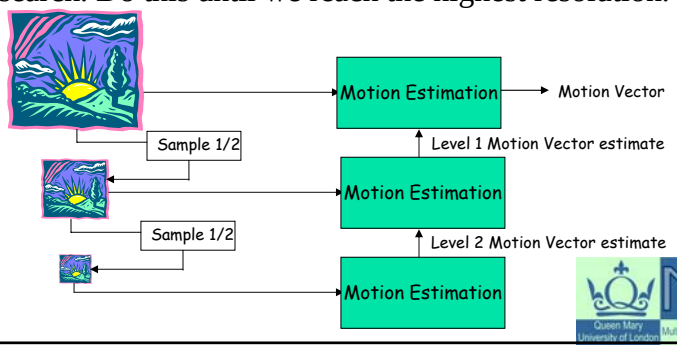
Algorithm 6: Hierarchical or multiresolution

None of them lead to the same full search result!

Hierarchical Motion Estimation (I)

Basic Idea:

- Form a collection of increasingly sub-sampled versions of the current and reference image
- Find a motion vector (*use one of the previous methods) using the lowest resolution images.
- Repeat the same in the higher resolution image using the previous (lower) answer as the initial point for search. Do this until we reach the highest resolution.



Hierarchical Motion Estimation (II)

Make 2 progressively low-resolution and downsampled versions of the current frame and the reference frame

- Let macroblock of reference frame be located at (x,y)
- Corresponding macroblocks are located in (x/2,y/2) and (x/4,y/4) for Level 1 and Level 2
- Let the size of the Level 0 macroblock be 16 X 16
- Let the motion vector have a dynamic range of ±p pixels

- Estimate motion vector from the Level 2 image, using a macroblock of 4 x 4 and a search space of [-p/4,p/4].
  - Let MAE be minimized at (u_2, v_2)

Hierarchical Motion Estimation (III)

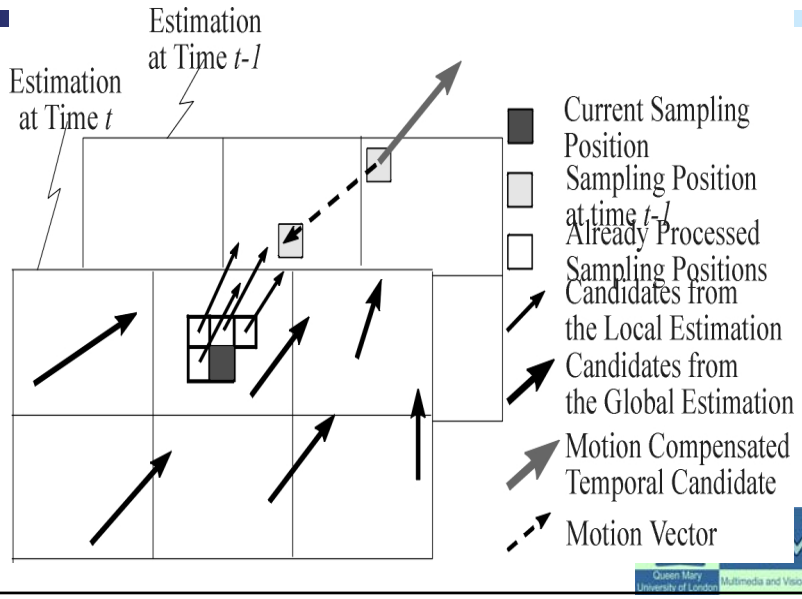
At Level 1, perform a motion vector search on 8 x 8 macroblocks

- The search is centered at $(x/2+2u_2, y/2+ 2v_2)$
- The search space is [-1,1]
- Let the minimal MAE be at (u_1, v_1)

At Level 0, perform a motion vector search on 16 x 16 macroblocks

- The search is centered at $(x+2u_1, y+ 2v_1)$
- The search space is [-1,1]
- Let the minimal MAE be at $(u_0, v_{0\,v})$

Motion Estimation with Prediction



Matching Criteria

- MAD
- MSD
- Cross-Correlation
- Normalized Cross-Correlation
- Any of these with weighting factor

$$MADI(d_l) = \frac{1}{mn}\sum_m\sum_n \left| I_l^{(t)}(z) - I_r^{(t)}(z + d_l(z,t)) \right| + \alpha \left\| d - d_p \right\|$$

Motion Estimation Example

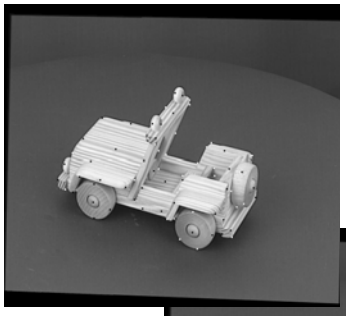


Image at time *t*

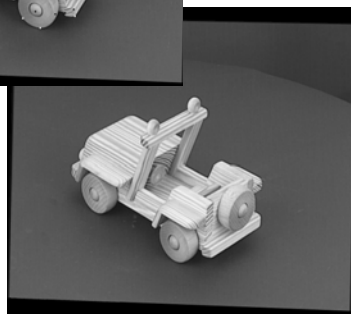
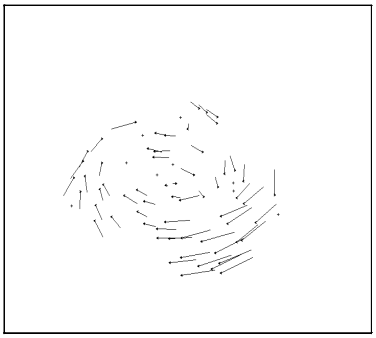


Image at time *t+1*



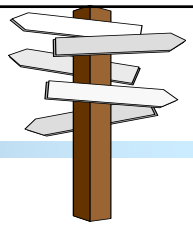
Sparse motion vectors

Motion Estimation (II)

Other faster but less accurate algorithms:
Algorithm 3: Diamond search
Algorithm 4: Hexagonal search
Algorithm 5: Cross-search
Algorithm 6: Hierarchical or multiresolution

None of them lead to the same full search result!

Road Map



- **JPEG**
- **Video**
- **Motion Compensation**
- **MPEG**

MPEG

"Moving Picture Coding Experts Group", standards body for delivery of video and audio.

MPEG-1 Target: VHS quality on a CD-ROM or Video CD (VCD) (352 x 240 + CD audio @ 1.5 Mbits/sec)

MPEG-2 allows different levels and profiles

Both standards have four parts:

- Video: Defines the video compression decoder
- Audio: Defines the audio compression decoder
- System: Describes how various streams(video, audio or generic data) are multiplexed and synchronized.
- Conformance: Defines a set of tests designed to aid in establishing that particular implementations conform to the design.

MPEG: Video Encoding

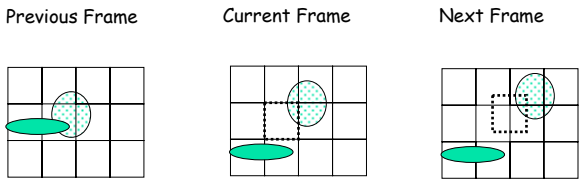
The MPEG standards

- do not define an encoding process
- define syntax of the coded stream
- define a decoding process

MPEG

Some macroblocks need information that is not present in the previous reference frame.

- Maybe, such information is available in a succeeding frame!

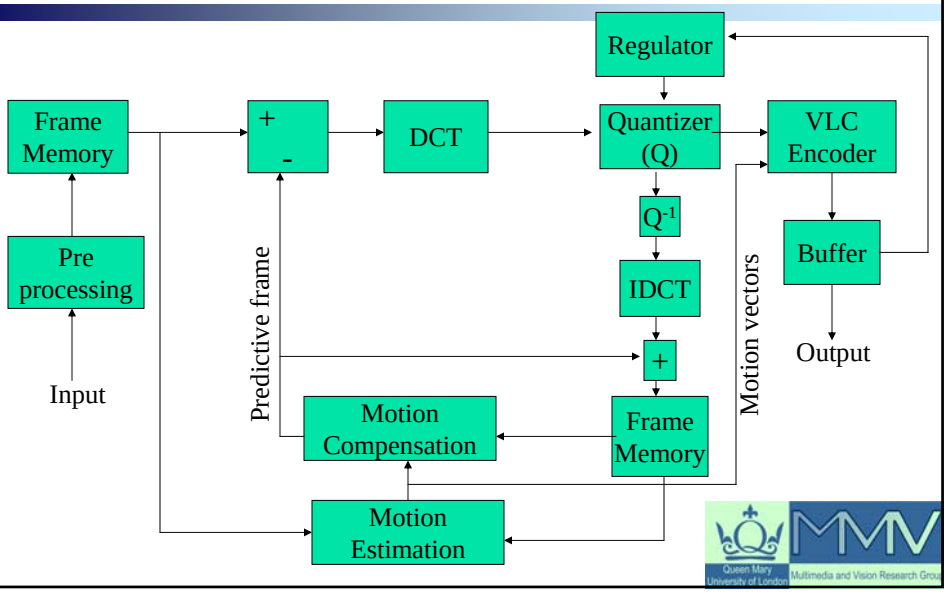


Add a third frame type (*B-frame*): To form a B-frame, search for matching macroblocks in both *past* and *future* frames.

Typical pattern is IBBPBBPBB IBBPBBPBB IBBPBBPBB

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

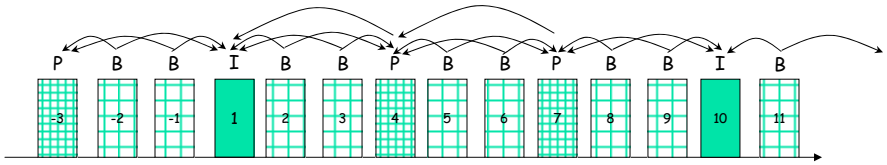
MPEG: Video Encoding



Bitstream order vs. Display order

Bitstream (Transmit) order:

1(I), 4(P), 2(B), 3(B), 7(P), 5(B), 6(B), 10(I), 8(B), 9(B)



Frame and Macroblock Prediction Types (I)

Anchor frame: *A frame that can be used for prediction*

Macroblock Type	Prediction
Nonpredicted Macroblock	**None**
Backward-predicted macroblock	**References temporally nearest subsequent anchor frame**
Forward-predicted macroblock	**References temporally nearest previous anchor frame**
Bidirectionally predicted macroblock	**Averages predictions from temporally nearest, previous and subsequent anchor frames**

Frame and Macroblock Prediction Types (II)

Frame Type	Anchor Frame	Macroblock Type
I-frame	Yes	Nonpredicted
P-frame	Yes	Nonpredicted, Forward predicted
B-frame	No	Nonpredicted, Forward predicted, Backward predicted, Bidirectionally predicted

Bidirectional Prediction

- B-frames allow effective prediction of uncovered background, areas of the current picture that were not visible in the past and visible in the future.
- B-frames can provide for interpolation equivalent to an even finer degree than half-pixel (1/4 pixel for example).
- If good prediction is available in both the previous and subsequent anchor frames, then averaging the two predictors reduces noise and hence increases efficiency.
- A ratio of 5:3:1 between the number of bits spent on I, P and B frames is quite common.
- Errors in B-frames tend to limit the effect to that B-frame only.

MPEG-4

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