

# *Digital Video Compression Fundamentals and Standards*

Wei-Yi Wei

E-mail: r97942024@ntu.edu.tw

Graduate Institute of Communication Engineering

National Taiwan University, Taipei, Taiwan, ROC

## **Abstract**

*Over the past decades, digital video compression technologies have become an integral part of the way we create, communicate and consume visual information. Digital video communication can be found today in many application sceneries such as broadcast services over satellite and terrestrial channels, digital video storage, wires and wireless conversational services and etc. The data quantity is very large for the digital video and the memory of the storage devices and the bandwidth of the transmission channel are not infinite, so it is not practical for us to store the full digital video without processing. For instance, we have a 720 x 480 pixels per frame, 30 frames per second, total 90 minutes full color video, then the full data quantity of this video is about 167.96 G bytes. Thus, several video compression algorithms had been developed to reduce the data quantity and provide the acceptable quality as possible as can. This paper starts with an explanation of the basic concepts of video compression algorithms and then introduces several video compression standards.*

## **1. Introduction**

Why an image can be compressed? The reason is that the correlation between one pixel and its neighbor pixels is very high, or we can say that the values of one pixel and its adjacent pixels are very similar. This is called the intraframe correlation in video compression because it is the correlation in a single frame. Once the correlation between the pixels is reduced, we can reduce the storage quantity. The image compression method is also applied to compression video. However, there still exists temporal correlation. The video is composed of a large number of still images, and due to the images are taken at short time distance, the two neighboring images are similar. Therefore, we know that there exists high correlation between the images or frames in the time direction. The correlation in the time direction is called the interframe correlation. If we can efficiently reduce the interframe correlation, then

video compression can be achieved. Several methods to reduce the interframe and intraframe correlation will be introduced in the following sections.

## 2. Video Quality Measure

In order to evaluate the performance of video compression coding, it is necessary to define a measure to compare the original video and the video after compressed. Most video compression systems are designed to minimize the *mean square error* (*MSE*) between two video sequences  $\Psi_1$  and  $\Psi_2$ , which is defined as

$$MSE = \sigma_e^2 = \frac{1}{N} \sum_t \sum_{x,y} [\Psi_1(x, y, t) - \Psi_2(x, y, t)]^2 \quad (1)$$

where  $N$  is the total number of frames in either video sequences.

Instead of the *MSE*, the *peak-signal-to-noise ratio* (*PSNR*) in decibel (dB) is more often used as a quality measure in video coding, which is defined as

$$PSNR = 20 \log_{10} \frac{255}{MSE} \quad (2)$$

It is worth noting that one should compute the *MSE* between corresponding frames, average the resulting *MSE* values over all frames, and finally convert the *MSE* value to *PSNR*.

## 3. The Three Dimensional Discrete Cosine Transform

We apply the *DCT* to reduce the spatial correlation and achieve compression in *JPEG*. A straightforward idea is to expand the 2-D *DCT* to the 3-D *DCT* for video sequences to remove the temporal correlation. The 3-D *DCT* is defined as

### Forward DCT

$$F(x, y, t) = \frac{8}{N^3} C(u)C(v)C(w) \sum_{t=0}^{N-1} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \Psi(x, y, t) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \cos \left[ \frac{\pi(2y+1)v}{2N} \right] \cos \left[ \frac{\pi(2t+1)w}{2N} \right]$$

for  $u = 0, \dots, N-1$ ,  $v = 0, \dots, N-1$  and  $w = 0, \dots, N-1$

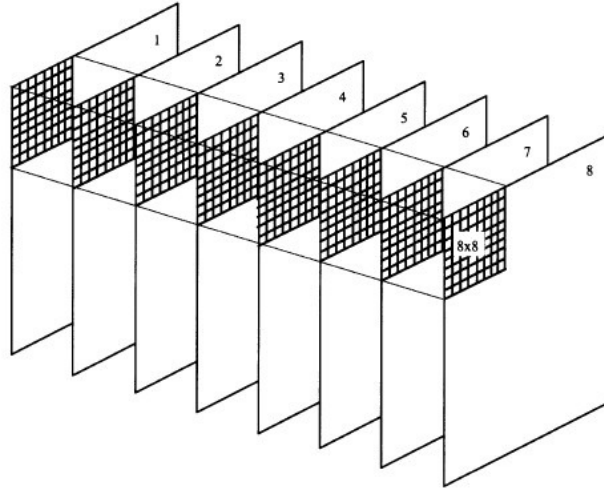
$$\text{where } N = 8 \text{ and } C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

### Inverse DCT

$$\Psi(x, y, t) = \frac{8}{N^3} \sum_{w=0}^{N-1} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)C(w)F(u, v, w) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \cos \left[ \frac{\pi(2y+1)v}{2N} \right] \cos \left[ \frac{\pi(2t+1)w}{2N} \right]$$

for  $x = 0, \dots, N-1$ ,  $y = 0, \dots, N-1$  and  $t = 0, \dots, N-1$  (4)

Through the 3-D forward *DCT*, we can remove the spatiotemporal correlation. The typical example is 3-D *DCT* which is as shown in Fig. 3-1.



**Fig. 3-1 The 3-D DCT of 8x8x8**

## 4. Motion-Compensated Predictive Coding

However, the performance is not efficient to compress the fast changing video, so other methods to remove the spatiotemporal redundancy are proposed for video compression. The most famous method is the block matching algorithm, or motion estimation and motion compensation method. The block matching algorithm divides the current frame and the previous frame into several macroblocks, comparing the blocks in the two frames and trying to search for the best matched pairs for each block.

The dissimilarity  $D(s,t)$  (sometimes referred to as error, distortion, or distance) between two images  $\Psi_n$  and  $\Psi_{n-1}$  is defined as follows

$$D(s,t) = \sum_{V_y=1}^p \sum_{V_x=1}^q M[\Psi_n(x,y), \Psi_{n-1}(x+V_x, y+V_y)] \quad (5)$$

where  $M(u,v)$  is a metric that measure the dissimilarity between the two arguments  $u$  and  $v$ .

There are several types of matching criteria and two most frequently used is MSE and MAD, which is defined as follows:

- *Mean square error (MSE):*  $M(u,v) = (u - v)^2$  (6)

- *Mean absolute difference (MAD):*  $M(u,v) = |u - v|$  (7)

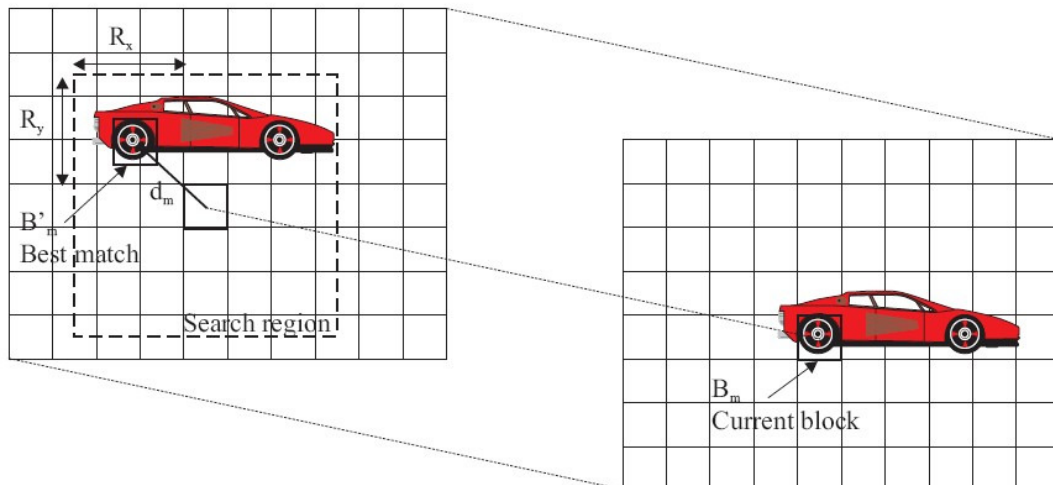
A study based on experimental works reported that the matching criterion does not significantly affect the search. Hence, the MAD is preferred due to its simplicity in implementation.

### 4.1 The Exhaustive Block-Matching Algorithm

Given a macroblock in the anchor block  $B_m$ , the motion estimation is to determine a matching block  $B_m'$  in the target frame such that the error  $D(s,t)$  between the two blocks is minimized. The most straightforward method is the *exhaustive block-matching algorithm (EBMA)*. The procedure of EBMA is shown in Fig. 4.1 and the block matching method can be defined as

$$MV = (V_x, V_y) = \arg \min_{(V_x, V_y) \in S} \sum_{(x, y) \in MB} M[\Psi_n(x, y), \Psi_{n-1}(x + V_x, y + V_y)] \quad (8)$$

where  $S$  is the search region and  $MV$  is the motion vector that minimize the distance.



**Fig. 4-1 The search procedure of the exhaustive block-matching algorithm**

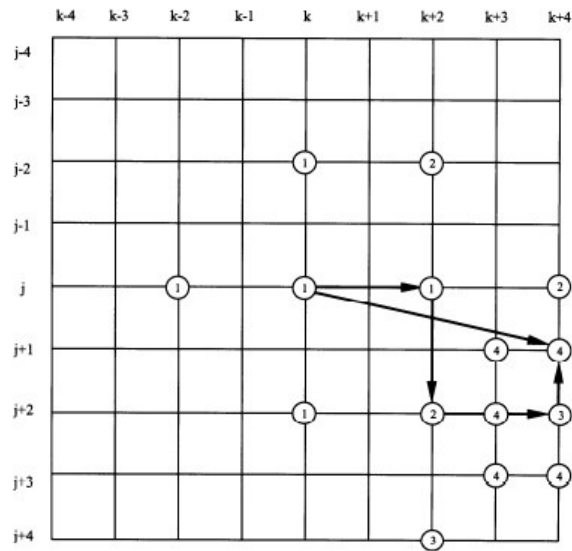
In the simplest case, the stepsize is one pixel in both vertical and horizontal direction, which is known as *integer-pel accuracy search*. Let the block size be  $N \times N$  pixels, and the search range be  $\pm R$  pixels in both vertical and horizontal direction. The total number of operations for a complete frame is  $M^2(2R+1)^2$ , which is a large amount of computation when  $M$  and  $R$  is large.

## 4.2 Fast Block-Matching Algorithms

As shown previous, the EBMA requires a large amount of computation. To speed up the search, various fast algorithms for block matching have been developed. The key to reducing the computation is reducing the number of search candidates. Various fast algorithms differ in the ways that they skip those candidates that unlikely to have small errors. The following subsections describe two of the most well-known fast algorithms.

### 4.2.1 2-D logarithm Search Method

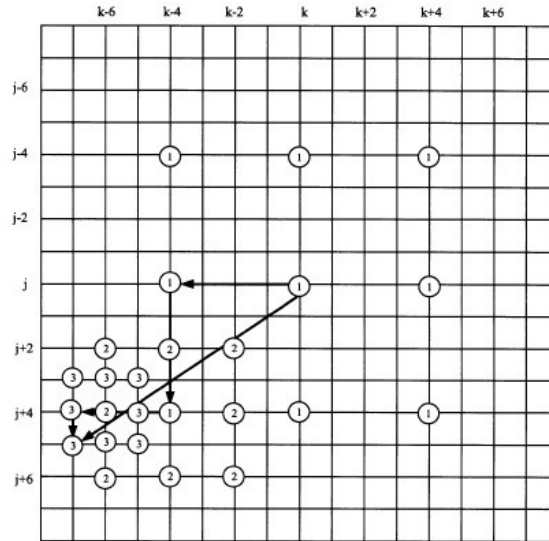
One most well-known and simple fast algorithm is the *2-D logarithm search*, which is shown in Fig. 4-2. The first step is to compute the matching criteria for five points in the search window. The one corresponding to the minimum dissimilarity is picked up as the winner. In the next step, surrounding this winner, another set of five points is selected in a similar fashion to that in the first step. This procedure continues until the final step, in which a set of candidate points are located in a  $3 \times 3$  2-D grid.



**Fig. 4-2 The 2-D logarithm search method**

## 4.2.2 Coarse-Fine Three Steps Search Method

This method is similar to 2-D logarithm search. The main difference is that this method includes only three steps of searching. The steps are as follows. First, compare a set of nine points that form a  $3 \times 3$  2-D grid structure. Second, the distances between the points in the  $3 \times 3$  2-D grid structure in the three step search decreases monotonically in step 2 and 3. Third, a total of only three steps are carried out. The full procedure is shown in Fig. 5-2.

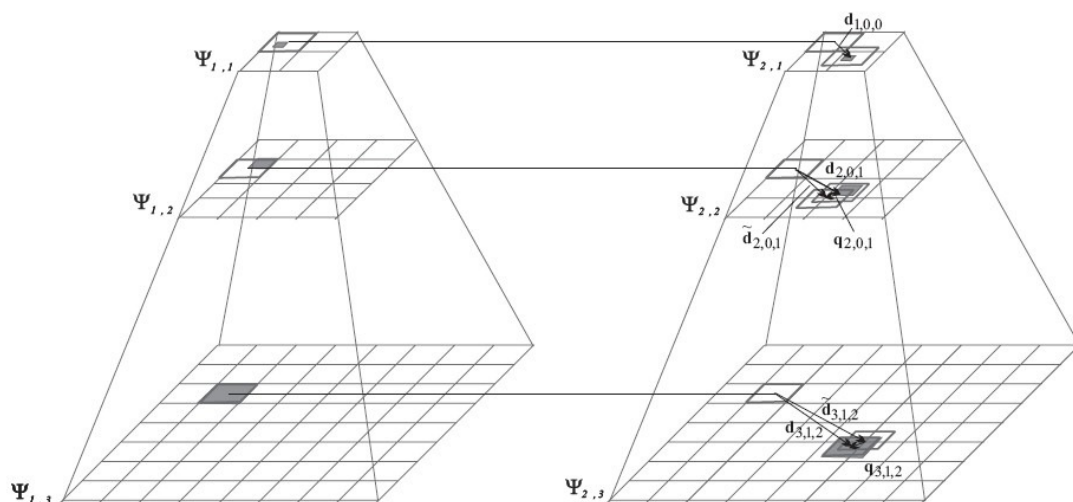


**Fig. 4-3 The Coarse-Fine Three-Step Search Method**

## 5 Multiresolution Motion Estimation

The previous section describes the fixed sized block matching method. On the contrary, the variable block size method is used in this section because it gives more efficient motion estimation than the fixed sized method. One well known example is the multiresolution structure, also known as a pyramid structure, is a very powerful computational configuration for various image and video processing tasks.

As illustrated in Fig. 5-1, pyramidal representations of the two raw image frames are derived, in which each level is a reduced-resolution representation of the lower level.



**Fig. 5-1 Illustration of the hierarchical block matching algorithm**

### General Formulation

Assume that the number of levels is  $L$ , with the  $L$ -th level being the original image. Let the  $l$ -th level images of the anchor and target frames be represented by  $\Psi_{t,l}(\mathbf{x}), \mathbf{x} \in \Lambda_l, t = 1, 2$ , where  $\Lambda_l$  is set of pixels at level  $L$ . Denote the total motion field obtained from levels 1 to  $l-1$  by  $\mathbf{d}_{l-1}(\mathbf{x})$ . At the  $l$ -th level, we first interpolate  $\mathbf{d}_{l-1}(\mathbf{x})$  to the resolution of level  $l$ , to produce initial motion estimation  $\tilde{\mathbf{d}}_l(\mathbf{x}) = U(\mathbf{d}_{l-1}(\mathbf{x}))$ , where  $U$  represents the interpolation operator.

We then determine the update  $\mathbf{q}_l(\mathbf{x})$  at this level such the error is minimized.

$$error = \sum_{\mathbf{x} \in \Lambda_l} |\Psi_{2,l}(\mathbf{x} + \tilde{\mathbf{d}}_l(\mathbf{x}) + \mathbf{q}_l(\mathbf{x})) - \Psi_{1,l}(\mathbf{x})|^p \quad (9)$$

The new motion field obtained after this step is

$$\mathbf{d}_l(\mathbf{x}) = \tilde{\mathbf{d}}_l(\mathbf{x}) + \mathbf{q}_l(\mathbf{x}) \quad (10)$$

Upon completion of successive refinements, the total motion at the finest resolution is

$$\mathbf{d}_l(\mathbf{x}) = \mathbf{q}_L(\mathbf{x}) + U(\mathbf{q}_{L-1}(\mathbf{x}) + U(\mathbf{q}_{L-2}(\mathbf{x}) + U(\dots + U(\mathbf{q}_1(\mathbf{x}) + \mathbf{d}_0(\mathbf{x})))) \quad (11)$$

The initial condition for this procedure is  $\mathbf{d}_0(\mathbf{x})=0$ . One can either directly specify the total motion  $\mathbf{d}(\mathbf{x})$ , or the motion updates at all levels  $\mathbf{q}_l(\mathbf{x}), l=1,2,\dots,L$ . The latter represents the motion in a layered structure, which is desired in applications requiring progressive retrieval of the motion field.

A practical example of multiresolution motion estimation is shown in Fig. 5-2

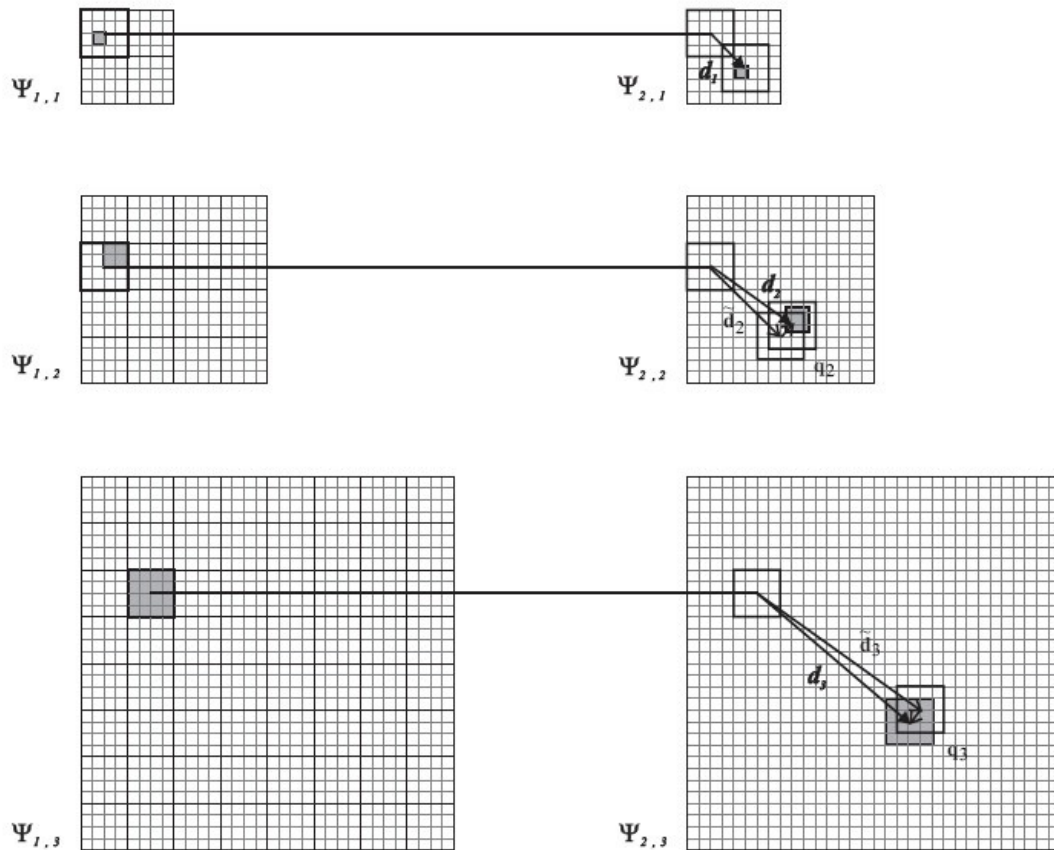


Fig. 5-2 An example of a three-level block motion estimation

## 6 MPEG-1

The main purpose of MPEG-1 video is to code moving image sequences or video signals. To achieve high compression ratio, both intraframe redundancy and interframe redundancy should be removed. There, the MPEG-1 video algorithm is mainly based on DCT and interframe motion compensation. The algorithm of the MPEG-1 will be described in the following subsections.

### 6.1 Layered Structure Based on Group of Pictures

In MPEG-1 and the video compression standards followed, the video sequence is first divided into a *group of pictures*, which is often called *GOP*. One example of GOP is shown in Fig. 6-1. Each GOP can consist of three types of frames:

- *Intracoded Frame (I-frame)*: I-frame is entirely coded in one frame by intraframe technique such as DCT. This type of frame no need for previous information.
- *Predictive Frame (P-frame)*: P-frame is coded using one-directional motion-compensated prediction from a previous frame, which can be either I-frame or P-frame. P-frame is generally referred to as inter-frame.



- *Bidirectional predictive frame (B-frame)*: B-frame is coded using bi-directional motion-compensated prediction from a previous frame or future frame. The reference can be either I-frame or P-frame. B-frame is also referred to as inter-frame.

The distance between two nearest I-frame is denoted by N, and the distance between the nearest I-frame and P-frame is denoted by M.

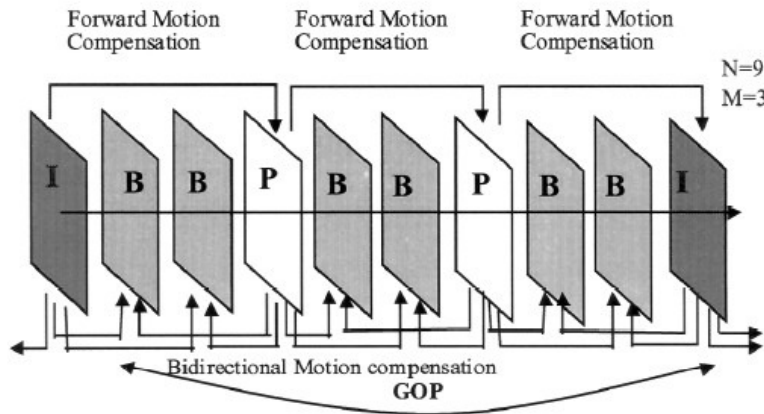


Fig. 6-1 A group of frames

## 6.2 The Encoder Structure of MPEG-1

The encoder structure of MPEG-1 is shown in Fig. 6-2.

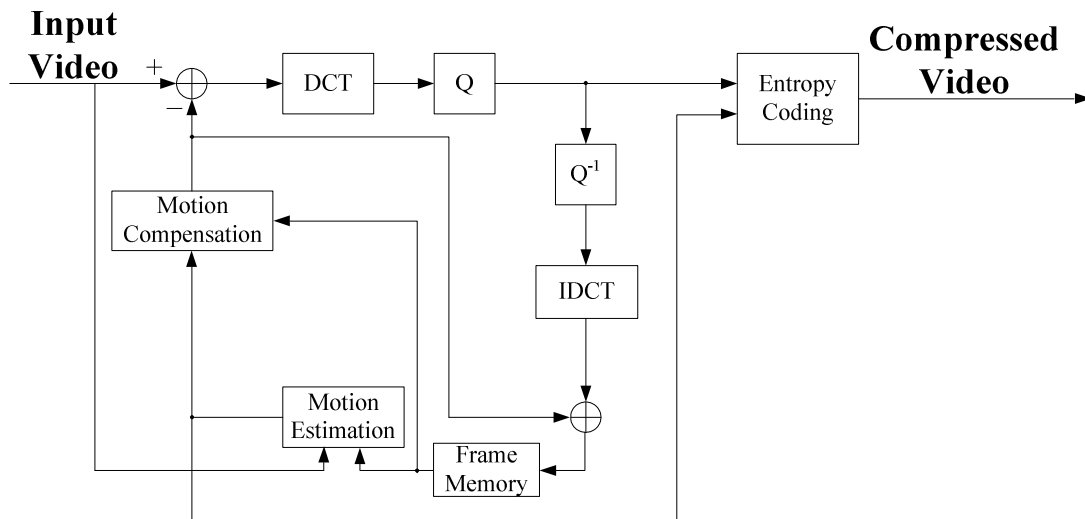


Fig. 6-2 The encoder structure of MPEG-1

The operation of encoding can be summarized as the following steps:

- 1 *Differential Coding*: The input image is subtracted from the predictive image, and the differential image is generated. This can remove temporal correlation. The operation can be represented as

$$D(t) = \psi(t) - \hat{\psi}(t) \quad (11)$$

$\psi(t)$  is the input image and  $\hat{\psi}(t)$  is the predictive image.

- 2 *DCT*: Although the temporal correlation is removed, the spatial redundancy still exists. Therefore, MPEG-1 exploits 2-D DCT to remove the spatial correlation.

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left[ \frac{\pi(2x+1)u}{2N} \right] \cos \left[ \frac{\pi(2y+1)v}{2N} \right]$$

for  $u = 0, \dots, N-1$  and  $v = 0, \dots, N-1$  (12)

$$\text{where } N = 8 \text{ and } C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k = 0 \\ 1 & \text{otherwise} \end{cases}$$

- 3 *Quantization*: The transform coefficients are then quantized. The function of quantization matrix is to quantize the high frequencies with coarse quantization steps because the human visual perception is less sensitive to the high frequencies. The bits saved for coding high frequencies are used for low frequencies. MPEG-1 defines two quantization table, the *intra quantizer weighting matrix* and the *nonintra quantization weighting matrix*. The two matrix is shown in (10).

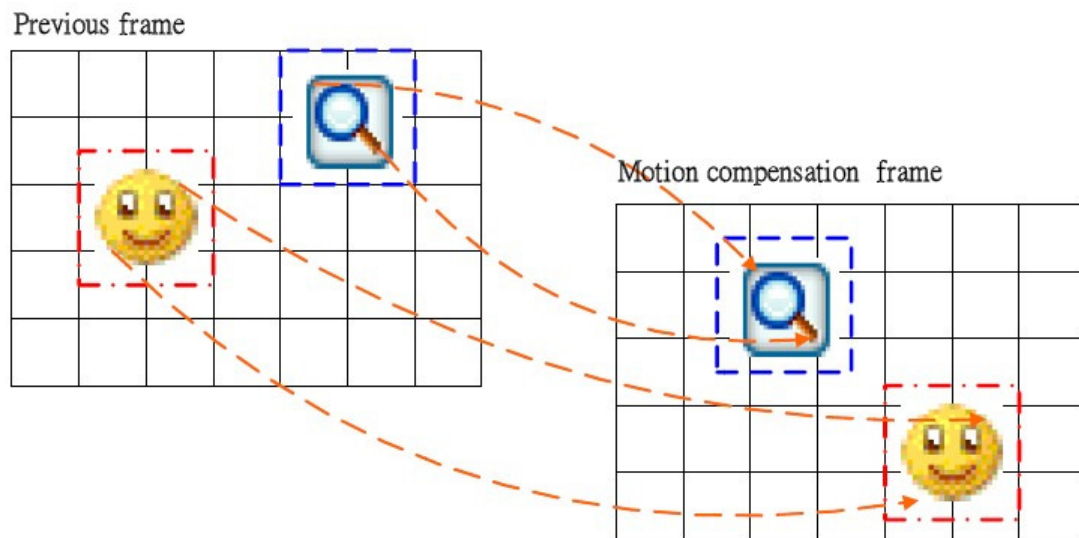
$$Q_{\text{intra}} = \begin{pmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{pmatrix} Q_{\text{intra}} = \begin{pmatrix} 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ 19 & 20 & 21 & 22 & 23 & 24 & 26 & 27 \\ 20 & 21 & 22 & 23 & 25 & 26 & 27 & 28 \\ 21 & 22 & 23 & 24 & 26 & 27 & 28 & 30 \\ 22 & 23 & 24 & 26 & 27 & 28 & 30 & 31 \\ 23 & 24 & 35 & 27 & 28 & 30 & 31 & 33 \end{pmatrix} \quad (13)$$

- 4 *Motion Estimation and Motion Compensation*: The quantization coefficients are dequantized and take the IDCT to reconstruct the approximation of the differential image. The encoder then takes motion estimation based on the input image and the reconstruction image to generate the motion vectors for each macroblock. Finally, motion compensation is executed based on the reconstruction image and the motion vectors to produce the new predictive image. For more details of motion estimation method, we can refer back to chapter4. The motion compensation is to map the macroblocks from the previous frame to the compensated frame as shown in Fig. 6-3. That is

$$\widehat{\Psi}_n(x, y) = \Psi_n(x + v_x(p, q), y + v_y(p, q)), (x, y) \in MB(p, q) \quad (14)$$

where  $\widehat{\Psi}_n(x, y)$  is the compensated image and  $\Psi_n(x, y)$  is the previous

reconstructed image.



**Fig. 6-3 The motion compensation**

## 7 MPEG-2

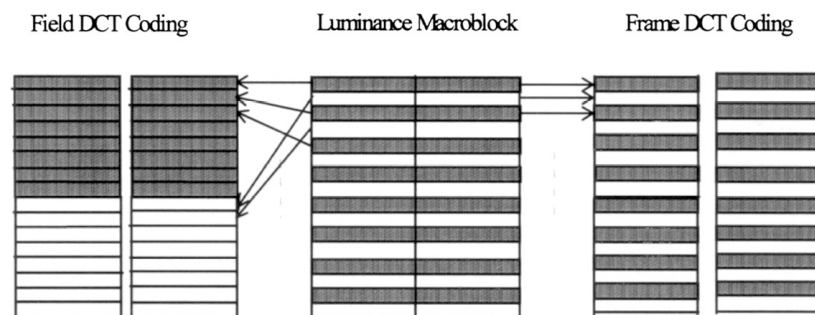
The basic coding structure of MPEG-2 is similar to that of MPEG-1, but MPEG-2 includes some features which are not in MPEG-1:

- Field/frame DCT coding.
- Downloadable quantization matrix and alternative scan order.
- Various picture sampling formats such as 4:4:4, 4:2:2 and 4:2:0.
- Field/frame prediction modes for supporting the interlaced video input.

In the following, the algorithm of MPEG-2 is introduced.

### 7.1 Field/Frame DCT Coding

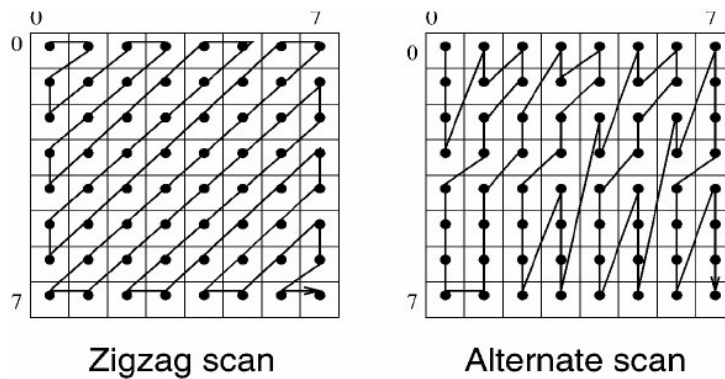
The interlaced material is to allow adaptive selection of the field/frame DCT coding as shown in Fig. 7-1. The middle is a luminance block and the black and white rectangles represent the top and bottom fields, respectively. The left is the *field DCT* contains only the pixels from the same fields. The right is the *frame DCT* which is seen in the previous sections. The field type DCT may be selected when the video experiences a large motion.



**Fig. 7-1 Frame and field DCT for interlaced video**

## 7.2 Alternative Scan Order

When the field DCT is selected for transform, the conventional zigzag scan for zero run length coding may not be useful. Therefore, the alternative scan order was proposed. In MPEG-2 standard, there is a flag that can be set for an alternative scan of DCT blocks, instead of using the zigzag scan. Depending on the spectral distribution, the alternative scan can yield run lengths that better exploits the multitude of zero coefficients. The zigzag scan order and alternative scan order is shown in Fig. 7-2.



**Fig. 7-2 Two different DCT scan order**

## 7.3 Downloadable Quantization Matrix

The MPEG-2 intra and inter quantization matrix is shown in (14).

$$Q_{\text{intra}} = \begin{pmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 27 & 29 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{pmatrix} \quad Q_{\text{inter}} = \begin{pmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{pmatrix} \quad (14)$$

## 7.4 The Encoder Structure of MPEG-2

Fig. 7-3 shows the encoder structure of MPEG-2. As can be seen from the picture, the main encoder structure of MPEG-2 is similar to that of MPEG-1 besides the SNR enhancement Encoder. This base encoder is used to code low resolution layer and the SNR enhancement encoder is used to encode the high resolution layer. For the up-sampled lower layer, an additional prediction mode is available in the MPEG-2

encoder. This is the flexible technique in terms of bit rate ratios, and the enhancement layer can be used in high-quality service.

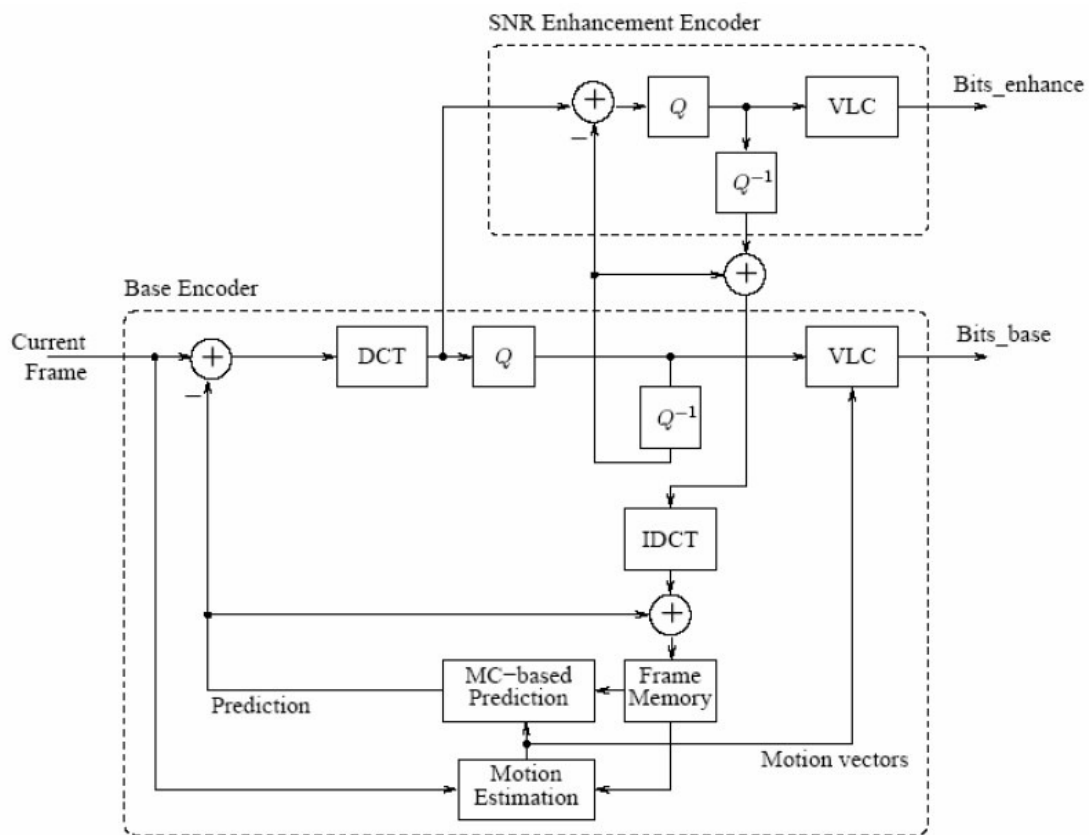


Fig. 7-3 The encoder structure of MPEG-2

## 8 H.264/AVC

There is no single element in the H.264 encoder provides the majority of significant improvement in compression efficiency in relation to the previous standards. Basically, each element is not far from each of the previous standards. However, they are optimized in H.264/AVC. It adds up the small contribution of each element to obtain significant gain.

### 8.1 Remove Temporal Redundancy

- Variable block size:** The size of the macroblock is fixed in the previous standard such as  $8 \times 8$  or  $16 \times 16$ . In order to improve the flexibility of comparison and reduce the error of comparison, H.264/AVC adopts 7 types of blocks for selection ( $16 \times 16$ ,  $16 \times 8$ ,  $8 \times 16$ ,  $8 \times 8$ ,  $8 \times 4$ ,  $4 \times 8$ ,  $4 \times 4$ ) as shown in Fig. 8-1. In the fast moving and changing area, we can adopt smaller blocks for high accuracy. In the slow moving and changing area, we can adopt larger blocks for storage saving.

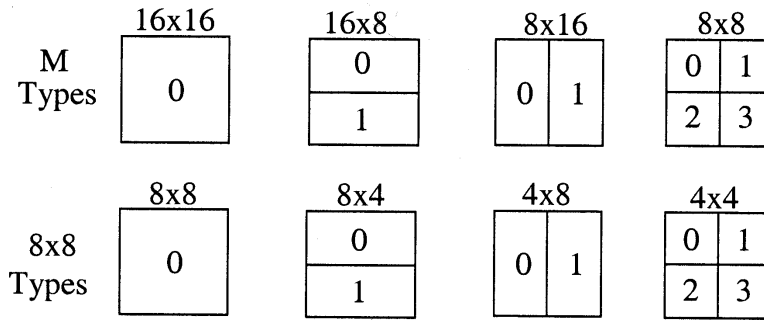


Fig. 8-1 7 types of macroblocks for ME and MC

- **1/4-pel resolution motion estimation:** The largest resolution is 1/2-pel in the previous standards. H.264/AVC extends the 1/2-pel to 1/4-pel to improve the accuracy. The missing pixels can be interpolated by the interpolation filter.
- **Multiple reference frames:** The previous standards only adopt one previous or future frame for reference. However, the neighboring frames are not the most similar in some cases such as Fig. 8-2. Therefore, H.264/AVC can adopt multiple frames for reference (up to 31 frames). Besides, the B-frame can be reference frame in H.264/AVC because the B-frame is closer to the target frame in many situations. One special case is shown in Fig. 8-3.

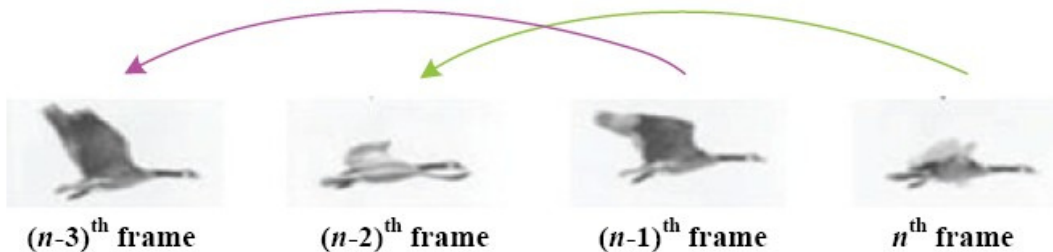


Fig. 8-2 The special case of frame reference

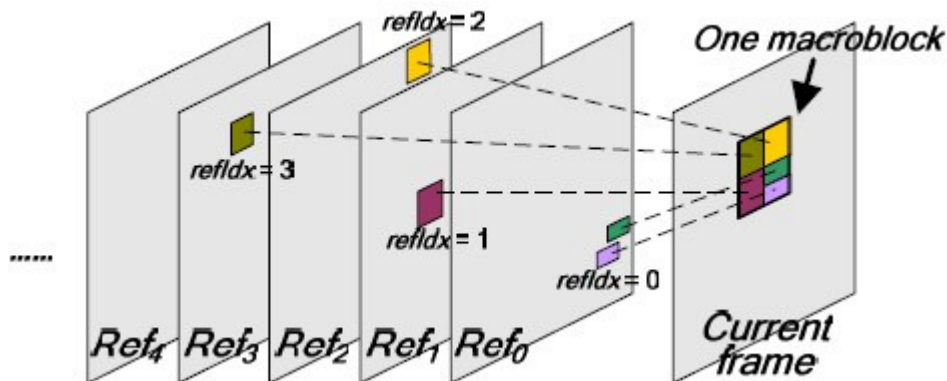


Fig. 8-3 Multiple reference frames

## 8.2 Remove Spatial Redundancy

- **Transform:** In H.264/AVC, the transform operation adopts a 4×4 block instead of a 4×4 DCT, a *separable integer transform* with similar properties is used. The transform matrix is given as

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \quad (15)$$

If the input is a 4×4 block A, then the transform coefficients are  $CXC^T$ .

- **Quantization:** H.264/AVC provides 52 quantization step factors for selection. The quantization step increases exponentially, and the difference between two neighboring steps is about 12%. This design makes H.264/AVC suitable for different kinds of environment.
- **Intra Prediction:** In contrast to the past video compression standard, the spatial redundancy is removed by transform coding only. However, H.264/AVC can predict the similarity between the neighboring pixels in one frame in advance, and exploit transform coding to remove the redundancy. There are nine prediction directions as shown in Fig. 8-4. The missing “2” direction is the DC prediction where one value is used to predict the entire 4×4 block.

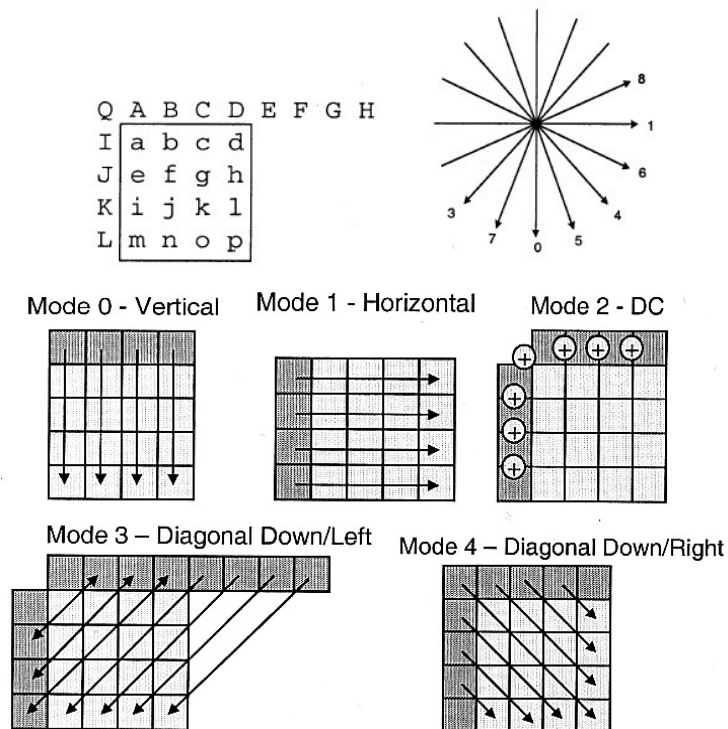


Fig. 8-4 Intra 4×4 prediction

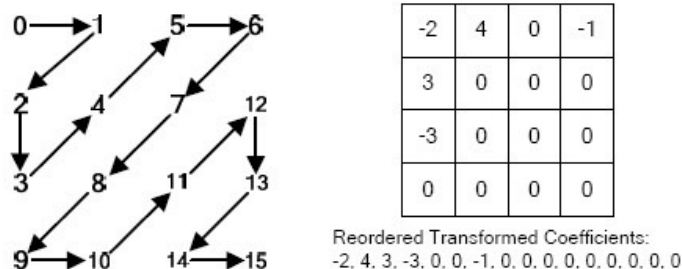
### 8.3 Remove Coding Redundancy

- **exp\_Golomb code:** H.264/AVC provides several kinds of coding methods. The simpler entropy coding method uses a single infinite-extent codeword table for all syntax elements except the quantized transform coefficients. Thus, instead of designing a different VLC table for each syntax element, only the mapping to the single codeword table is customized according to the data statistics. The single codeword table chosen is an exp-Golomb code, which is shown in Table 8-1 with very simple and regular decoding properties.

**Table 8.1 Unsigned exp-Golomb Code**

Code Number		Codeword
0	=>1	1
1	=>10	010
2	=>11	011
3	=>100	00100
4	=>101	00101
5	=>110	00110
6	=>111	00111
7	=>1000	0001000

- **CAVLC (Context-Adaptive Variable Length Coding):** In H.264/AVC, the quantized transform coefficients will be coded by an efficient method called CAVLC. Every 4x4 coefficients will be rearranged in zigzag scan order as shown in Fig. 8-5. After zigzag scanning, the low frequency parts locate on the left side and the high frequency parts locate on the right side. Based on the statistical behavior, the VLC table can assign the corresponding code length to the value of the coefficients.



**Fig. 8-5 The zigzag scan order and the reordered coefficients**

- **CABAC (Context-Adaptive Binary Arithmetic Coding)**





Read AVI file	mov = aviread(filename)
---------------	-------------------------

例如我們現在想要讀取一支名為「example」的檔案，則可輸入

```
=>> mov=aviread('example.avi');
```

此時所有有關的像素資訊都會儲存在mov變數當中。接下來我們想要讀取第一個frame出來進行處理，則可輸入

```
=>> f=mov(1).cdata;
```

此時有關第一張frame的資訊即可儲存在f變數當中，運用同樣的方式就可以讀取其他的frame出來進行處理了。

## 9.2 YUV 檔案之讀取與處理

YUV檔案為最常使用於視訊壓縮的標準格式，因其屬於無失真的壓縮格式(不包含Luminance和Chrominance的subsampling)，就好比BMP檔案為影像壓縮的常用標準格式。在YUV檔案中不像其他以壓縮的視訊格式，其僅包含了視訊的像素質，並沒有其他額外的Header等檔案資訊包含於其中。因此，對於一352×288、4:2:0、200 frames的CIF格式YUV檔案而言，其所需的記憶體空間為 $352 \times 288 \times (1 + 0.25 + 0.25) \times 200 = 30412800 \text{ bytes} = 29.7 \text{ Mbytes}$ 。

但是遺憾的是，MATLAB並沒有提供函式讓使用者讀取YUV的檔案個是進行讀取與寫入，所以相關的檔案讀取要由使用者自行想辦法設計程式去將檔案讀入。所幸YUV檔案中並不包含複雜的Header資訊，在讀取檔案程式的撰寫並不算很困難，在本報告當中即提供的一支可用的程式，能夠對YUV的檔案進行讀取和寫入。

### 1) 讀取YVU檔案

```
% Read YUV-Data from File
% filename: The filename of the input yuv file
% width   : The width of the input yuv video
% height  : The height of the output yuv video
% format  : The subsampling format, it can be '400', '411', '420', '422',
'444'
function YUV=yuv_read(filename,width,height,format)
end
```

本程式的主要使用資訊如表格所示。使用時只要輸入(a)檔名(b)每個frame的寬度(c)每個frame的高度(d)該檔案儲存的subsampling格式，即可將檔案的每個frame讀入並且以cell array的形式儲存在輸出變數當中。例如輸入

```
=>> YUV=yuv_read('test.yuv',352,288,'420');
```

因為test.yuv有199張frames，所以輸出的YUV變數為1×199的cell陣列。此時如果想將第一張frame抽取出來做處理，則可輸入

```
=>> frame=YUV{1};
```

抽取出來的像素資訊及儲存在frame當中，此時frame的大小為352×288×3。注意在讀取YUV檔案的過程中我們就已經把4:2:0的格式upsample到4:4:4了，所以Luma和Chroma的大小都是352×288。

## 2) 儲存YVU檔案

```
% Save YUV-Data to File
% filename: The filename of the input yuv file
% width   : The width of the input yuv video
% height  : The height of the output yuv video
% format  : The subsampling format, it can be '400', '411', '420', '422',
'444'
function yuv_save(YUV,filename,width,height,format)
end
```


本程式的主要使用資訊如表格所示。其使用的方式和其一讀檔程式大同小異，使用者可以依照需求輸入(a)檔名(b)每個frame的寬度(c)每個frame的高度(d)該檔案欲儲存的subsampling格式。例如想將之前讀入的YUV像素資訊儲存起來，可以輸入

```
=>> yuv_save(YUV,'test_new.yuv',352,288,'420');
```

代程式執行完畢即可完成存檔。

## 3) 播放YVU檔案

MATLAB同樣沒有提供使用者播放YUV檔案的功能，所以如何將處理過後的視訊檔案進行播放呢？目前網路上有眾多免費的YUV檔案播放軟體，例如YUV Display即是一個很好用的播放軟體，其使用方法如下

A) 首先點選圖示開始程式，可看到以下畫面。

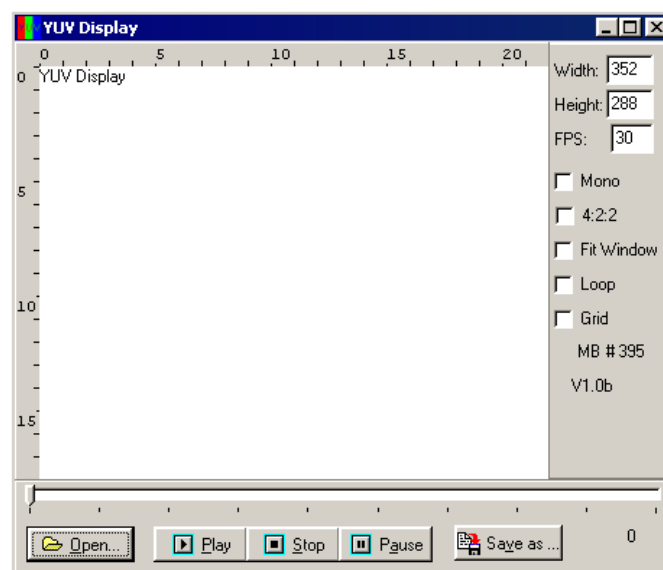


Fig 9-1 YUV Display軟體開啟畫面

B) 接著點選OPEN可看到以下畫面，選取所欲開啟的YUV檔案之後，選取該讀入的檔案格式，例如報告當中所使用的CIF檔案(352×288, 4:2:0)格式，點選YUV Files，點選開啟就可順利把檔案讀入。

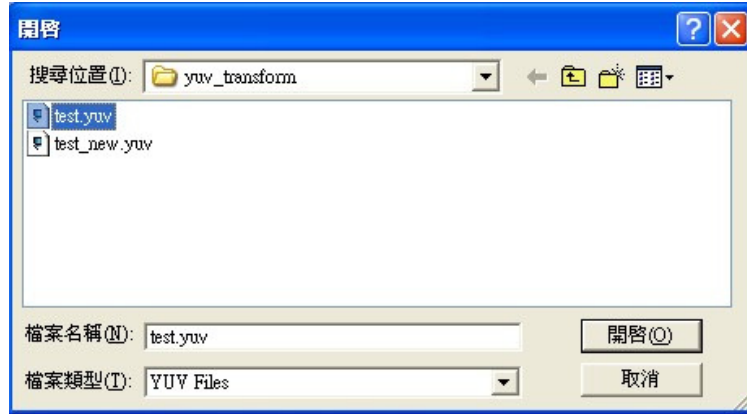


Fig. 9-2 檔案開啟畫面

C) 檔案順利讀入後就可看到以下畫面，如果選取的Subsampling格式正確，則所顯示的畫面為彩色無異狀，如圖9-3所示。

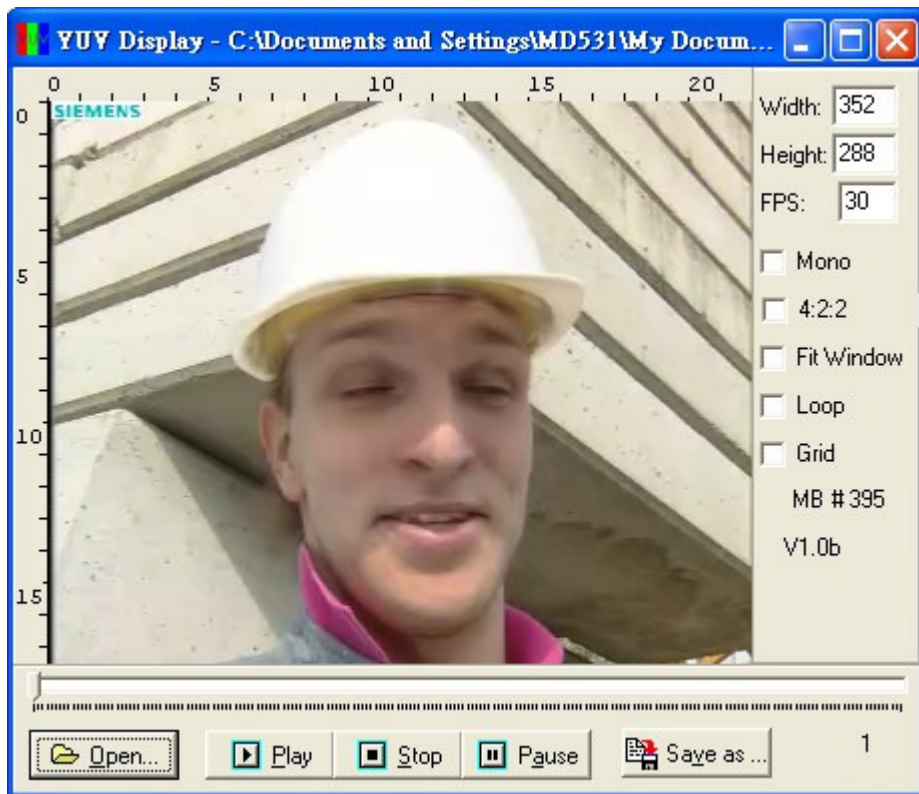


Fig. 9-3 檔案開啟成功畫面

如果選取的格式錯誤，則會產生異樣如圖9-4所示。所以正確的Subsampling格是

要選取好才可正確把畫面播放出來。

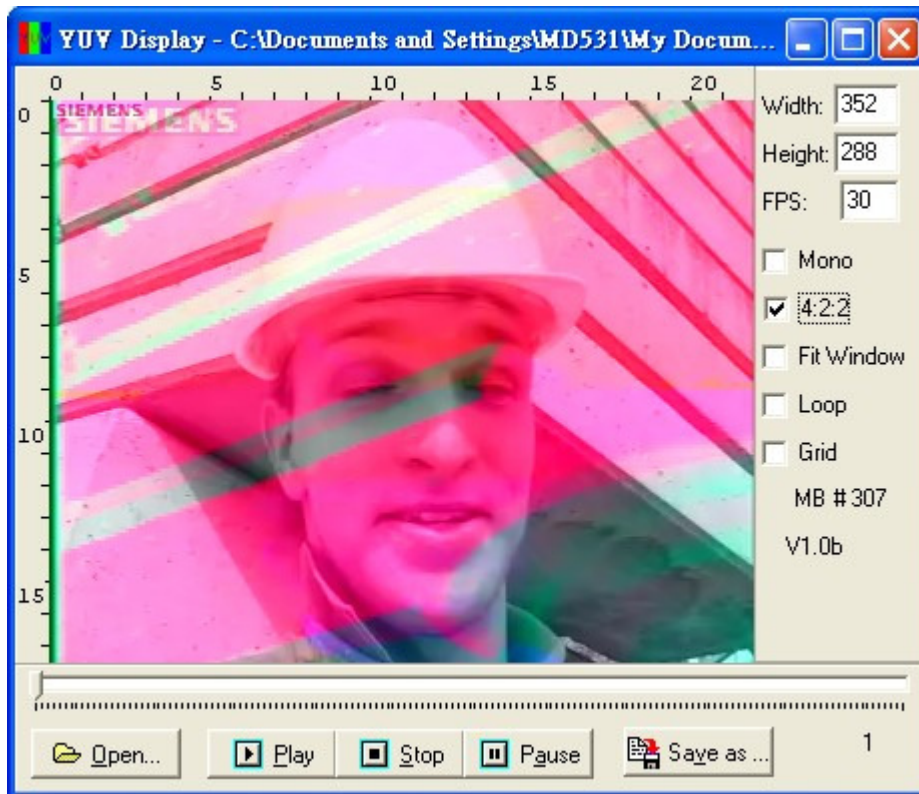


Fig. 9-4 檔案格式選取錯誤畫面

D) 如果像要擷取畫面，只要點選Save as輸入想要儲存的檔名後按儲存即可。

## 10 Conclusion and Future Work

In this paper, we have introduced the fundamental concepts of video compression and the characteristics of various video compression standards. Although the existed video compression standards can compress the video effectively, it still leaves room for improvement. For example, for reducing the temporal and spatial redundancy, Block Matching Algorithm and DCT has been exploited. Both of these two methods can reduce the temporal and spatial redundancy, respectively. However, they will lead to the annoying unnecessary blocking artifact. In H.264, the de-blocking filter is exploited for smooth the block artifact and recovers the video as possible as can. The performance of the de-blocking filter in H.264 is good, but it still leaves room for improvement. Therefore, the more outstanding adaptive de-blocking filter for removing blocking artifact may be a trend.

On the other hand, the Exhausted BMA needs a large amount of computation. In H.264, the concept of variable block size has been proposed, which increases the computation cost more. Therefore, the fast algorithm for BMA and fast mode decision

for selecting block size is need. The entropy coding method is also a hot topic of video and image compression field. After removing the temporal and spatial redundancy, the entropy encoder plays the role of the unit that converts the original pixels values into bitstream. The more powerful the encoder, the more compression ratio can be achieved.

These are a few of the many challenges in video compression to be fully resolved and may affect the compression performance in the years to come.

## 11 Reference

- [1] Yun Q. Shi and Huifang Sun, “*Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*”, CRC press, 2000.
- [2] Yao Wand, Jorn Ostermann and Ya-Qin Zhang, “*Video Processing and Communications*”, Prentice Hall, 2007.
- [3] Richardson, Lain E. G., “*Video Codec Design: Developing Image and Video Compression Systems*”, John Wiley & Sons Inc, 2002.
- [4] Barry G, Haskell, Atul Puri and Arun N. Netravali, “*Digital Video : An Introduction to MPEG-2*”, Boston : Kluwer Academic, 1999.
- [5] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard”, *IEEE Trans. on Circuits and systems for video Technology*, vol. 13, no. 7, pp. 560-576, July 2003.
- [6] G. Sullivan and T. Wiegand, “Video Compression - From Concepts to the H.264/AVC Standard”, *Proceedings of the IEEE*, Special Issue on Advances in Video Coding and Delivery, December 2004.
- [7] 酒井善則、吉田俊之 共著，白執善 編譯，“*影像壓縮技術*”，全華，2004.