The University Of Sheffield.

EEE6081 (EEE421)
Visual Information Engineering (VIE)
Topic 7 – Image and video coding

- Image and video representation
  - Data rates and available bandwidth
  - Redundant and Irrelevant data
- Image compression
- Video compression
- Motion estimation
- Scalable coding

Dr. Charith Abhayaratne
c.abhayaratne@sheffield.ac.uk

Reference Books

- Multimedia and Communications Technology
  - Steve Heath
    - Chapters 4 (General video), 5 (Image compression) and 6 (Video compression).

- Digital Image Processing
  - R. C. Gonzalez and R. E. Woods (Second Edition)
    - Chapters 1 (Introduction), 2 (Digital image fundamentals), 7 (Wavelets and Multi-resolution processing) and 8 (Image compression)

- Wavelets and subband coding
  - Martin Vetterli [e-version is available online]
    - Chapters 7.3 (Image Compression) and 7.4 (Video Compression).
    - Chapters 1 and 3 for wavelet transforms

How to find the bit rate?

W pixels/line
H Lines/frame

H x W pixels/frame (This is the pixels in Y channel)

S Chrominance sub sampling factor
S=3 for 4:4:4 S=2 for 4:2:2 S=1.5 for 4:2:0

N bits/pixel (bpp)

S x N x H x W bits/frame

F frames/sec

S x N x H x W x F bits/sec

Derive the bit rate for 4:2:2 format 4CIF video!

Y channel resolution of 4CIF video = 576 x 720

Considering N= 8 bits per pixel per colour channel

Cb and Cr resolutions using 4:2:2 sampling S= 2

Memory per frame = 576 x 704 x 2 x 8 = 6,488,064 bits

Memory to store 90 minutes of video (at 50 frames per sec)

6,488,064 x 50 =324,403,200 bits/s

For 90 minutes 324,403,200 x 60 x 90 = 204 G Bytes

How many DVDs are required to record this programme?
(A single layer DVD can store only about 4.5 Gbytes per disk)

Another Example

Derive bit rate for video transmission over mobile networks using QCIF 4:2:0 format

Mobile phones display resolution
180 x 144

Typical Frame rate is 6.25 fps

Bits/frame = 180 x144 x 1.5 x 8 ~ 304 K bits

At 6.25 frames/s ~ 1.9 Mbits/s

What is the available mobile phone network bandwidth?

What can we do to further reduce the data rate?

Why is it necessary to compress data?

Data rate requirements:

1. The data rate for 4CIF resolution 4:2:2 video =

2. The data rate for transmitting QCIF 4:2:0 and 6.25 fps video over a mobile communication link =

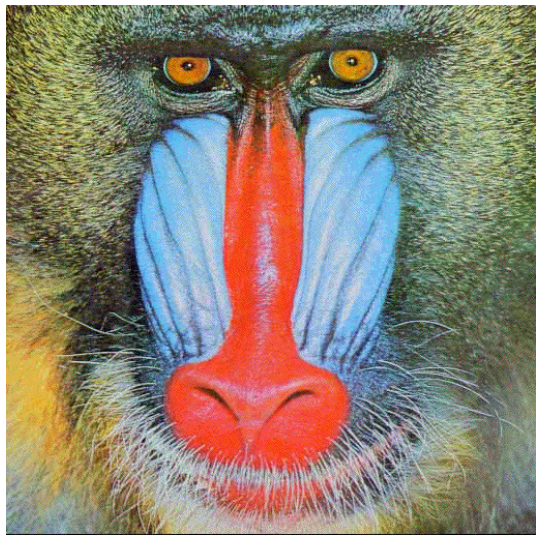
What is available:

Capacity of a DVD -	~4.5 Gbytes
Standard modem -	~20K bits/sec
Broadband modem -	~1M bits/sec
Mobile phone -	~128K bits/sec

The available bandwidth and storage space is limited.

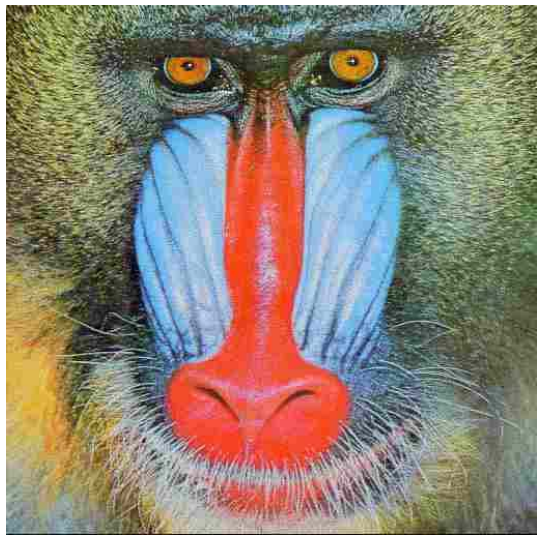
Solution is data compression

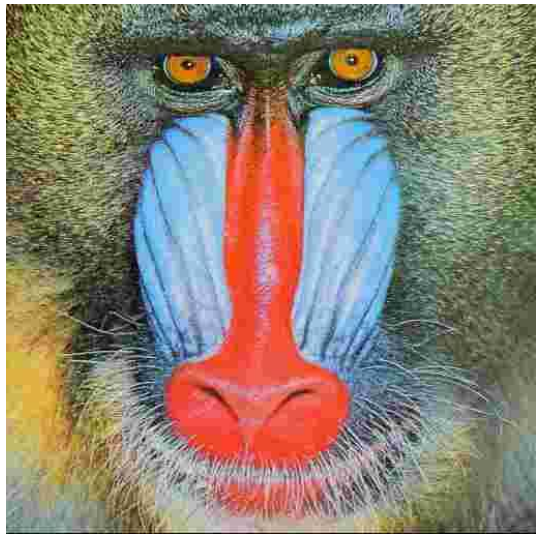
Image Compression Examples



512 x 512 pixels (8 bit colour)
262144 bytes

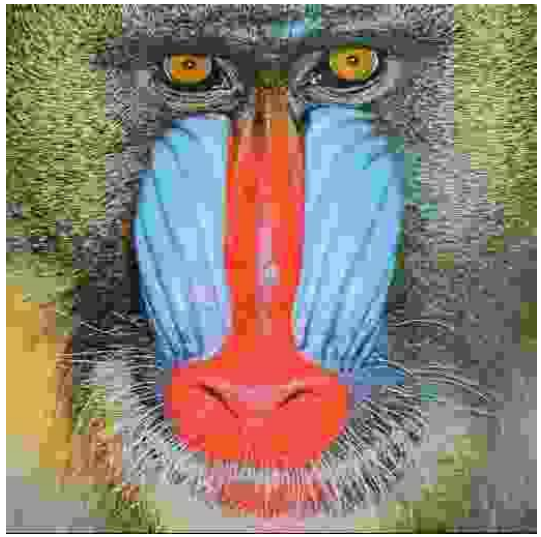
10:1 Compression
26,200 bytes





20:1 Compression
13,100 bytes

40:1 Compression
6,600 bytes



Why is data compression possible?

Usual digital representations are redundant. For compression remove data redundancies. Data redundancy can be of several forms.

1. Coding Redundancy

We know images use N-bit (usually N=8 for monochrome or for each colour channel) codewords to represent images/video pixels per colour channel. This is a fixed-length code representation.

What are the disadvantages of using fixed length codes?

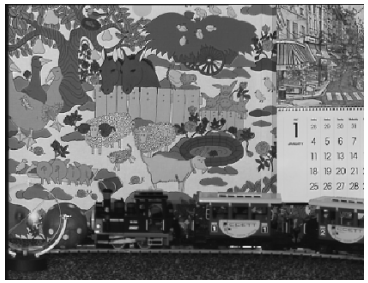
What is the solution?

2. Inter-sample (inter-pixel) redundancy

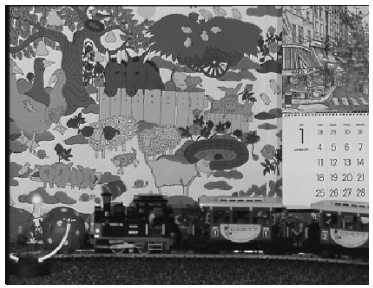
The spatial and temporal (for video) correlations with neighbouring pixels. Usually, the neighbouring pixels have similar gray-levels.

Spatial redundancy

Temporal redundancy



Frame 1

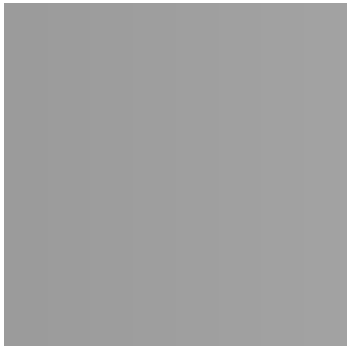


Frame 2

If data is correlated the inter-sample redundancy is high. By decorrelating data inter-sample redundancy can be removed.

3. Psycho-visual redundancy

Consider these 8x8 pixel blocks. Visually they look the same.

A 

B 

The actual image matrices are:

$$A=\begin{pmatrix} 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \\ 156 & 157 & 158 & 159 & 160 & 161 & 162 & 163 \end{pmatrix}$$

$$B=\begin{pmatrix} 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \\ 160 & 160 & 160 & 160 & 160 & 160 & 160 & 160 \end{pmatrix}$$

Which block has the higher entropy?

How can we obtain B from A?

Image/video Redundancy

Inter-pixel Redundancy:

The spatial and temporal (for video) correlations with neighbouring pixels. Usually, the neighbouring pixels have similar gray-levels.

Psychovisual Redundancy:

The eye does not response with equal sensitivity to all visual information. Certain information has less relative importance than other information in normal visual processing. This information is said to be psychovisually redundant.

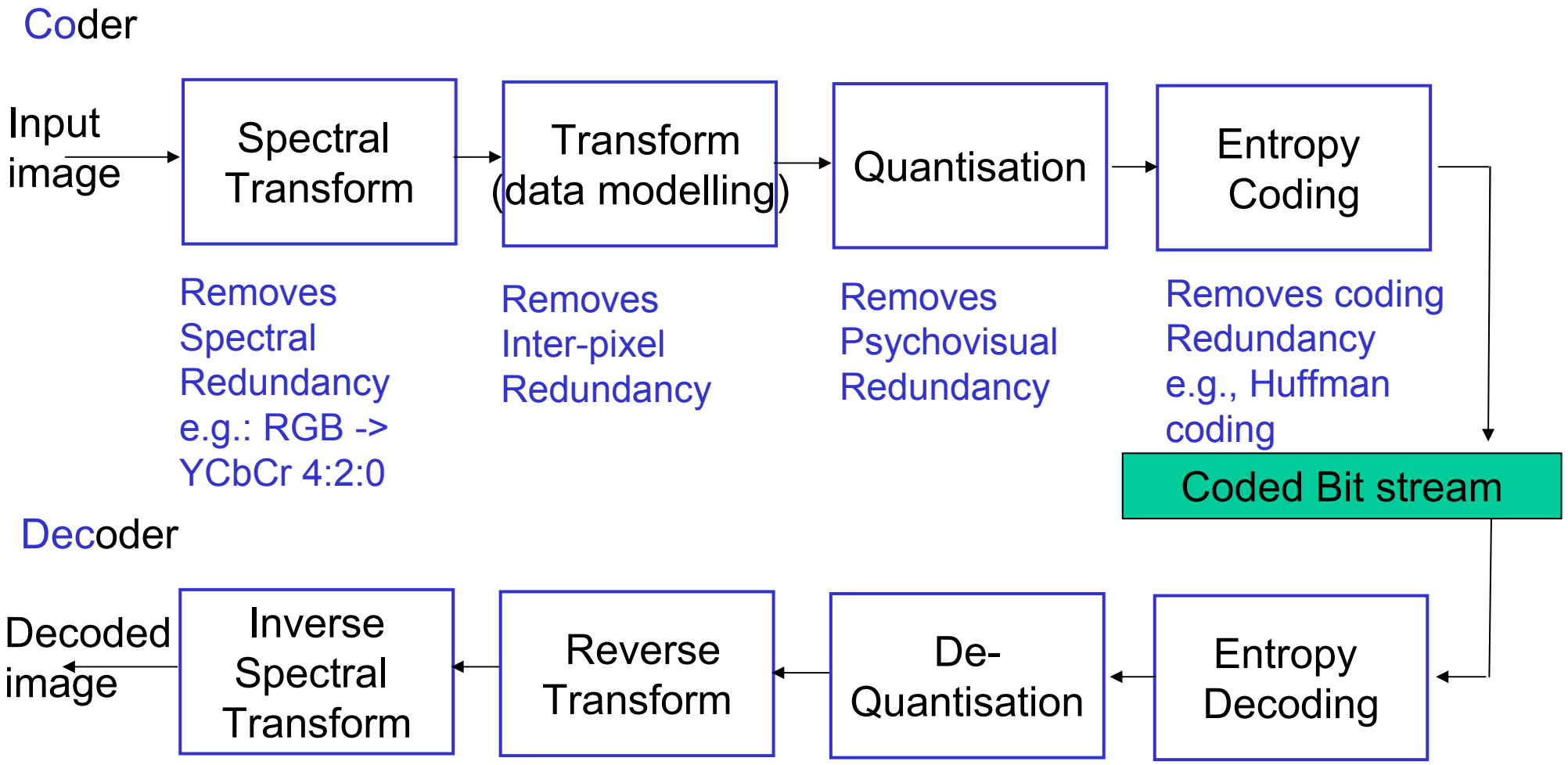
Coding Redundancy (R):

Based on the probability of occurrence for a symbol

Spectral Redundancy:

The correlation among different spectral bands. For example, the redundancy in RGB bands.

Image Compression Model



Coder-Decoder pair is called Codec

Removing spatial redundancy

We need to decorrelate the image data.

-One approach is to develop a model and use this model to predict that data and replace data values with the prediction error

- e.g.,
  - Linear Pedictive coding (for speech)
  - Differential Pulse Coded Modulation (DPCM)

Similarly for images 2D DPCM can be used.

But there are disadvantages in using DPCM

Better solution is to use

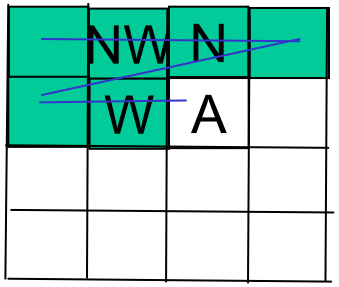
-The Discrete Cosine Transform (DCT)

-The Wavelet Transform

An example of a 2D template

$A_p = A-(aW+bN+cNW)$

$a+b+c=1;$



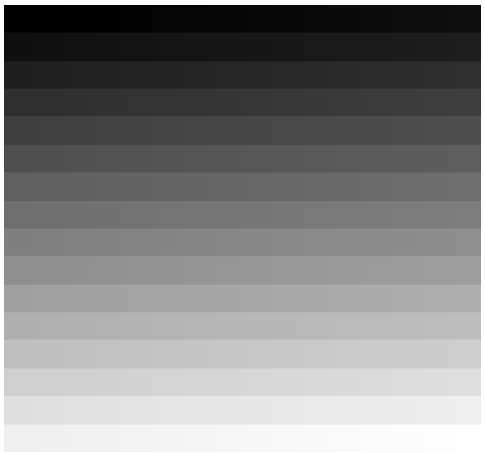
Removing spatial redundancy

These images use 256 gray levels. That means 8 bits per pixel (bpp).

The Shannon's entropy 7.46 bpp

Only one colour.

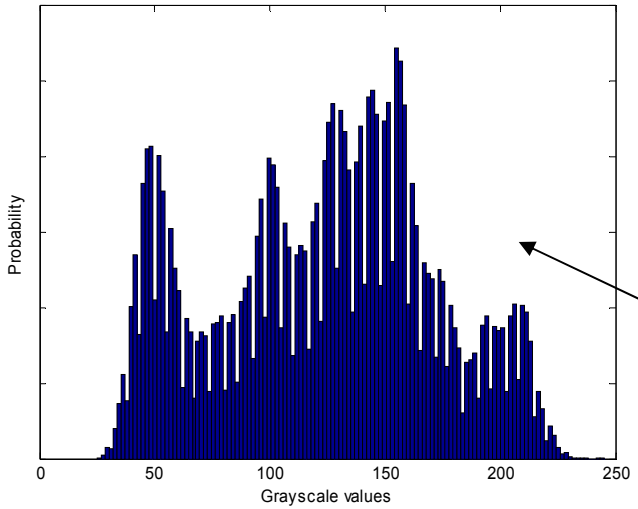
What is the entropy?



16x16 pixels
In total 256 pixels.

Each pixel represent
Each of the 256 colours.

What is the entropy?

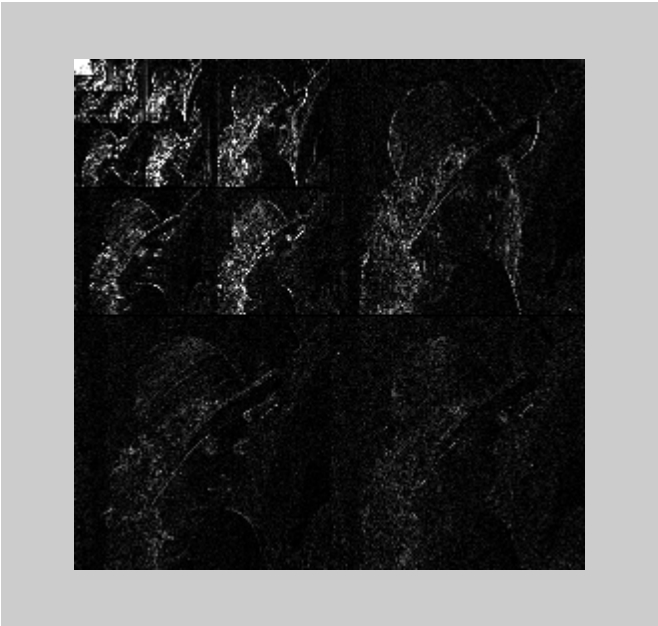


Probability distribution of Lena image.
We have to modify this distribution to one with a narrow peak and long tails to reduce entropy.

Removing inter-sample redundancy

For images using image transformations represent images in an efficient way.
e.g., The Discrete Cosine Transform, The Wavelet transform

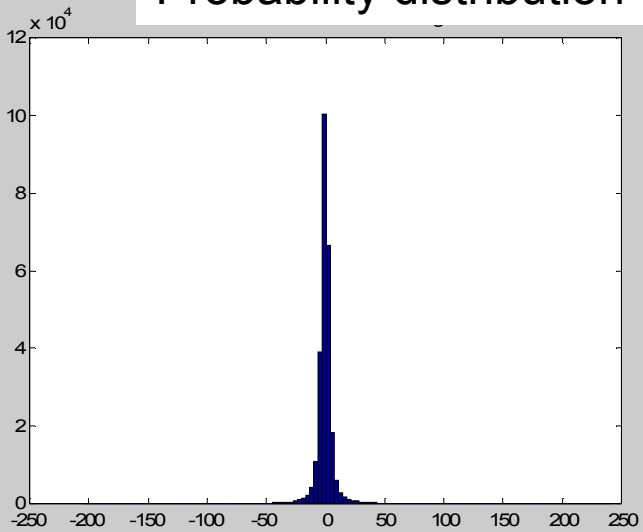
They remove inter-sample redundancy and compact energy into a fewer number of coefficients (low pass or low frequency coefficients)



Gray scale representation of the transformed Lena image

Entropy 4.35 bpp

Probability distribution



Removing psycho-visual redundancy

We can reduce the bit resolution by further quantising the transformed image values.

Quantisation of a transformed pixel x using a quantisation factor Q is

$$x_q = \text{Round}\{ x/Q \}$$

Fewer bits need to represent x_q

Rounding operation discards information. Usually the appropriate Q values are determined by the HVS limits.

De-quantisation $x_r = x_q \, Q$

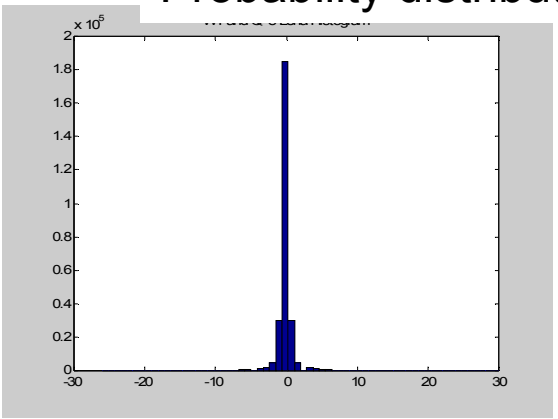
Note that x_r and x are not the same. This is where image coding process adds errors to pixel values.

Quantisation error is $x - x_r$

Removing psycho-visual redundancy

Grey scale representation of Q=8 quantised Lena transform coefficients

Probability distribution



Decoded Image for Q=8

Entropy 1.38 bpp

Decoded Image for Q=16

Entropy= 0.08 bpp

An Example

8x8 image block

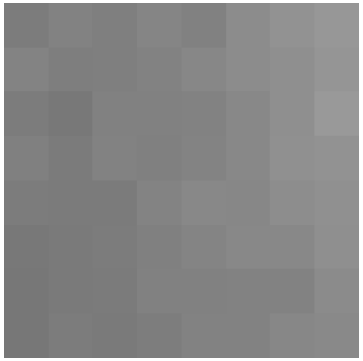
125	131	128	134	130	141	147	152
132	127	128	131	136	141	144	150
125	121	130	130	131	137	144	154
129	124	131	129	132	137	145	147
125	124	124	132	137	136	142	145
121	123	125	129	133	137	137	144
120	123	124	130	130	131	131	140
120	125	123	126	131	131	136	138

Image Transform coefficients using 2D - DCT

133	12	3	11	0	4	8	6
-5	-2	3	6	-5	-1	5	6
-2	-2	4	2	1	6	2	5
0	-5	-1	-1	4	4	0	5
1	-1	3	-3	-11	-3	-6	1
3	0	1	-3	-1	-2	-1	-8
-2	-1	-1	-3	3	-4	5	4
1	-2	0	1	2	-3	-1	-7

Quantised Transform coefficients

16	4	0	1	0	0	0	0
2	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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



Decompression

Compression

Which of these 3 matrices has the lowest entropy?

Lossless vs. Lossy coding

Can reduce the file size (or data rate) as most data representations are not optimally efficient.

Two basic forms of compression – **lossless** and **lossy**

- Possible to reconstruct exactly the original data.
- No information is discarded (does not include quantisation process).
- Useful for Scientific imaging, where the exact pixel values are required to do further image analysis.
- E.g., JPEG-LS
- Only low compression gains. (e.g., 1.5:1 or 2:1.
- The compressed file sizes are closer to entropy of the image.

- Possible only to reconstruct an approximation of the original data.
- Information is discarded.
- The limits of the HVS are exploited.
- High compression ratios.
- 10:1 visually lossless. (The rate at which just noticeable difference –JND- between the original and the compressed image is seen)
- e.g. JPEG

JPEG Compression

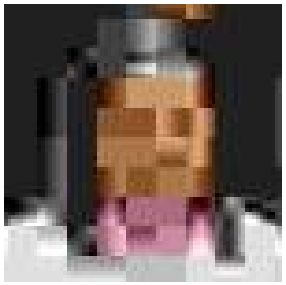

20:1


40:1


60:1


80:1


100:1


120:1

The latest standard JPEG2000 is much better
File formats: .jp2 and .j2k

Can we do better with video?

176 x 112 pixels at 8 bits/colour at 10 frames/s

Raw data rate = 4,730 k bit/s

Playing = 14.4 k bit/s

Compressed = 330:1 (0.3%)

Introduction to video compression

Can consider video as a simple sequence of digitized pictures

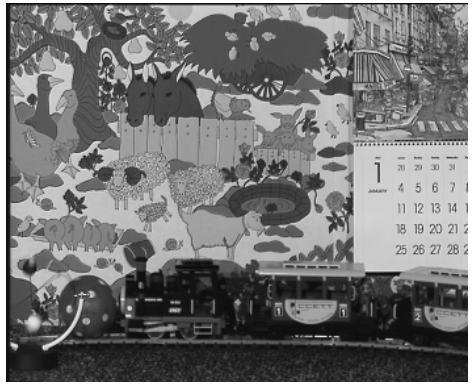
So just transmit a sequence of separately compressed JPEG images
This approach is known as Motion JPEG.
This is called Intra only coding (I frames).

Compression is limited as reasonable compression ratio per JPEG is ~20:1

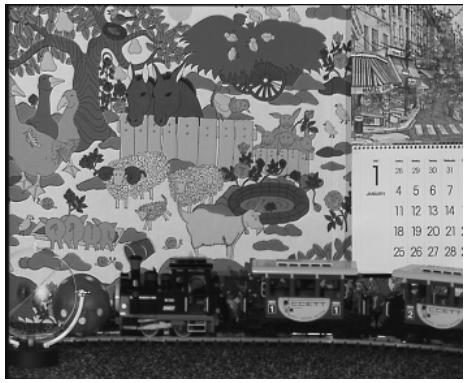
Need to use the temporal redundancy between frames (pictures) – often little change between two consecutive frames.

Video Motion

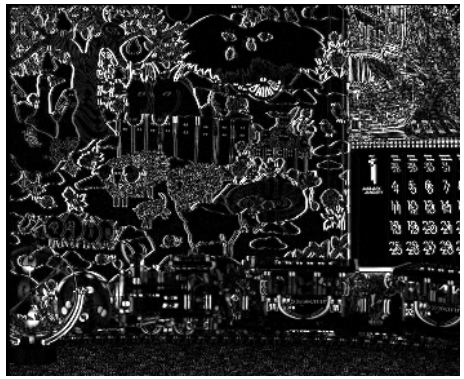
- Can transmit frame (picture) differences.



F1 7.61 bpp



F2 7.61 bpp



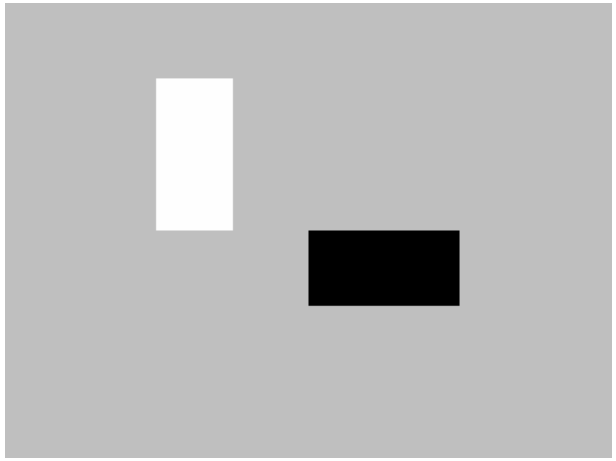
F2-F1 6.48 bpp

- Changes in consecutive pictures are due to
  - Consistent motion of large regions of image.
    - Horizontal, vertical and rotational
  - Camera motion
  - Zooming in and out
  - Scene and shot changes
  - Entry or exit of objects

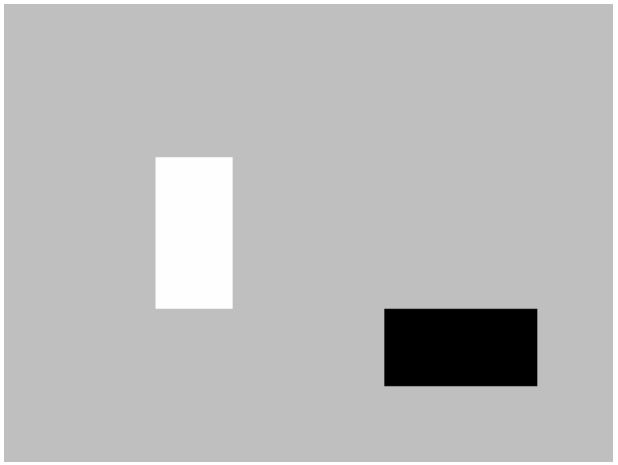
Video Motion

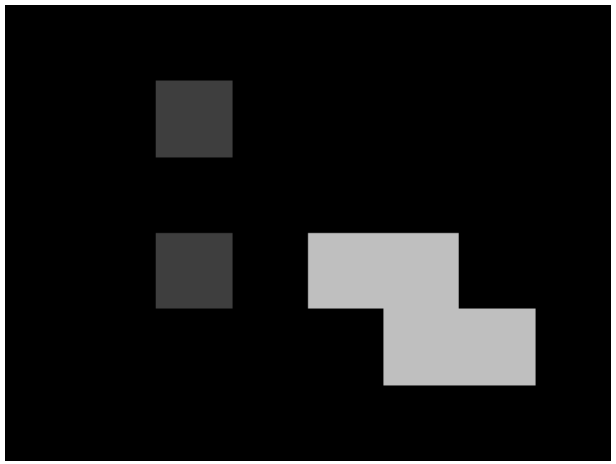
Consider two frames: A. --- Current frame (the frame to be predicted).
B. --- A previously encoded frame (the reference frame).

B



A

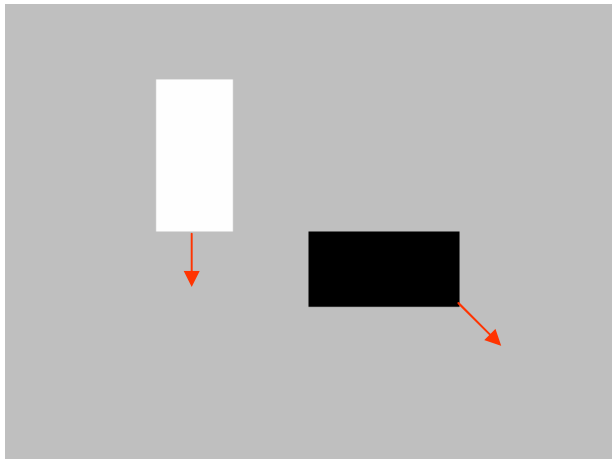




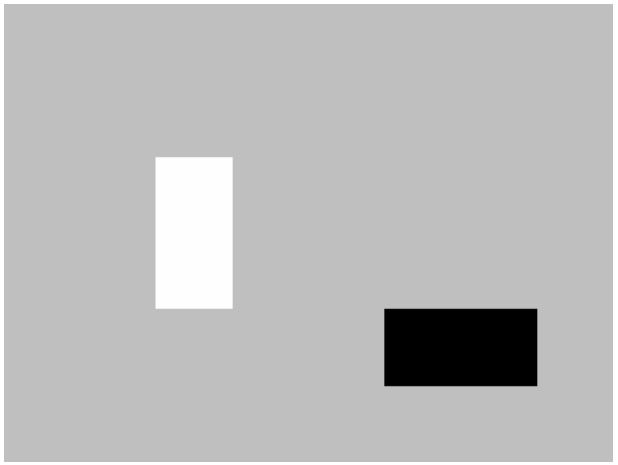
Frame difference
C= A-B (no change in background)
C is the prediction error.
Now we can write
A= C+B;
Since B is the previously coded frame. We only need to encode C to transmit Frame A to the receiver.

Video Motion

B



A



Now if we know how the displacements of the objects, we can do a better prediction.
C= A – f(B, displacement values),

f is called Motion compensated prediction.
The process to find displacement values is called Motion Estimation.

C is the prediction error.
A= C + f(B, displacement values),

Since B is the previously coded frame. We only need to encode C and displacement values to transmit Frame A to the receiver (C is zero if displacement values are accurate).

Motion compensated prediction (MCP)

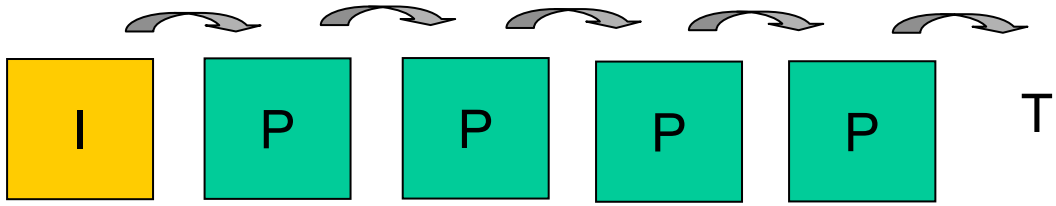
For complex scenes, it is difficult to estimate motion for each object.

Instead we partition each frame into smaller non-overlapping blocks, estimate displacement for each of the block. In this case it is difficult to accurately estimate displacement for all blocks. Hence prediction error (C) is not exactly zero.

C=F2- f(F1, displacements)

New entropy is 5.01 bpp

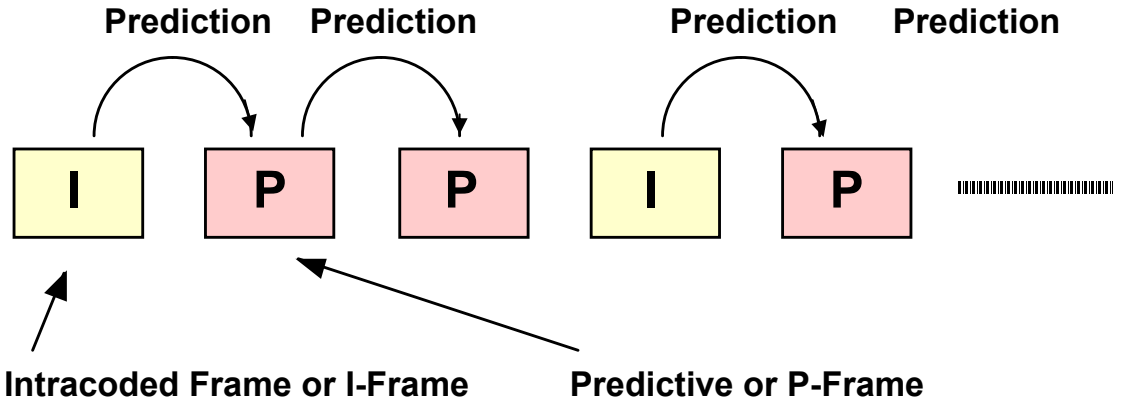
Intra frame (I) = A frame encoded as a still image.
Predicted frame (P) = A frame encoded using MCP. -- known as P frames



This is Motion compensated DPCM.

Is this a good strategy?
Why?

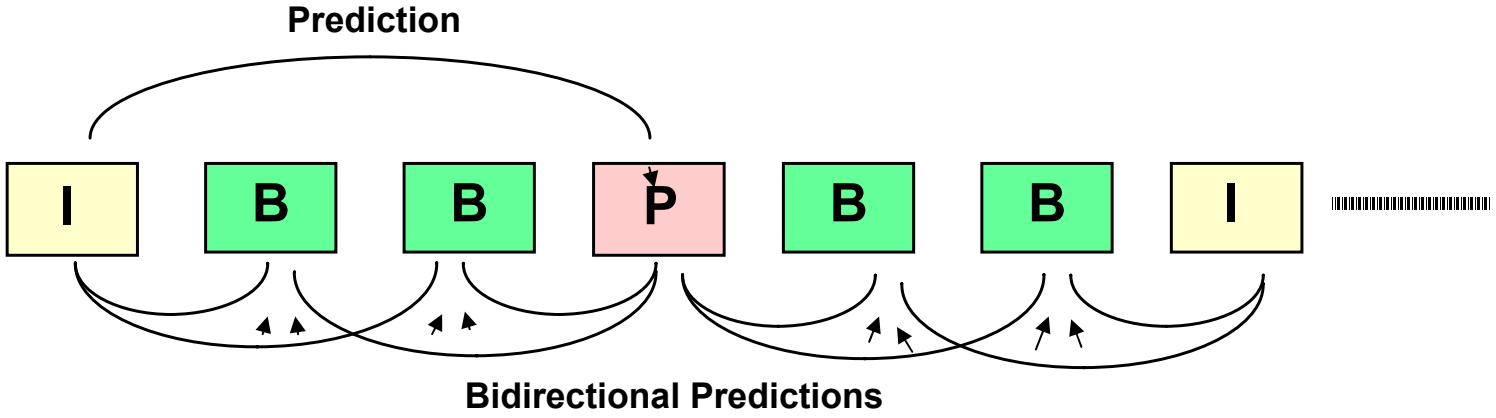
A better Strategy



- **Intra Frames** - encoded without reference to any other frame. Treated as separate picture and use JPEG type algorithm (normally) on separate Y, C_b and C_r images.
- Level of compression is fairly low. Must be the first frame of clip (and probably the first frame of a new scene).
- Need to resend a new I-frame occasionally as cannot work just with frame differences if an error occurs. An I-frame is coded, typically, at every 4-12 frames.
- Encoding of a **P-frame** depends on content of preceding I-frame or preceding P-frame. Use combination of motion estimation (displacement values) and motion compensated prediction - hence much higher compression.
- At the same time P- frames are highly computationally complex due to motion estimation.

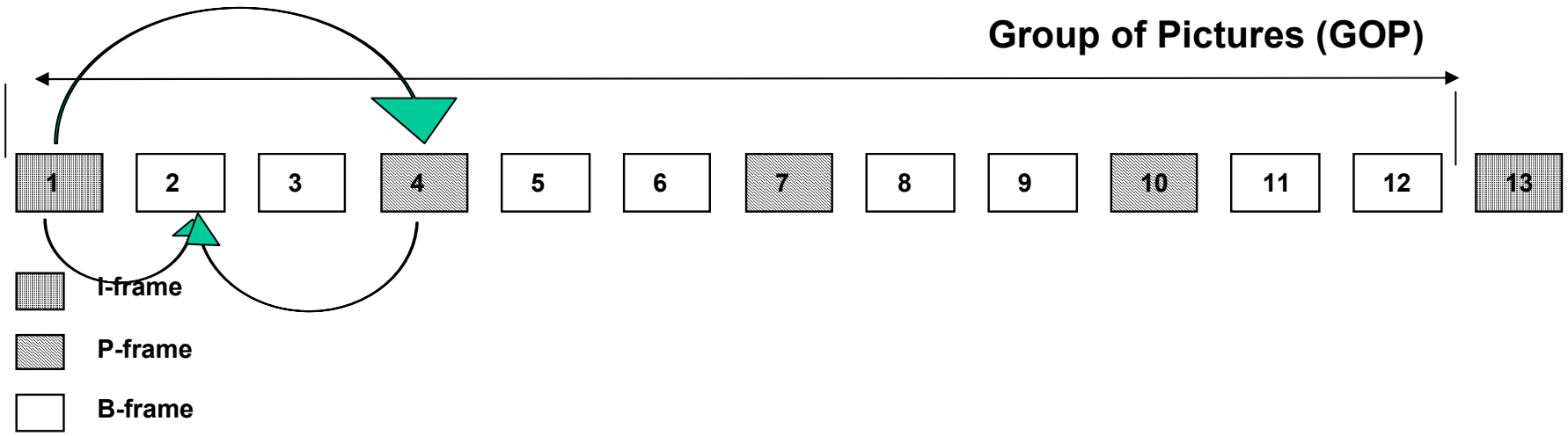
B-Frames

- Simple P-frames are fine for videoconferencing (head and shoulder shots - little movement). How about for general movies.
- Use **B-frames** - predict from past and future frames. Gives better estimate (Thus extremely high compression). Compressed heavily because they are not used as reference frames for further predictions.
- but cannot have fully real-time operation.



- B-frames have to be decoded using succeeding P- or I-frames.
- Increases coding delay. High complexity due to two motion estimations per frame.
- Need to reorder the incoming coded frames and require buffering to permit reordering for final uncompressed video stream.

Frame Reordering

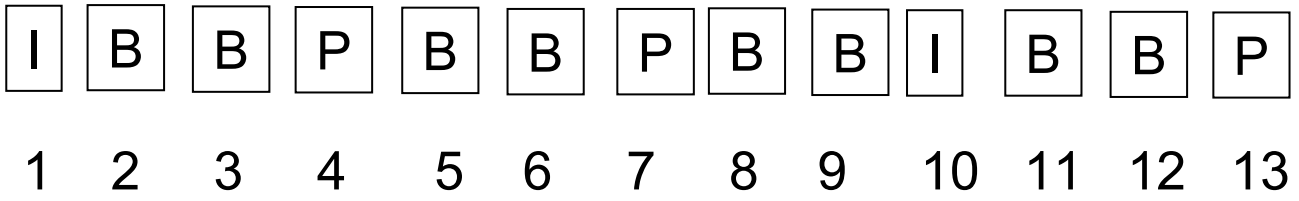


A GOP is group of pictures, that contain only one I-Frame.
GOP size = (number of frames between two I frames) + 1

Display order = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, …….. Etc.

What is the coding / decoding order? What is the buffer size?

Summary



Display order of first 13 frames are shown above.

A GOP is a group of pictures that contains only one I-Frame.

What is the GOP size?

What is the coding / decoding order?

How many frames should be kept in the buffer at any given time?

Summary

I P B frames

How are they generated/coded?

Are they used as reference frames for generating other frames?

How much compression can be tolerated?

Summary

Four different frame organisations:

1. Intra only: I I I I I I I I I I I I I ….
2. Motion compensated DPCM: I P P P P P P P P P …
3. Same as (2) but with occasional Intra frames: I P P P P P I P P P P …
4. Same as (3) but with B frames: I B B P B B I B B P B …

Which arrangement can get the highest compression?

Lowest compression?

Has the highest computational complexity?

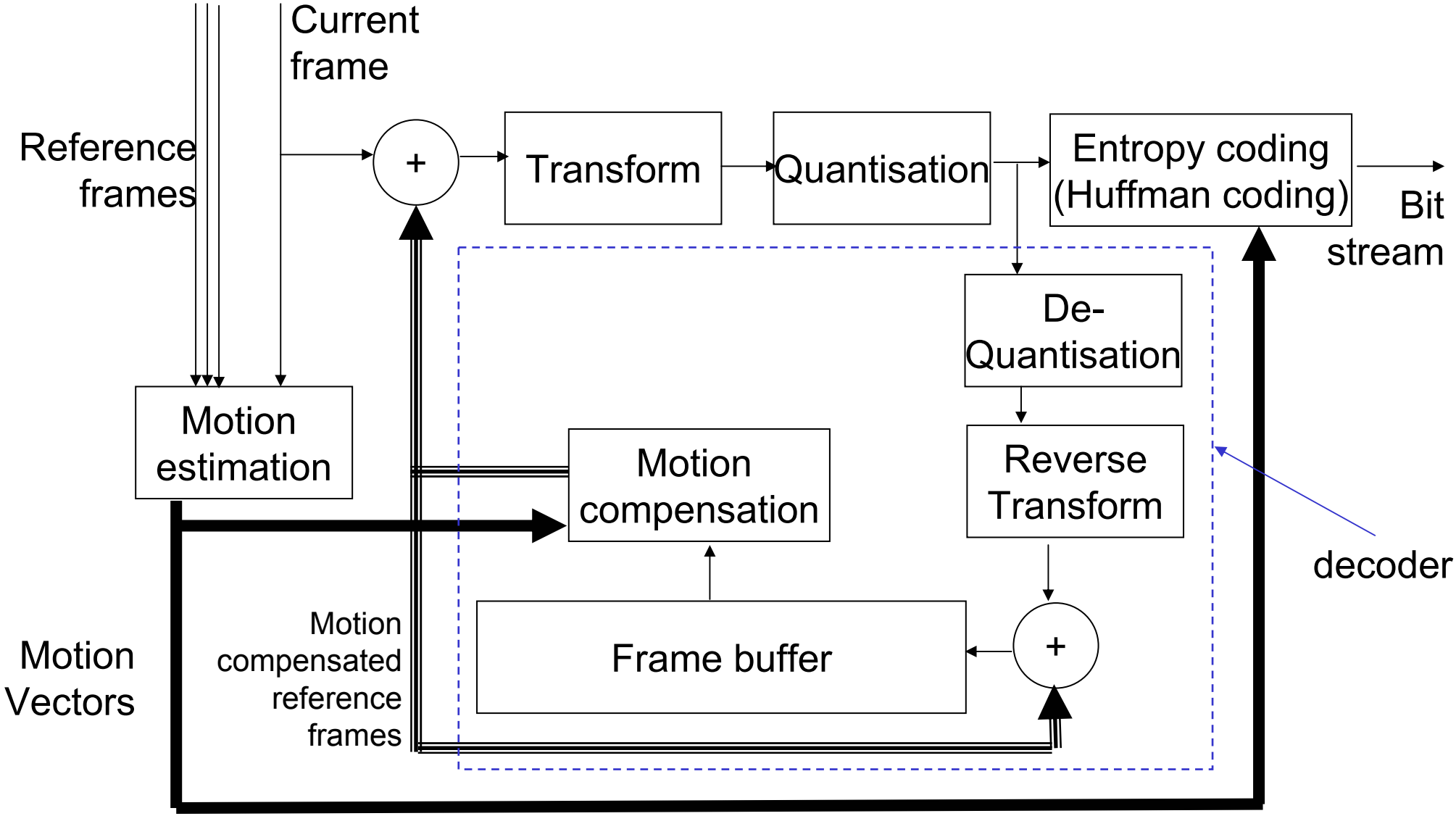
Lowest computational complexity?

Most susceptible for propagation of errors?

Least susceptible for propagation of errors?

With the lowest coding/decoding delay?

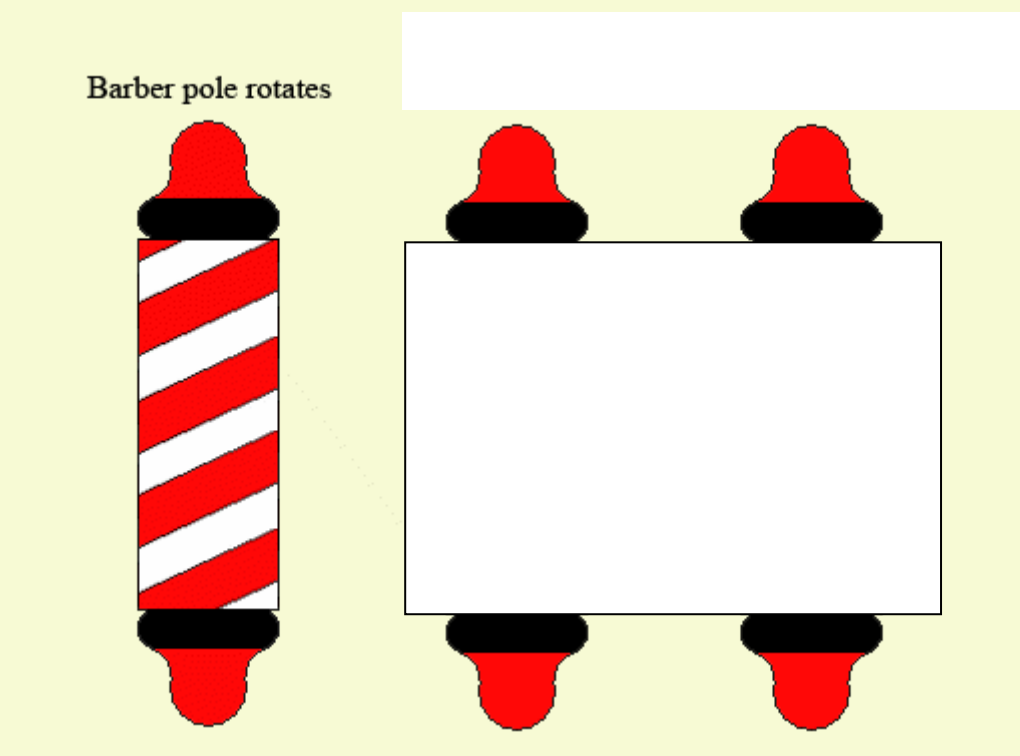
Video compression architecture (encoder of an MPEG-2 codec)



Motion-Compensated Prediction Framework

- Transform-based coding on motion-compensated prediction error (residue)
- Popular transformations: block DCT or wavelet
- Closed-loop DPCM to prevent error propagation (drifting)
- Usually block-translation motion model is employed

- All international video coding standards are based on this coding framework
  - Video teleconferencing: H.261, H.263, H.263++, H.26L/H.264
  - Video archive & play-back: MPEG-1, MPEG-2 (in DVDs) MPEG-4

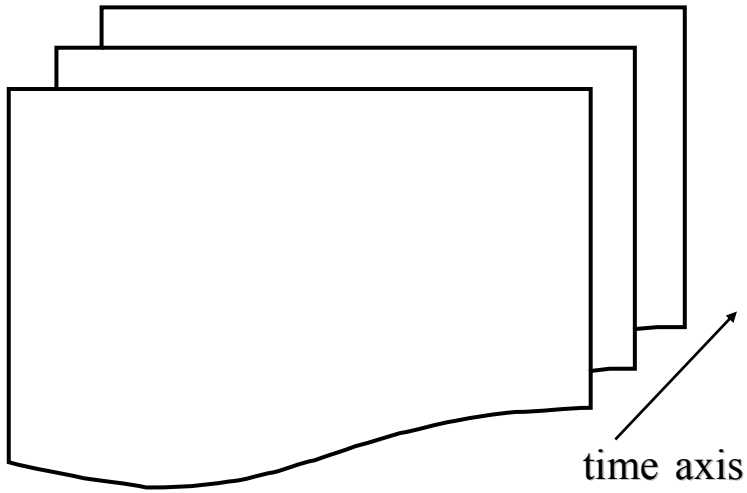

Barber pole rotates

Motion Estimation

- Goal: extract correlation between adjacent video frames to improve compression efficiency
- General problem statement:

Given the current frame $C(x, y)$
and the reference frame $R(x,y)$,
find functions $f(x, y)$ and $g(x, y)$ to
minimize

$$E = d\{C(x,y), R(f(x,y), g(x,y))\}$$



- *d* is the difference
- Practical motion model: small objects or regions moving in translational fashion

$$f(x, y) = x - dx; \quad g(x, y) = y - dy$$

- dx and dy are the amounts of displacement due to motion.
- Block-based motion estimation (BME) and compensation (BMC)

Motion Models

- Translation

$$\begin{cases} f(x,y) = x - d_x \\ g(x,y) = y - d_y \end{cases}$$

The most commonly used model

- Affine

$$\begin{cases} f(x,y) = a_{00}x + a_{01}y + d_x \\ g(x,y) = a_{10}x + a_{11}y + d_y \end{cases}$$

- Bilinear

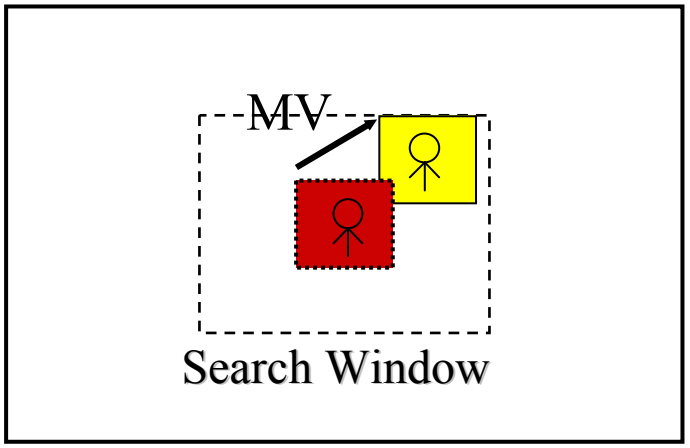
$$\begin{cases} f(x,y) = a_{00}x + a_{01}y + a_{02}xy + d_x \\ g(x,y) = a_{10}x + a_{11}y + a_{12}xy + d_y \end{cases}$$

- Perspective

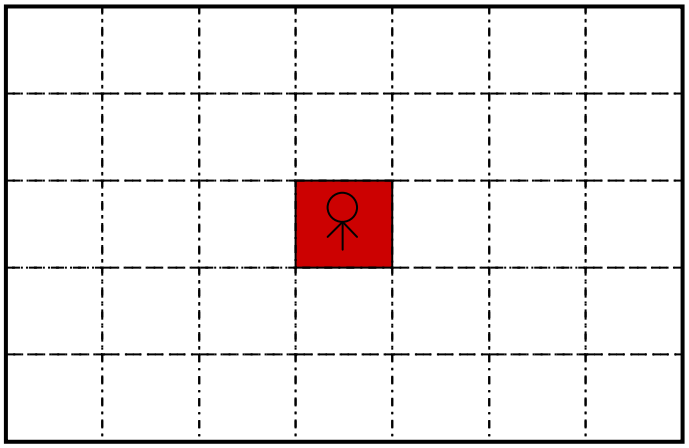
$$\begin{cases} f(x,y) = \dfrac{a_{00}x + a_{01}y + a_{02}}{a_{20}x + a_{21}y + a_{22}} \\ g(x,y) = \dfrac{a_{10}x + a_{11}y + a_{12}}{a_{20}x + a_{21}y + a_{22}} \end{cases}$$

Block Motion Estimation (BME)

- Partition current frame into small non-overlapped blocks called macro-blocks (MB)
- For each block, within a search window, find the motion vector (displacement) that minimizes a pre-defined mismatch error
- For each block, motion vector and prediction error (residue) are encoded



Reference Frame



Current Frame

BME: Error Measure

- Sum of absolute differences (L-1 Norm based)

$$SAD(dx,dy) = \sum_{dx=-w}^{w} \sum_{dy=-w}^{w} \left| \sum_{i=0}^{b-1} \sum_{j=0}^{b-1} C(x+b, y+b) - R(x+b+dx, y+b+dy) \right|$$

search range

current block

reference block

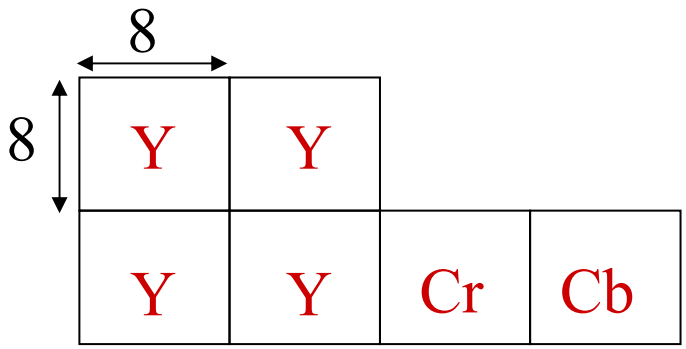
- Sum of squared errors (similar to mean-squared error) (L-2 norm based)

$$SSE(dx,dy) = \sum_{dx=-w}^{w} \sum_{dy=-w}^{w} \left| \sum_{i=0}^{b-1} \sum_{j=0}^{b-1} C(x+b, y+b) - R(x+b+dx, y+b+dy) \right|^2$$

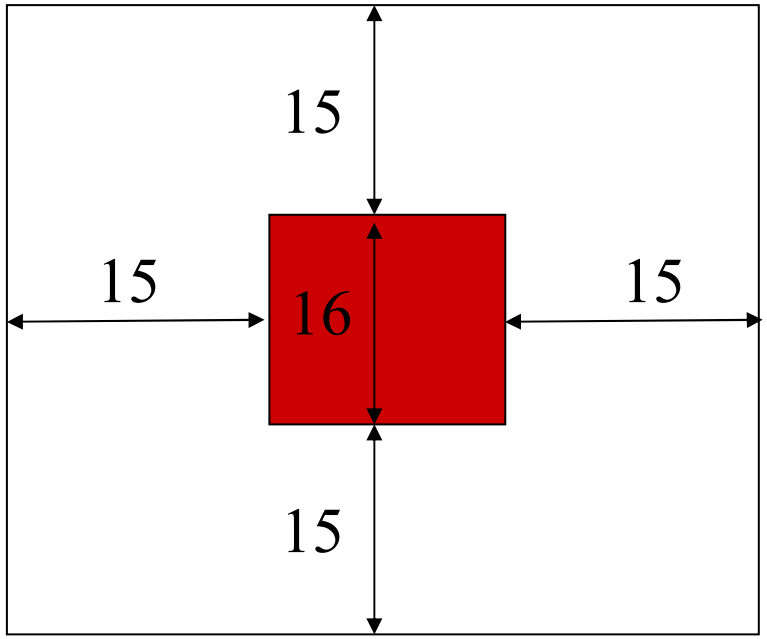
- Note:
  - Other norms, correlation measure have been tested
  - Approximately same coding performance
  - SAD is less complex for some hardware architectures

Common Settings

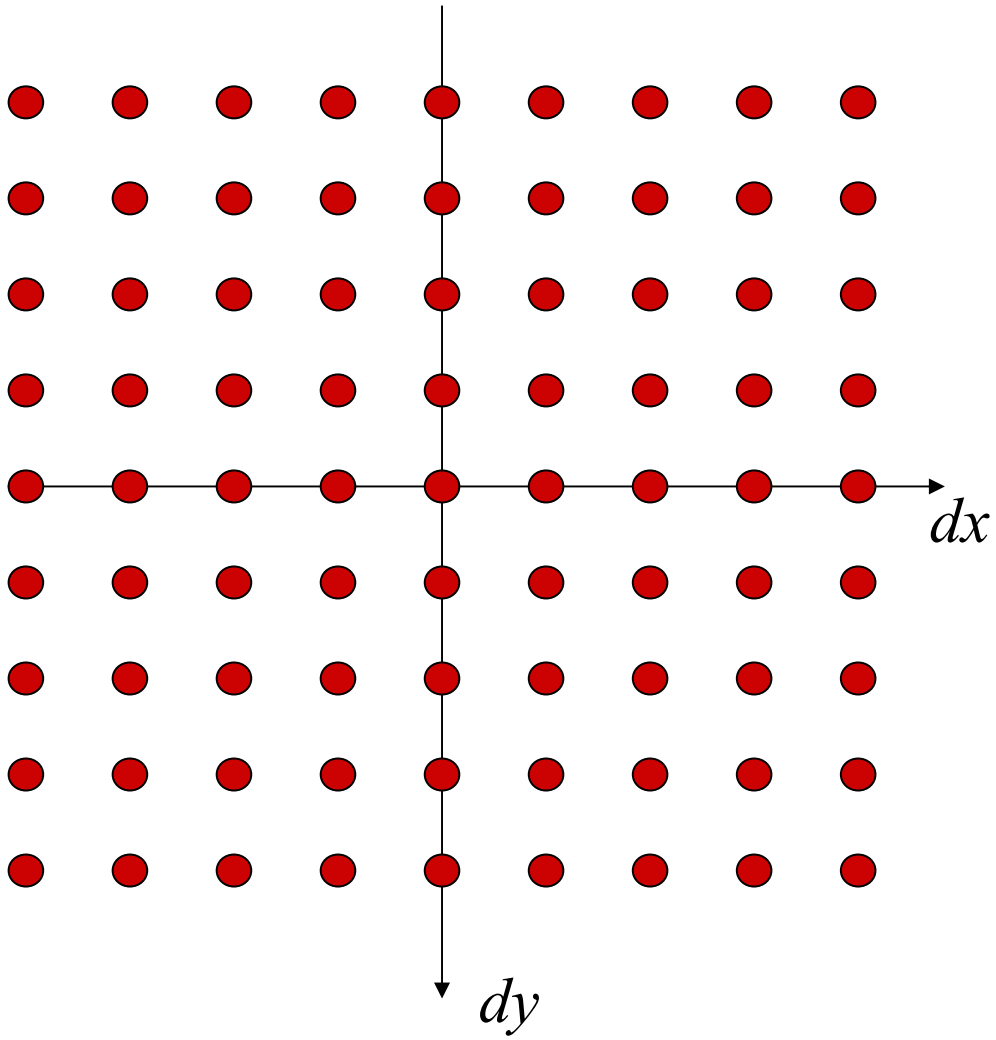
- Macro-block
  - Luminance: 16x16, four 8x8 blocks
  - Chrominance: two 8x8 blocks (in 4:2:0)
  - Motion estimation only performed for luminance



- Motion vector range
  - [-15, 15], i.e., w=15
  - How many bits?



Search Strategies 1

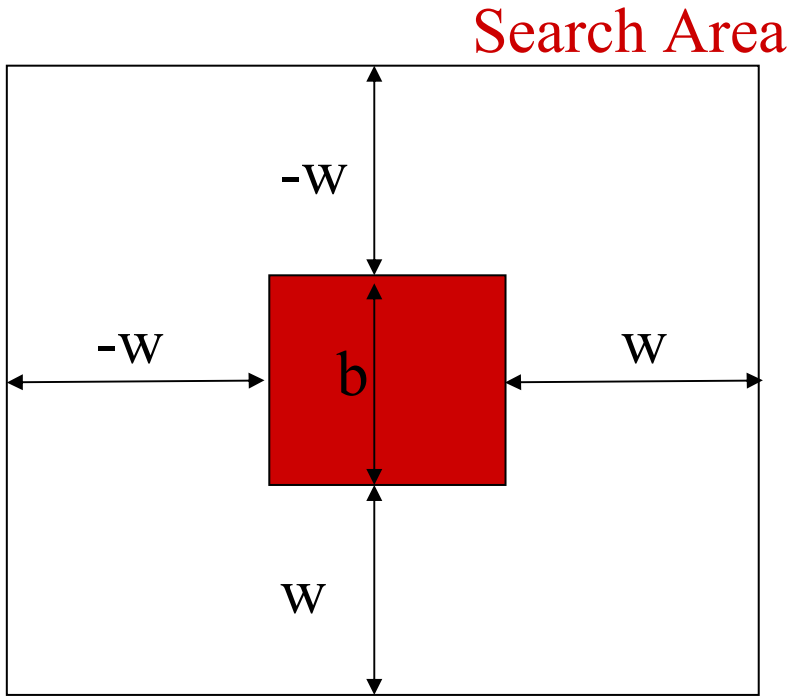


Exhaustive Search

- All possible MV candidates within the search range are investigated
- Very computationally expensive
- Optimal, parallel computable

Exhaustive search complexity

- Search area -- number of possible search positions for a pixel of the current block
  -
- Number of pixels in a block
  -
- Total number of blocks in a frame of M x N pixels.
  -



- Complexity =
  =

Exhaustive search complexity

- Complexity = $T(2w+1)^2$ search operations,
  - Where $T = MN$ Total number of pixels.
  - An operation is typically ((|subtract|) or ((subtract)^2)) and additions

- For a given frame,
  - Complexity is proportional to $(2w+1)^2$

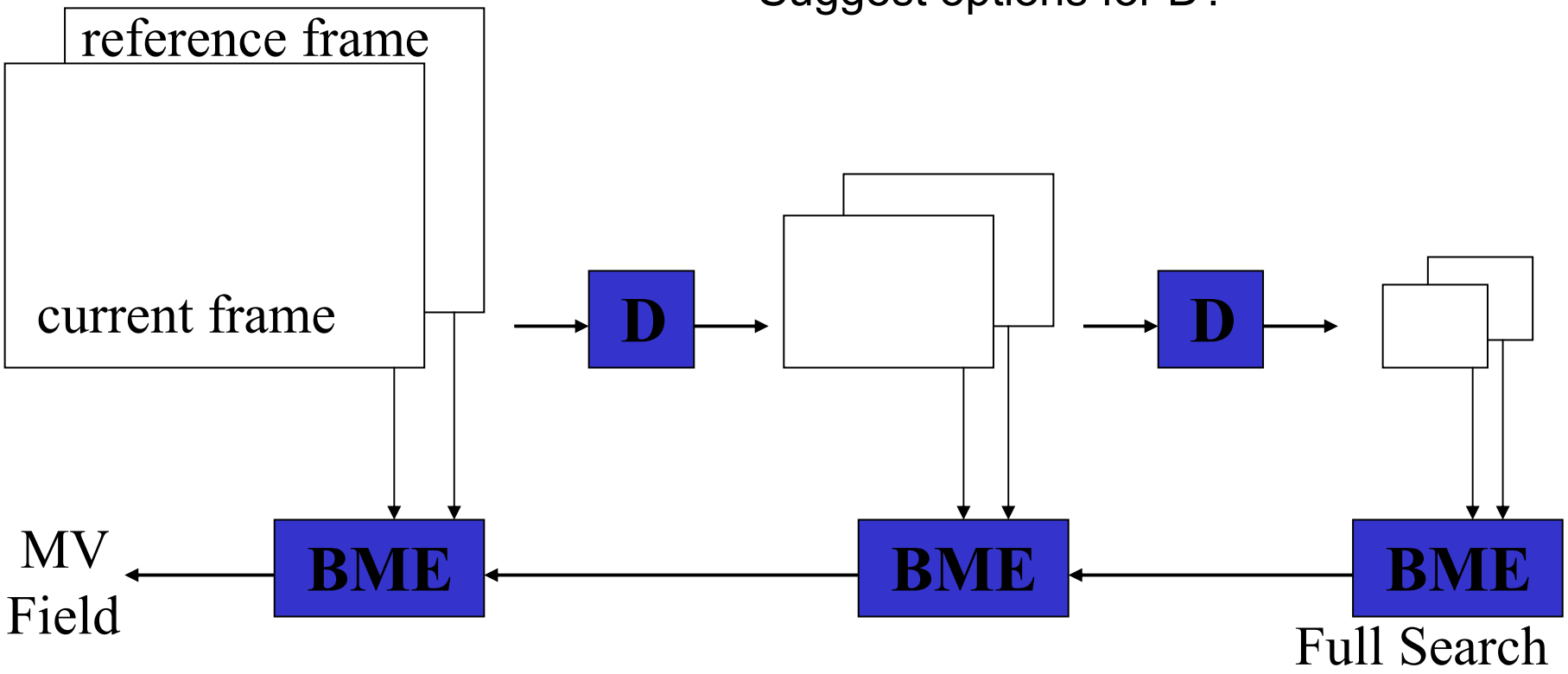
- When w=15 the complexity is 961 T.

- Can we use MRA to reduce the complexity?

Search Strategies 2

Multi-resolution or Hierarchical Search

Suggest options for D?



MRA based search complexity

- Complexity = T x $(2w+1)^2$ operations using exhaustive search
- For MRA based at each resolution level the total number of pixels are reduced by a factor of 4
- Complexity (for s levels of down sampling

$$= (2w_{-s}+1)2\,T/(4^S) + (2w_{-(s-1)}+1)^2\,T/(4^{S-1}) + \ldots\ldots + (2w_{-1}+1)^2\,T/4 + (2w_0+1)^2\,T$$

$w_{-s} = w/(4^S)$ *and*

$w_{-(s-1)} = \ldots\ldots = w_{-1} = w_0 = 1$ *to retain the same prediction accuracy as before.*

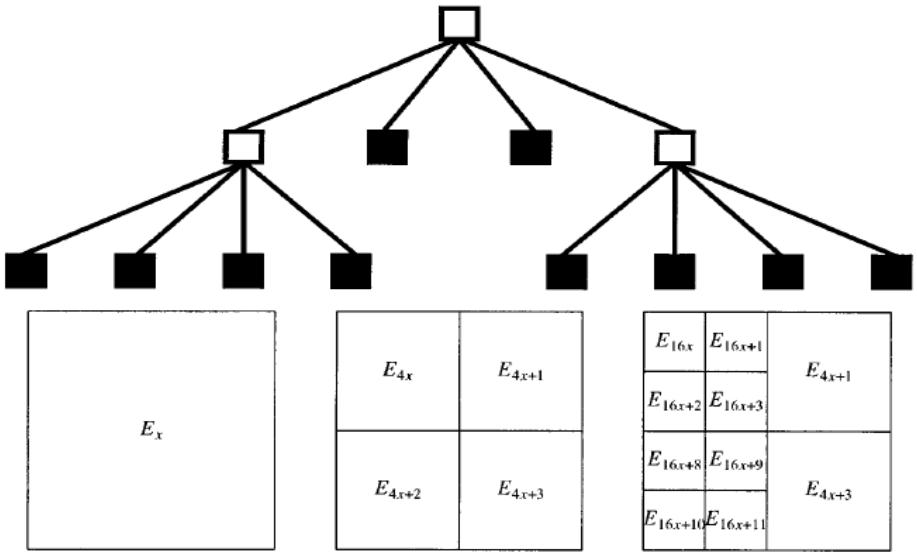
- Compute the complexity when s=2
- What are the advantages and disadvantages of this scheme?

- So far we have discussed:
  - BME with fixed block sizes [i.e., the same b]
  - Search range -w to w, where w is an integer value

- But sometimes, when the current frame is partitioned into fixed size blocks, a block can include parts from objects moving with different velocities [magnitudes and directions]. This can lead to an error of motion estimation in that block.

- Solutions:
  - Variable block sizes
  - Sub pixel accuracy motion vectors

Variable block size BME



Subdivide each macro-block into 4 blocks.

Then perform BME for each sub block.

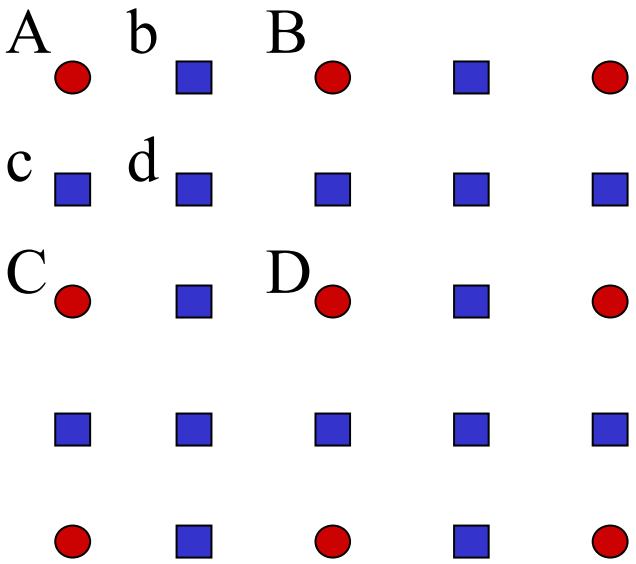
Perform this until a predefined minimum block size is achieved and create a full decomposition quad tree.

Then use rate [bits required for motion vectors] and distortion [the prediction error] to merge nodes of the quad tree by scanning the tree from bottom to top.

Sub-Pixel Motion Estimation

- Sub-pixel motion vector resolution
- Use linear/bilinear interpolation to fill in sub-pixels

- Trade-offs: motion accuracy versus MV bit-rate and complexity increase
- H.264 video codec uses down to 1/4-accuracy, maybe even 1/8



integer pixel

half pixel

$b = \text{round}[(A+B)/2]$

$c = \text{round}[(A+C)/2]$

$d = \text{round}[(A+B+C+D)/4]$

Image / Video Codec Performance Evaluation

The performance of codecs is usually evaluated using the rate-distortion (R-D) measurements.

1. Rate: This is a measure of the amount of compression. The most commonly used metric is the compression ratio (CR) .

There are many way to measure the compression ratio. A generic way to define it is as a ratio of "Original parameters" and "new parameters" for the compressed image/video.

$$CR = \frac{\text{Parameter value of Uncompressed data}}{\text{the value of the same parameter of Compressed data}}$$

Some parameters: data rates, total bits (file size)

data rates = bits per pixel (bpp) for images.
bits per second (bps) for video.

Rate-distortion measures

Distortion: In R-D analysis, distortion metrics measure the difference between the original image and the compressed image.

The simplest method is to compare two images/video pixel by pixel and compute the mean square error (mse). For an original image I_1 with WxH pixels and the decoded image I_2 the MSE is:

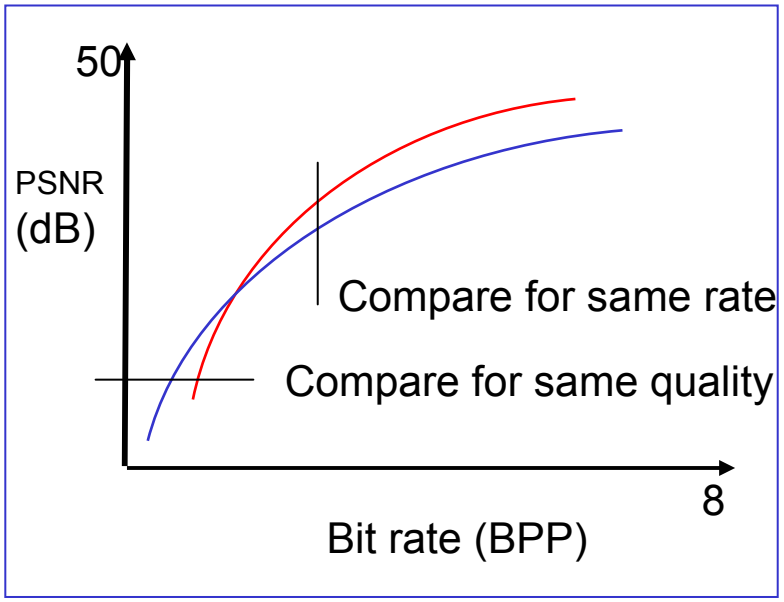
$$MSE(I_1, I_2) = \underbrace{\frac{1}{W \times H}}_{\text{Mean}} \sum_{i=1}^{W} \sum_{j=1}^{H} \underbrace{\left(I_2(i,j) - I_1(i,j)\right)^2}_{\text{(squared (error))}}$$

As usual we represent this as a signal power to noise power ratio.

For visual applications, we define Peak Signal to Noise Ratio (PSNR).

$$PSNR = 10 \log_{10} \left(\frac{(\text{Peak signal value})^2}{MSE} \right)$$

For an N-bit image the peak signal value is 2^N-1.
e.g. For an 8 bit image the peak signal value is



What is the MSE and PSNR for a lossless coded image?

Is MSE or PSNR, a good metric for measuring image quality?

Desirable features of image and video codecs

- Good coding performances.
  - i.e., Rate-distortion performances.
  - e.g., [ ] rmse, [ ] r PSNR, [ ] bit rate, [ ] compression ratio.
- Speed – fast (low complexity) coders and decoders.
- Robustness to transmission errors (e.g., due to packet losses)
- Low buffer requirements.
- Availability of both software and hardware based implementations.
- Scalability. – ability to decode with parameters different to the coding parameters.

Multimedia usage

Various network bandwidths

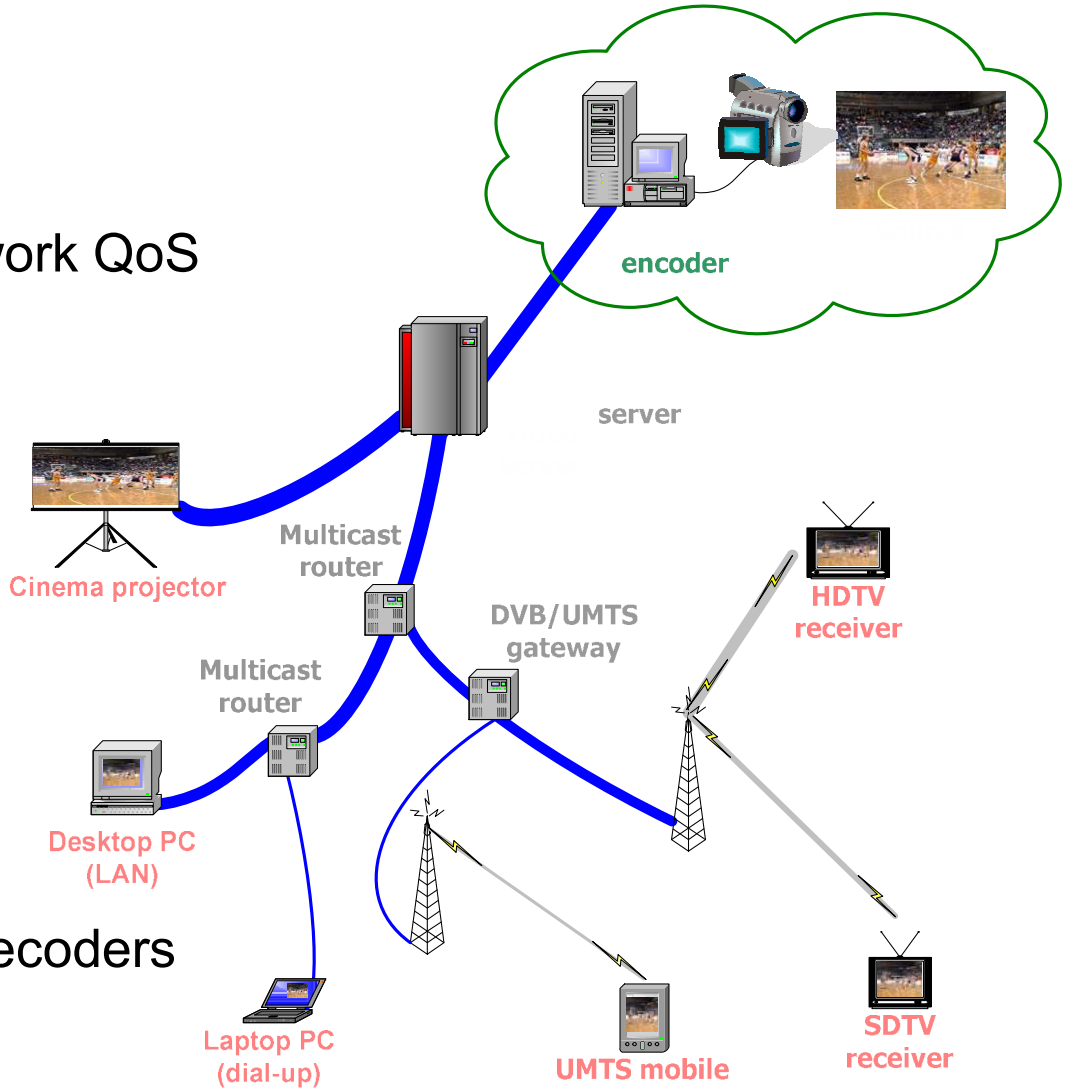
Various network QoS

Various transmission medium (noisy channels) – packet losses, bit wise errors

Display resolutions

Memory and power availability at decoders

Different usage requirements.



Visual contents needs to be adapted to cater these needs

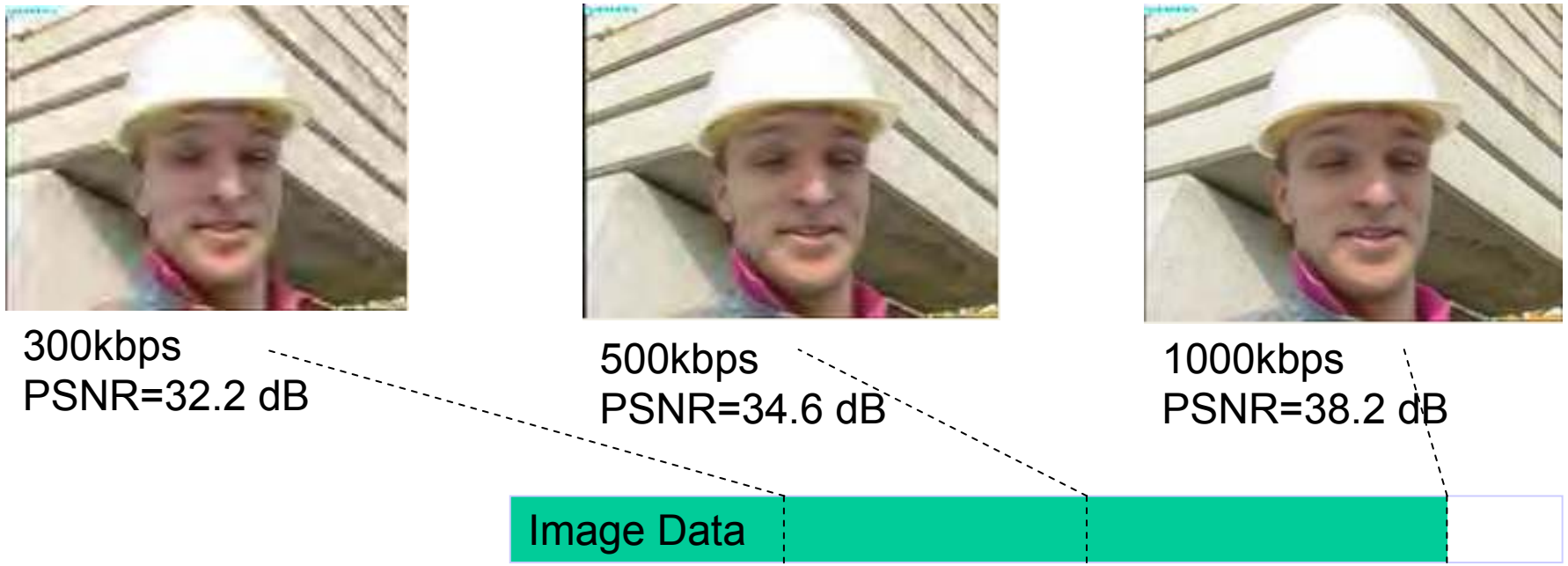
Scalable coding

- Objective:- Encode once for the highest resolution at the highest quality and then decode in many ways.

- Types of scalabilities
  - Quality scalability (also known as bit rate scalability, PSNR scalability)
  - Resolution scalability (also known as spatial scalability)
  - Temporal scalability (also known as frame rate scalability)

Quality Scalability

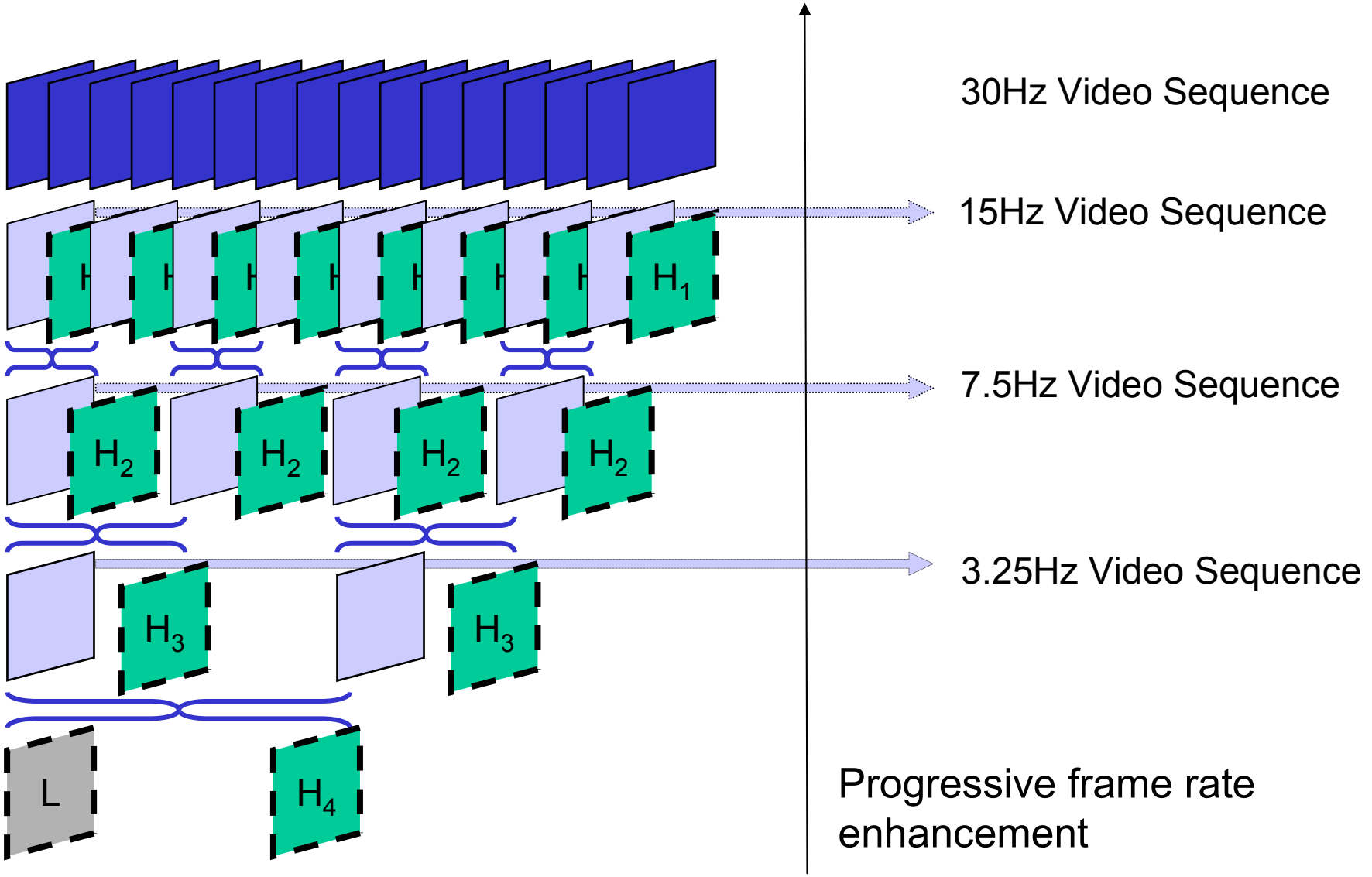
- Progressive quality increment



Resolution Scalability

Progressive resolution enhancement

Temporal Scalability



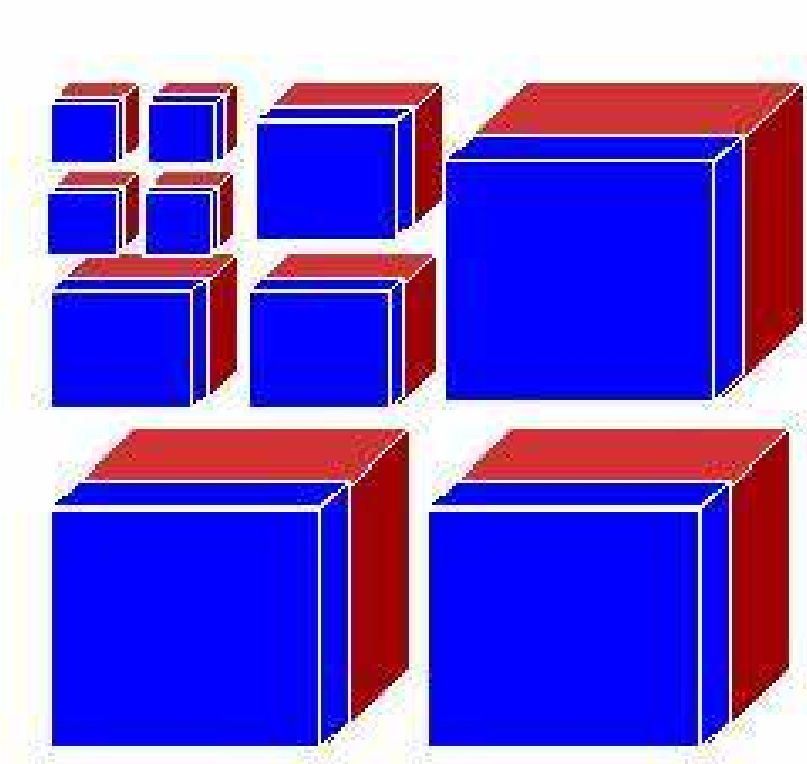
Scalability for 1D signals

- Use 1D wavelet transforms
- Resolution scalability – by discarding high pass sub bands at different wavelet decomposition levels.
- Quality scalability – by discarding bit planes
- Refer to MATLAB example and tutorial question.
- Can combine bit plane and sub band discarding to achieve joint resolution-quality scalability.

Scalability for images

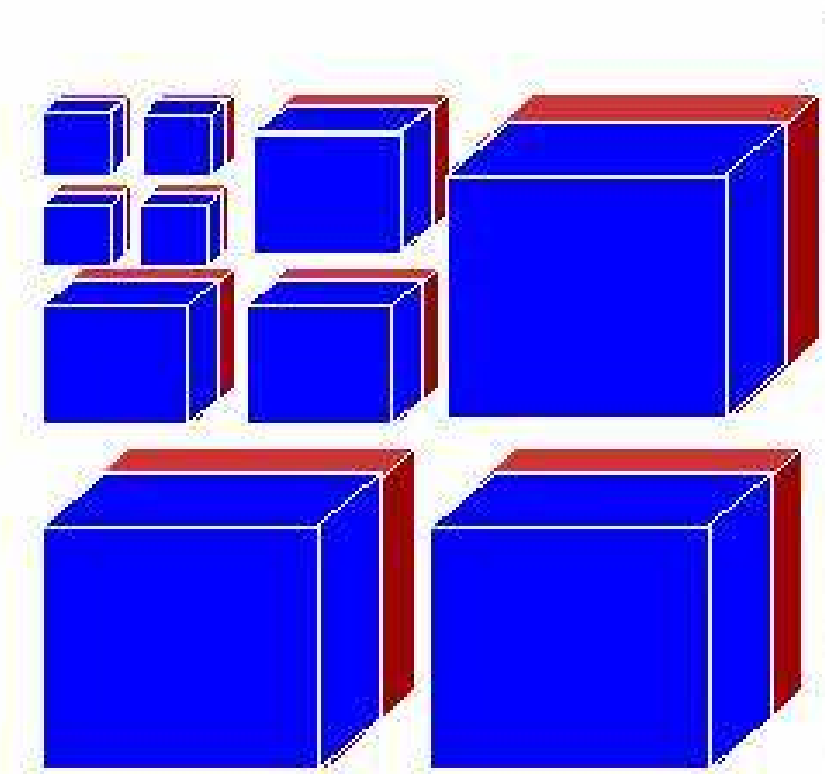
- Use 2D wavelet transforms
- Resolution scalability – by discarding high pass sub bands at different wavelet decomposition levels.
- Quality scalability – by discarding bit planes
- Can combine bit plane and sub band discarding to achieve joint resolution-quality scalability.

Quality scalability for Images



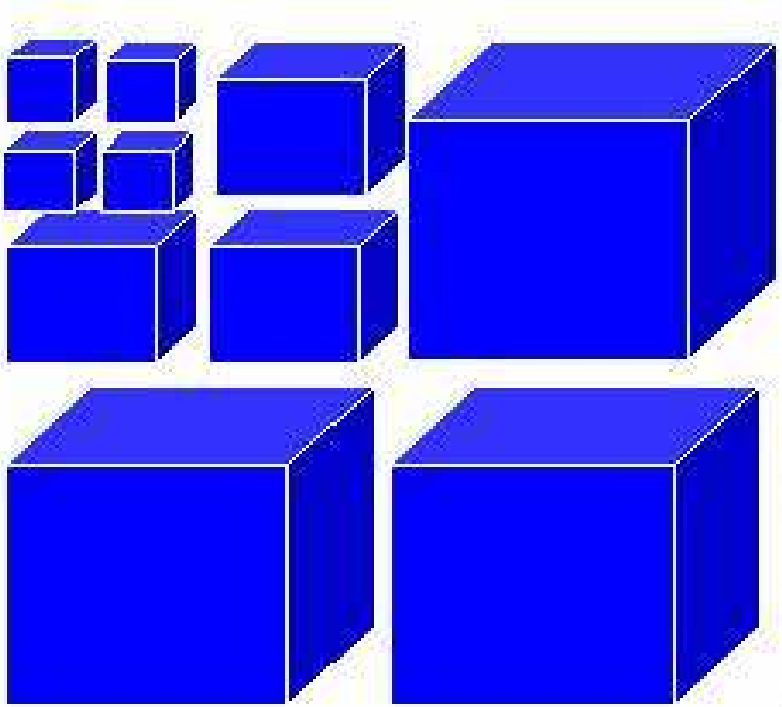
Choose only the most significant bit planes from wavelet coefficients. (heavy quantisations)

Quality scalability for Images



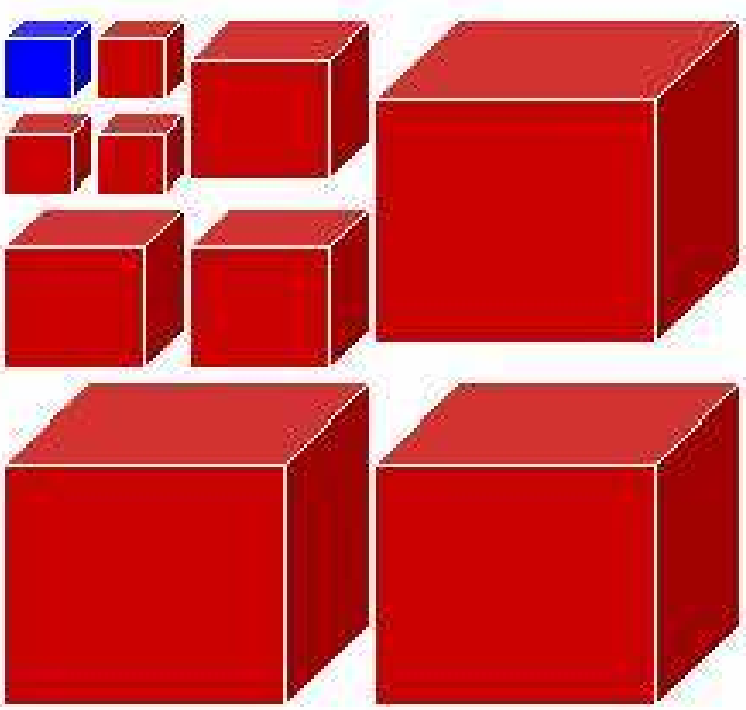
Include more bit planes to the previous decoded image (medium quantisation)

Quality scalability for Images



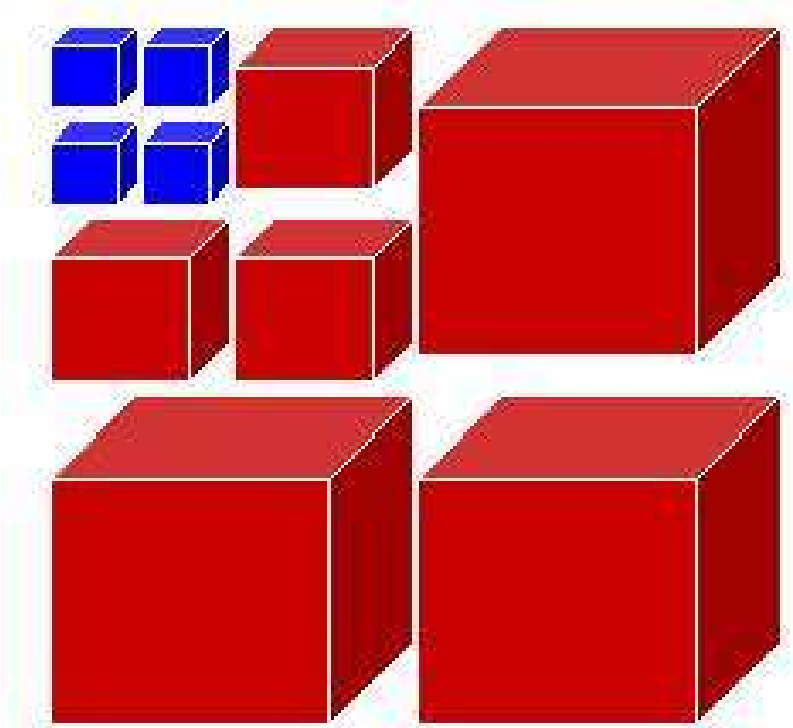
Full quality using all bit planes

Resolution scalability for Images



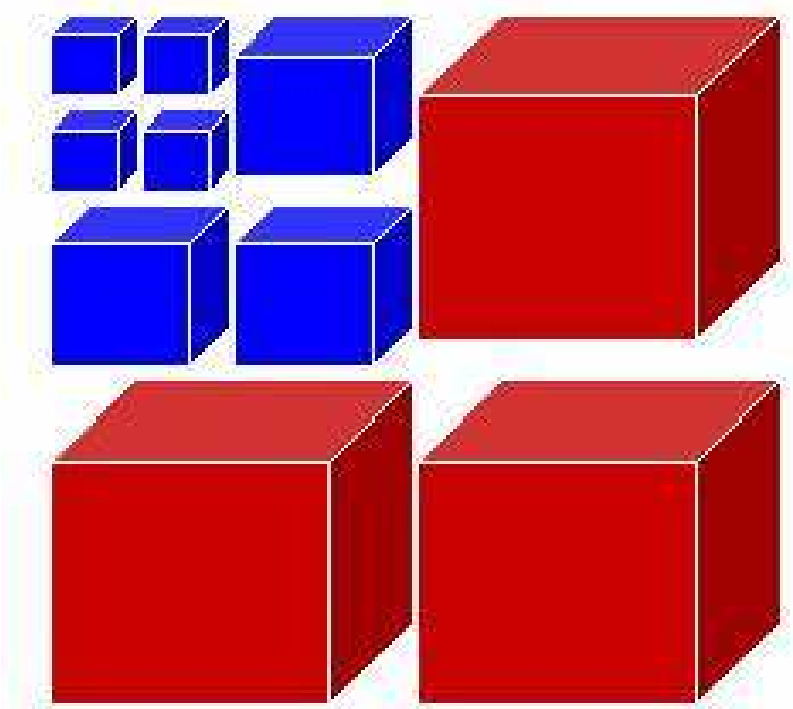
1/64th of the full resolution.
By keeping only the LL sub band after 3 levels of wavelet decomposition

Resolution scalability for Images



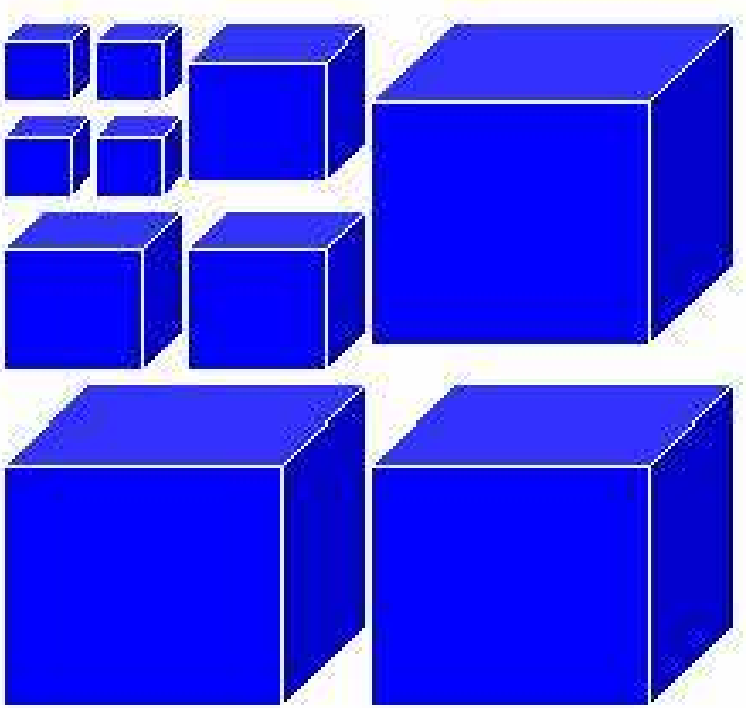
1/16th of the full resolution.
By keeping only the LL sub band after 2 levels of wavelet decomposition

Resolution scalability for Images



1/4th of the full resolution.
By keeping only the LL sub band after 1 level of wavelet decomposition

Resolution scalability for Images



Scalability for video

- Use 3D wavelet transforms
- Resolution scalability – by discarding high pass sub bands at different wavelet decomposition levels.
- Quality scalability – by discarding bit planes
- Temporal scalability – by discarding high pass temporal sub bands at different wavelet decomposition levels.
- We can combine bit plane and sub band discarding to achieve joint temporal-resolution-quality scalability.

Scalability for video

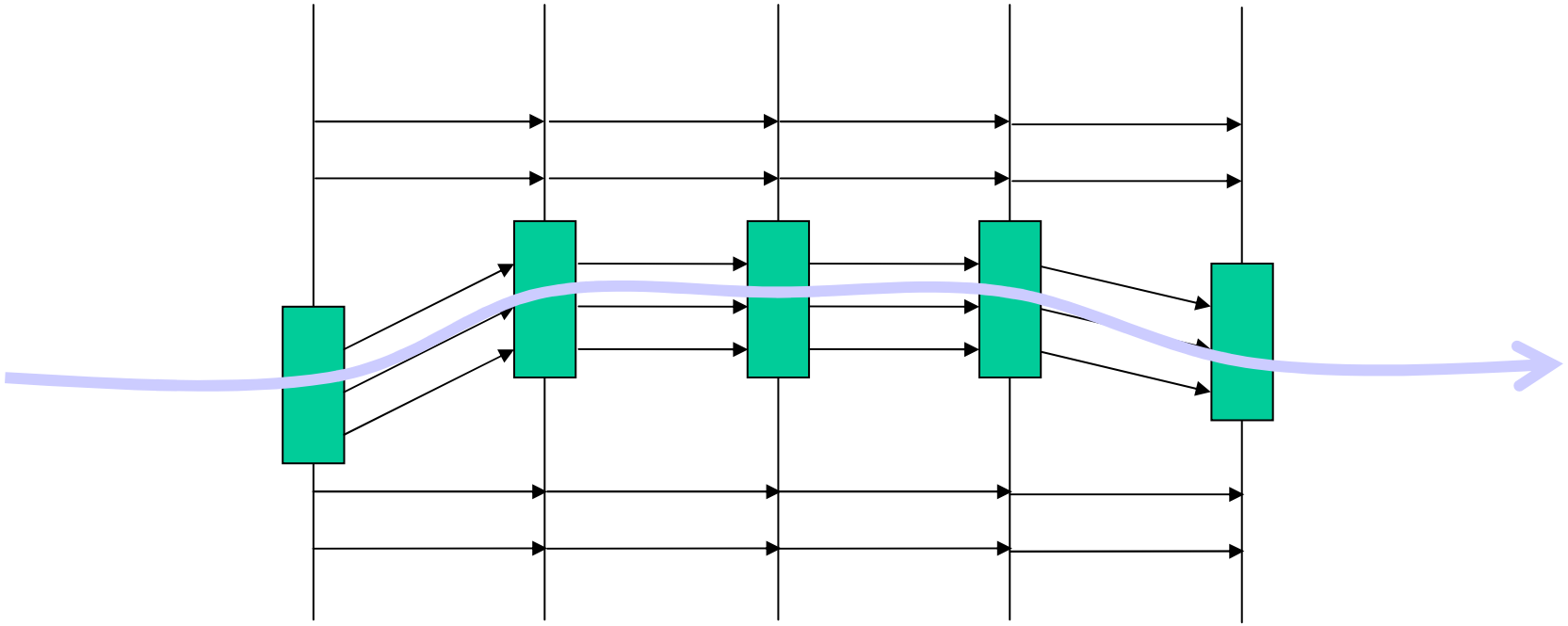
- Use 3D wavelet transforms - How to realise 3D wavelet transforms?

- Separable approach
  - temporal transform + 2D transform (t+2D)
  - 2D transform + temporal transform (2D+t)

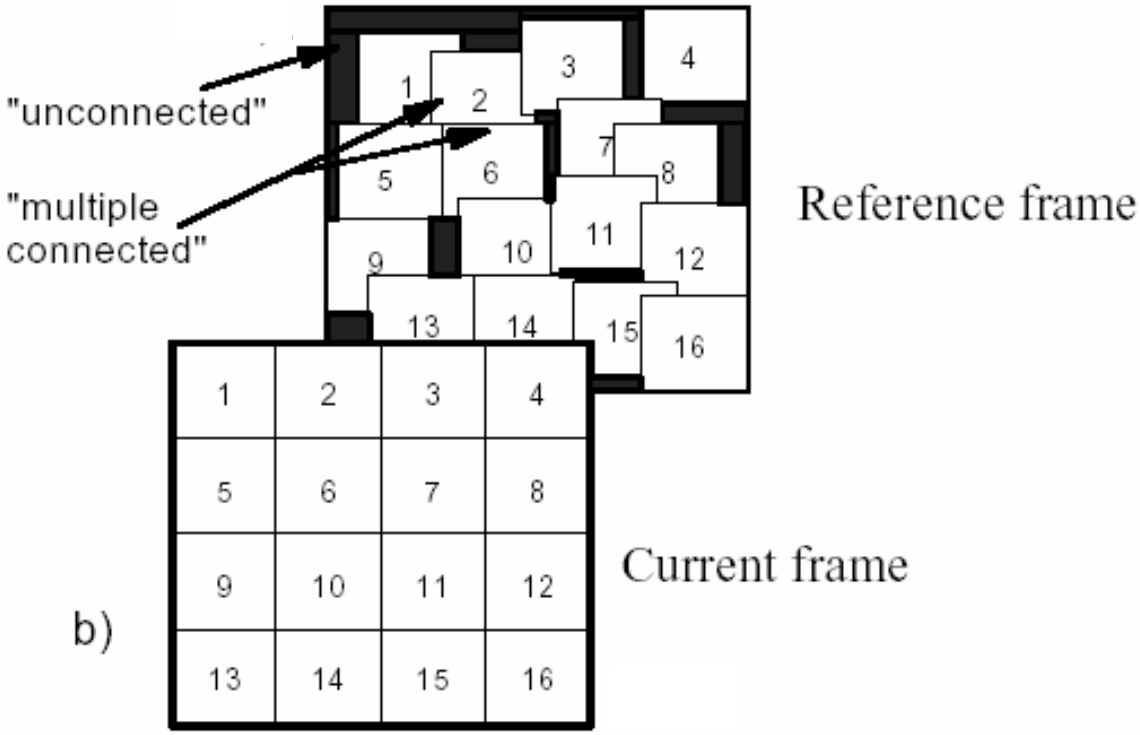
- How to do the temporal transform.
  - Can use any wavelet transform
  - But there is problem with motion
  - Therefore it has to take the motion into account
  - Motion compensated temporal filtering (MCTF)

MCTF

MCTF = Motion Compensated Temporal Filtering



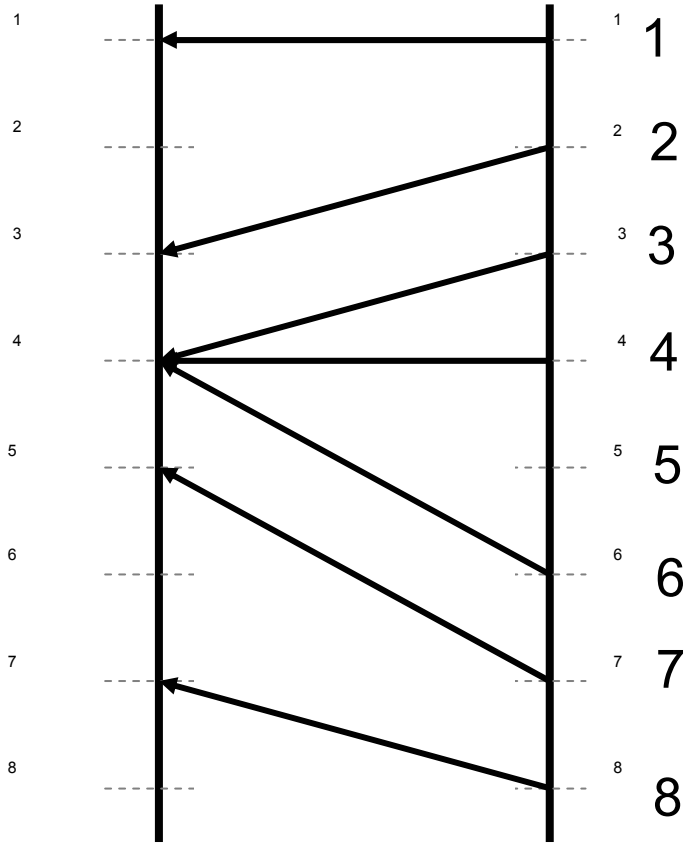
Temporal filtering should follow the motion trajectory based on motion estimation.



Three Types of connectivity of pixels in the reference frame to the current frame when motion estimation is concerned:

1. Unconnected
2. Single connected (1-to-1 matching)
3. Multiple connected (1-to-manymatching)

Reference Frame (R) A B Current Frame (C)



What are the connectivity for each of pixels on the reference frame?

Scalability for video

- Now we can use lifting by considering the connectivity.

- P lifting steps make the current frame a high pass frame (Haar transform) - Motion compensated temporal lifting

Scalability for video

- U lifting steps make the reference frame a low pass frame (Haar transform) - Motion compensated temporal lifting

For single connected pixels:

For multiple connected pixels:

For unconnected pixels:

t+2D for video