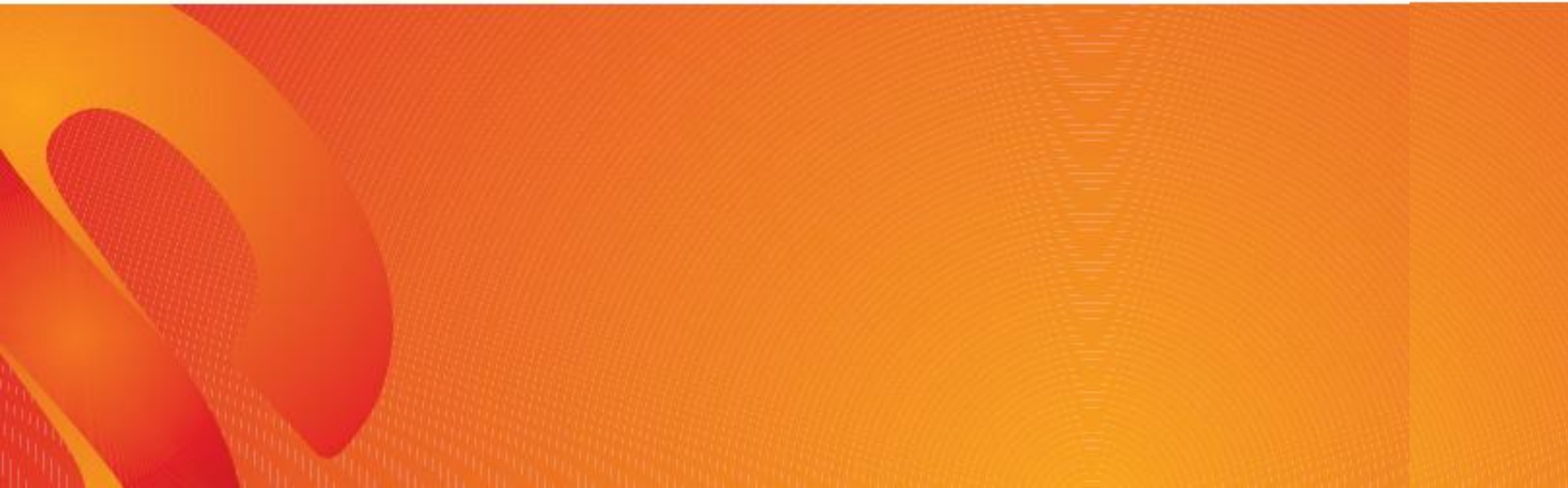
Basic Video Compression Techniques

Lecturer：Jun Xiao
（肖俊）
College of Software and Technology

Content

- Introduction to video compression
- Video compression based on motion compensation
- Search for motion vectors
- H.261
- H.263

1. Introduction to Video Compression



1 Introduction

- Why we need video compression?
  - Uncompressed video data could be extremely large
  - Pose some problems for network communication

  - High-Definition Television (HDTV)
    - 1920x1080
    - 30 frames per second (full motion)
    - 8 bits for each three primary colors

  → Total 1.5 Gb/sec!

  - Each cable channel is 6 MHz
    - Max data rate of 19.2 Mb/sec
    - Reduced to 18 Mb/sec w/audio + control …

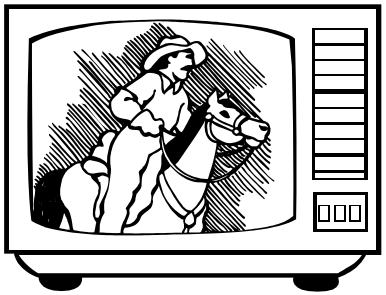
  →Compression rate must be 83:1!

Throwing away some information

- Redundant Spatial Information?
- Color Information?
- Redundant Temporal Information?

Spatial Information

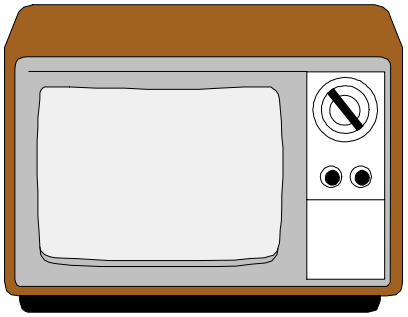
- European broadcast TV standard



625 Half-lines

417 lines

- Resolution is reduced to 352 (width) by 288 (height) pixels
  - **S**ource **I**nput **F**ormat (SIF)



288 pixels

352 pixels

Color

- Human perception is most sensitive to luminance (brightness) changes
- Colour is less important e.g. a black and white photograph is still recognisable
- RGB encoding is wasteful – human perception tolerates poorer colour.
- Use YUV and only encode chrominance (UV) at half resolution in each direction (176 by 144) – Quarter SIF) This gives 0.25 data for U and V compared to Y

Throwing away color

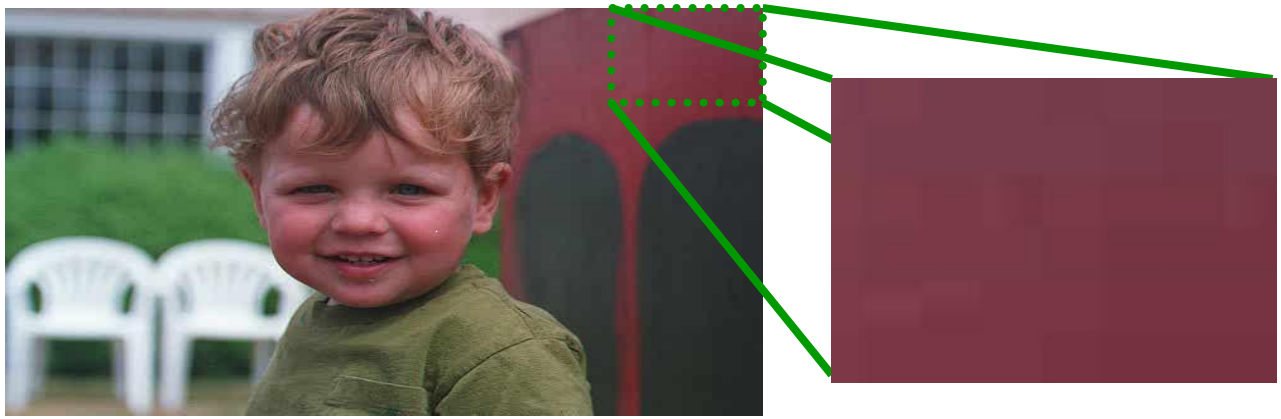
- After throwing away all this information, we still have a data rate of (assuming 8 bits per YUV):
  - Y = (352*288) * 25 * 8 = 20.3 Mb/s
  - U = (352/2 * 288/2) * 25 * 8 = 5.07 Mb/s
  - V = (352/2 * 288/2) * 25 * 8 = 5.07 Mb/s
  - TOTAL (for video) = 30.45 Mb/s
  - MPEG 1 audio runs at 128Kb/s
  - Video CD - Target is 1.5Mb/sec
  - Space for video = 1.5 – 0.128Mb/s = 1.372Mb/s
- So now use compression to get a saving of 22:1

Spatial Compression by JPEG

- A video is a sequence of images – and images can be compressed
- JPEG uses lossy compression – typical compression ratios are 10:1 to 20:1
- We could just compress images and send these

1 Introduction

- Feasibility of video compression
  - Frames in the same scene are very similar, so video data have temporal redundancy
  - Even static images can be compressed at large compression ratio, say nothing of video



Spatial redundancy

1 Introduction

950

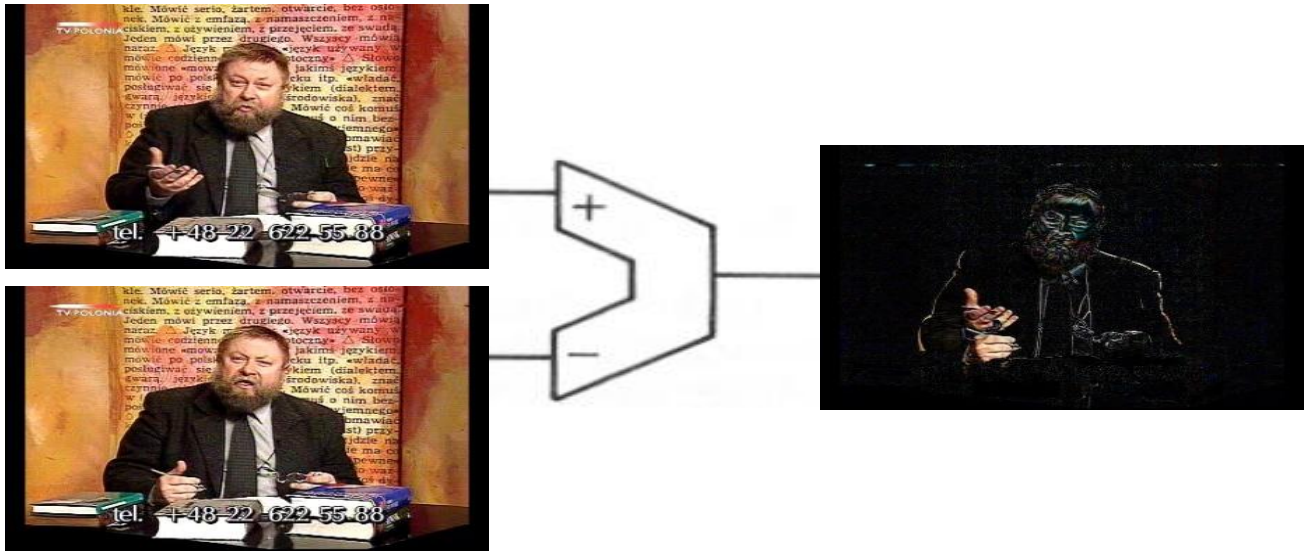
951

952

temporal redundancy

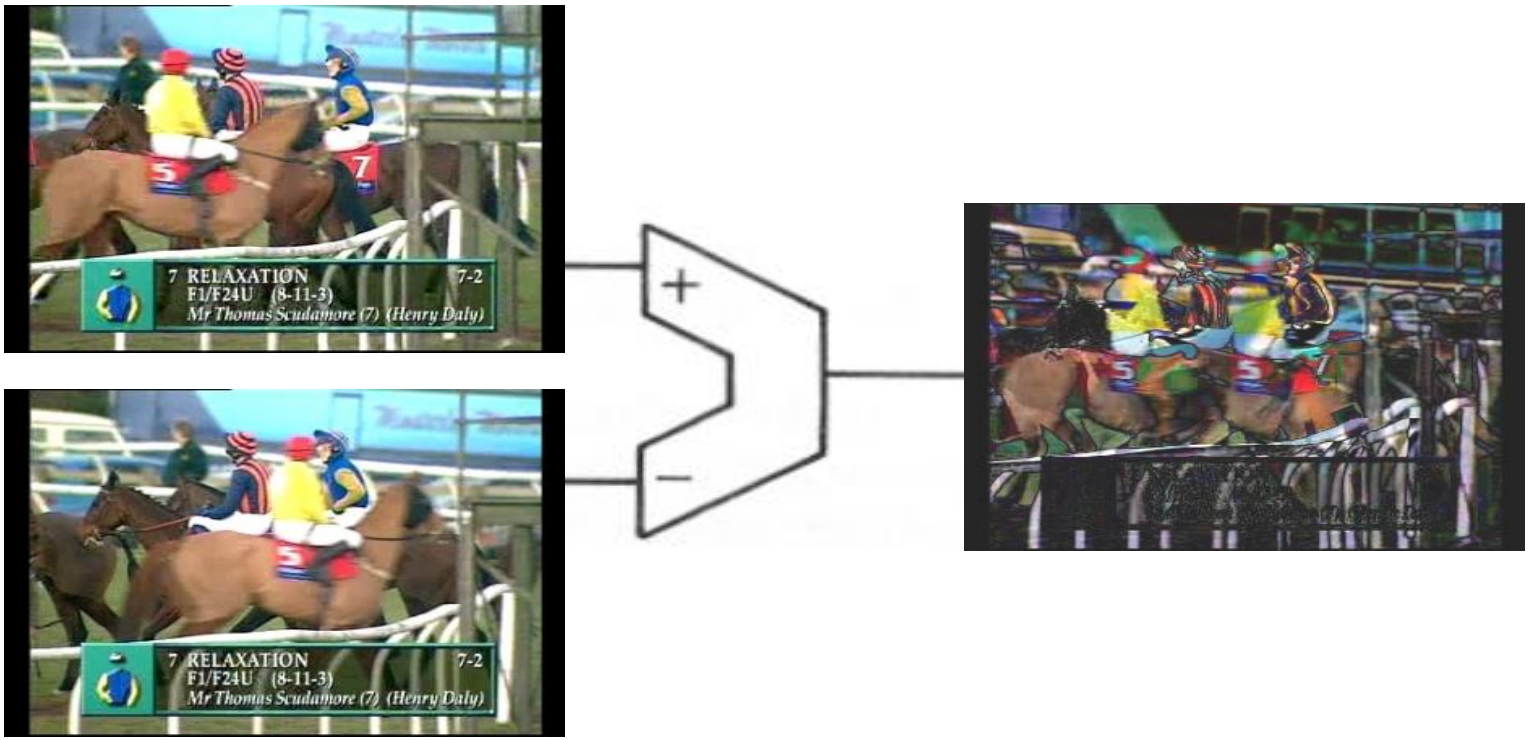
1 Introduction

- Video is a sequence of images stacked in the temporal dimension
- The most simple method: Predictive Coding
  - Subtract images in time order
  - Code the residual error



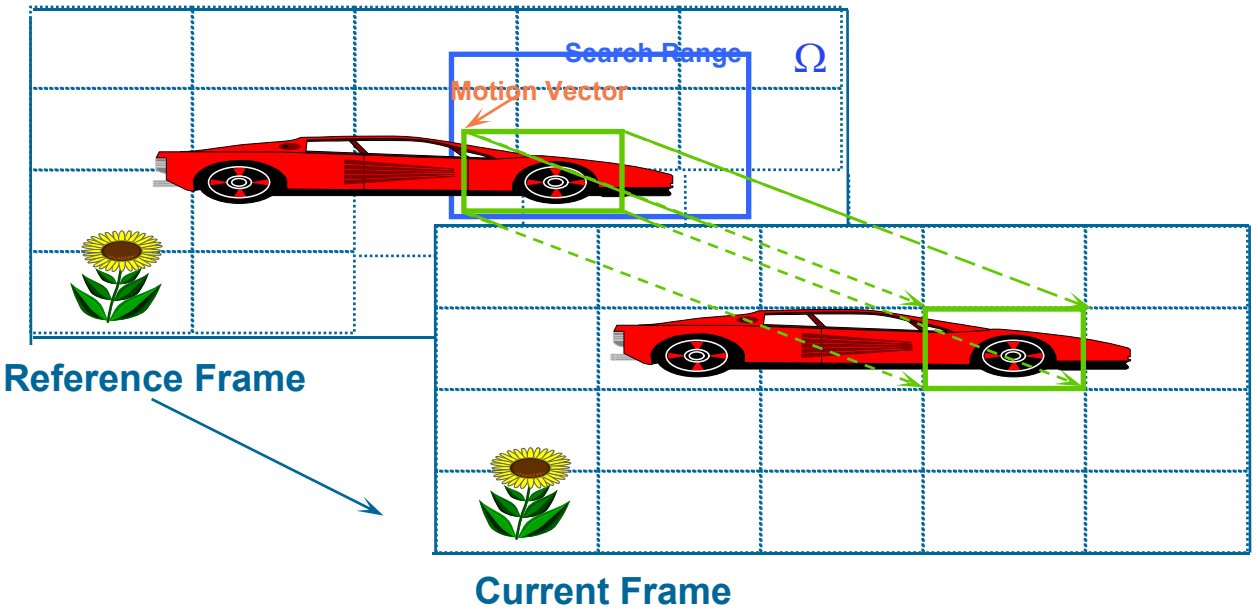
1 Introduction

- Difference coding is good, but often an object will simply change position between frames.
- DCT coding not as good as for ‘sparse’ difference image.

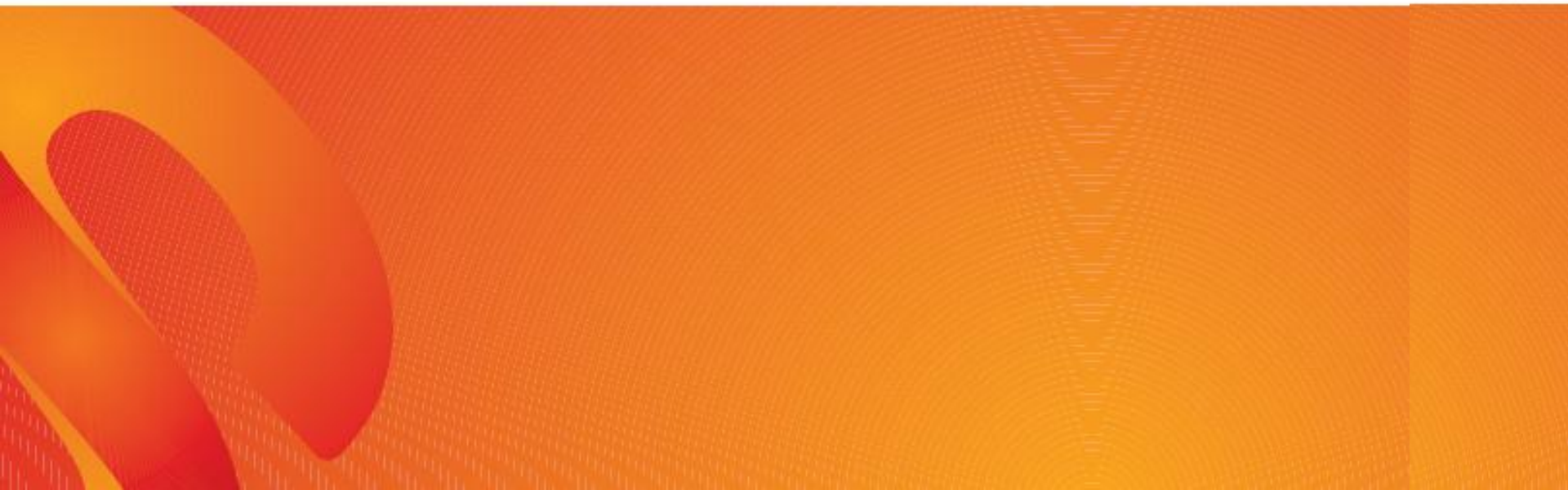


1 Introduction

- Better methods
  - Search for the right parts of the image to subtract from the previous frame.
  - Motion Estimation
  - Motion Compensation



2. Video Compression Based on Motion Compensation



2.1 Temporal Redundancy

- A video: a sequence of images in temporal dimension
- Consecutive frames are usually similar
  - The video has significant temporal redundancy
  - Not every frame coded independently
  - Difference between adjacent frames are coded
- Main cause of difference between frames
  - Camera or object motion
- Motion generators can be compensated
  - Detecting the displacement of corresponding pixels or regions
  - Measuring their differences (motion compensation MC)

2.1 Temporal Redundancy

- Both spatial and temporal redundancy exist in moving sequential pictures

- Principles of moving pictures encoding： reduce spatial redundancy and temporal redundancy
  - Intra-Frame: similar as JPEG
  - Inter-Frame: based on motion prediction and compensation
    - P frame、B frame
    - Multi-frame references（H.264）

2.2 Motion Compensation

Basic idea of ***Motion Compensation***:

- Many "moving" images or image sequences consist of a static background with one or more moving foreground objects. We can get coding advantage from this.
- we code the first frame by baseline JPEG and use this frame as ***reference image***.
- Treat the second image block by block and compare each block with the blocks in the reference image.
- For blocks that have identical block in reference image, we only send a special code instead of whole code.
- For other blocks, we just encode them as usual.

2.2 Motion Compensation

- The three main steps
  - Motion estimation: motion vector search
  - Motion-compensation-based prediction
  - Derivation of the prediction error

- Each image is divided into *macroblocks* of size N x N.
  - By default, N = 16 for luminance images. For chrominance images, N = 8 if 4:2:0 chroma subsampling is adopted.

2.2 Motion Compensation

- 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* **MV**.
  - Figure 10.1 shows the case of *forward prediction* in which the Reference frame is taken to be a previous frame.

2.2 Motion Compensation

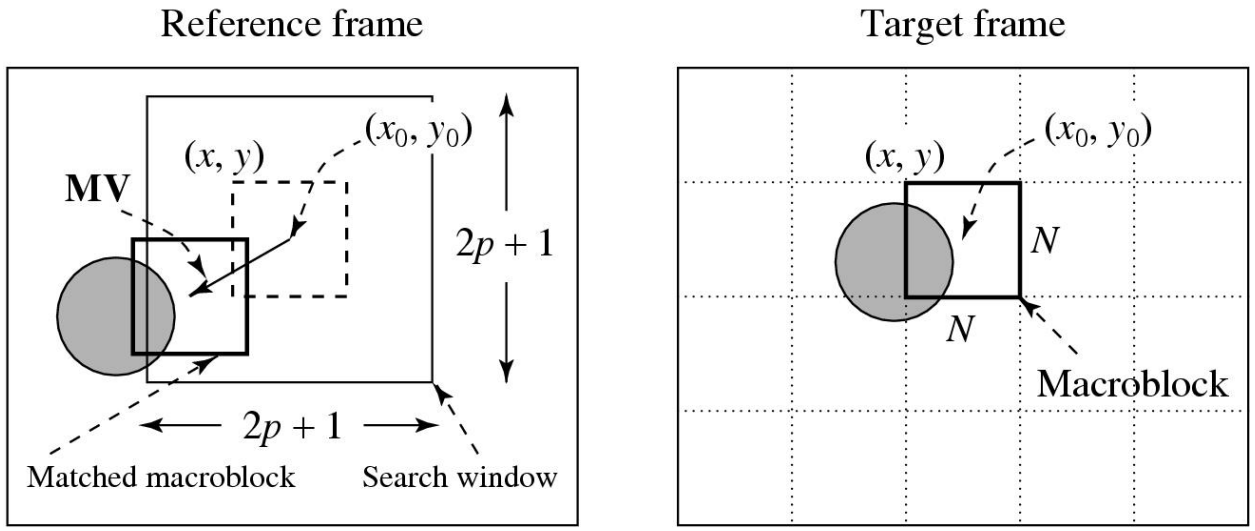
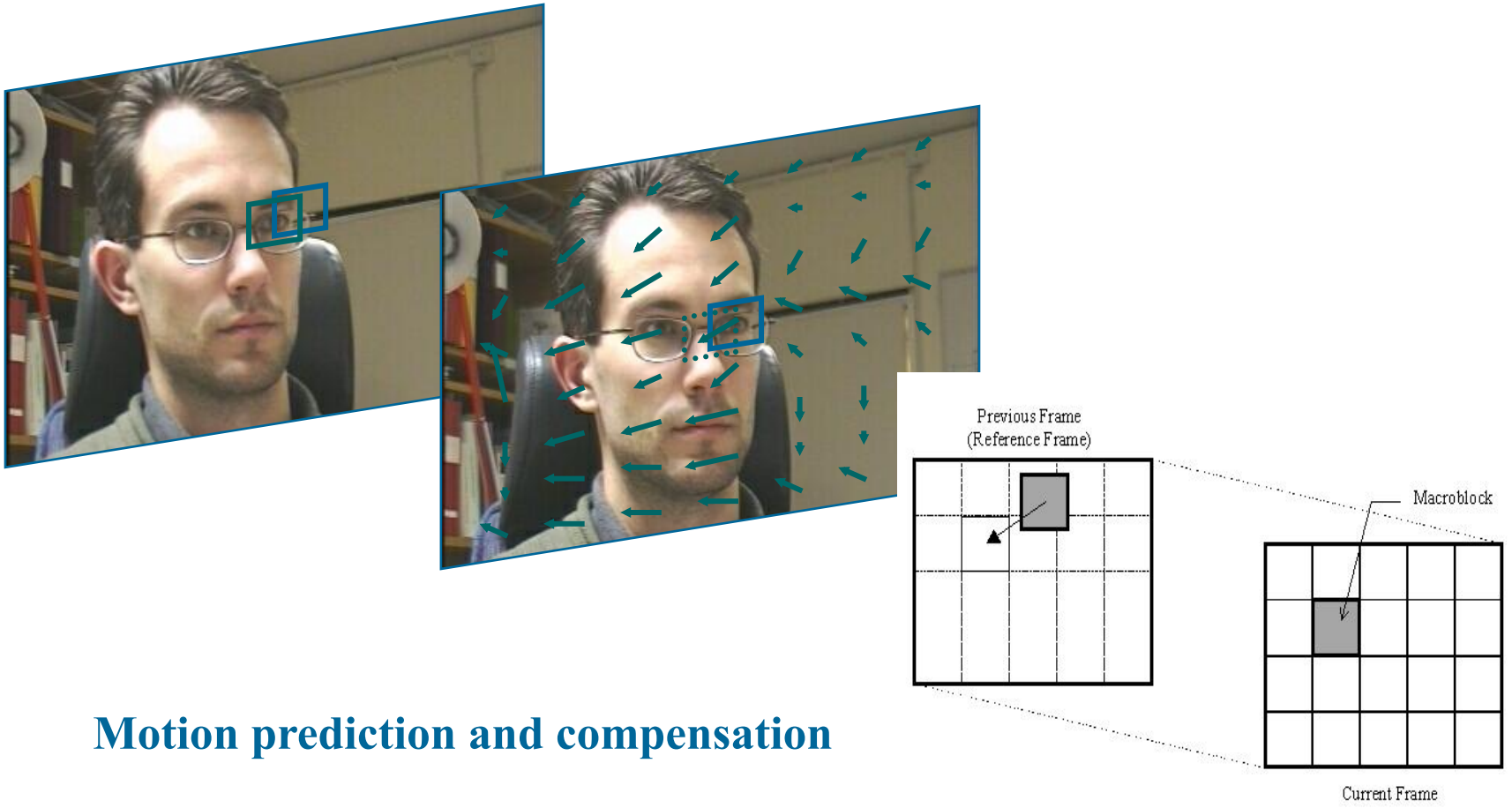


Fig. 10.1: Macroblocks and Motion Vector in Video Compression.

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

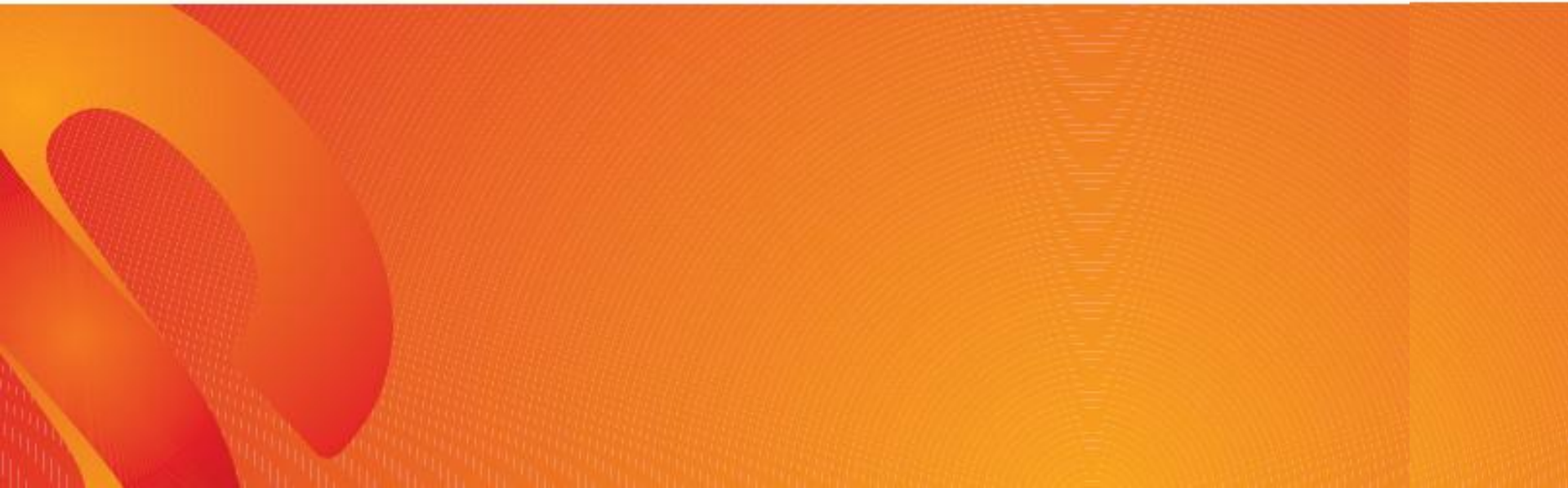
  This makes a search window of size $(2p + 1) \times (2p + 1)$.

2.2 Motion Compensation

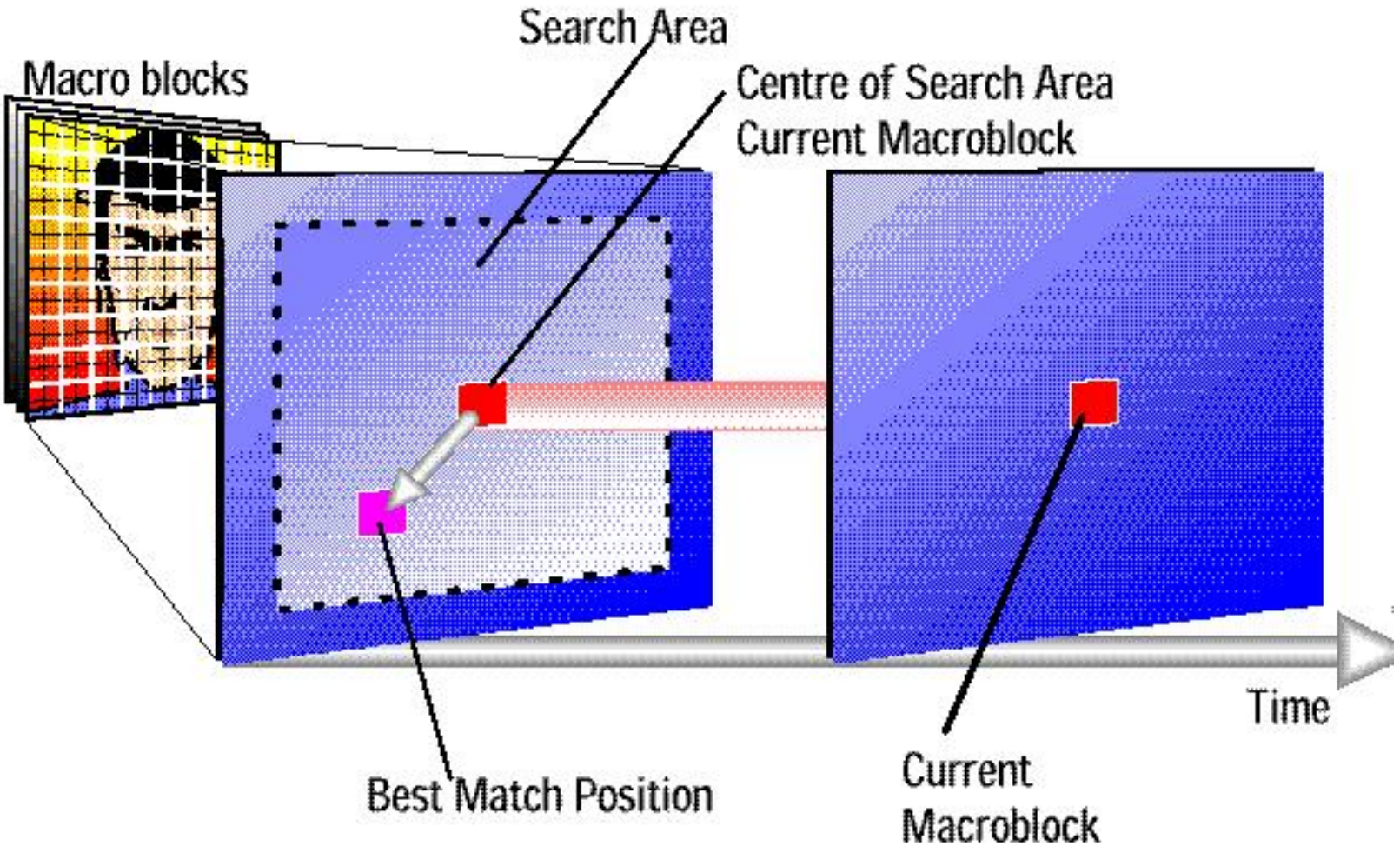


Motion prediction and compensation

3. Search For Motion Vectors



Search For Motion Vectors



3.1 Criteria of matching

- Motion vector（MV）search: a matching problem, called correspondence problem
- Horizontal and vertical displacement i, j are in the range [-p,p], a search window of size （2p+1）*（2p+1）
- The goal: find (i,j) minimize the distance between two macroblocks

$$C(x+k, y+l): pixel\ in\ target\ frame\ macroblock$$

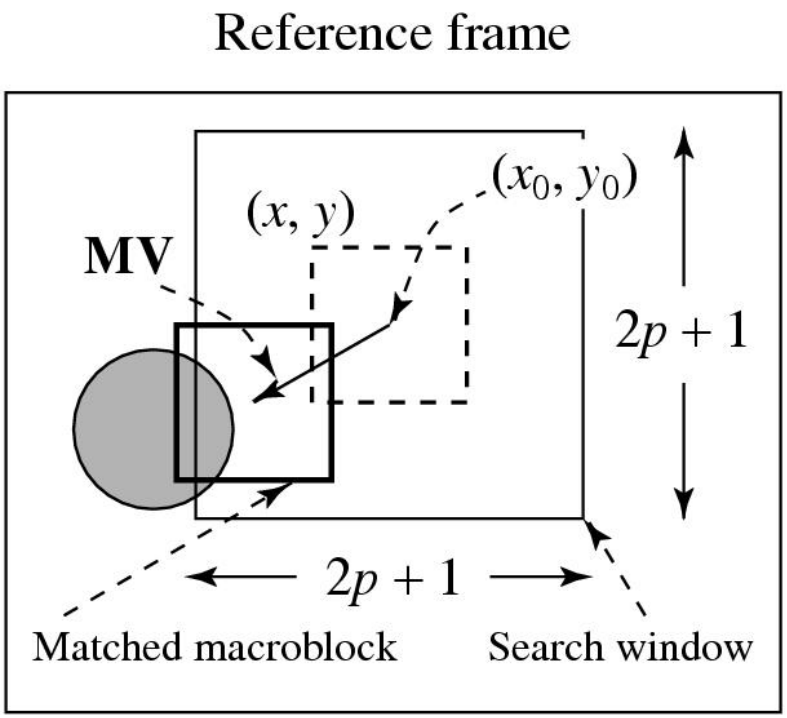
$$R(x+i+k, y+j+l): pixel\ in\ reference\ macroblock$$

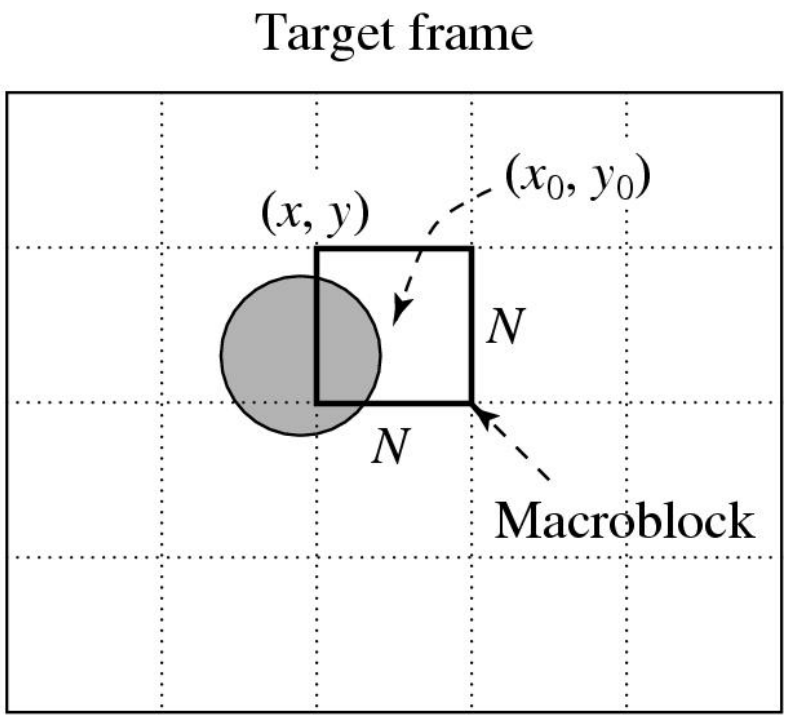
$$when\ motion\ vector\ is (i, j)$$

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

$$(u,v) = [(i,j) \,|\, MAD(i,j) \quad is\ minimum,\ \ i \in [-p,p], j \in [-p,p]]$$

3.1 Criteria of matching





3.2 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 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 $(2p + 1) \quad (2p + 1) \quad N^2 \quad 3$ $O(p^2 N^2)$.**

3.2 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
```

3.3 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.**

3.3 2D-Logarithmic-search

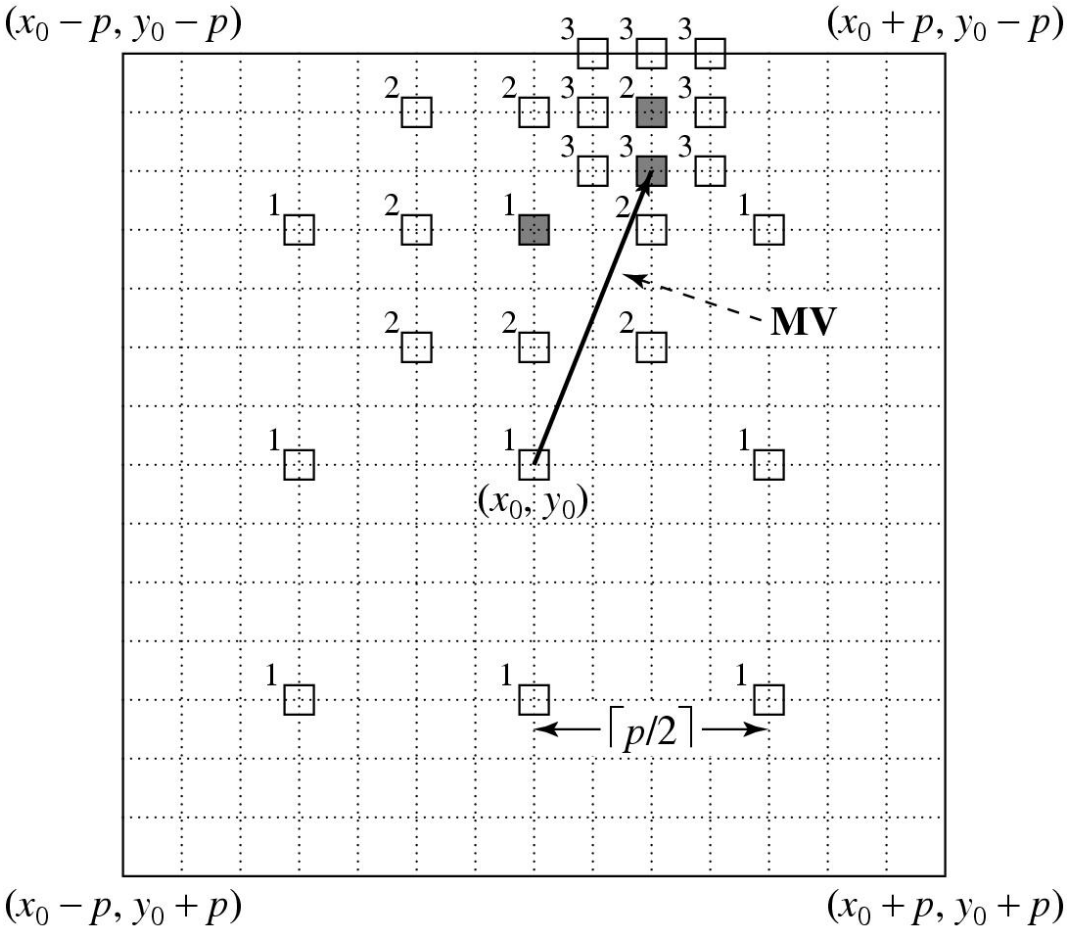


Fig. 10.2: 2D Logarithmic Search for Motion Vectors.

3.3 2D-Logarithmic-search

begin

offset = ;

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

3.3 2D-Logarithmic-search

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

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

3.4 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.

3.4 Hierarchical Search

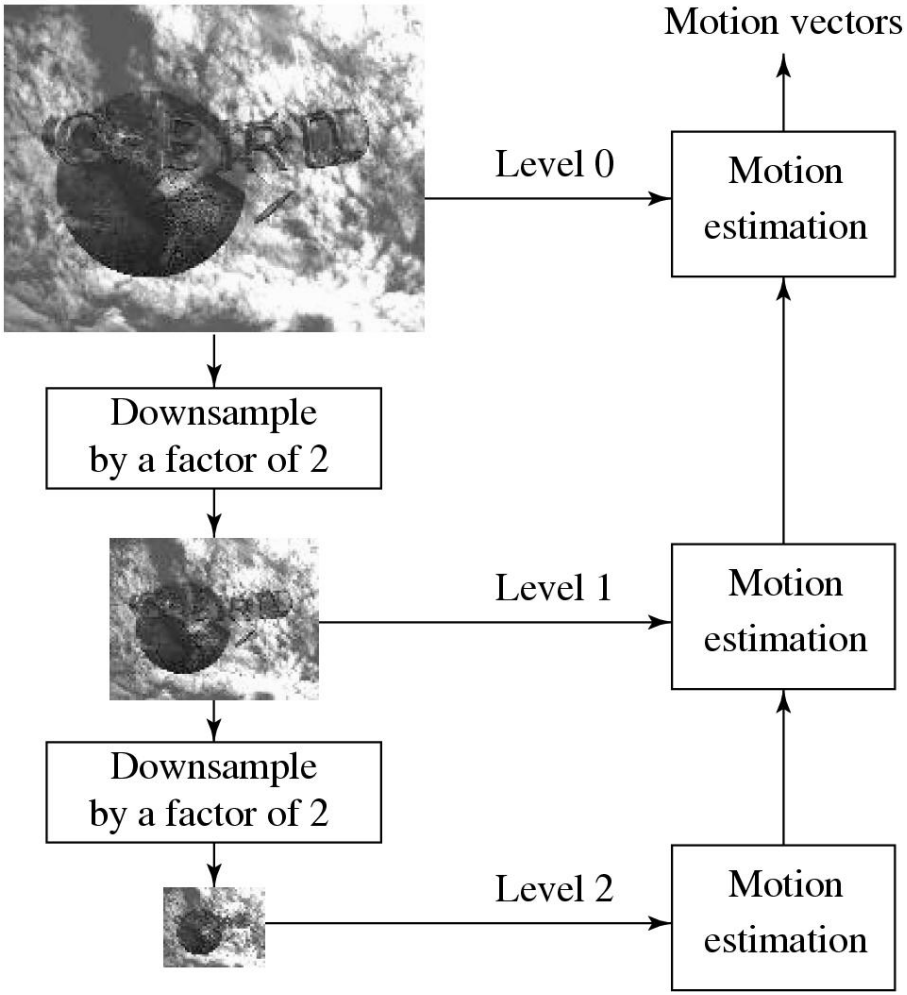


Fig. 10.3: A Three-level Hierarchical Search for Motion Vectors.

3.4 Hierarchical Search

- 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 can be outlined as follows:

3.4 Hierarchical Search

PROCEDURE 10.3 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 method to get initial estimated $\mathbf{MV}(u^k, v^k)$ at Level k;

while last ≠ TRUE

{

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

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

if $k = 1$ then last = TRUE;

$k = k - 1$;

Assign (x^k_0, y^k_0) 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

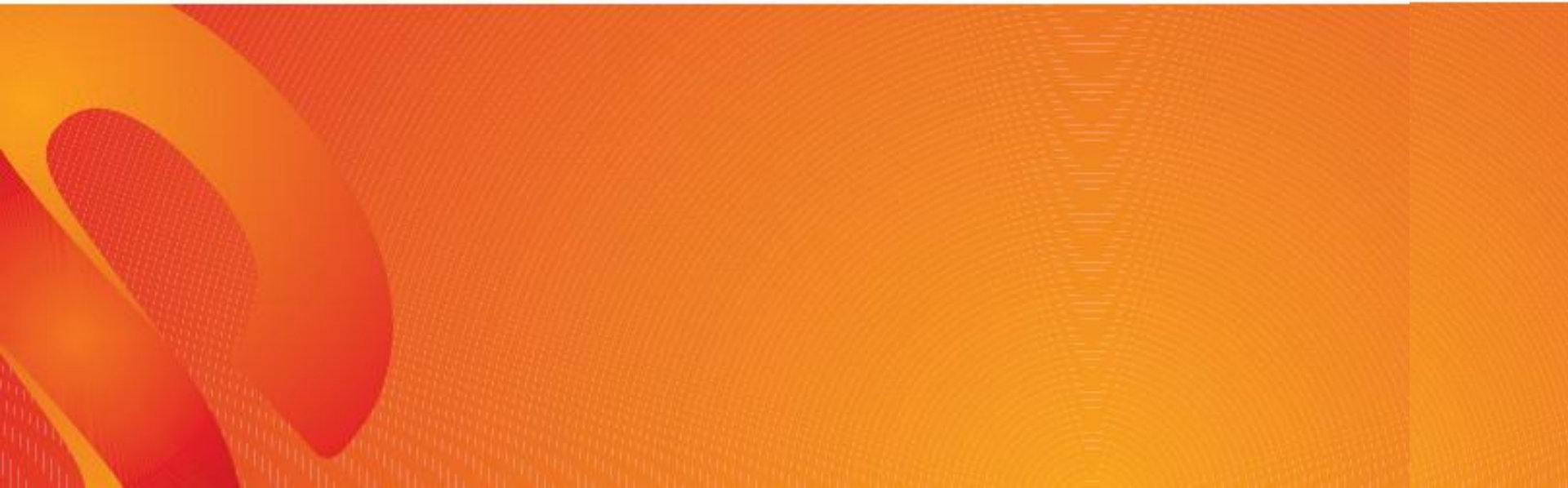
Search Method	OPS_per_second for 720×480 at 30 fps	
	$p = 15$	$p = 7$
Sequential search	29.89×10^9	7.00×10^9
2D Logarithmic search	1.25×10^9	0.78×10^9
3-level Hierarchical search	0.51×10^9	0.40×10^9

3.5 Other methods

- Search of motion vector is one of major steps in video compression, many methods are proposed to improve the efficiency.

- Some methods are listed as follows：
  - Three Step Search： TSS
  - Conjugate Direction Search： CDS
  - Cross Search Algorithm： CSA
  - New TSS： NTSS
  - Four-Step Search： FSS
  - Diamond Search Algorithm： DS
  - Adaptive Block Matching Algorithm： ABMA

4. H.261



4.1 Overview of H.261

- **H.261**: An earlier digital video compression standard (formed in 1990), 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 x 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 bidirectional video conferencing.

4.1 Overview of H.261

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

4.1 Overview of H.261

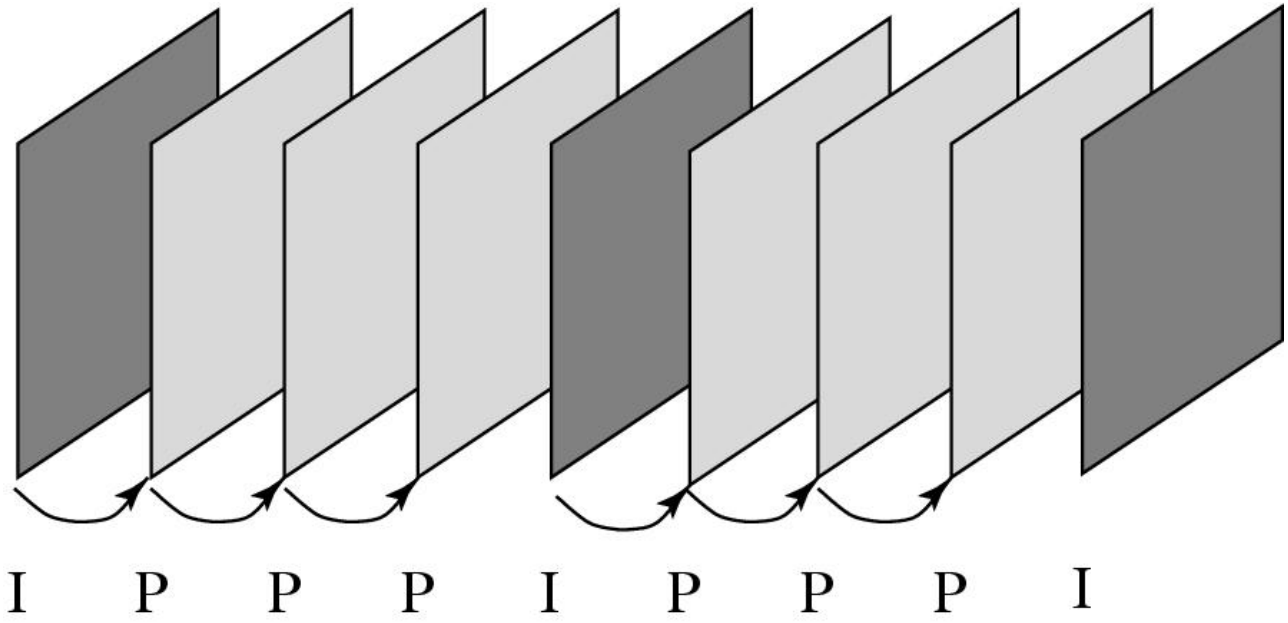


Fig. 10.4: H.261 Frame Sequence.

4.1 Overview of H.261

- 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 $\pm$ 15 pixels, i.e., p = 15.**

4.2 Intra-Frame Coding

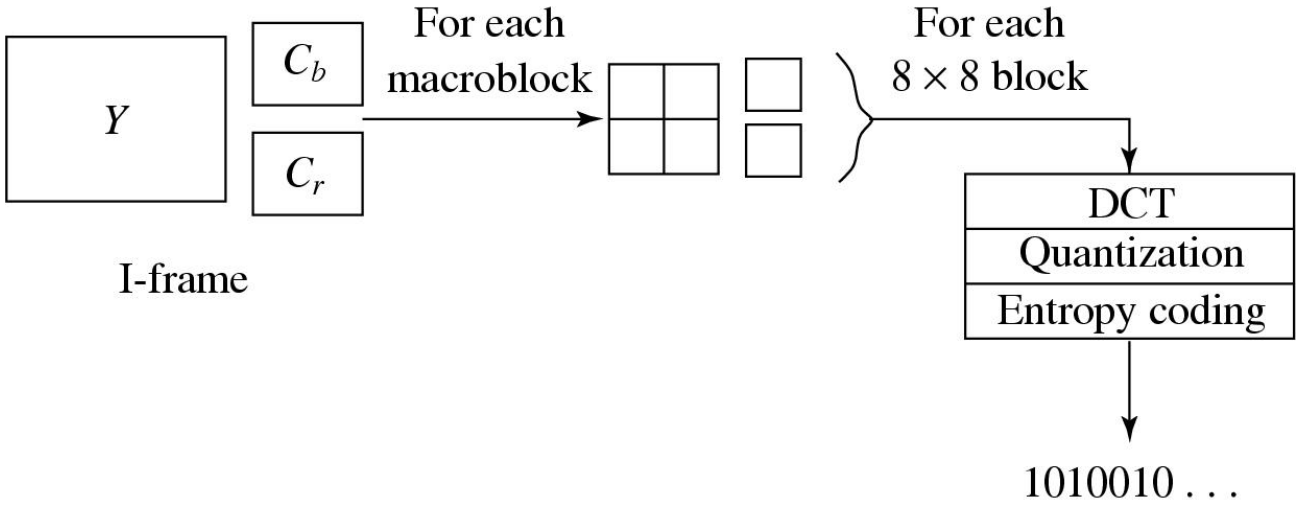
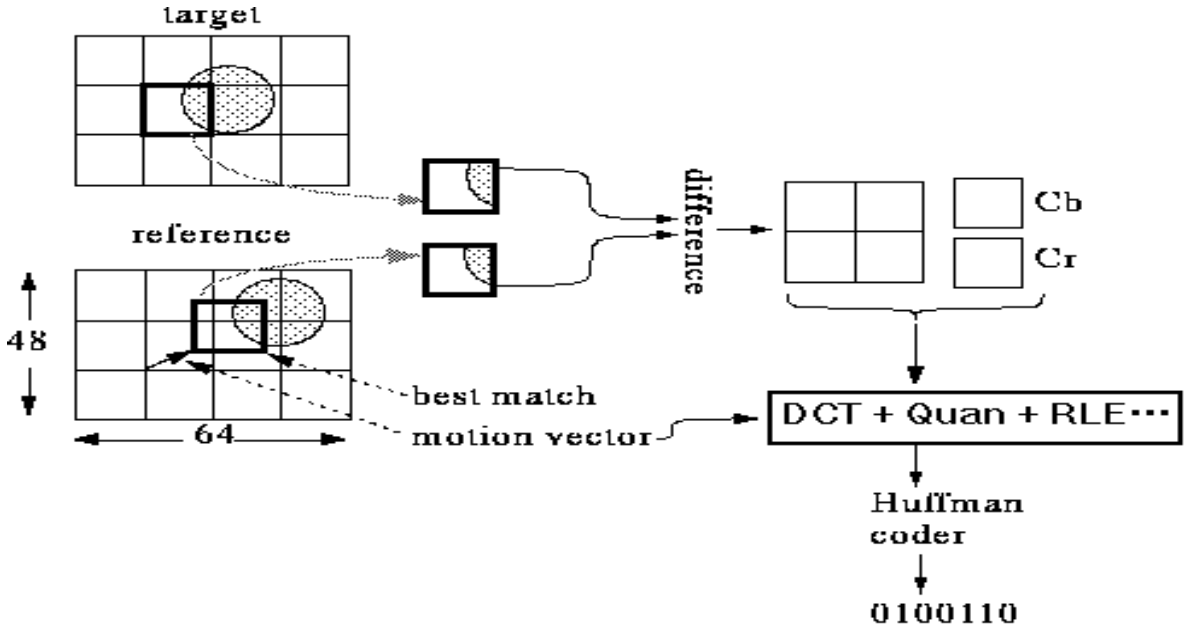


Fig. 10.5: I-frame Coding.

- **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.

4.3 Inter-Frame predictive Coding



- 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 x 8 blocks go through DCT, quantization, zigzag scan and entropy coding procedures.

4.3 Inter-Frame predictive Coding

- 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:

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

4.4 Quantization in H.261

- Quantization uses a constant (*step_size*) for all DCT coefficient in a macroblock, *step_size* is one of the even value from 2 to 62.
- In intra mode, step_size=8 is always used for DC coefficient.

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

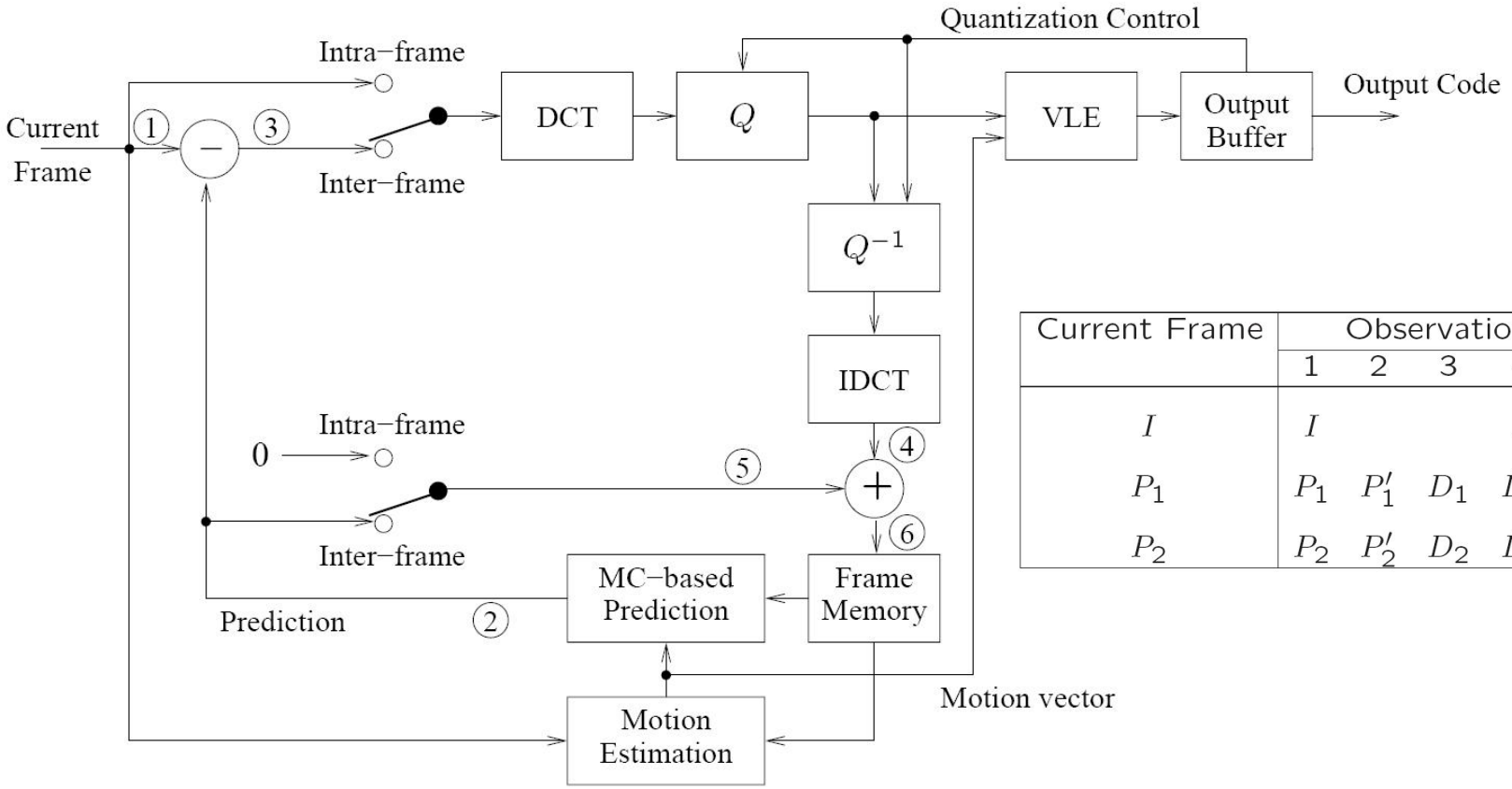
- For all other coefficients

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

- scale is an integer in the range of [1, 31]

4.5 H.261 encoder and decoder

- Encoder and data flow



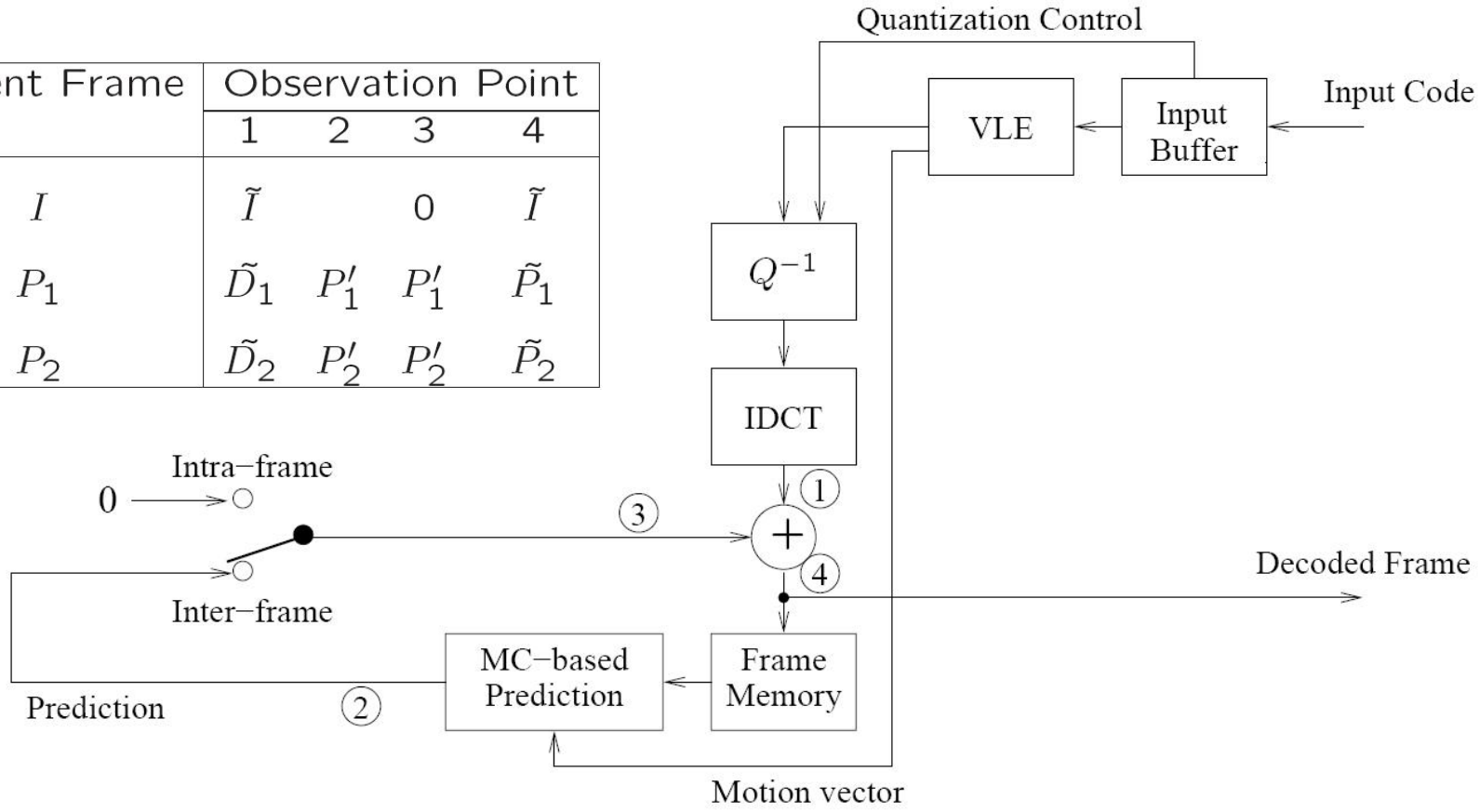
Current Frame	Observation Point					
	1	2	3	4	5	6
I	I			$\tilde{I}$	0	$\tilde{I}$
P_1	P_1	P_1'	D_1	$\tilde{D}_1$	P_1'	$\tilde{P}_1$
P_2	P_2	P_2'	D_2	$\tilde{D}_2$	P_2'	$\tilde{P}_2$

(a) Encoder

4.5 H.261 encoder and decoder

- Decoder and data flow

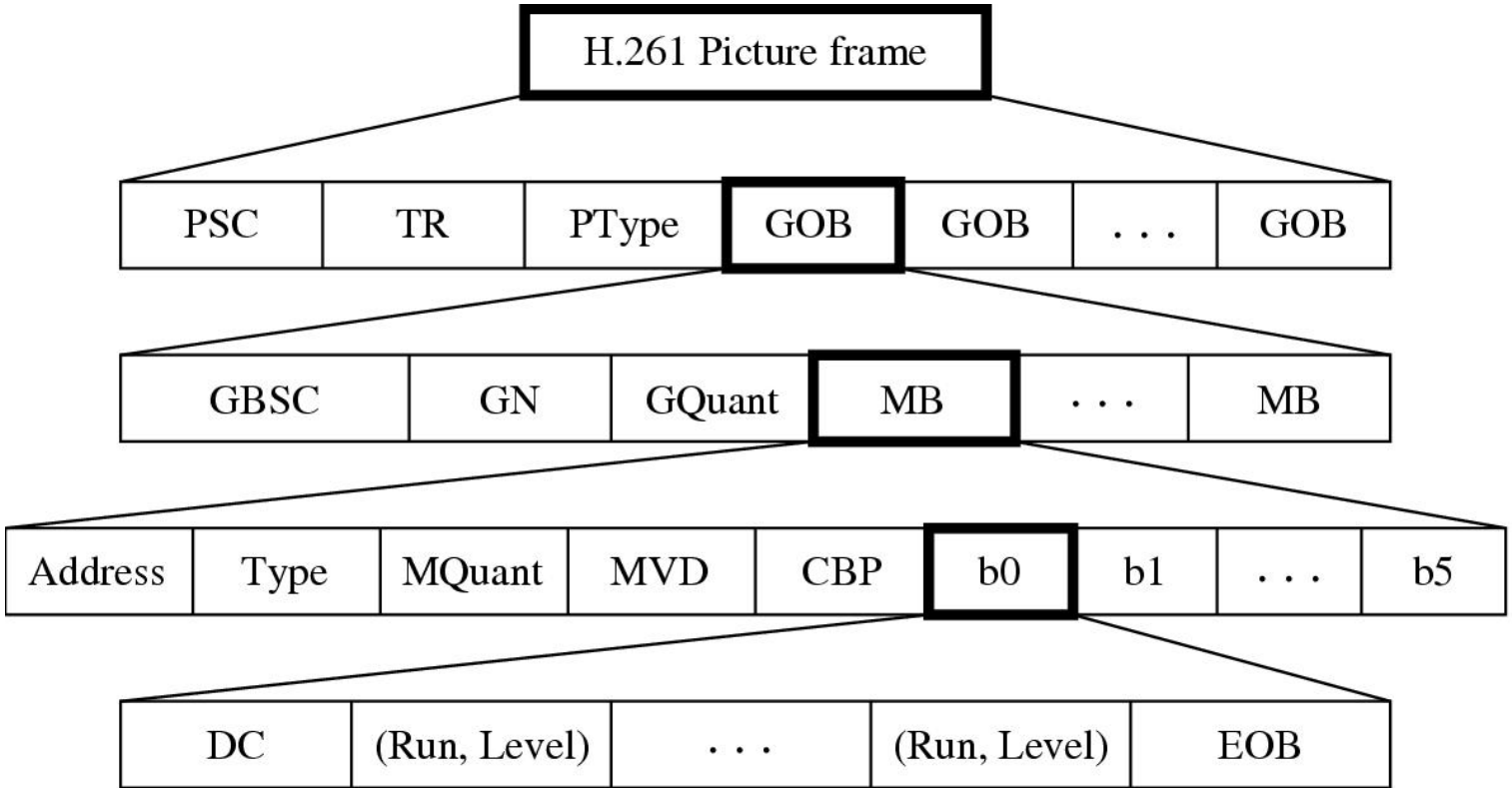
Current Frame	Observation Point			
	1	2	3	4
I	$\tilde{I}$		0	$\tilde{I}$
P_1	$\tilde{D_1}$	P_1'	P_1'	$\tilde{P_1}$
P_2	$\tilde{D_2}$	P_2'	P_2'	$\tilde{P_2}$



(b) Decoder

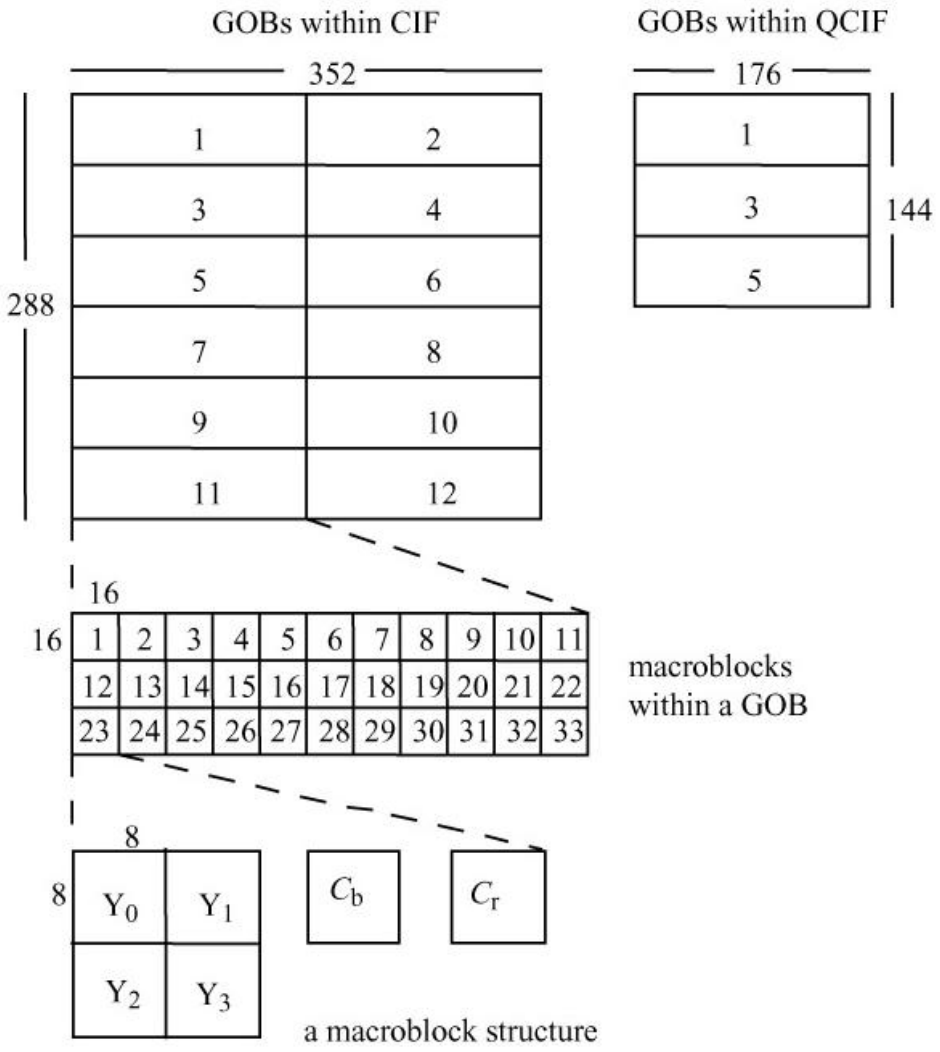
4.6 H.261 Video Bitsteam Syntax

- H.261 video bitsteam syntax ： four layers
  - Picture, Group of Blocks, Macroblock and Block



4.6 H.261 Video Bitsteam Syntax

- In H.261, Frame （Picture） is the highest layer



4.6 H.261 Video Bitsteam Syntax

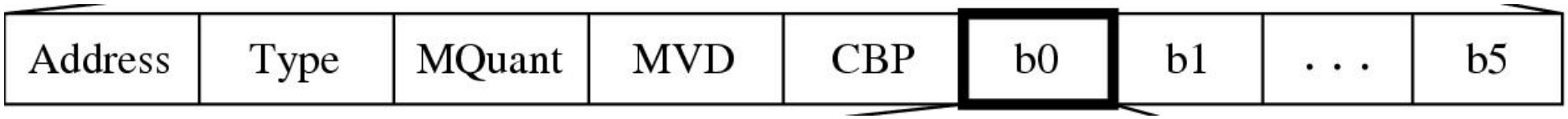
- The Picture Layer
  - PSC (Picture Start Code) delineates boundaries between pictures. TR (Temporal Reference) provides a time-stamp for the picture.

4.6 H.261 Video Bitsteam Syntax

- **The GOB Layer:** H.261 pictures are divided into regions of 11 x 3 macroblocks, each of which is called a Group of Blocks (GOB).
  - 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).
  - 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).

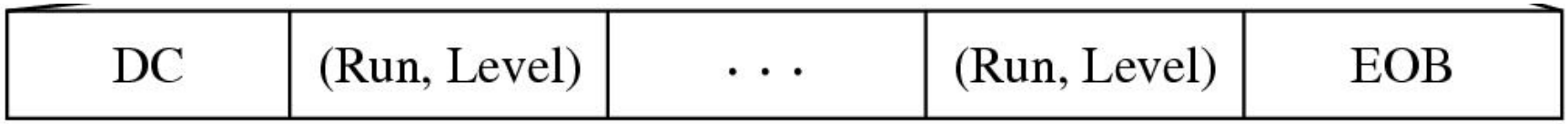
4.6 H.261 Video Bitsteam Syntax

- 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).



4.6 H.261 Video Bitsteam Syntax

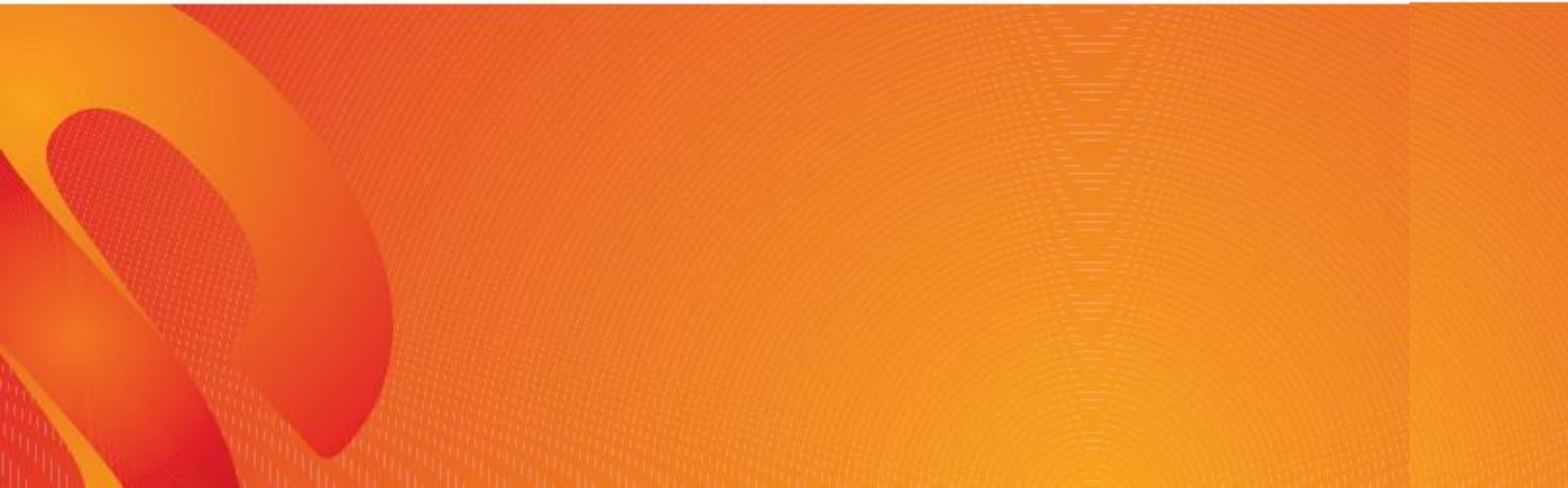
- The Block layer:
  - For each 8 x 8 block, the bitstream starts with DC value, followed by pairs of length of zerorun (Run) and the subsequent non-zero value (Level) for ACs, and finally the End of Block (EOB) code. The range of Run is [0; 63]. Level reflects quantized values — its range is [−127,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



4.6 H.261 Video Bitsteam Syntax

- Syntax of H.261 video bitstream
  - PSC (Picture Start Code)
  - Ptype (Picture Type)
  - GBSC (GOB Start Code)
  - Gquant (GOB Quantizer)
  - Mquant (MB Quantizer)
  - CBP (Coded Block Pattern)
  - TR (Temporal Reference)
  - GOB (Group of Blocks)
  - GN (Group Number)
  - MB (Macroblock)
  - MVD (Motion Vector Data)
  - EOB (End of Block)

5. H.263



5.1 overview of H.263

- An improve standard for videoconferencing and other audiovisual services on PSTN, adopted by ITU-T Study Group 15 in 1995
- Aims at low bit-rate communications at bit-rates of less than 64 kbps.

- H.263 supports sub-QCIF, 4CIF和16CIF, GOBs don't have a fixed size.

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

5.2 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 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).

5.2 Group of Blocks (GOB)

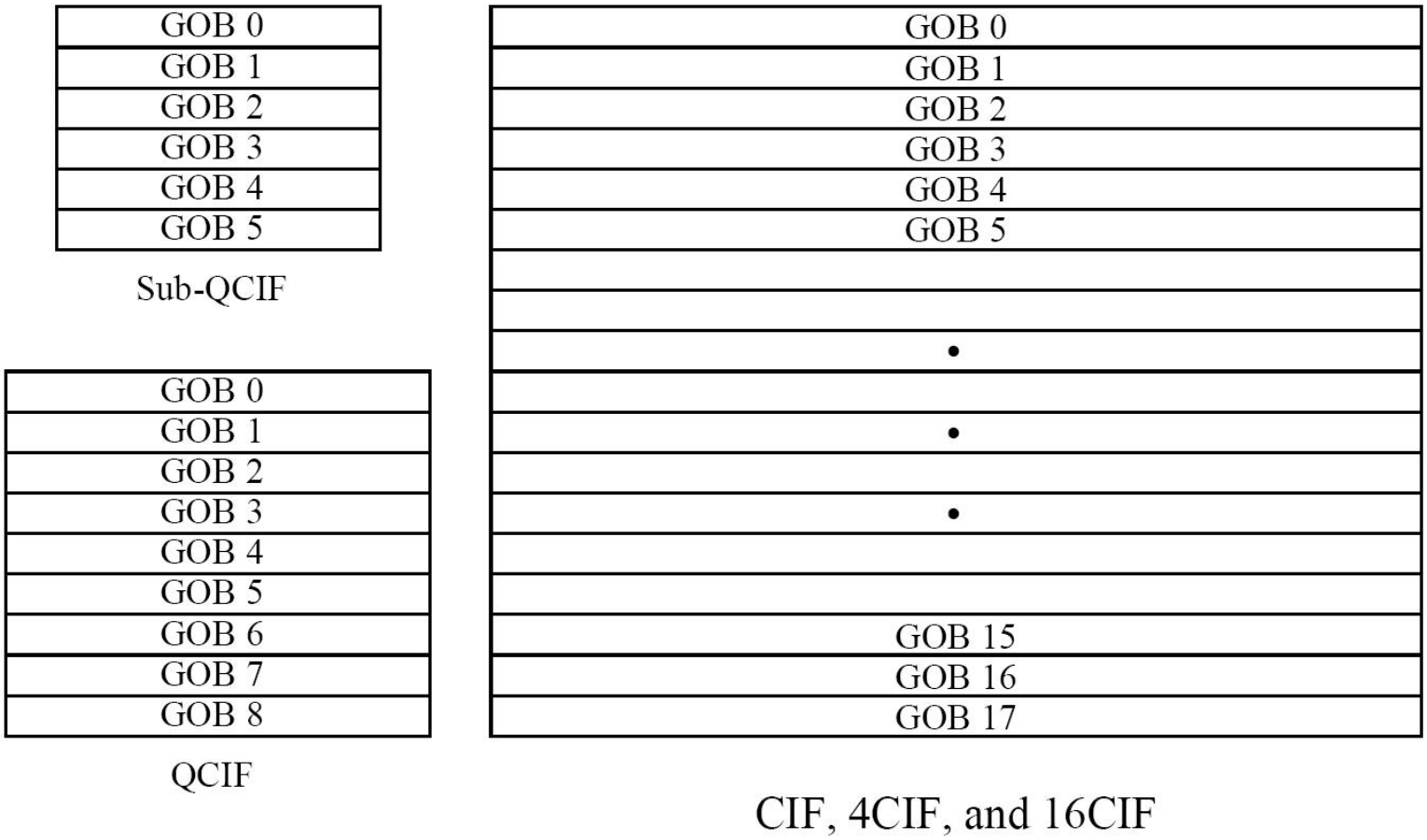


Fig. 10.10 Arrangement of GOBs in H.263 Luminance Images.

5.3 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. 10.11 (a)).

- For the Macroblock with MV(u, v):

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

	MV2	MV3
MV1	**MV**	

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

5.3 Motion Compensation in H.263

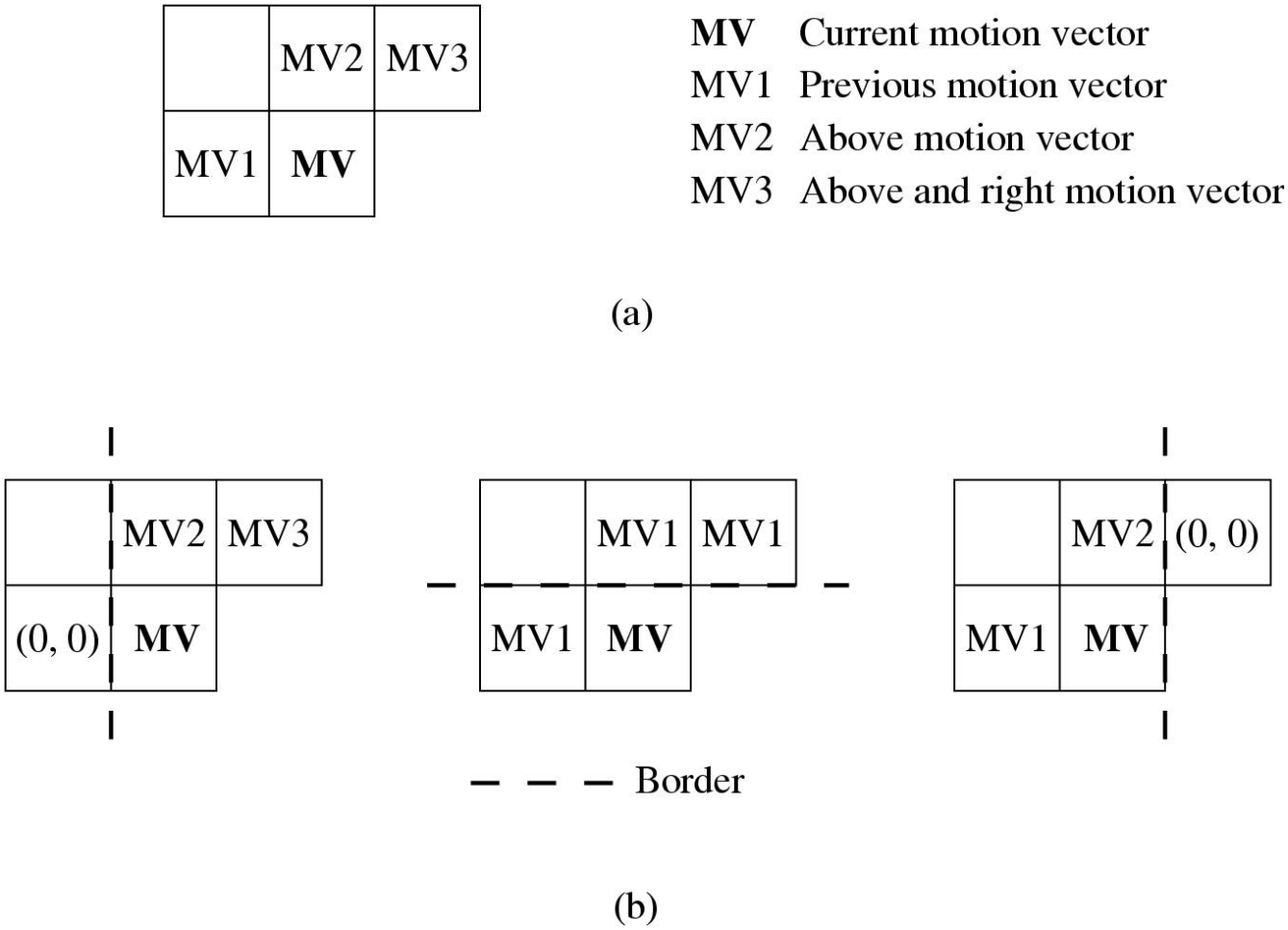


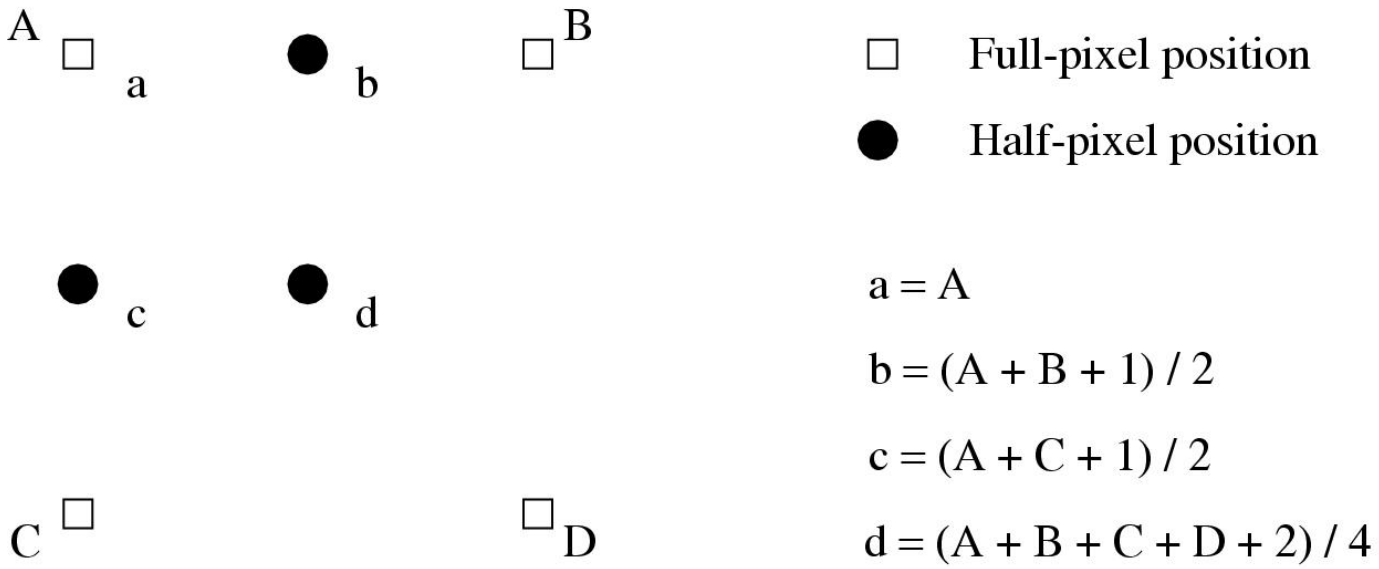
Fig. 10.11 Prediction of Motion Vector in H.263.

5.3 Motion Compensation in H.263

- 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 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.

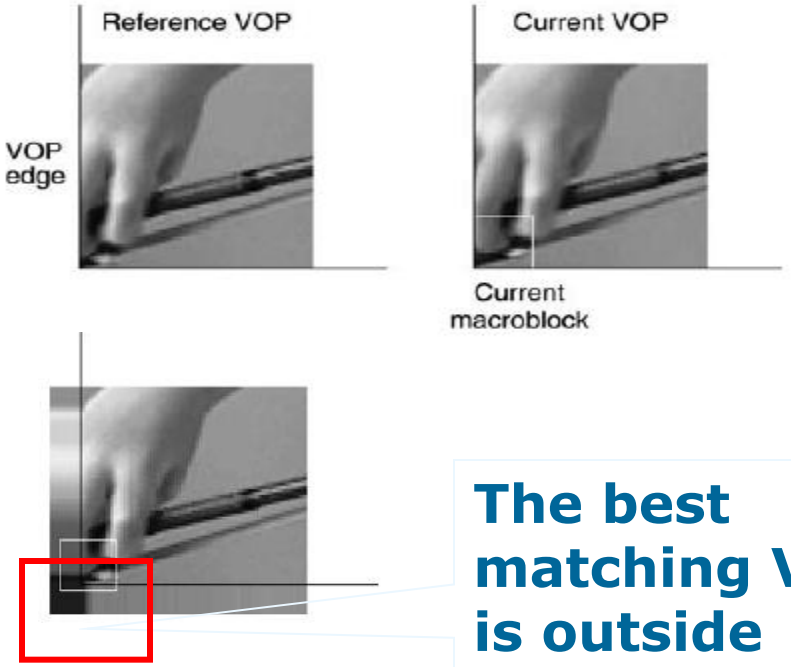


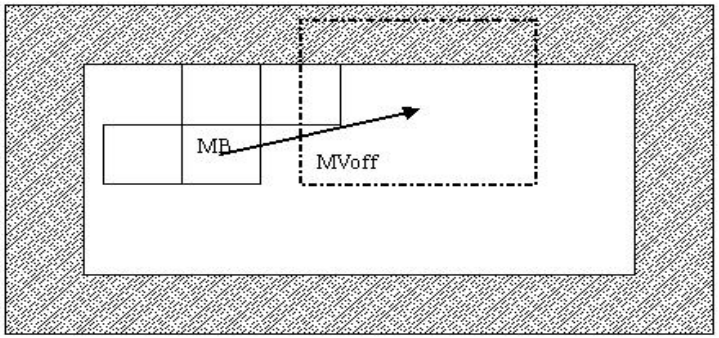
5.4 Optional H.263 Coding Modes

- Negotiable options besides its core algorithm：
  - Unrestricted motion vector mode.
  - Syntax-based arithmetic coding mode
  - Advanced prediction mode (4 MV for a macroblock)
  - PB-frames mode

5.4 Optional H.263 Coding Modes

- Unrestricted Motion Vectors
  - Referenced not restricted by image boundary
- Fulfilled by enlarging the coding image edges for a certain size(e.g. macroblock size)





5.4 Optional H.263 Coding Modes

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.

- 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.

3. **Advanced prediction mode:**

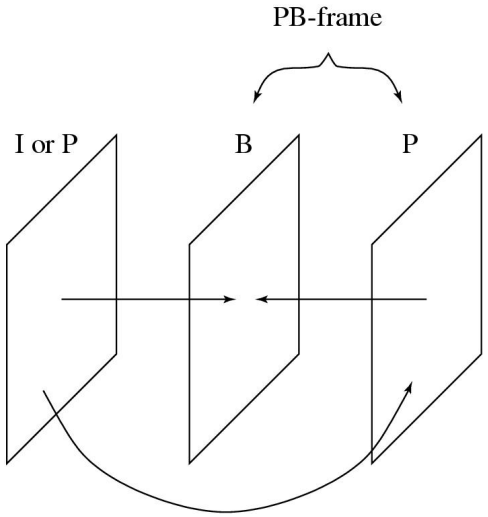
- In this mode, the macroblock size for MC is reduced from 16 to 8.

- Four motion vectors (from each of the 8 x 8 blocks) are generated for each macroblock in the luminance image.

5.4 Optional H.263 Coding Modes

4. **PB-frames mode:**

- **In H.263, a PB-frame consists of two pictures being coded as one unit, as shown Fig. 10.13.**
- **The use of the PB-frames mode is indicated in PTYPE.**
- **The PB-frames mode yields 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.**



5.5 H.263+ and H.263++

- Second version of H.263
  - Redefine the unrestricted motion vector mode
  - Slice structure is used to replace GOB
  - Implements Temporal,SNR,and Spatial scalabilities
  - Supports improved PB-frames mode
  - Use deblocking filtersto reduce blocking effects
- H.263++: the new extension
  - Enhanced reference picture selection（ERPS）
  - Data partition slice (DPS)
  - Additional supplemental enhancement information

The End

Thanks!

Email: junx@cs.zju.edu.cn