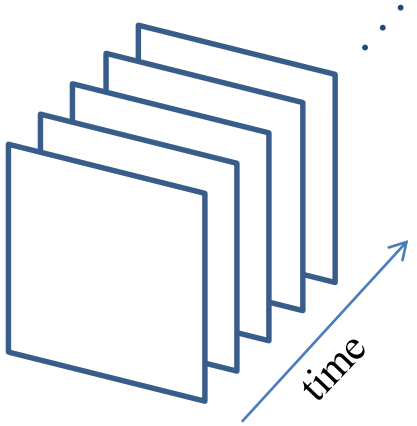
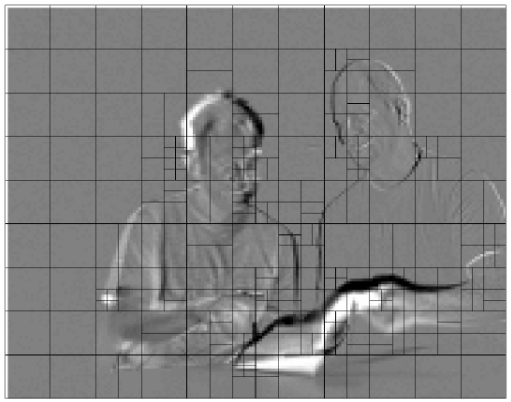
Basic Video Compression Techniques

- Introduction to Video Compression
- Video Compression with Motion Compensation
- Search for Motion Vectors
- H.261
- H.263

Introduction to Video Compression

- A video consists of a time-ordered sequence of frames, i.e., images.
- An obvious solution to video compression would be predictive coding based on previous frames.
- Compression proceeds by subtracting images: subtract in time order and code the residual error.
- It can be done even better by searching for just the right parts of the image to subtract from the previous frame.





diff image
白色代表有motion

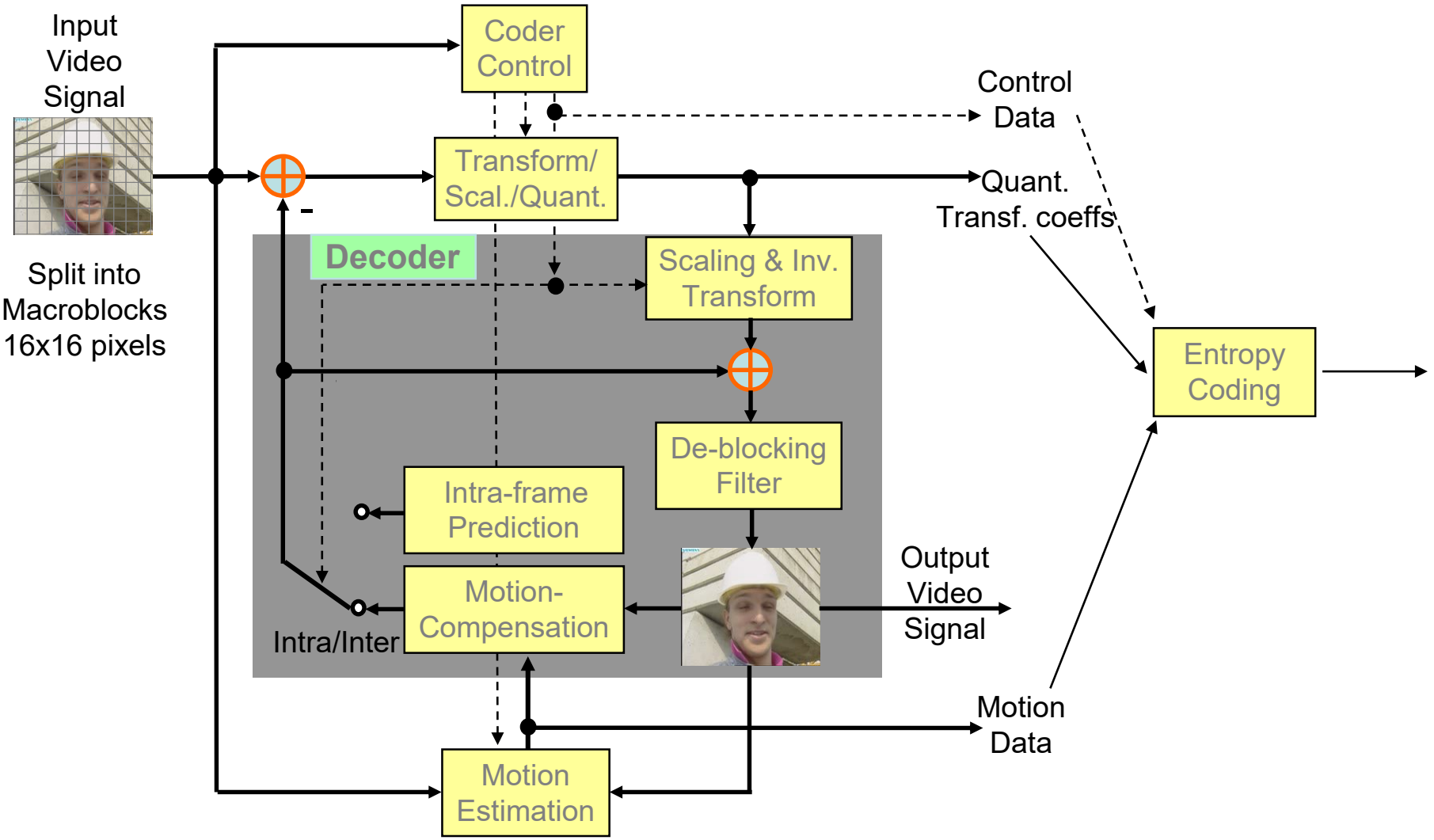
Video Compression with Motion Compensation

- Consecutive frames in a video are similar
- **Temporal redundancy** is exploited so that each video frame need not be coded independently as a new image.
- The difference between the current frame and the reference frame(s) in the sequence will be coded — small values and low entropy, good for compression.

找出物体的運動

- Steps of video compression based on ***Motion Compensation*** *(MC)*
  1. Motion Estimation (motion vector search)
  2. MC-based Prediction
  3. Compute the prediction error, i.e., the difference

Basic Coding Structure



↳ video 和 JPEG 這不太一樣的地方，其他的精神一樣

Motion Estimation

- Each image is divided into *macroblocks* of size N x N.
  - Default N = 16
  - Normally, ME is only performed on luminance images
- Motion compensation
  - Performed at the macroblock level
  - The current frame is referred to as *target frame*.
  - A match is sought between the macroblock in the Target Frame and the most similar macroblock in previous and/or future frame(s) referred to as *reference frame*(*s*).
  - The displacement of the reference macroblock to the target macroblock is called a *motion vector* $\mathbf{MV}$.

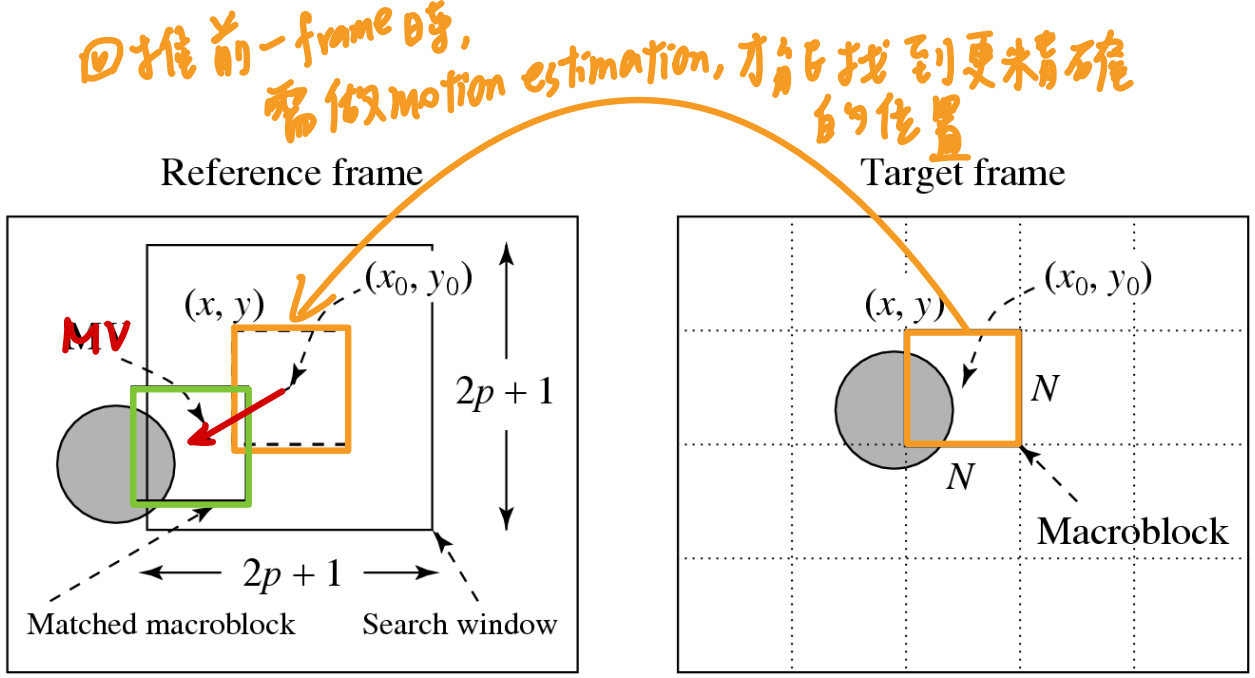


Fig. 10.1: Macroblocks and Motion Vector in Video Compression.

- Figure 10.1 shows the case of forward prediction in which the reference frame is the previous frame.
- MV search is usually limited to a small immediate neighborhood — both horizontal and vertical displacements in the range $[-p, p]$.
- Size of the search window in this case is $(2p + 1) \times (2p + 1)$.

- The difference between two macroblocks can be measured by the *mean absolute difference* (MAD) or sum of *absolute difference* (SAD)

$$MAD(i,j) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1}\left|C(x+k,y+l) - R(x+i+k,y+j+l)\right| \quad (10.1)$$

N — size of the macroblock,
k and l — indices for pixels in the macroblock,
i and j — horizontal and vertical displacements,
$C(x+k, y+l)$ — pixels in macroblock in Target frame,
$R(x+i+k, y+j+l)$ — pixels in macroblock in Reference frame.

- The goal of the search is to find a vector (i, j) as the motion vector $\mathbf{MV} = (\mathbf{u}, \mathbf{v})$ such that $MAD(i, j)$ is minimum

$$(u,v) = \left[(i,j) \mid MAD(i,j)\right] \text{ is minimum, } i \in [-p,p],\ \mathrm{j} \in [-p,p]\Big] \quad (10.2)$$

Exhaustive Search

- Search the whole $(2p + 1)$ x $(2p + 1)$ window in the reference frame (also referred to as full search).
- In each step, a macroblock centered at a pixel within the window is compared to the macroblock in the target frame pixel by pixel and their respective *MAD* is computed using (10.1).
- The vector (i, j) that offers the least MAD is designated as the $\mathbf{MV}$ (u, v) for the macroblock in the target frame.
- Computationally expensive — assuming each pixel comparison requires three operations (subtraction, absolute value, addition), the cost for obtaining a motion vector for a single macroblock is $(2p + 1) \cdot (2p + 1) \cdot N^2 \cdot 3 \Rightarrow O(p^2 N^2)$.

- *If* p=16, N=16, and image size=720x480, a total of 29.89x10^9 operations is required

PROCEDURE Motion-vector:exhaustive-search

```
begin
    min_MAD = A LARGE NUMBER;          /* Initialization */
    for i = −p to p
        for j = −p to p
        {
            curr_MAD = MAD(i, j);
            if curr_MAD < min_MAD
            {
                min_MAD = curr_MAD;
                u = i;      /* Get the coordinates for MV. */
                v = j;
            }
        }
end
```

2D Logarithmic Search

- A cheaper, suboptimal but effective solution
- Involve an iterative procedure akin to binary search
- Initially only nine locations in the search window are used as seeds for a MAD-based search; they are marked ‘1’ in the figure.
- After the one that yields the minimum MAD is located, move the center of the new search region to it and reduce the step-size (“offset”) to half.
- In the next iteration, the nine new locations are marked ‘2’ and so on.

2D Logarithmic Search for Motion Vectors.

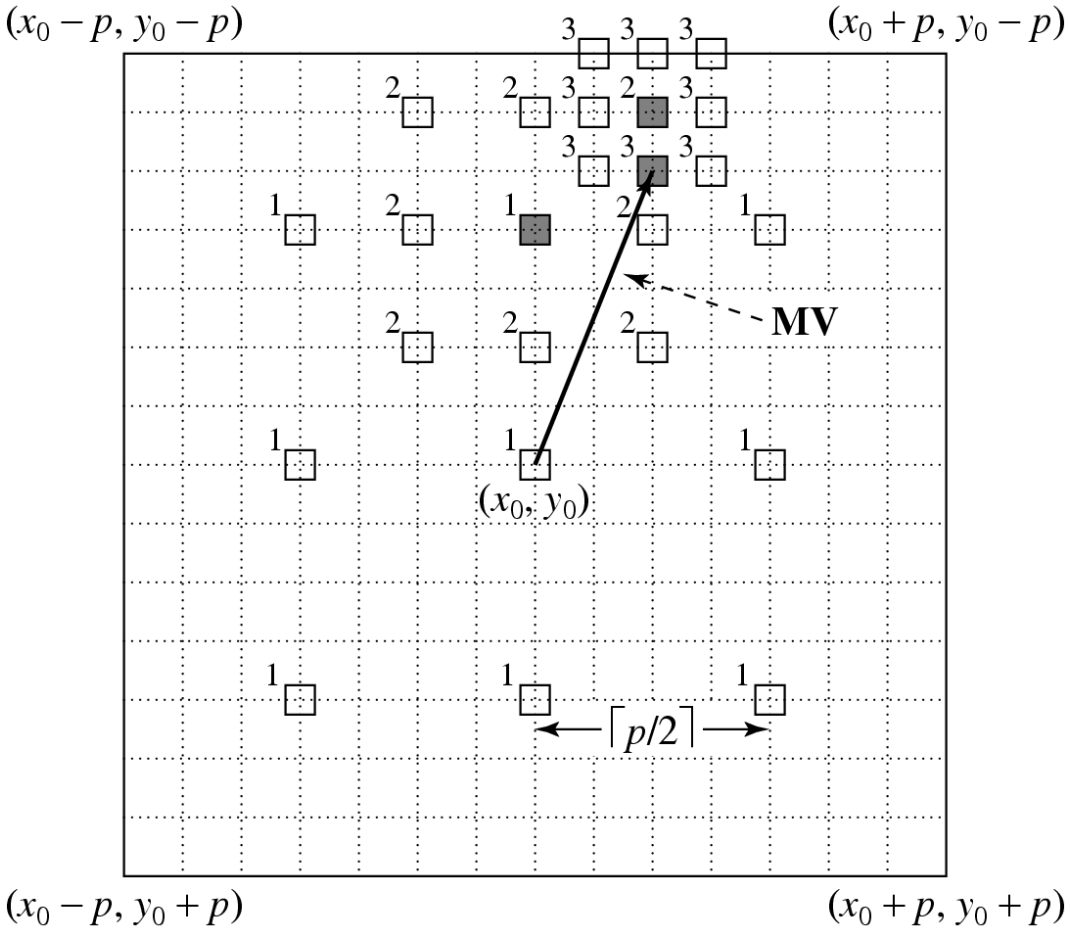


Fig. 10.2

PROCEDURE Motion-vector:2D-logarithmic-search

begin

offset = $\lceil \frac{p}{2} \rceil$;

Specify nine macroblocks within the search window in the reference frame, they are centered at (x_0, y_0) and separated by offset horizontally and/or vertically;

while last ≠ TRUE

{

Find one of the nine specified macroblocks that yields minimum MAD; if offset = 1 then last = TRUE;

offset = ⌈ offset/2 ⌉;

Form a search region with the new offset and new center found;

}

end

- Number of macrobloacks to compare:

  $9(\lceil \log_2 p \rceil + 1) \longrightarrow 8(\lceil \log_2 p \rceil + 1),$

  since the comparison that yields the last iteration can be reused.

- Using the same example as in the previous subsection, the total number of operations per second is dropped to:

$$\begin{aligned} OPS\_per\_second &= \left(8 \cdot (\log_2 p + 1) + 1\right) \cdot N^2 \cdot 3 \cdot \frac{720 \times 480}{N \cdot N} \cdot 30 \\ &= \left(8 \cdot \log_2 15 + 9\right) \times 16^2 \times 3 \times \frac{720 \times 480}{16 \times 16} \times 30 \\ &\approx 1.25 \times 10^9 \end{aligned}$$

Hierarchical Search

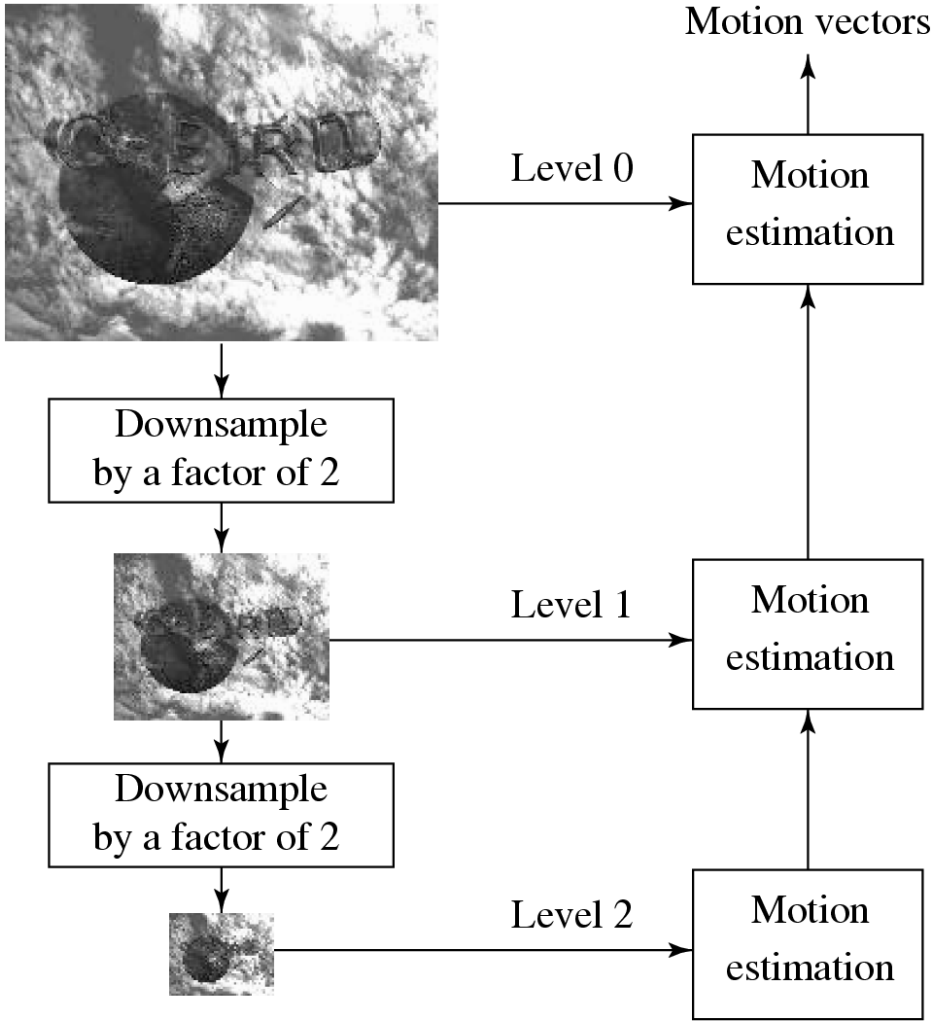
- The search can benefit from a hierarchical (multiresolution) approach in which initial estimation of the motion vector can be obtained from images with a significantly reduced resolution.

$\Rightarrow$ Construct a multi-level image structure, perform a motion estimation at the coarsest level, and pass the resulting motion estimates to the next level as initial estimate.

- Figure 10.3 shows a three-level hierarchical search in which the original image is at Level 0, images at Levels 1 and 2 are obtained by down-sampling from the previous levels by a factor of 2, and the initial search is conducted at Level 2.

- Since the size of the macroblock is smaller and p can also be proportionally reduced, the number of operations required is greatly reduced.

A Three-level Hierarchical Search for Motion Vectors.



Hierarchical Search (cont'd)

- Given the estimated motion vector (u^k, v^k) at Level k, a 3 x 3 neighborhood centered at $(2 \cdot u^k, 2 \cdot v^k)$ at Level $k - 1$ is searched for the refined motion vector.
- The refinement is such that at Level $k - 1$ the motion vector (u^{k-1}, v^{k-1}) satisfies

$$2u^k - 1 \le u^{k-1} \le 2u^k + 1$$

$$2v^k - 1 \le v^{k-1} \le 2v^k + 1$$

- Let (x^k_0, y^k_0) denote the center of the macroblock at Level k in the target frame. The procedure for hierarchical motion vector search for the macroblock centered at (x^0_0, y^0_0) in the target frame is outlined in Procedure 10.3.

PROCEDURE Motion-vector:hierarchical-search

begin

// Get macroblock center position at the lowest resolution Level k

$x^k_0 = x^0_0 / 2^k$; $y^k_0 = y^0_0 / 2^k$;

Use sequential (or 2D Logarithmic) search to get the initial estimate $\mathbf{MV}(u^k, v^k)$ at Level k;

while last ≠ TRUE

{

Find one of the nine macroblocks that yields minimum MAD at Level $k-1$ centered at (x, y),

$2(x^k_0+u^k) - 1 \le x \le 2(x^k_0+u^k) + 1$;

$2(y^k_0+v^k) - 1 \le y \le 2(y^k_0+v^k) + 1$;

if $k = 1$ then last = TRUE;

$k = k - 1$;

Assign new center location and MV to (x^k_0, y^k_0) and (u^k, v^k);

}

end

Comparison of Computational Cost of Motion Vector Search based on examples

Search Method	OPS_per_second for 720×480 at 30 fps	
	$p = 15$	$p = 7$
Sequential search	29.89×10^9	7.00×10^9
2D Logarithmic search	1.25×10^9	0.78×10^9
3-level Hierarchical search	0.51×10^9	0.40×10^9

- Remark: In addition to computational cost, accuracy of motion estimate is another important factor to consider in practice.

H.261

- **H.261**: An earlier digital video compression standard, its principle of MC-based compression is retained in all later video compression standards.
- Designed for videophone, video conferencing and other audiovisual services over ISDN.
- Supports bit-rates of p x 64 kbps, where p ranges from 1 to 30 (hence also known as p * 64).
- Requires that the delay of the video encoder be less than 150 msec so that the video can be used for real-time bidirectional video conferencing.

* WHY video有 delay但像JPEG就沒有

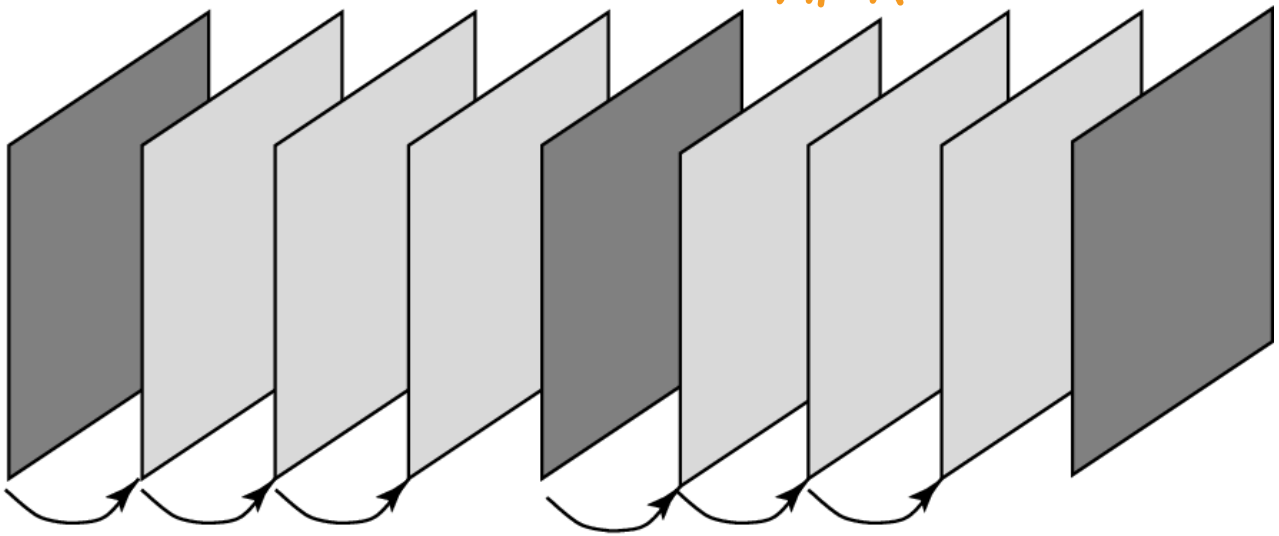
Video Formats Supported by H.261

Video format	Luminance image resolution	Chrominance image resolution	Bit-rate (Mbps) (if 30 fps and uncompressed)	H.261 support
QCIF	176×144	88×72	9.1	required
CIF	352×288	176×144	36.5	optional

chrominance較低
是因為人眼較不敏感

H.261 Frame Sequence

每隔一陣子就要插一個 I frame
∵若有 error，才不會一直傳遞下去，但∵P需要的bits遠小於I frame，
的以I frame也不能太多，要找平衡.



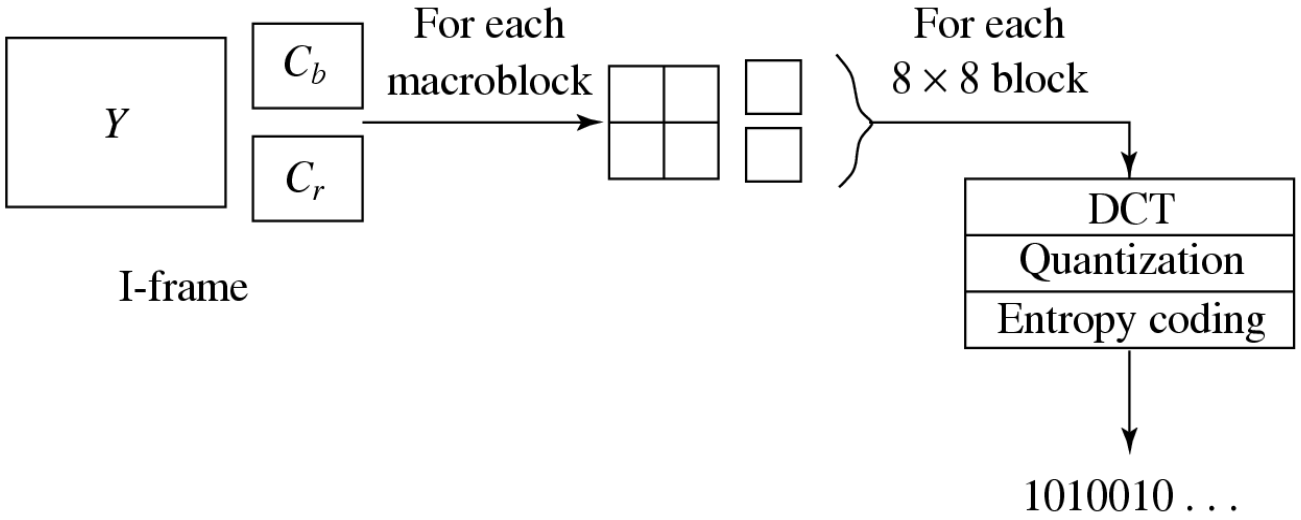
intra coding frame
(自己跟自己encode)

predictive coding frame
(I frame做完找才predict)

H.261 Frame Sequence

- Two types of image frames are defined: intra-frames (**I-frames**) and inter-frames (**P-frames**)
  - I-frames are treated as independent images. Transform coding method similar to JPEG is applied within each I-frame, hence "Intra".
  - P-frames are not independent: coded by a forward predictive coding method (prediction from a previous P-frame or a previous I-frame).
  - **Temporal redundancy removal** is included in P-frame coding, whereas I-frame coding performs only **spatial redundancy removal**.
  - To avoid propagation of coding errors, an I-frame is usually sent a couple of times in each second of the video.
- Motion vectors in H.261 are always measured in units of full pixel and they have a limited range of ± 15 pixels, i.e., $p = 15$.

I-frame Coding



- **Macroblocks** are of size 16 x 16 pixels for the Y frame, and 8 x 8 for Cb and Cr frames, since 4:2:0 chroma subsampling is employed. A macroblock consists of four Y, one Cb, and one Cr 8 x 8 blocks.
- For each 8 x 8 block a DCT transform is applied, the DCT coefficients then go through quantization zigzag scan and entropy coding.

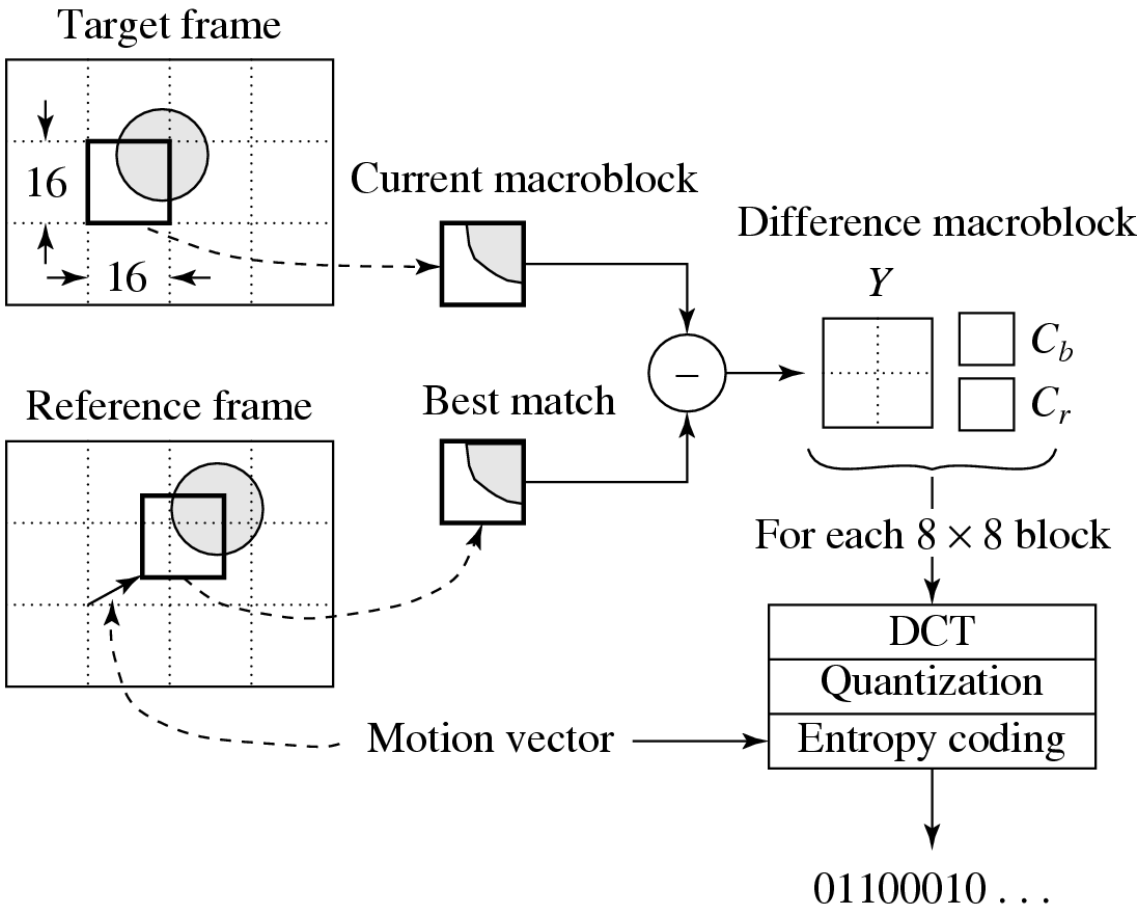
P-frame Coding

- Figure 10.6 shows the H.261 P-frame coding scheme based on motion compensation
- For each macroblock in the Target frame, a motion vector is determined by, for example, one of the search methods discussed earlier.
- After the prediction, a *difference macroblock* representing the *prediction error* is computed.
- Each of these 8 x 8 blocks go through DCT, quantization, zigzag scan and entropy coding procedures.

- The P-frame coding encodes the difference macroblock (not the Target macroblock itself).
- Sometimes, a good match cannot be found, i.e., the prediction error exceeds a certain acceptable level.
  - The MB itself is then encoded (treated as an Intra MB) and in this case it is termed a *non-motion compensated MB*.
- For a motion vector, the difference **MVD** is sent for entropy coding:

$$\mathbf{MVD} = \mathbf{MV}_{\mathbf{Preceding}} - \mathbf{MV}_{\mathbf{Current}} \qquad (10.3)$$

H.261 P-frame Coding Based on Motion Compensation



Quantization in H.261

- The quantization in H.261 uses a constant, called *step_size*, for all DCT coefficients within a macroblock except the intra dc coefficient.
- If we use *DCT* and *QDCT* to denote the DCT coefficients before and after the quantization, then for DC coefficients in Intra mode:

$$\text{DC:} \quad QDCT = round\left(\frac{DCT}{step\_size}\right) = round\left(\frac{DCT}{8}\right) \tag{10.4}$$

for all other coefficients:

$$\text{AC:} \quad QDCT = \left\lfloor \frac{DCT}{step\_size} \right\rfloor = \left\lfloor \frac{DCT}{2 * scale} \right\rfloor \tag{10.5}$$

$scale$ — an integer in the range of [1, 31].

H.261 Encoder and Decoder

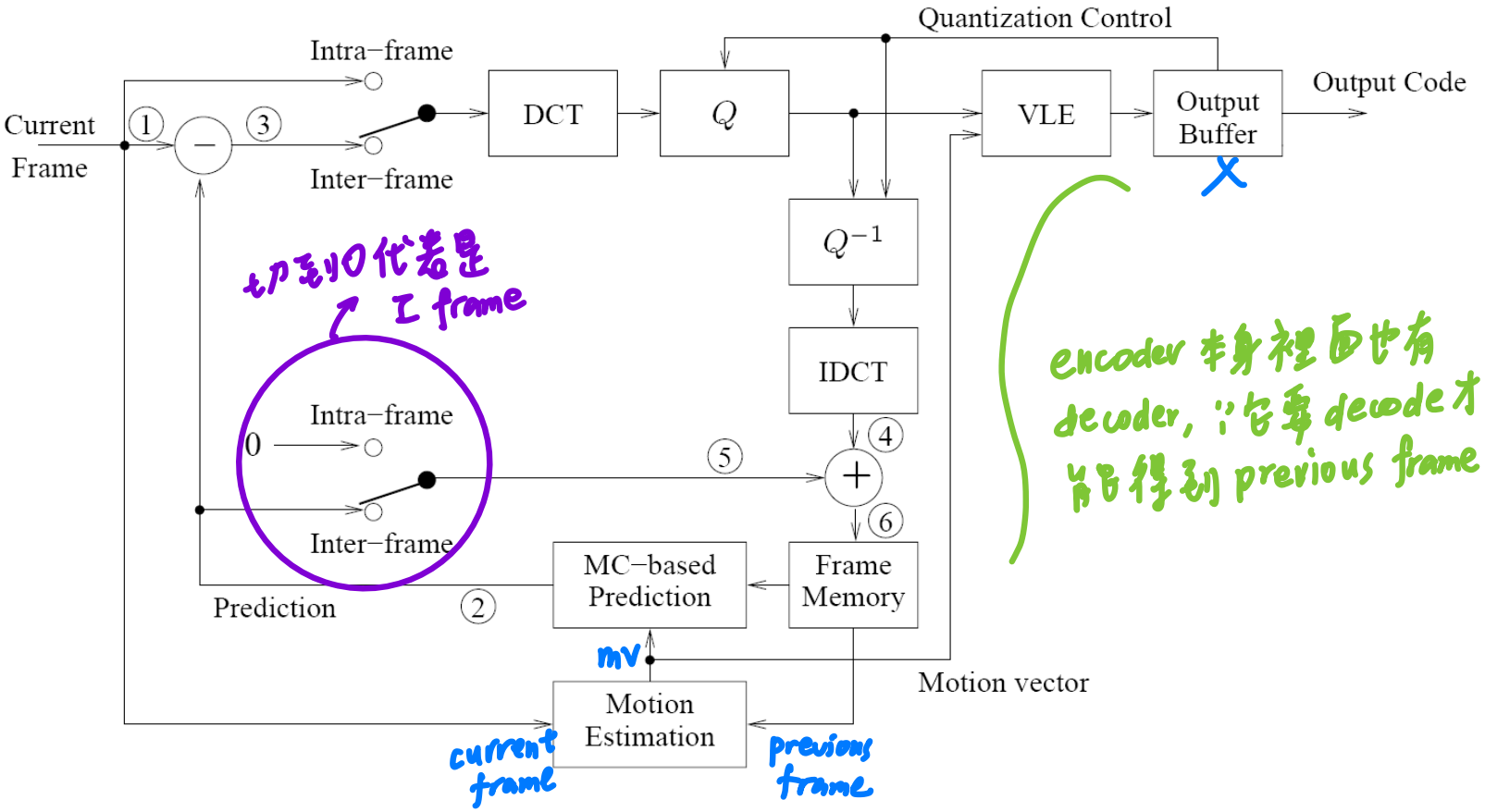
- Fig. 10.7 shows a relatively complete picture of how the H.261 encoder and decoder work.

  A scenario is used where frames I, P_1, and P_2 are encoded and then decoded.

- Note: decoded frames (not the original frames) are used as reference frames in motion estimation.

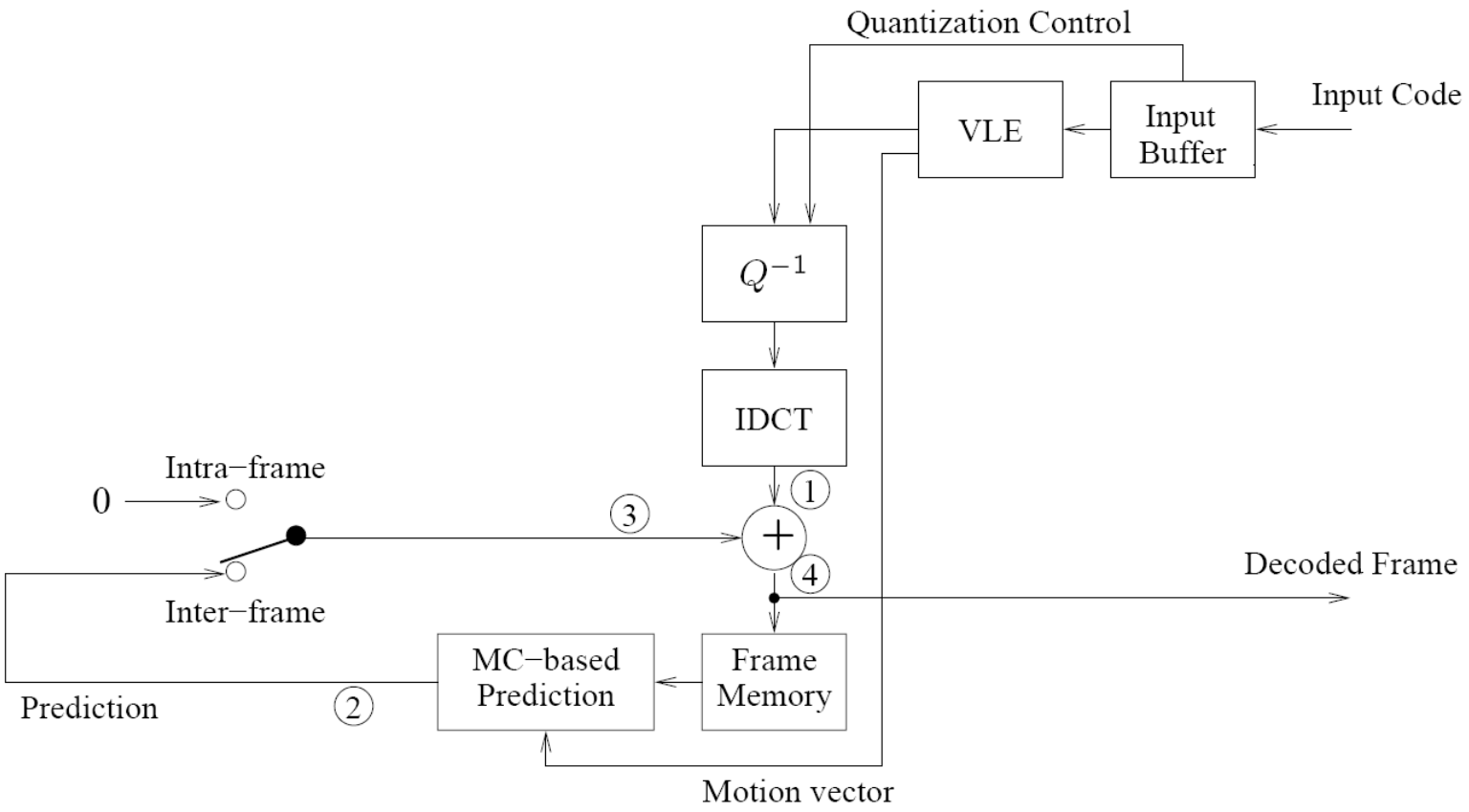
- The data that goes through the observation points indicated by the circled numbers are summarized in Tables 10.3 and 10.4.

H.261 Encoder and Decoder



(a) Encoder

H.261 Encoder and Decoder



(b) Decoder

Data Flow at the Observation Points in H.261 Encoder

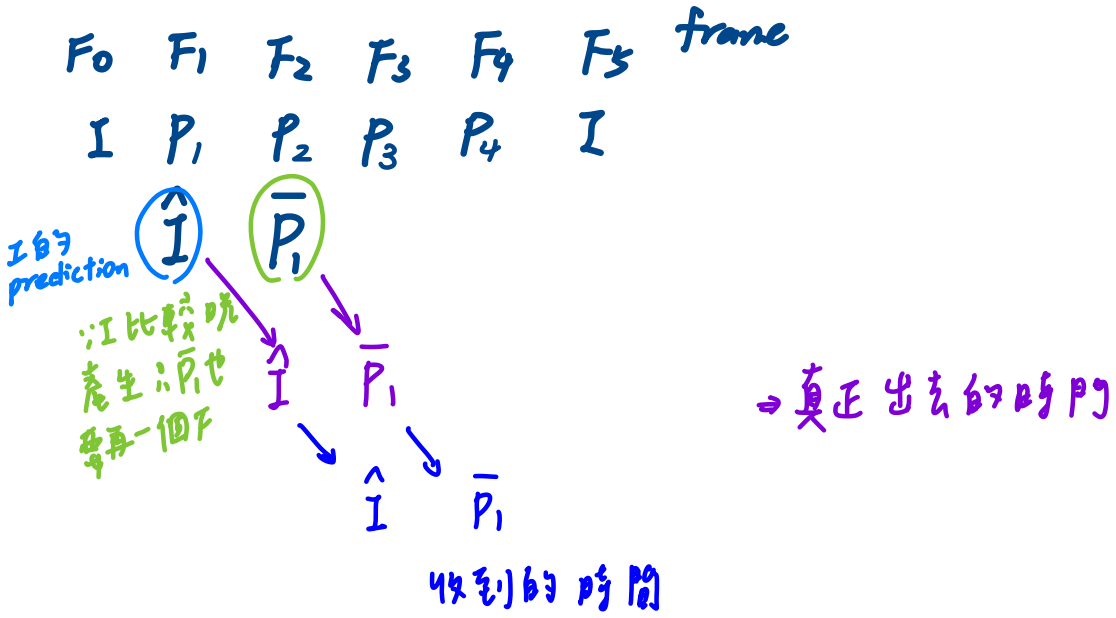
Current Frame	Observation Point					
	1	2	3	4	5	6
I	I			$\tilde{I}$	0	$\tilde{I}$
P_1	P_1	P_1'	D_1	$\tilde{D}_1$	P_1'	$\tilde{P}_1$
P_2	P_2	P_2'	D_2	$\tilde{D}_2$	P_2'	$\tilde{P}_2$

Notation
I, P : original data
P' : prediction data
$\tilde{I}, \tilde{P}$: decoded data
D : difference

Data Flow at the Observation Points in H.261 Decoder

Current Frame	Observation Point			
	1	2	3	4
I	$\tilde{I}$		0	$\tilde{I}$
P_1	$\tilde{D}_1$	P_1'	P_1'	$\tilde{P}_1$
P_2	$\tilde{D}_2$	P_2'	P_2'	$\tilde{P}_2$

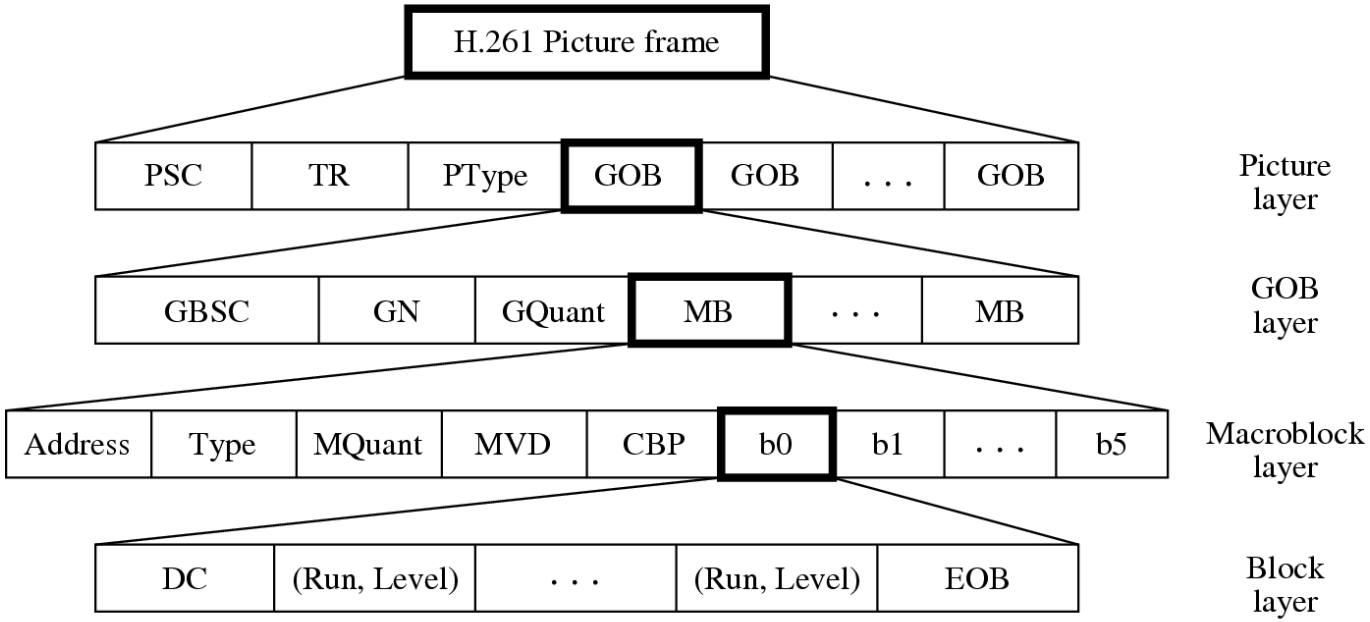
I P P P P I


F0 F1 F2 F3 F4 F5
frame
I P1 P2 P3 P4 I
Î
P̄1
I自己prediction
比較時產生P̄1也要再一個下
Î
P̄1
Î
P̄1
收到的時間
→真正出去的時間

Syntax of H.261 Video Bitstream

- Four layers: Picture, Group of Blocks (GOB), Macroblock, and Block.
  1. **The Picture layer**: PSC (Picture Start Code) delineates boundaries between pictures. TR (Temporal Reference) provides a time-stamp for the picture.
  2. **The GOB layer:** Each picture is divided into Groups of Blocks (GOB), each of which contains 11 x 3 macroblocks.
     - For instance, the CIF image has 2 x 6 GOBs, corresponding to its image resolution of 352 x 288 pixels. Each GOB has its Start Code (GBSC) and Group number (GN).

Syntax of H.261 Video Bitstream



PSC	Picture Start Code	TR	Temporal Reference
PType	Picture Type	GOB	Group of Blocks
GBSC	GOB Start Code	GN	Group Number
GQuant	GOB Quantizer	MB	Macroblock
MQuant	MB Quantizer	MVD	Motion Vector Data
CBP	Coded Block Pattern	EOB	End of Block

- In case a network error causes a bit error or the loss of some bits, H.261 video can be recovered and resynchronized at the next identifiable GOB.
- GQuant indicates the Quantizer to be used in the GOB unless it is overridden by any subsequent MQuant (Quantizer for Macroblock). GQuant and MQuant are referred to as $scale$ in Eq. (10.5).

3. **The Macroblock layer:** Each Macroblock (MB) has its own Address indicating its position within the GOB, Quantizer (MQuant), and six 8 x 8 image blocks (4 Y, 1 Cb, 1 Cr).

4. **The Block layer:** For each 8 x 8 block, the bitstream starts with DC value, followed by pairs of length of zerorun (Run) and the subsequent non-zero value (Level) for ACs, and finally the End of Block (EOB) code. The range of Run is [0*;* 63]. Level reflects quantized values — its range is [−127*,* 127] and Level ≠ 0.

Arrangement of GOBs in H.261 Luminance Images

GOB 0	GOB 1
GOB 2	GOB 3
GOB 4	GOB 5
GOB 6	GOB 7
GOB 8	GOB 9
GOB 10	GOB 11

CIF

GOB 0
GOB 1
GOB 2

QCIF

H.263

- H.263 is an improved video coding standard for video conferencing and other audiovisual services transmitted on Public Switched Telephone Networks (PSTN).
- Aims at low bit-rate communications at bit-rates of less than 64 kbps.
- Uses predictive coding for inter-frames to reduce temporal redundancy and transform coding for the remaining signal to reduce spatial redundancy (for both Intra-frames and inter-frame prediction).

H.263 Video Coding Standard

- H.263 is the video coding standard in H.323/H.324
- Developed later than H.261, can accommodate computationally more intensive options
  - Initial version (H.263 baseline): 1995
  - H.263+: 1997
  - H.263++: 2000
- Result: Significantly better quality at lower rates
  - Better video at 18-24 Kbps than H.261 at 64 Kbps
  - Enable video phone over regular phone lines (28.8 Kbps) or wireless modem

Improvements over H.261

- Better motion estimation
  - half-pel accuracy motion estimation with bilinear interpolation filter
  - Larger motion search range [-31.5,31], and unrestricted MV at boundary blocks
  - More efficient predictive coding for MVs (median prediction using three neighbors)
  - overlapping block motion compensation (option)
  - variable block size: 16x16 -> 8x8, 4 MVs per MB (option)
  - use bidirectional temporal prediction (PB picture) (option)
- 3-D VLC for DCT coefficients
  - (runlength, value, EOB)
- Syntax-based arithmetic coding (option)
  - 4% savings in bit rate for P-mode, 10% saving for I-mode, at 50% more computations
- The options, when chosen properly, can improve the PSNR 0.5-1.5 dB over default at 20-70 kbps range.

Video Formats Supported by H.263

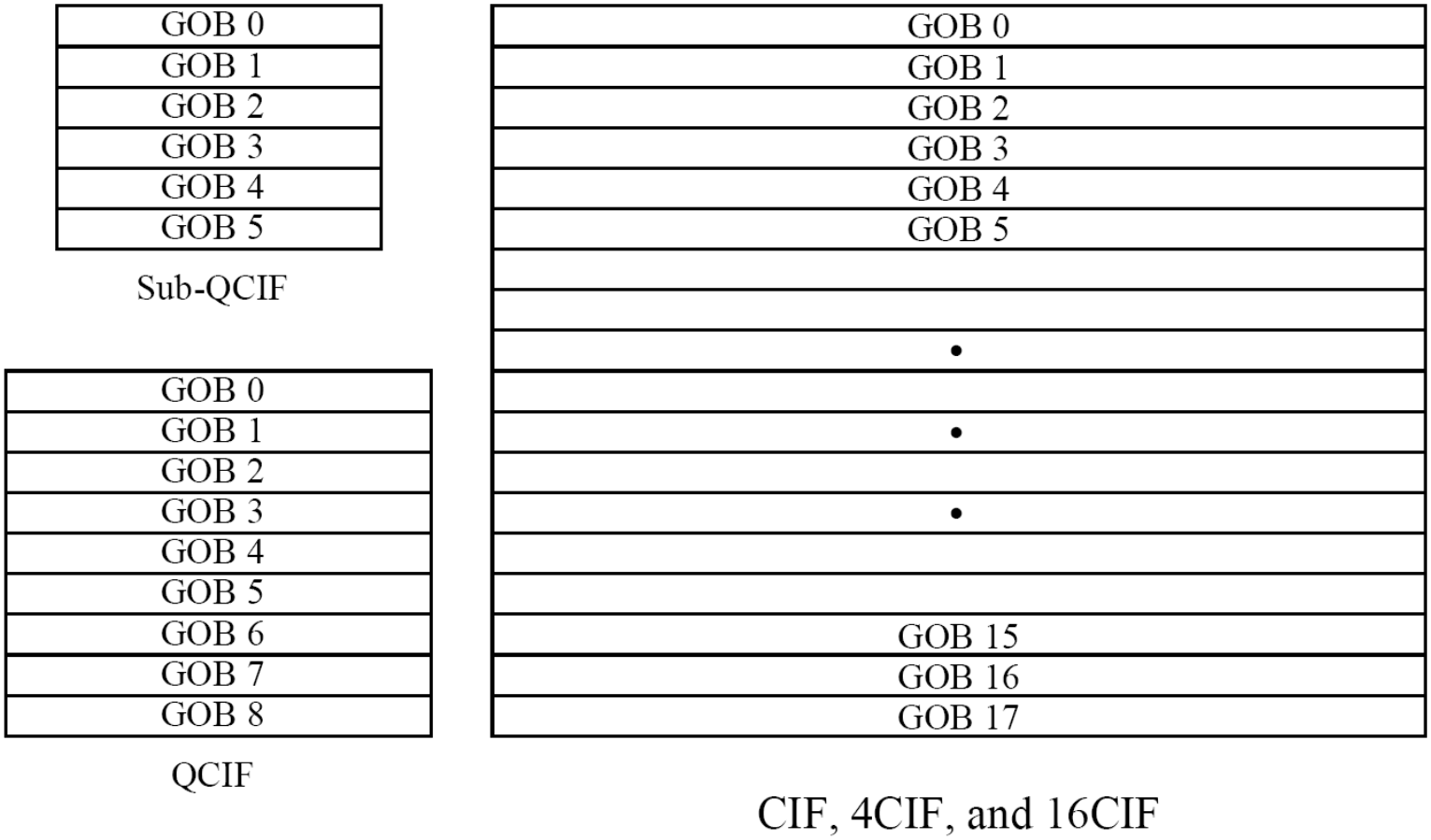
Video format	Luminance image resolution	Chrominance image resolution	Bit-rate (Mbps) (if 30 fps and uncompressed)	Bit-rate (kbps) BPPmaxKb (compressed)
sub-QCIF	128×96	64×48	4.4	64
QCIF	176×144	88×72	9.1	64
CIF	352×288	176×144	36.5	256
4CIF	704×576	352×288	146.0	512
16CIF	$1,408 \times 1,152$	704×576	583.9	1024

BPP: bitrate per picture

H.263 & Group of Blocks (GOB)

- As in H.261, H.263 standard also supports the notion of Group of Blocks (GOB).
- The difference is that GOBs in H.263 do not have a fixed size, and they always start and end at the left and right borders , respectively, of the picture.
- As shown in Fig. 10.10, each QCIF luminance image consists of 9 GOBs and each GOB has 11 x 1 MBs (176 x 16 pixels), whereas each 4CIF luminance image consists of 18 GOBs and each GOB has 44 x 2 MBs (704 x 32 pixels).

Arrangement of GOBs in H.263 Luminance Images.



Motion Compensation in H.263

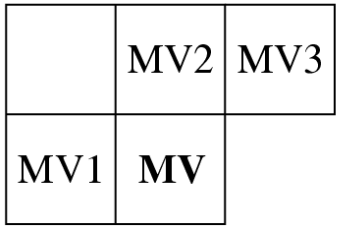
- The horizontal and vertical components of the **MV** are predicted from the median values of the horizontal and vertical components, respectively, of **MV1**, **MV2**, **MV3** from the "previous", "above" and "above and right" MBs (see Fig. (a) on next page).

- For the Macroblock with $\mathbf{MV}(u, v)$:

$$u_p = median(u_1, u_2, u_3),$$
$$v_p = median(v_1, v_2, v_3). \qquad (10.6)$$

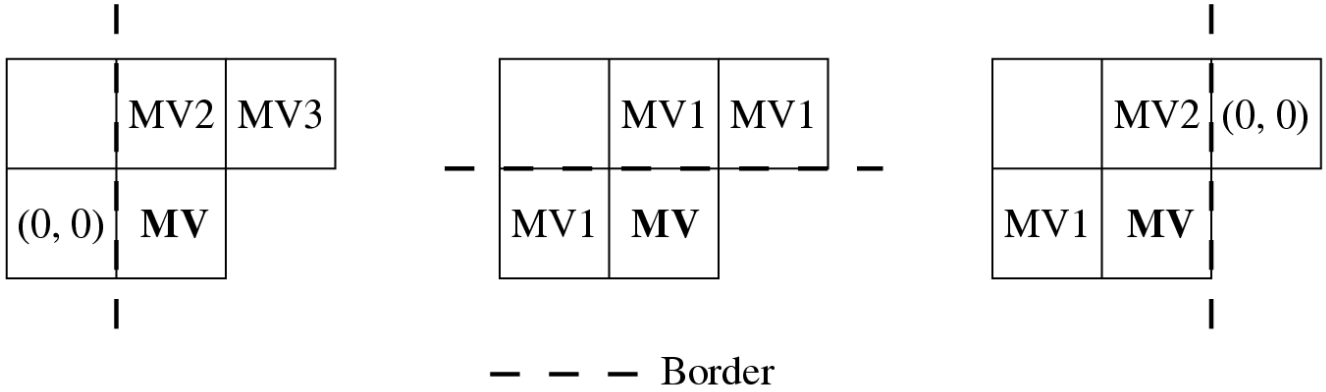
Instead of coding the $\mathbf{MV}(u, v)$ itself, the error vector $(\delta u, \delta v)$ is coded, where $\delta u = u - u_p$ and $\delta v = v - v_p$.

Prediction of Motion Vector in H.263.



MV Current motion vector
MV1 Previous motion vector
MV2 Above motion vector
MV3 Above and right motion vector

(a)



(b)

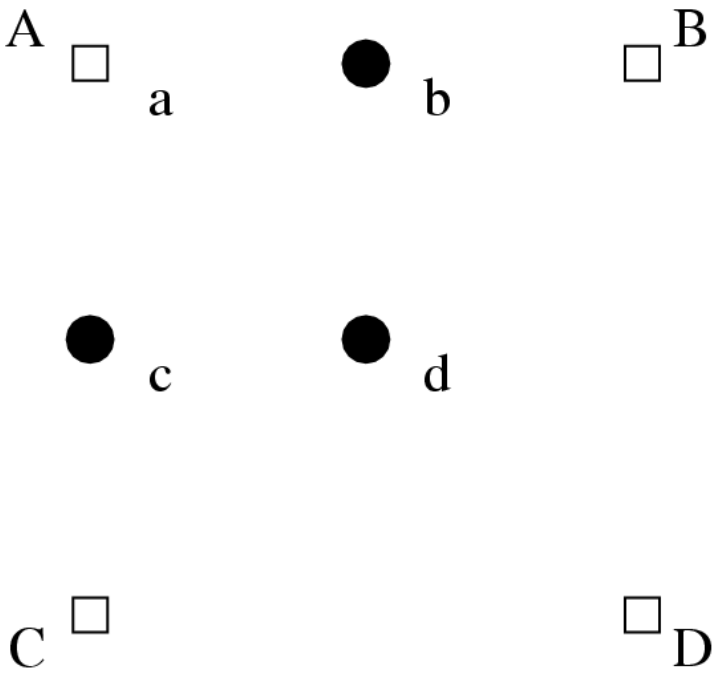
Half-Pixel Precision

- In order to reduce the prediction error, **half-pixel precision** is supported in H.263 vs. full-pixel precision only in H.261.

  - The default range for both the horizontal and vertical components u and v of $\mathbf{MV}(u, v)$ are now [−16, 15.5].

  - The pixel values needed at half-pixel positions are generated by a simple **bilinear interpolation** method, as shown in Fig. 10.12.

Half-Pixel Prediction by Bilinear Interpolation in H.263.



Full-pixel position

Half-pixel position

$a = A$

$b = (A + B + 1) / 2$

$c = (A + C + 1) / 2$

$d = (A + B + C + D + 2) / 4$

Optional H.263 Coding Modes

- H.263 specifies many negotiable coding options in its various Annexes.

  Four of the common options are as follows:

  1. **Unrestricted motion vector mode:**

  - The pixels referenced are no longer restricted to be within the boundary of the image.

  - When the motion vector points outside the image boundary, the value of the boundary pixel that is geometrically closest to the referenced pixel is used.

  - The maximum range of motion vectors is [-31.5, 31.5].

2. **Syntax-based arithmetic coding mode:**

- As in H.261, variable length coding (VLC) is used in H.263 as a default coding method for the DCT coefficients.
- VLC must generate an integer number of bits for each symbol. By Arithmetic coding, such a restriction is removed, resulting in bitrate saving of 4% for inter-frames and 10% for intra-frames in this mode.
- Similar to H.261, the syntax of H.263 is also structured as a hierarchy of four layers. Each layer is coded using a combination of fixed length code and variable length code.
- All VLC operations are replaced with arithmetic operations.
- Each different string of symbols to be coded may have a different distribution (model). The syntax-based arithmetic coder switches the model on the fly according to the syntax.

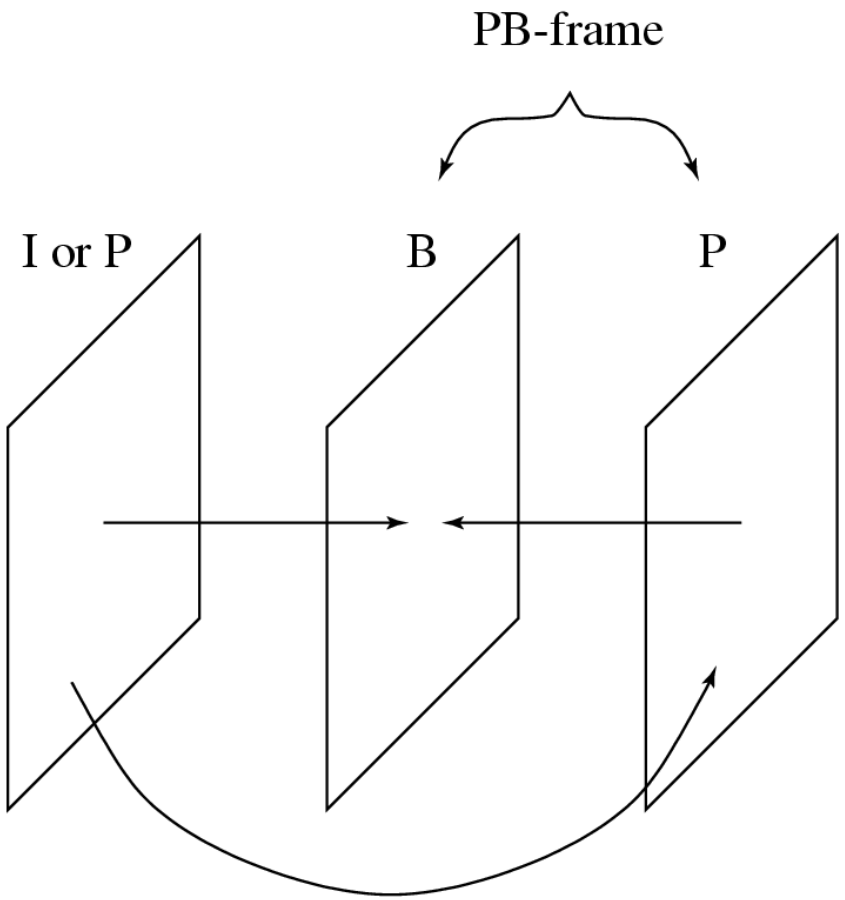
3. **Advanced prediction mode:**

- In this mode, the block size for MC is reduced from 16 to 8, and four motion vectors (from each of the 8 x 8 blocks) are generated for each macroblock in the luminance image.
- Each pixel in a prediction block takes a weighted sum of three predicted values based on the motion vector of the current block and two out of the four motion vectors from the neighboring blocks (1 from left or right block and 1 from the block above or below)

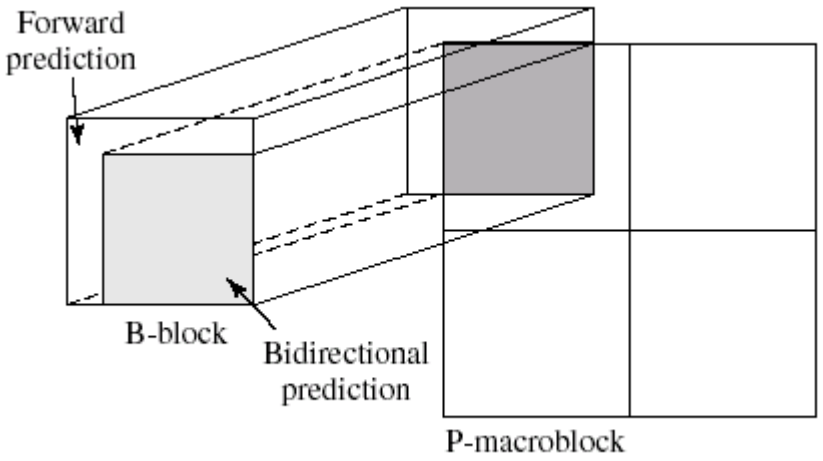
4. **PB-frames mode:**

- A PB-frame consists of two pictures being coded as one unit
- The use of the PB-frames mode is indicated in PTYPE
- Generates satisfactory results for videos with moderate motions
- Under large motions, PB-frames do not compress as well as B-frames and an improved new mode has been developed in Version 2 of H.263

PB-frame in H.263.



PB-Picture Mode

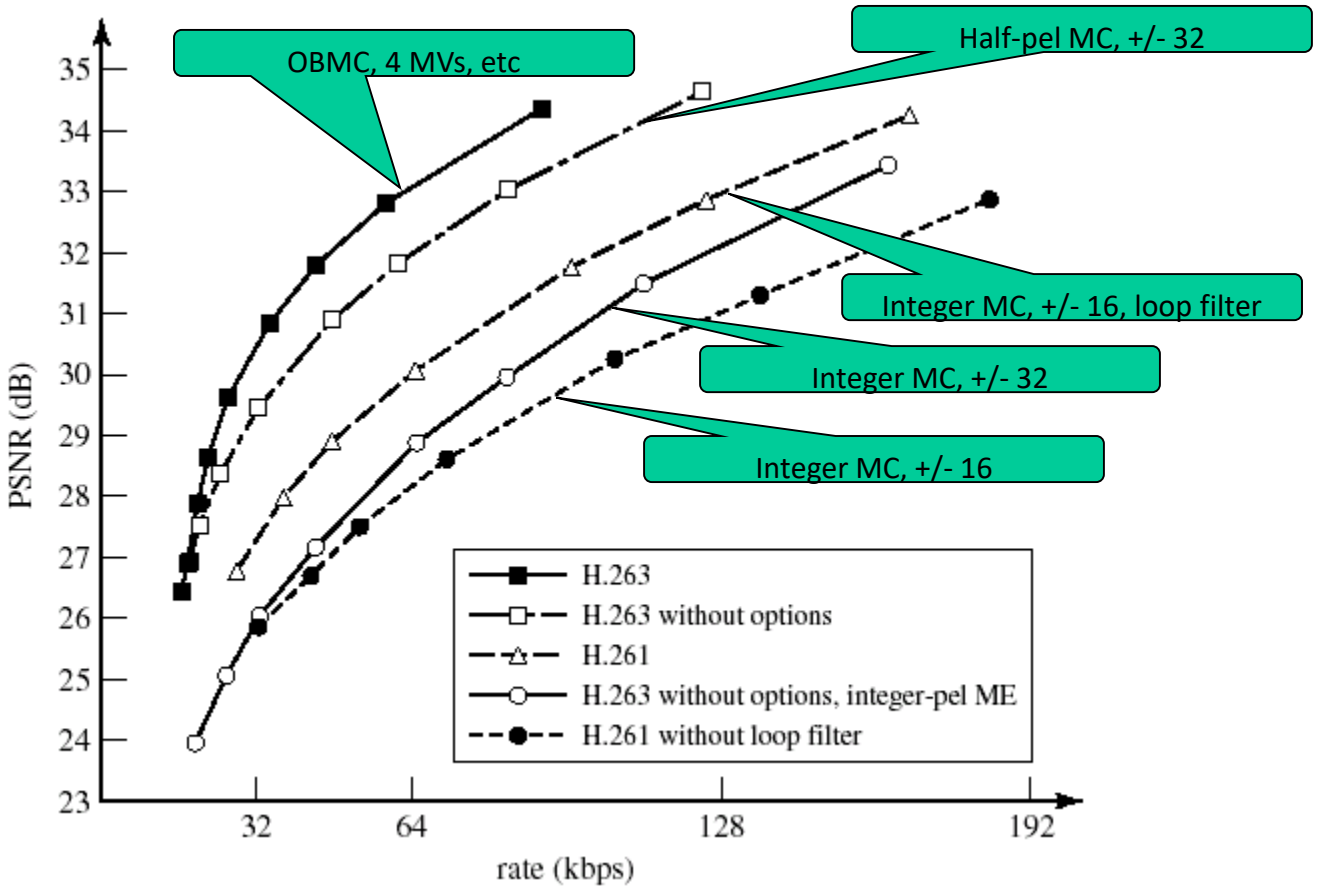


PB-picture mode codes two pictures as a group. The second picture (P) is coded first, then the first picture (B) is coded using both the P-picture and the previously coded picture. This is to avoid the reordering of pictures required in the normal B-mode. But it still requires additional coding delay than P-frames only.

In a B-block, forward prediction (predicted from the previous frame) can be used for all pixels; backward prediction (from the future frame) is only used for those pels that the backward motion vector aligns with pels of the current MB. Pixels in the “white area” use only forward prediction.

An improved PB-frame mode was defined in H.263+, that removes the previous restriction.

Performance of H.261 and H.263



Forman, QCIF, 12.5 Hz

ITU-T Multimedia Communications Standards

Network	System	Video	Audio	Mux	Control
PSTN	H.324	H.261/3	G.723.1	H.223	H.245
N-ISDN	H.320	H.261	G.7xx	H.221	H.242
B-ISDN/ATM	H.321	H.261	G.7xx	H.221	Q.2931
	H.310	H.261/2	G.7xx/MPEG	H.222.0/1	H.245
QoS LAN	H.322	H.261/3	G.7xx	H.221	H.242
Non-QoS LAN	H.323	H.261	G.7xx	H.225.0	H.245