Chapter 10
Basic Video Compression Techniques

10.1 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.

10.2 Video Compression with Motion Compensation

- Consecutive frames in a video are similar — temporal redundancy exists.
- **Temporal redundancy** is exploited so that not every frame of the video needs to be coded independently as a new image.

  The difference between the current frame and other frame(s) in the sequence will be coded — small values and low entropy, good for compression.
- Steps of Video compression based on *Motion Compensation (MC)*:
  1. Motion Estimation (motion vector search).
  2. MC-based Prediction.
  3. Derivation of the prediction error, i.e., the difference.

Motion Compensation

- Each image is divided into *macroblocks* of size $N \times N$.
  - By default, $N = 16$ for luminance images. For chrominance images, $N = 8$ if 4:2:0 chroma subsampling is adopted.
- Motion compensation is performed at the macroblock level.
  - The current image 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* $\mathbf{MV}$.
  - Figure 10.1 shows the case of *forward prediction* in which the Reference frame is taken to be a previous frame.

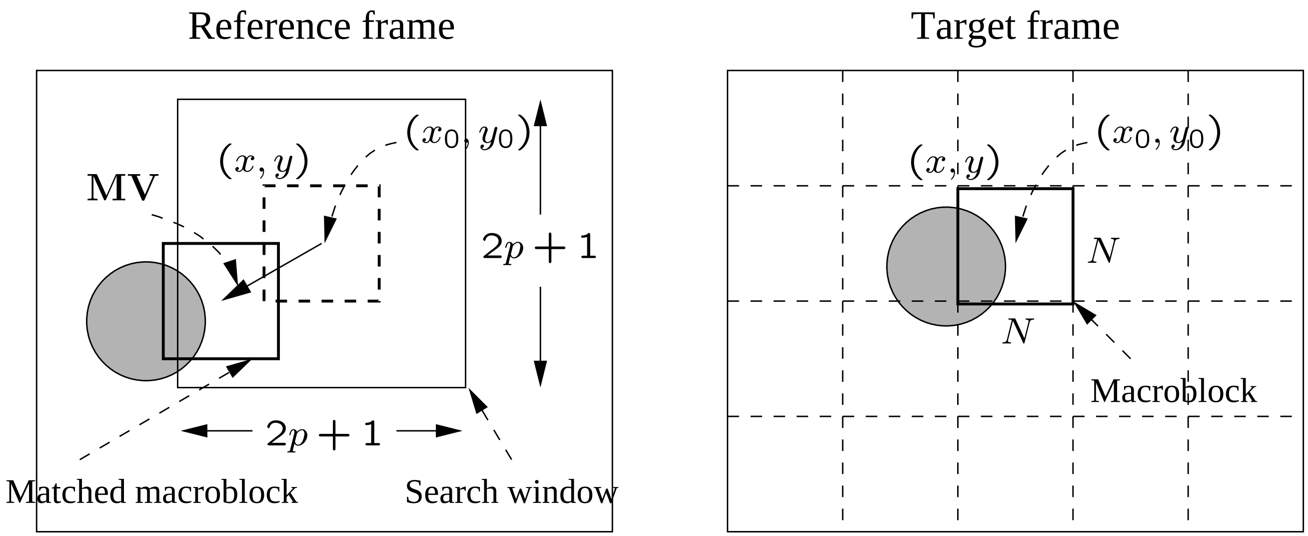


Fig. 10.1: Macroblocks and Motion Vector in Video Compression.

- MV search is usually limited to a small immediate neighborhood — both horizontal and vertical displacements in the range $[-p, p]$.

  This makes a search window of size $(2p + 1) \times (2p + 1)$.

10.3 Search for Motion Vectors

- The difference between two macroblocks can then be measured by their *Mean Absolute Difference (MAD)*:

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

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:

$$(u, v) = [\ (i, j) \mid MAD(i, j) \text{ is minimum}, \ i \in [-p, p], \ j \in [-p, p]\ ] \qquad (10.2)$$

Sequential Search

- **Sequential search**: sequentially search the whole $(2p+1) \times (2p+1)$ window in the Reference frame (also referred to as Full search).
  - a macroblock centered at each of the positions within the window is compared to the macroblock in the Target frame pixel by pixel and their respective MAD is then derived using Eq. (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.
  - sequential search method is very costly — assuming each pixel comparison requires three operations (subtraction, absolute value, addition), the cost for obtaining a motion vector for a single macroblock is $(2p+1) \cdot (2p+1) \cdot N^2 \cdot 3 \Rightarrow O(p^2N^2)$.

PROCEDURE 10.1 Motion-vector:sequential-search

```
begin
  min_MAD = LARGE_NUMBER;        /* Initialization */
  for i = -p to p
    for j = -p to p
      {
        cur_MAD = MAD(i, j);
        if cur_MAD < min_MAD
          {
            min_MAD = cur_MAD;
            u = i;      /* Get the coordinates for MV. */
            v = j;
          }
      }
end
```

2D Logarithmic Search

- **Logarithmic search**: a cheaper version, that is suboptimal but still usually effective.

- The procedure for 2D Logarithmic Search of motion vectors takes several iterations and is akin to a binary search:
  - As illustrated in Fig.10.2, initially only nine locations in the search window are used as seeds for a MAD-based search; they are marked as ‘1’.
  - After the one that yields the minimum MAD is located, the center of the new search region is moved to it and the step-size (”offset”) is reduced to half.
  - In the next iteration, the nine new locations are marked as ‘2’, and so on.

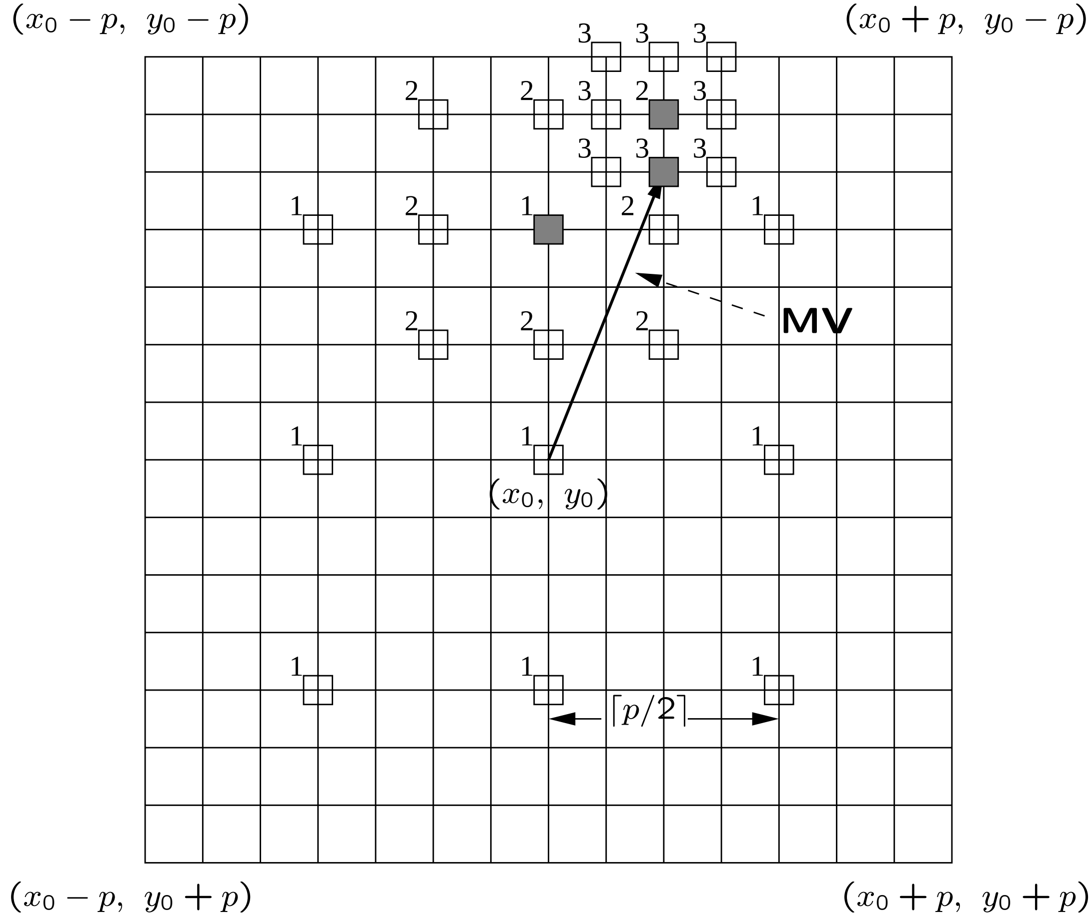


Fig. 10.2: 2D Logarithmic Search for Motion Vectors.

PROCEDURE 10.2 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 $\neq$ TRUE

{

Find one of the nine specified macroblocks that yields minimum MAD;

if offset $= 1$ then last $=$ TRUE;

offset $= \lceil$offset$/2\rceil$;

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

}

end

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

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

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.

- Figure 10.3: 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.

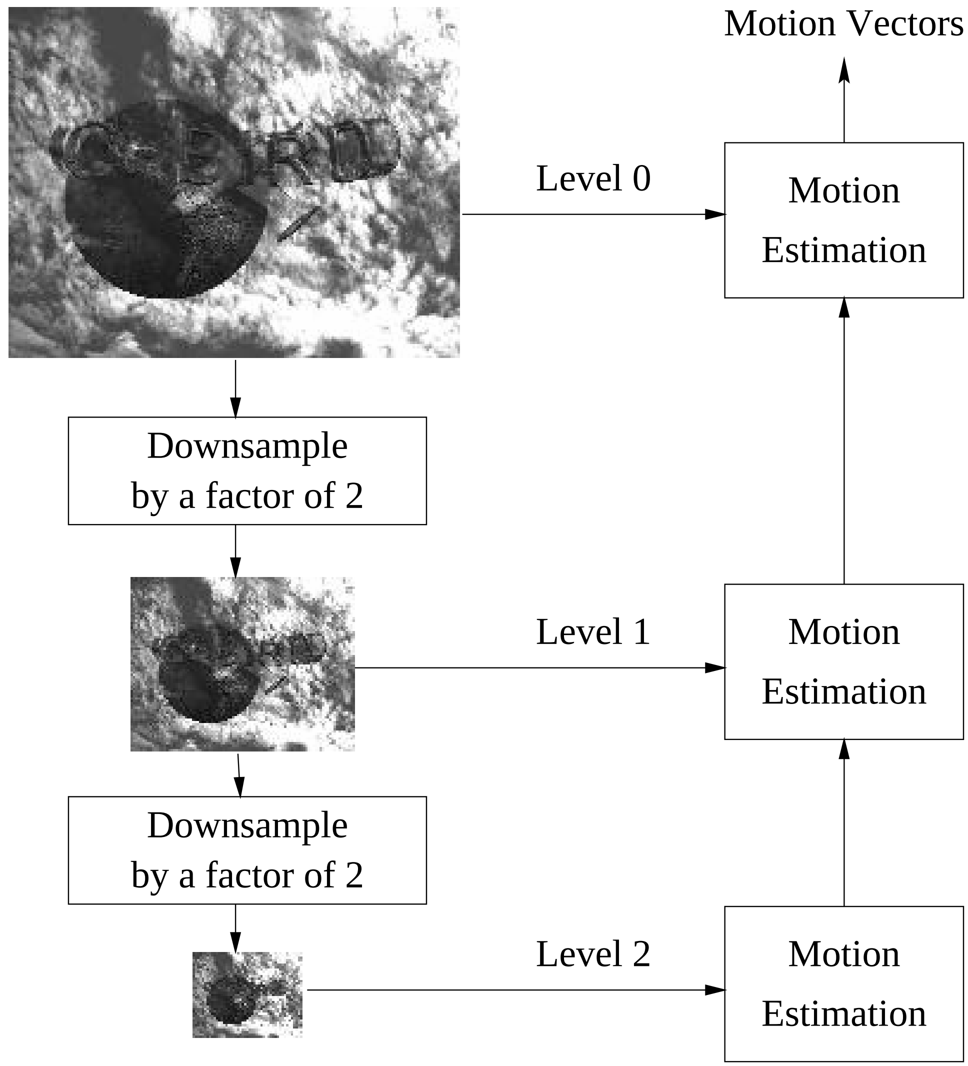


Fig. 10.3: A Three-level Hierarchical Search for Motion Vectors.

Hierarchical Search (Cont'd)

- Given the estimated motion vector (u^k, v^k) at Level k, a 3×3 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:

$$(2u^k - 1 \le u^{k-1} \le 2u^k + 1, \quad 2v^k - 1 \le v^{k-1} \le 2v^k + 1)$$

- Let (x_0^k, y_0^k) 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 can be outlined as follows:

PROCEDURE 10.3 Motion-vector:hierarchical-search

begin

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

$x_0^k = x_0^0/2^k$; $\quad y_0^k = y_0^0/2^k$;

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

while last $\neq$ TRUE

{

Find one of the nine macroblocks that yields minimum MAD at Level $k-1$ centered at

$(2(x_0^k + u^k) - 1 \leq x \leq 2(x_0^k + u^k) + 1, \;\; 2(y_0^k + v^k) - 1 \leq y \leq 2(y_0^k + v^k) + 1)$;

if $k = 1$ then last = TRUE;

$k = k - 1$;

Assign (x_0^k, y_0^k) and (u^k, v^k) with the new center location and MV;

}

end

Table 10.1 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

10.4 H.261

- **H.261**: An earlier digital video compression standard, its principle of MC-based compression is retained in all later video compression standards.
  - The standard was designed for videophone, video conferencing and other audiovisual services over ISDN.
  - The video codec supports bit-rates of $p \times 64$ kbps, where *p* ranges from 1 to 30 (Hence also known as $p * 64$).
  - Require that the delay of the video encoder be less than 150 msec so that the video can be used for real-time bi-directional video conferencing.

ITU Recommendations & H.261 Video Formats

- H.261 belongs to the following set of ITU recommendations for visual telephony systems:

  1. H.221 — Frame structure for an audiovisual channel supporting 64 to 1,920 kbps.
  2. H.230 — Frame control signals for audiovisual systems.
  3. H.242 — Audiovisual communication protocols.
  4. H.261 — Video encoder/decoder for audiovisual services at $p \times 64$ kbps.
  5. H.320 — Narrow-band audiovisual terminal equipment for $p \times 64$ kbps transmission.

Table 10.2 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

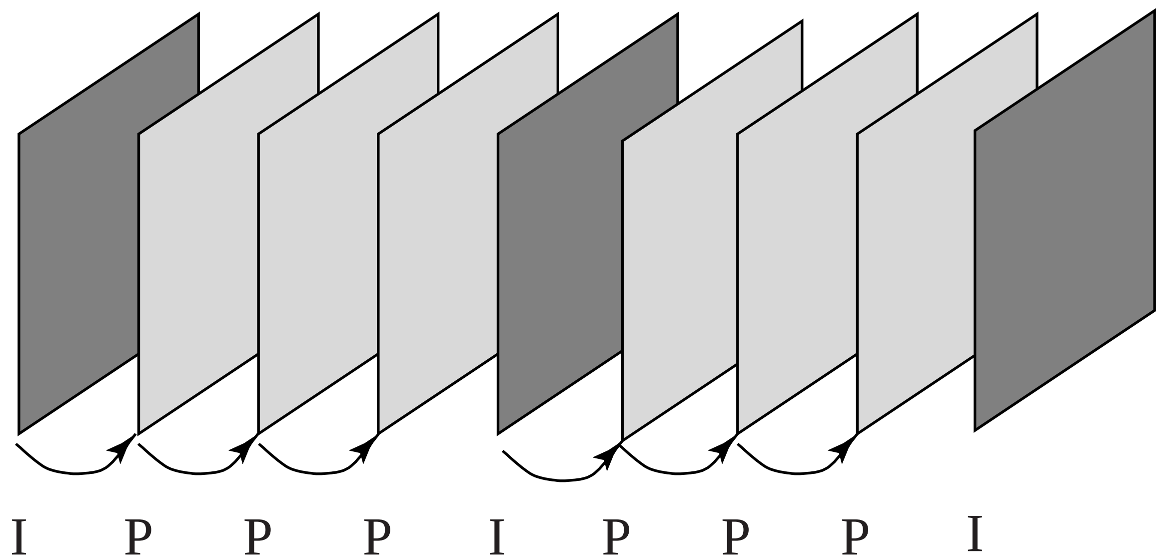


Fig. 10.4: H.261 Frame Sequence.

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 is allowed — not just from 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., $p = 15$.

Intra-frame (I-frame) Coding

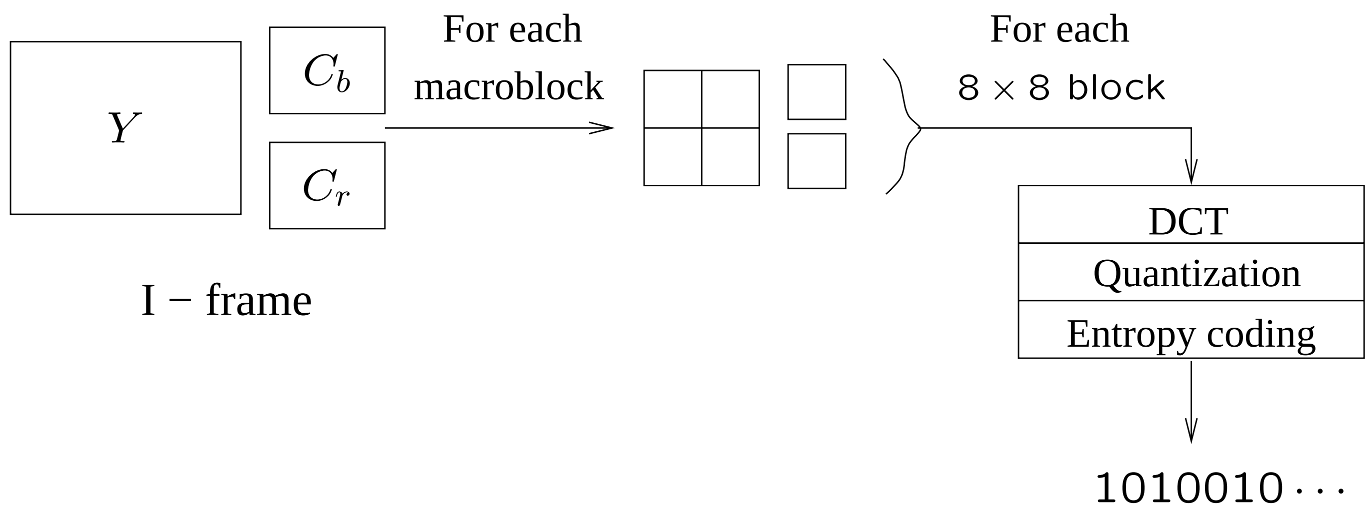


Fig. 10.5: I-frame Coding.

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

Inter-frame (P-frame) Predictive 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 allocated by one of the search methods discussed earlier.
  - After the prediction, a *difference macroblock* is derived to measure the *prediction error*.
  - Each of these 8×8 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 motion vector, the difference $\mathbf{MVD}$ is sent for entropy coding:

$$\mathbf{MVD} = \mathbf{MV}_{\mathbf{Preceding}} - \mathbf{MV}_{\mathbf{Current}} \tag{10.3}$$

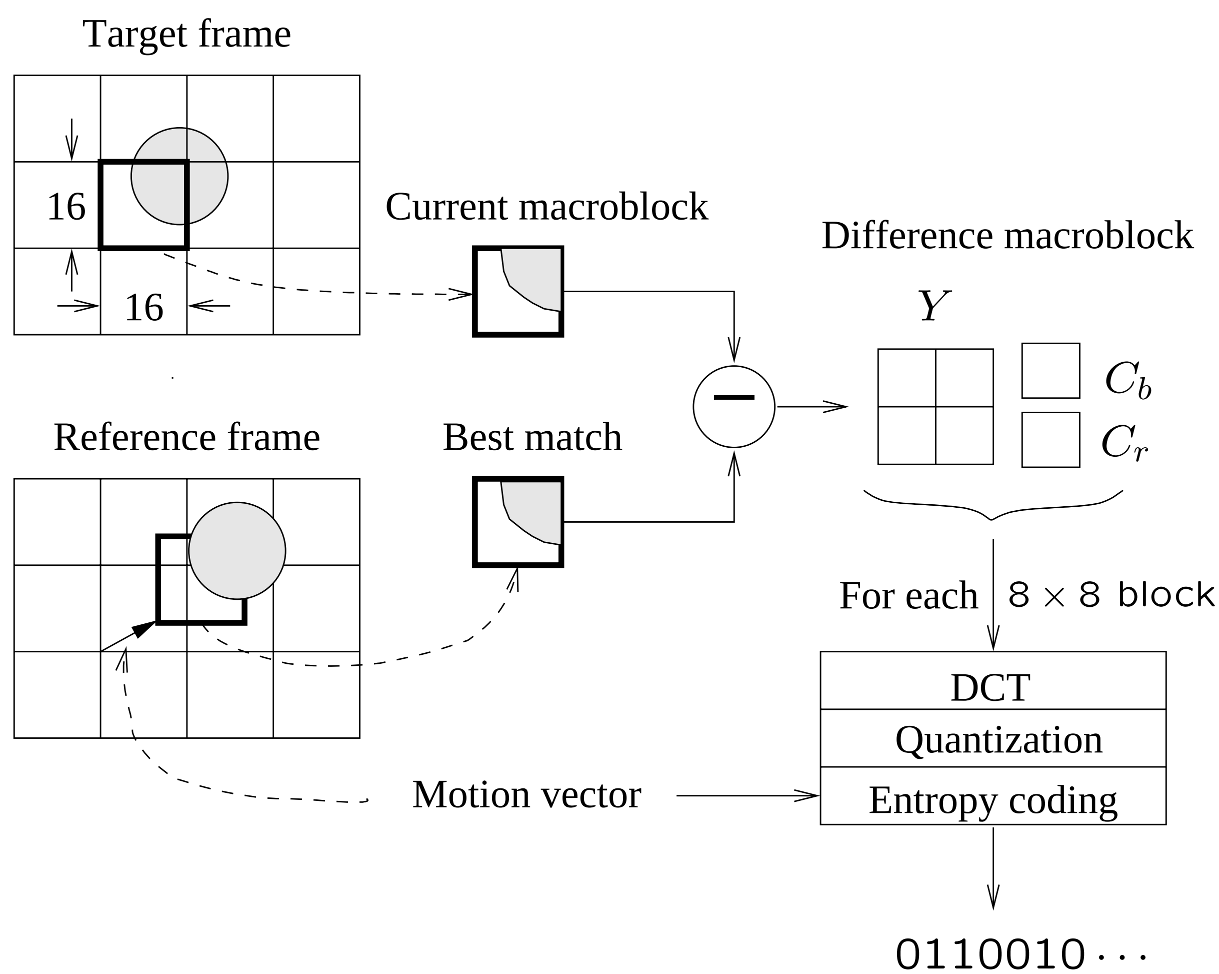


Fig. 10.6: H.261 P-frame Coding Based on Motion Compensation.

Quantization in H.261

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

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

for all other coefficients:

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

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

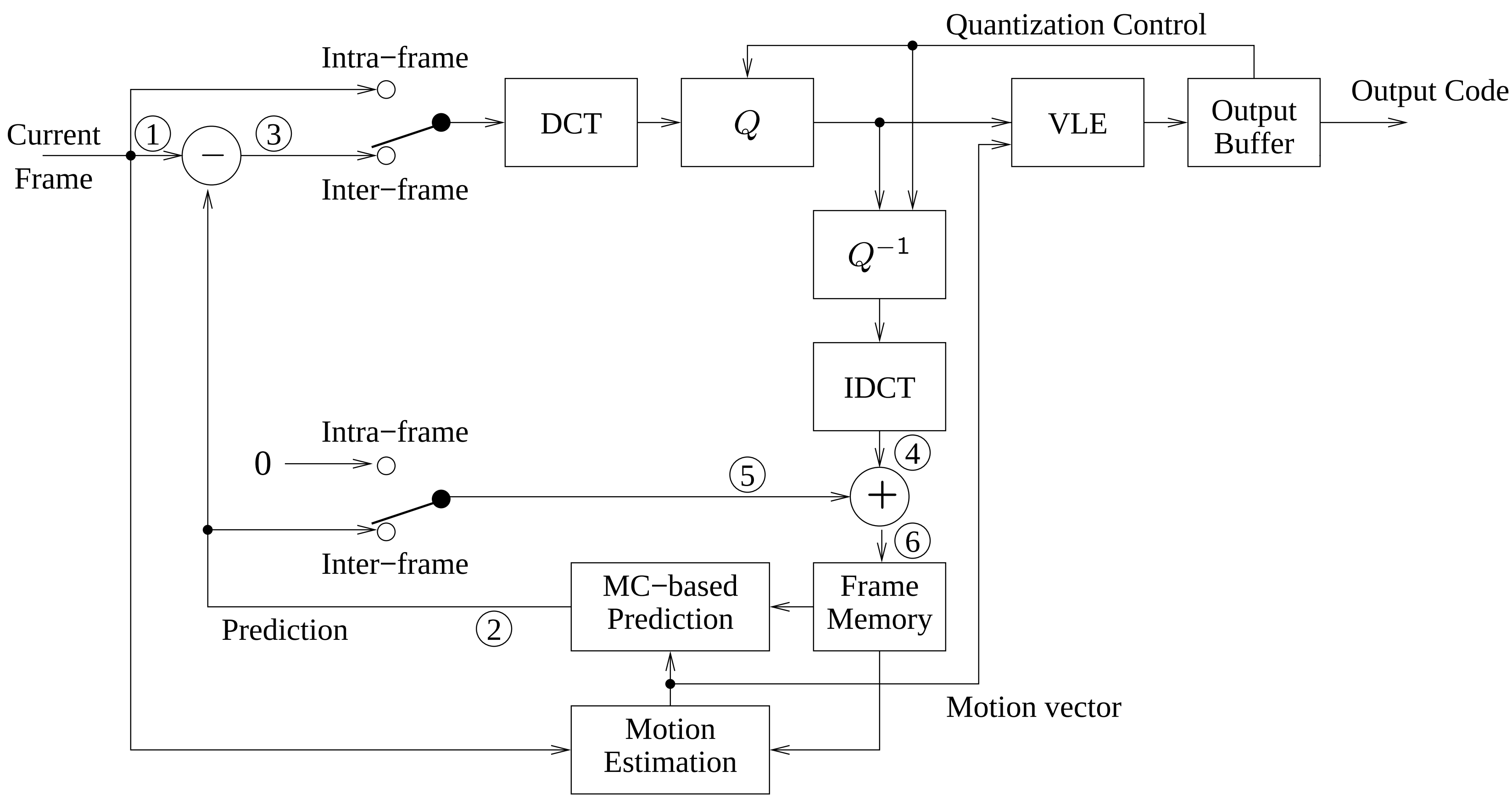
H.261 Encoder and Decoder

- 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.

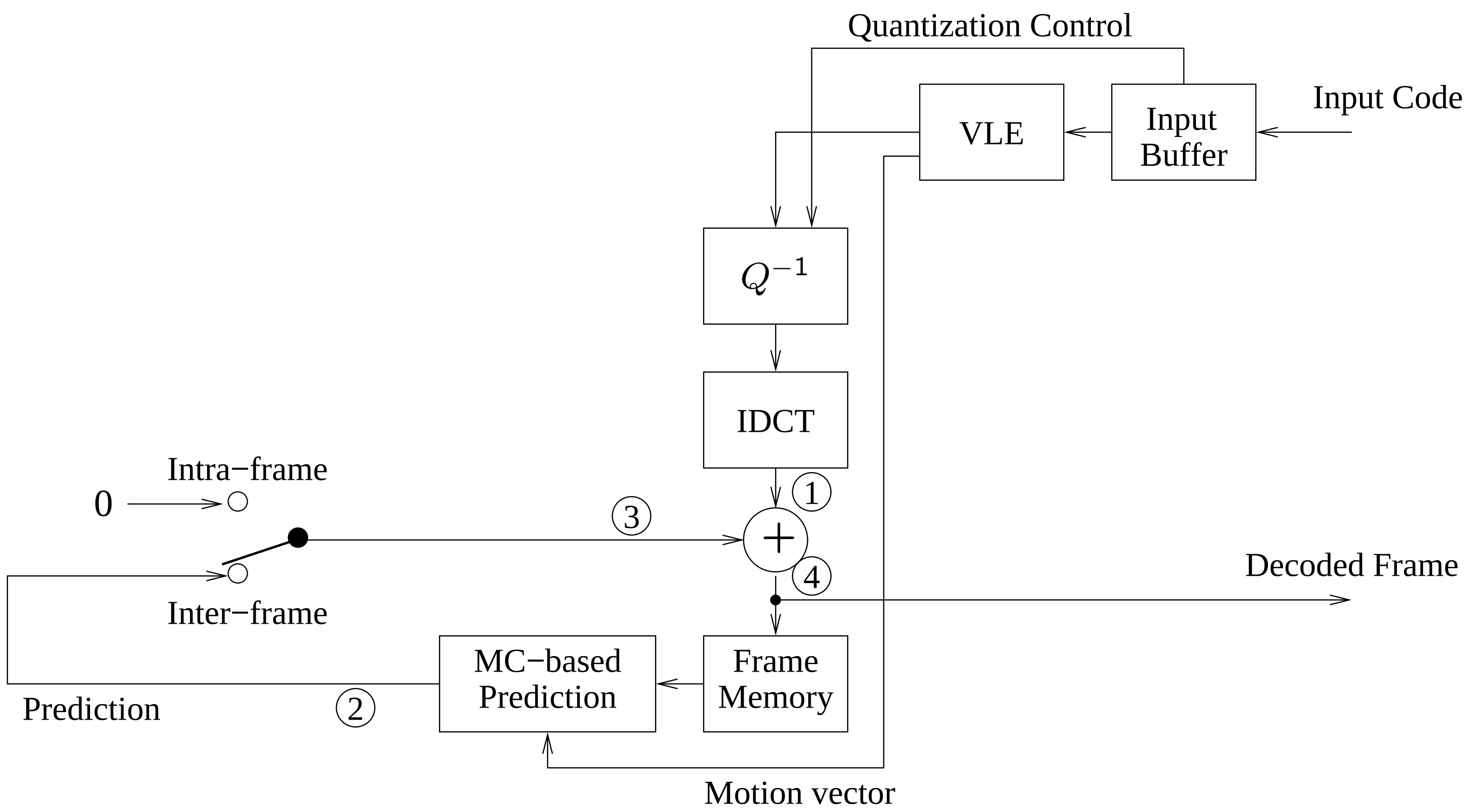
- Note: decoded frames (not the original frames) are used as reference frames in motion estimation.

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



(a) Encoder

Fig. 10.7: H.261 Encoder and Decoder.



(b) Decoder

Fig. 10.7 (Cont'd): H.261 Encoder and Decoder.

Table 10.3: 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'	$\tilde{P}_2$

Table 10.4: 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'	$\tilde{P}_2$

10.6 Further Exploration

- **Text books:**
  - *A Java H.263 decoder* by A.M. Tekalp
  - *Digital Video and HDTV Algorithms and Interfaces* by C.A. Poynton
  - *Image and Video Compression Standards* by V. Bhaskaran and K. Konstantinides
  - *Video Coding: An introduction to standard codecs* by M. Ghanbari
  - *Video processing and communications* by Y. Wang et al.

- **Web sites:** ⟶ Link to Further Exploration for Chapter 10.. including:
  - Tutorials and White Papers on H.261 and H263
  - H.261 and H.263 software implementations
  - An H263/H263+ library
  - A Java H.263 decoder