ELSEVIER

Signal Processing: *Image Communication* 6 (1994) 173–187

SIGNAL PROCESSING: IMAGE COMMUNICATION

# Real-time software-based moving picture coding (SBMPC) system

Ho-Chao Huang, Ja-Ling Wu*

*Communications and Multimedia Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, ROC*

Received 23 November 1992

**Abstract**

A novel moving picture coding system, called the software-based moving picture coding system, is presented in this paper. In this coding system, two new techniques, modified block truncation codes and multiresolution-in-time sampling, are used for real-time encoding and decoding of the moving pictures. The modified block truncation codes can process image data with three times faster and two times higher compression ratio than those of the traditional ones. The multiresolution-in-time sampler samples the image blocks with variant sampling rates, which depend on the activities of image blocks. This technique can make the coding process faster and the compression ratio higher. By using these techniques, the software-based moving picture coding system can compress moving pictures (video) under the real-time constraints of video applications, such as videophone/videoconference, without using any expensive compression chips.

*Key words:* Data compression; Video coding; Real-time video compression

## 1. Introduction

Most existing moving picture coding (MPC) techniques, such as MPEG1 [6], H. 261 [4] and its preceding reference model RM8 [2], have been designed to compress data by both reducing the temporal correlation through the motion compensation [7] (MC) and DPCM [5] techniques, and reducing the spatial correlation through the transform coding, such as DCT [1]. The major problems of the MC/DCT-based techniques are the relatively poor performance in dealing with sharp edges and the relatively high time complexity of both encoders and decoders. Because of the poor performance of dealing with sharp edges, the ripple effect around the sharp edges is significantly observable, and because of the high time complexity of those codecs, software implementation, such as video conference and playback system, in currently common PCs is not possible at real-time speed. As a result, the cost of the video application systems is very high, and the implementation of those systems is relatively difficult.

In this paper, a real-time software-based moving picture coding (SBMPC) system is proposed. In the SBMPC system, two new techniques, the modified block truncation code and multiresolution-in-time (MIT) sampling, are used. The block truncation coding [3] (BTC) is considered as an edge-preserving algorithm for image coding, and is a relatively fast algorithm as compared to the transform coding. In order to fit the requirements of the SBMPC system, high speed and high compression ratio,

* Corresponding author.


*SSDI* 0923-5965(94)E0014-A

a modified version of BTC is used. The modified BTC not only preserves the sharp edges of image data but runs faster and achieves a higher compression ratio than the normal BTC. In order to get higher compression ratio, the MIT sampling technique is included in the SBMPC system. The MIT sampling technique analyzes the temporal activity of the image sequence, and assigns a higher sampling rate to the blocks with higher degree of activity. The compression ratio and the frame rate that can be obtained by using these new techniques are 20 to 25 and about 14 frames per second, respectively, performed on an IBM PC-AT, with $128 \times 128$ frame size. The resultant quality of the compressed video is in our opinion good enough for videophone/videoconference applications.

In Section 2, we first review the traditional MC/DCT-based techniques and the BTC algorithm briefly. Section 3 describes the SBMPC system, the modified BTC, and the MIT sampling techniques. Section 4 shows the experimental results of the proposed system. Finally, some discussions and conclusions are presented in Section 5.

## 2. Preliminary reviews

We first review the MC/DCT-based MPC in Section 2.1, and then point out the major problems of this coding scheme; actually, these are what the SBMPC system is designed to solve. The BTC technique for image coding is reviewed in Section 2.2, because it is one of the major components of the SBMPC system.

### *2.1. Review of MC/DCT-based MPC and its related problems*

The basic idea of MPC is to reduce the bit-rate by using the decorrelation process to remove spatial and temporal redundancies, and using the entropy coding to remove codeword redundancies. A typical decorrelation process of an MPC usually contains two parts: a prediction coding part for temporal redundancy removal and a transform coding part for spatial redundancy removal. A typical entropy coder of an MPC usually consists of a run-length coder and a Huffman coder. The run-length coder removes the quantized transformed coefficients with zeros in succession, and the Huffman coder is used for representing the run-length coded tuples, which is known to be of minimal redundancy in theory.

The MC/DCT-based MPC, such as H.261, uses the DPCM technique with the motion compensation techniques as the prediction coder, and the DCT as its transform coder. The block diagram of the H.261 encoder is shown in Fig. 1.

Although the MC/DCT-based MPC has the capability of encoding the moving pictures with relatively high compression ratio and picture quality, the following problems of this coding scheme still need to be solved.

#### *2.1.1. Time complexity problem*

This is the major problem of the MC/DCT-based MPC. Enormous operations are needed in order to perform the functions of the motion estimation and the DCT. Even though the fast algorithms of MC and DCT are used, it is still almost impossible to implement a real-time MC/DCT-based MPC by software.

Traditionally, some special compression chips have been developed to perform the operations such as DCT and motion estimation in order to meet the real-time requirement of the MC/DCT-based MPC. This hardware solution for a real-time MPC has some disadvantages:

1. The high price of these compression chips will make it too expensive to implement practical video applications.
2. The development of device drivers and video systems will be more difficult by using a hardware solution than its software counterpart.
3. Flexibility in installation, maintenance, and upgrading is lost when fixed, hardwired implementations are applied.

The disadvantages listed above may form a high threshold for the introduction of low-cost video application. Therefore, we believe that the use of hardware chips does not seem a very good solution for developing low-cost MPC systems.

#### *2.1.2. Sharp edge problem*

The second problem of the MC/DCT-based MPC is the sharp edge problem. The basic idea of

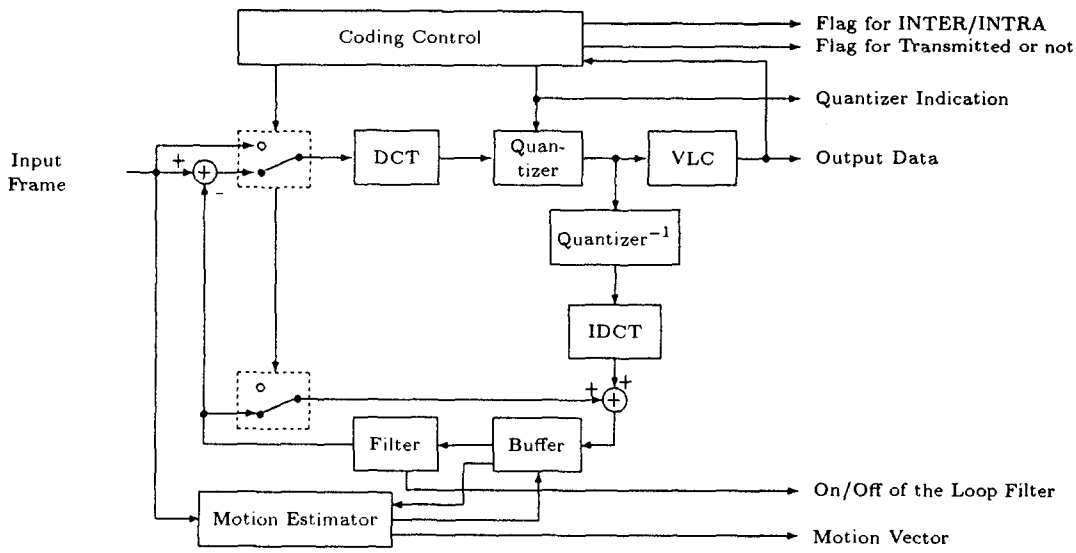


Fig. 1. Block diagram of the CCITT H.261 encoder.

transform coding, such as DCT, is to analyze the frequency distributions of a signal. In a slow-varying signal, most of the energy of this signal will be concentrated at the first few coefficients in the transform domain, and the high-frequency coefficients can be ignored without losing important information. However, for a signal containing sharp edges, some values of the high-frequency coefficients cannot be ignored and these coefficients must be quantized for further processing. The distortions caused by the quantizer will make the reconstructed signals contain obvious ripple effects at the flat regions and blurring effects at the sharp edges. An image signal, which consists of several objects, will contain many sharp edges at the boundaries of objects, and the effects of sharp edges made by DCT and quantizer will degrade the quality of the reconstructed image notably. In Fig. 2, the original signal contains a sharp edge, and then the ripple effects (which are caused by the quantizer) of the reconstructed signal are obvious.

## 2.2. Review of BTC

In BTC, an image is first divided into $n \times n$ nonoverlapping blocks, and then these blocks are quantized by a two-level block-dependent quantizer. Both the threshold and the two reconstruction levels of the quantizer are adaptive in response to the local statistics of a block. An example of BTC is given in Fig. 3.

### 2.2.1. Advantages of BTC

The advantages of BTC are listed below, and these are the reasons why the BTC was chosen as the basic algorithm of the proposed SBMPC system:

1. The coding speed of BTC is much faster, at least five times faster in our experience, than that of the transform coding.
2. The BTC is an edge-preserving coding algorithm, and can compress images, which contain many sharp edges, with high compression ratio and good image quality.

### 2.2.2. Limitations of BTC

Even though the BTC has some good features as described above, there are some inborn defects existing in BTC, and this is why we did not apply BTC directly in the SBMPC system:

1. It is not easy to adapt the compression ratio of BTC with fixed block size (where the quality (distortion) of the reconstructed image is also fixed).

| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| original signal | 5 | 5 | 5 | 5 | 5 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 | 35 |
| transform coefficient | 36 | -16 | -9 | -1 | 3 | 4 | 1 | -2 | -3 | -1 | 1 | 2 | 1 | 0 | -2 | -2 |
| quantized coefficient | 36 | -16 | -8 | 0 | 4 | 4 | 0 | 0 | -4 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |
| reconstructed signal | 8 | 4 | 8 | 2 | 5 | 29 | 36 | 31 | 37 | 37 | 33 | 36 | 32 | 37 | 42 | 29 |

(a)

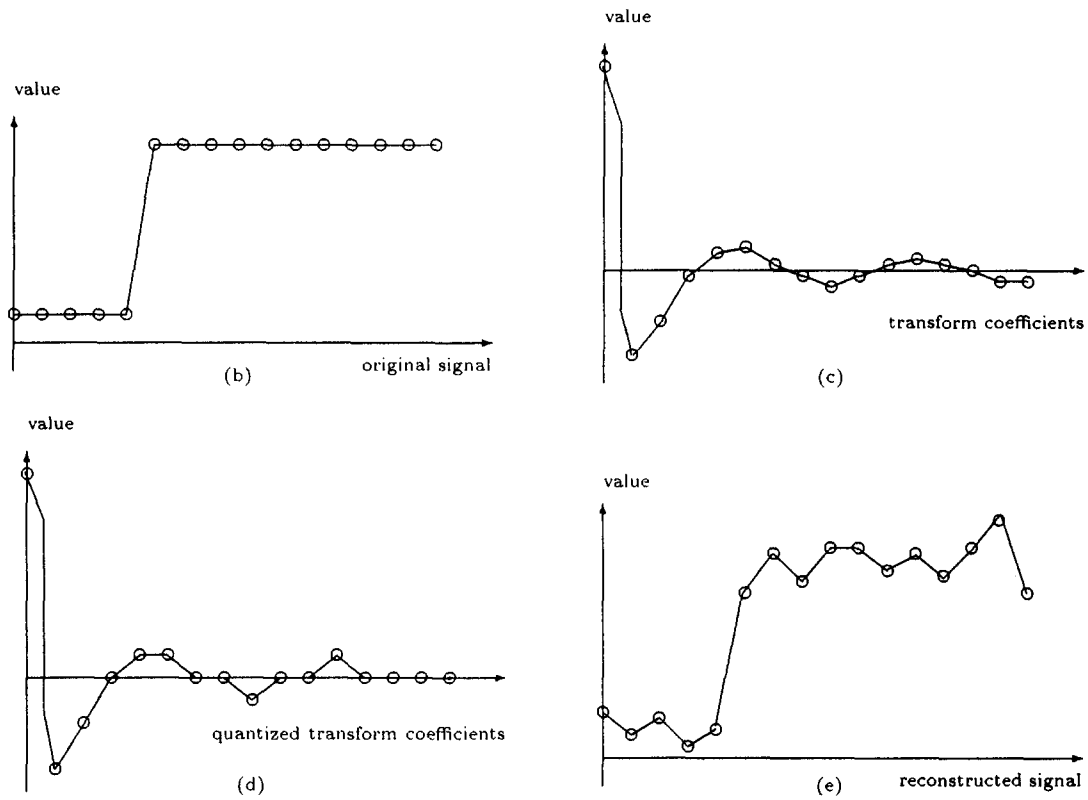


Fig. 2. Example of the DCT coding for a signal with a sharp edge: (a) shows the values of the original signal, the transformed coefficients, the quantized transformed coefficients, and the reconstructed signal; (b), (c), (d), and (e) show the waveforms of the original signal, the transformed coefficients, the quantized transformed coefficients, and the reconstructed signal, respectively.

$$X = \begin{bmatrix} 176 & 172 & 169 & 166 \\ 167 & 164 & 102 & 77 \\ 165 & 155 & 70 & 67 \\ 100 & 79 & 72 & 65 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \hat{X} = \begin{bmatrix} 167 & 167 & 167 & 167 \\ 167 & 167 & 79 & 79 \\ 167 & 167 & 79 & 79 \\ 79 & 79 & 79 & 79 \end{bmatrix}$$

(a) (b) (c)

Fig. 3. An example of BTC: (a) shows the original $4 \times 4$ image block; (b) shows the quantized bit map, the threshold of the quantizer is the mean value of the image block, and the two reconstruction levels, respectively, are 79 and 167; (c) shows the reconstructed image block.

2. The compression ratio of BTC is not high enough to meet the bit-rate requirement of a real MPC application. If the original image is of 8 bits/pixel (bpp) and is divided into $4 \times 4$ blocks, then the bit-rate of the compressed data is 2 bpp and the corresponding compression ratio is 4 : 1 only.
3. Although the coding speed of BTC is faster than that of the DCT, it is still not fast enough to meet the real-time requirement of the most urgent video applications, such as the videophone and videoconference systems.

## 3. The SBMPC system

In the previous sections, we have pointed out the problems of the MC/DCT-based MPC. The goal of the SBMPC system is not only to solve the time complexity problem of the traditional MC/DCT-based approach by proposing a new high-speed coding architecture instead of hardware chips solutions, but to obtain a high-quality compressed video with high compression ratio and edge-preserving (object-preserving) characteristics. The modified BTC and the MIT sampling techniques, which can be considered as kernels of the SBMPC system, will now be introduced.

### 3.1. The modified BTC

In the modified BTC, the input image is first segmented into blocks of size $8 \times 8$, and then the pixels within each block are subsampled. This subsample process is necessary because it can both accelerate the computation and achieve higher compression ratio. Furthermore, an efficient algorithm is used for computing the quantizer's threshold and the two reconstruction levels of each subsampled block, and then the pixels are quantized by a two-level quantizer. The decoder of the modified BTC reconstructs the quantized pixels, interpolates the pixels (which were eliminated by the decimation process), synthesizes and generates the output image. The bit-rate of the modified BTC is 44 bits per $8 \times 8$ image block (32 bits for the bit map and 6 bits for each reconstruction level). The block diagram of the proposed modified BTC is shown in Fig. 4.

#### 3.1.1. Blocker

The Blocker of the modified BTC is used to divide the original image data $OX(n)$ into blocks $X(n)$ with size $8 \times 8$.

#### 3.1.2. Subsampler

The image blocks generated by the Blocker are then sent to the Subsampler. Let $X(n)$ and $SX(n)$ be the input and output data of the Subsampler, respectively. The function that is performed by the Subsampler is shown as follows:

$$SX(n) = X(k \times n), \quad k \text{ is a positive integer.} \tag{1}$$

In the SBMPC system, $k$ is set to 3 so as to get the information of each horizontal and vertical scan line. Fig. 5 shows an example of the function performed by the Subsampler.

#### 3.1.3. QP decider

The QP Decider is used for deciding the quantizer's threshold QT and the two reconstruction levels, RL(0) and RL(1), of the two-level quantizer. In order to accelerate the computation speed of the QP Decider, we use $SX(n)$, which was generated by the Subsampler, instead of the original image blocks $X(n)$ as the input data of the QP Decider.

The parameters QT, RL(0) and RL(1) are defined as

$$\mathrm{QT} = \frac{1}{\lceil 64/k \rceil} \sum_{0 \leqslant i < \lceil 64/k \rceil} SX(i), \tag{2}$$

$$\mathrm{RL}(0) = \frac{1}{\sum_{0 \leqslant i < \lceil 64/k \rceil} \delta_{SX(i) < \mathrm{QT}}} \times \sum_{0 \leqslant i < \lceil 64/k \rceil} \delta_{SX(i) < \mathrm{QT}} SX(i), \tag{3}$$

$$\mathrm{RL}(1) = \frac{1}{\sum_{0 \leqslant i < \lceil 64/k \rceil} \delta_{SX(i) \geqslant \mathrm{QT}}} \times \sum_{0 \leqslant i < \lceil 64/k \rceil} \delta_{SX(i) \geqslant \mathrm{QT}} SX(i), \tag{4}$$

where $k$ was defined in the Subsampler, $\delta_f = 1$ when $f$ is true, otherwise, $\delta_f = 0$, and $\lceil r \rceil$ returns the closest integer which is greater than or equal to $r$.

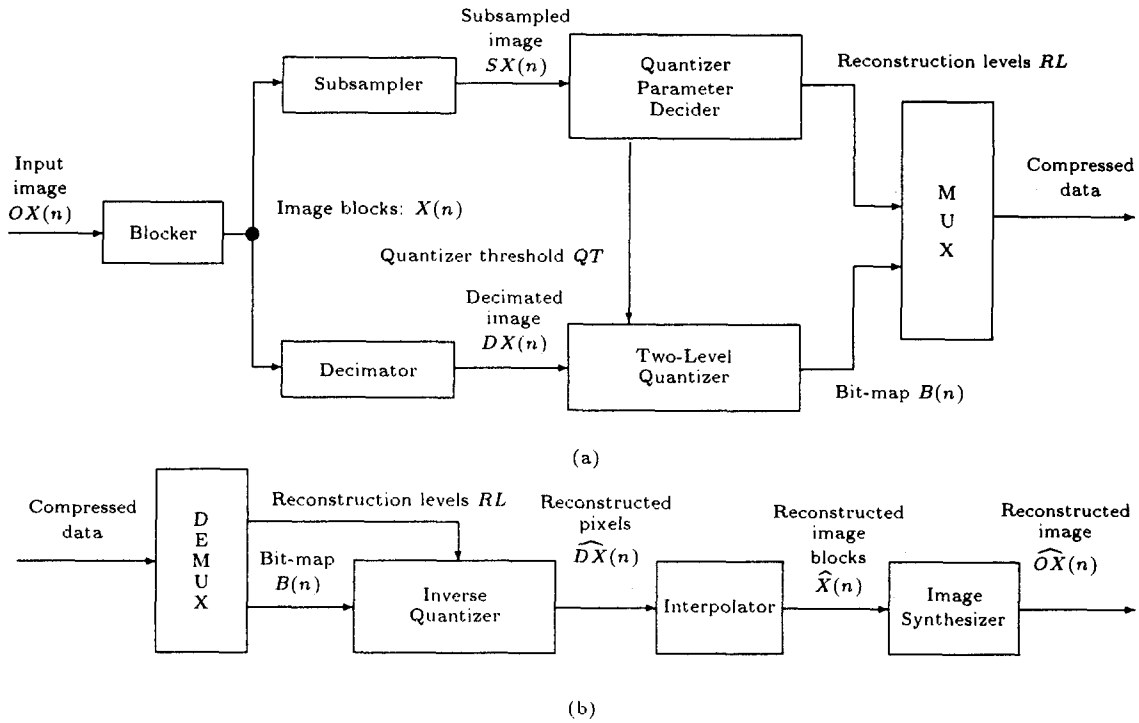


Fig. 4. Block diagram of the modified BTC: (a) encoder; (b) decoder.

$$X = \begin{bmatrix} \boxed{x_{00}} & x_{01} & x_{02} & \boxed{x_{03}} & x_{04} & x_{05} & \boxed{x_{06}} & x_{07} \\ x_{10} & \boxed{x_{11}} & x_{12} & x_{13} & \boxed{x_{14}} & x_{15} & x_{16} & \boxed{x_{17}} \\ x_{20} & x_{21} & \boxed{x_{22}} & x_{23} & x_{24} & \boxed{x_{25}} & x_{26} & x_{27} \\ \boxed{x_{30}} & x_{31} & x_{32} & \boxed{x_{33}} & x_{34} & x_{35} & \boxed{x_{36}} & x_{37} \\ x_{40} & \boxed{x_{41}} & x_{42} & x_{43} & \boxed{x_{44}} & x_{45} & x_{46} & \boxed{x_{47}} \\ x_{50} & x_{51} & \boxed{x_{52}} & x_{53} & x_{54} & \boxed{x_{55}} & x_{56} & x_{57} \\ \boxed{x_{60}} & x_{61} & x_{62} & \boxed{x_{63}} & x_{64} & x_{65} & \boxed{x_{66}} & x_{67} \\ x_{70} & \boxed{x_{71}} & x_{72} & x_{73} & \boxed{x_{74}} & x_{75} & x_{76} & \boxed{x_{77}} \end{bmatrix} \quad \text{(a)}$$

$$SX = \begin{bmatrix} x_{00} & x_{03} & x_{06} & x_{11} & x_{14} & x_{17} & \cdots & x_{71} & x_{74} & x_{77} \end{bmatrix} \quad \text{(b)}$$

Fig. 5. An example of an execution process of the Subsampler with $k = 3$: (a) is the input image data, and the framed pixels will be selected by the Subsampler; (b) is the output data.

The calculated QT is used by the Two-Level Quantizer (cf. Section 3.1.5), and the RLs are sent to the Inverse Quantizer (cf. Section 3.1.6) for reconstructing the image pixels.

### 3.1.4. Decimator

The Decimator performs vertical and horizontal decimation functions, which behave similarly to the subsample function of the Subsampler. These functions are defined as follows.

The horizontal decimation function is

$$DX_h(n) = X(d \times n), \quad d \text{ is a positive number}, \quad (5)$$

and the vertical decimation function is

$$DX_v(n) = X\left(\left\lfloor \frac{n}{w} \right\rfloor \times d \times w + (n \bmod w)\right), \quad (6)$$

where $w$ is the width of the image block, $\lfloor r \rfloor$ returns the closest integer which is less than or equal to $r$, and the index $n$ is a $w \times w$ block that counts from left to right and top to bottom. The examples of the execution results performed by these functions are shown in Fig. 6.

The objective of the Decimator is to reduce the bit-rate of the compressed data and to accelerate the computation of the Two-Level Quantizer. In order to support a wide spatiotemporal spectrum without resolution loss in the horizontal or vertical directions only, the decimation function subsamples the pixels along the horizontal and vertical directions image by image, alternatively. In the SBMPC system, the decimation factor $d$ is set to 2. The output data of the Decimator are then sent to the Two-Level Quantizer.

### 3.1.5. Two-level quantizer

The Two-Level Quantizer quantizes the decimated image $DX(n)$ obtained from the output of the Decimator. The quantizer's threshold QT, defined by Eq. (2), is used to perform the quantization process. The bit-map $B(n)$ comes from the output of the Two-Level Quantizer and is defined as

$$B(n) = \delta_{DX(n) \geq \mathrm{QT}}. \tag{7}$$

The bit-map $B(n)$ is then multiplexed with the reconstruction levels RL(0) and RL(1) and then sent to the MBTC decoder.

### 3.1.6. Inverse quantizer

The Inverse Quantizer performs an inverse function of the Two-Level Quantizer. It uses the reconstruction levels, RL(0), RL(1), and the bit-map $B(n)$ to reconstruct image pixels based on the following function:

$$\widehat{DX}(n) = \begin{cases} \mathrm{RL}(0) & \text{if } B(n) = 0, \\ \mathrm{RL}(1) & \text{otherwise}, \end{cases} \tag{8}$$

or in its equivalent representation form shown as

$$\widehat{DX}(n) = \delta_{B(n)=0} \times \mathrm{RL}(0) + \delta_{B(n)=1} \times \mathrm{RL}(1). \tag{9}$$

### 3.1.7. Interpolator

Because the image pixels reconstructed from the Inverse Quantizer are the decimated pixels, the pixels eliminated by the Decimator must be

$$X = \begin{bmatrix} \boxed{x_{00}} & x_{01} & \boxed{x_{02}} & x_{03} & \boxed{x_{04}} & x_{05} & \boxed{x_{06}} & x_{07} \\ \boxed{x_{10}} & x_{11} & \boxed{x_{12}} & x_{13} & \boxed{x_{14}} & x_{15} & \boxed{x_{16}} & x_{17} \\ \boxed{x_{20}} & x_{21} & \boxed{x_{22}} & x_{23} & \boxed{x_{24}} & x_{25} & \boxed{x_{26}} & x_{27} \\ \boxed{x_{30}} & x_{31} & \boxed{x_{32}} & x_{33} & \boxed{x_{34}} & x_{35} & \boxed{x_{36}} & x_{37} \\ \boxed{x_{40}} & x_{41} & \boxed{x_{42}} & x_{43} & \boxed{x_{44}} & x_{45} & \boxed{x_{46}} & x_{47} \\ \boxed{x_{50}} & x_{51} & \boxed{x_{52}} & x_{53} & \boxed{x_{54}} & x_{55} & \boxed{x_{56}} & x_{57} \\ \boxed{x_{60}} & x_{61} & \boxed{x_{62}} & x_{63} & \boxed{x_{64}} & x_{65} & \boxed{x_{66}} & x_{67} \\ \boxed{x_{70}} & x_{71} & \boxed{x_{72}} & x_{73} & \boxed{x_{74}} & x_{75} & \boxed{x_{76}} & x_{77} \end{bmatrix} \quad \text{(a)}$$

$$DX_h = \begin{bmatrix} x_{00} & x_{02} & x_{04} & x_{06} & x_{10} & x_{12} & \cdots & x_{72} & x_{74} & x_{76} \end{bmatrix} \quad \text{(b)}$$

$$X = \begin{bmatrix} \boxed{x_{00}} & \boxed{x_{01}} & \boxed{x_{02}} & \boxed{x_{03}} & \boxed{x_{04}} & \boxed{x_{05}} & \boxed{x_{06}} & \boxed{x_{07}} \\ x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\ \boxed{x_{20}} & \boxed{x_{21}} & \boxed{x_{22}} & \boxed{x_{23}} & \boxed{x_{24}} & \boxed{x_{25}} & \boxed{x_{26}} & \boxed{x_{27}} \\ x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\ \boxed{x_{40}} & \boxed{x_{41}} & \boxed{x_{42}} & \boxed{x_{43}} & \boxed{x_{44}} & \boxed{x_{45}} & \boxed{x_{46}} & \boxed{x_{47}} \\ x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\ \boxed{x_{60}} & \boxed{x_{61}} & \boxed{x_{62}} & \boxed{x_{63}} & \boxed{x_{64}} & \boxed{x_{65}} & \boxed{x_{66}} & \boxed{x_{67}} \\ x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77} \end{bmatrix} \quad \text{(c)}$$

$$DX_v = \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & \cdots & x_{65} & x_{66} & x_{67} \end{bmatrix} \quad \text{(d)}$$

Fig. 6. The execution results of the decimation functions with $k = 2$ and $w = 8$: (a) is the original image block and the framed pixels will be selected by the horizontal decimation function; (b) is the output data of the horizontal decimation function; (c) is the same input block but the framed pixels will be selected by the vertical decimation function; (d) is the execution result of the vertical decimation function.

recovered through an interpolation process. The Interpolator performs two functions: the horizontal and the vertical interpolation functions, which are used to reconstruct the image blocks decimated by the horizontal (cf. Eq. (5)) and vertical (cf. Eq. (6)) decimation functions, respectively.

The horizontal interpolation function is defined as

$$\hat{X}(n) = \widehat{X_h}(n) = \widehat{DX}\left(\left\lfloor \frac{n}{d} \right\rfloor\right), \tag{10}$$

where $d$ is a positive integer defined in Eq. (5), and the vertical interpolation function is

$$\hat{X}(n) = \widehat{X_v}(n) = \widehat{DX}\left(\left\lfloor \frac{n}{w \times d} \right\rfloor \times w + (n \bmod w)\right), \tag{11}$$

where $w$ is defined in Eq. (6).

### 3.1.8. Image synthesizer

Finally, the image blocks $\hat{X}(n)$ recovered from the Interpolator are synthesized to an output image $\widehat{OX}(n)$ through the Image Synthesizer.

## 3.2. MIT sampling

### 3.2.1. Motivations and MIT sampling

Although the compression ratio of the modified BTC is higher than that of the traditional BTC, it is not high enough to meet the bandwidth requirement of video coding. In order to obtain a higher compression ratio, a novel sampling technique, the MIT sampling, will be introduced in this section. Several observations which motivate the development of this sampling technique are presented first.

**Observation 1.** The adjacent images in an MPC system are similar to each other and the DPCM technique can remove most of the time redundancy in the image sequence.

Observation 1 shows the trivial fact that most of the time redundancy can be removed by using the DPCM technique. When the image sequence is slow variant, the residual images obtained from the output of the DPCM process contain very little information, which can be discarded. When the image sequence contains a scene change or some fast-moving objects, the residual images of the DPCM would contain a lot of information and must be coded through the modified BTC. The following observations can assist us to code the result images more efficiently.

**Observation 2.** The residual images of the DPCM process are segmented into blocks in the SBMPC system. The activities of these residual image blocks are variant from one block to another.

The activity, ACT, of an image block is defined as

$$\mathrm{ACT} = \mathrm{RL}(1) - \mathrm{RL}(0), \tag{12}$$

where RL(0) and RL(1) are defined in Eqs. (3) and (4), respectively.

**Observation 3.** The fast-changing residual blocks, such as those containing the outlines of moving objects, usually have higher ACTs than those of the residual blocks with slow variant. Therefore, the information contained by the blocks with higher ACTs is considered to be more important than that contained by the blocks with lower ACTs.

The coding strategies of MIT sampling are as follows. For moving objects, it is more important to trace the actions of the objects fast and clearly than to describe the contents of the objects in detail. The residual image blocks which contain a part of the moving objects usually have higher ACT values than those blocks which are a part of the background image, as stated in Observation 3. The MIT sampler codes the higher activity blocks with higher sampling rate in the time axis and codes the lower activity blocks with lower sampling rate. The basic concept of MIT sampling is shown in Fig. 7. In order to simplify MIT sampling, the decision on temporal resolution is based on ACT values in the DPCM residual blocks of one picture instead of a set of pictures, and the decision is taken based just on a single ACT value for each residual block.

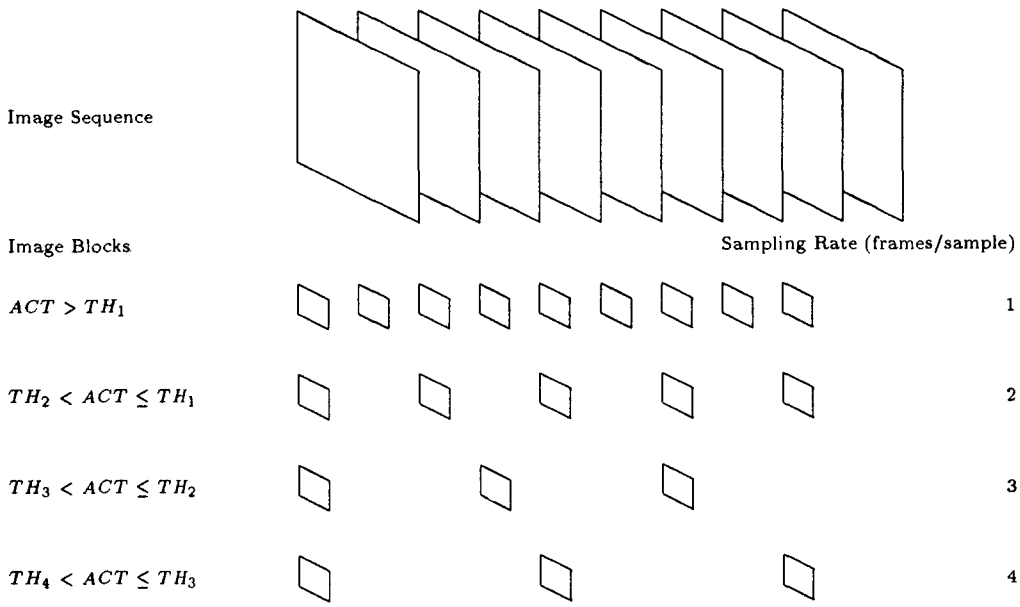


Fig. 7. The diagram to show the concept of MIT sampling, where $TH_1 > TH_2 > TH_3 > TH_4 > \cdots$.

Combining the modified BTC and the MIT sampling techniques, the SBMPC system can code the moving pictures with high coding speed and high compression ratio.

*3.2.2. The realization of the MIT sampler*

The MIT sampler is realized as follows. The residual image blocks, based on the corresponding activities, are classified into several classes. If all the blocks within the same activity class are sampled with the same time spacing, the so-obtained video sequence will become discontinuous. For this reason, instead of sequentially sampling these image blocks in the time axis, the SBMPC system subsamples the residual image blocks following a specified block sequence. For example, if the sampling rate of some activity class is 1/3 sample/block, only the 1st, 4th, 7th, ..., blocks in that class are coded. In order to sample these image blocks fairly, the starting sampled block is selected randomly, i.e. the MIT sampler samples the image sequences following the block sequence '1, 4, 7, ...', '2, 5, 8, ...', or '3, 6, 9, ...' with equal probability, in the above case.

MIT sampling has been realized by sampling the *image blocks within an image frame* instead of sampling the *same image blocks in the video sequence.* Thus, if the MIT sampler samples the blocks in a fixed phase, e.g. 1, 4, 7, ..., for all images, the blocks 2, 3, 5, 6, 8, ... will not be selected in all images. That is why we use the 'random phase' sampling instead of sampling at fixed phase.

The MIT sampler can also be realized without classifying the image blocks. In this case, the sampling function is defined as

$$\mathrm{MITS} = \delta_{((\mathrm{RL}(1)-\mathrm{QT})^2\sum_{0\leq i<\lceil 64/k\rceil}\delta_{SX(i)\geq \mathrm{QT}} + (\mathrm{QT}-\mathrm{RL}(0))^2\sum_{0\leq i<\lceil 64/k\rceil}\delta_{sx(i)<\mathrm{QT}})\mathrm{TH}_{\mathrm{MIT}} > \mathrm{rand}(R_{\mathrm{MIT}})}, \quad (13)$$

where the function rand($n$) returns a random number between 0 and $n-1$, $R_{\mathrm{MIT}}$ and $\mathrm{TH}_{\mathrm{MIT}}$ are the threshold values for controlling the MIT sampling, and the MIT sampler selects the image block when the function MITS returns 1, otherwise, the image block is not selected.

*3.3. The SBMPC encoder*

The block diagram of the SBMPC encoder is shown in Fig. 8. The input image is first partitioned

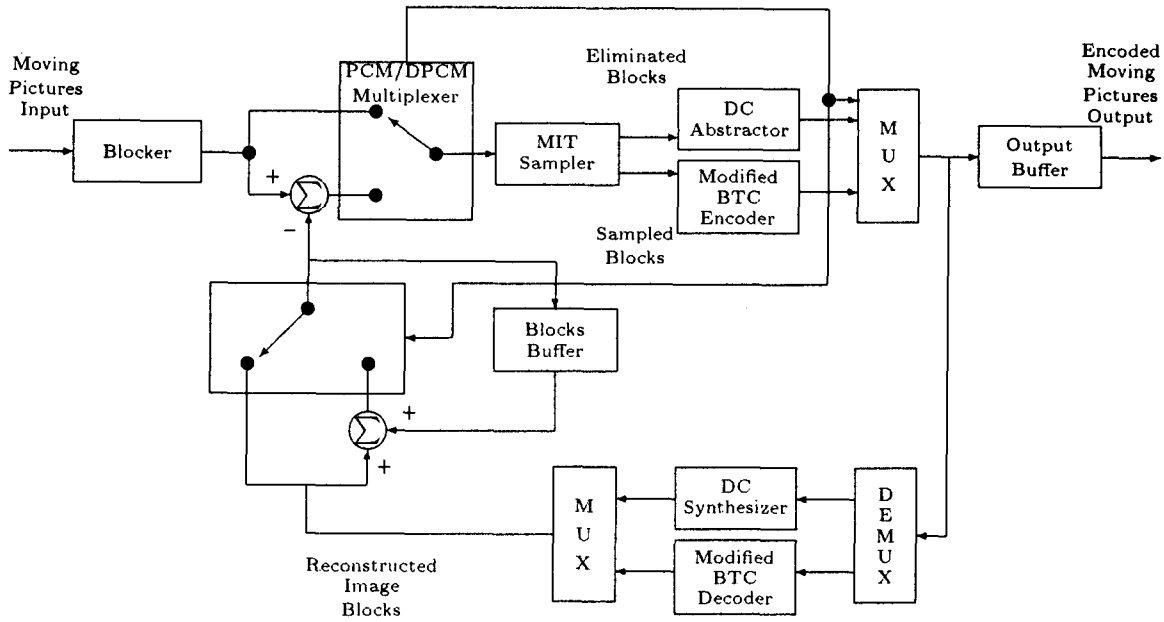


Fig. 8. The block diagram of the SBMPC encoder.

into image blocks by the Blocker. These image blocks are then sent into the PCM and DPCM simultaneously. For each input image block, two image blocks (one is the image block output from the PCM process and the other is the residual image block generated by the DPCM) will be obtained. The RL(0)s and RL(1)s of these two image blocks are calculated, called $RL_P(0)$, $RL_P(1)$, $RL_D(0)$ and $RL_D(1)$, respectively. Furthermore, the PCM/DPCM Multiplexer decides which image block must be sent to the MIT Sampler, based on the following function:

$$\text{Block} = \begin{cases} \text{Block}_{\text{PCM}} & \text{if } (RL_D(1) - RL_D(0)) - (RL_P(1) - RL_P(0)) > TH_{P/D}, \\ \text{Block}_{\text{DPCM}} & \text{otherwise}, \end{cases} \quad (14)$$

where Block is the output image block of the PCM/DPCM Multiplexer, $\text{Block}_{\text{PCM}}$ stands for the output image block of the PCM process, $\text{Block}_{\text{DPCM}}$ is the output image block of the DPCM process, and $TH_{P/D}$ is the threshold value for controlling the PCM/DPCM selection.

The MIT Sampler gets the image block sent by the PCM/DPCM Multiplexer and does or does not sample it based on the sampling rules described in Section 3.2. If the input image block is selected by the MIT Sampler, then the modified BTC algorithm is applied to encode this image block. Otherwise, the mean value of this image block is calculated.

For each image block, two control bits are needed: one indicates that this image block is a PCM or DPCM image block, the other indicates whether this image block is selected by the MIT Sampler or not. These control bits are multiplexed with the encoded data of a selected block or the mean value of an unselected block, and then sent to the SBMPC decoder.

### 3.4. The SBMPC decoder

Fig. 9 shows the block diagram of the SBMPC decoder. The encoded data is first demultiplexed according to the PCM/DPCM and the mean/MBTC control bits. When the PCM/DPCM control bit is set to '1', the DPCM technique is applied to this image block, and the decoder must add the pixel values in the image block of the

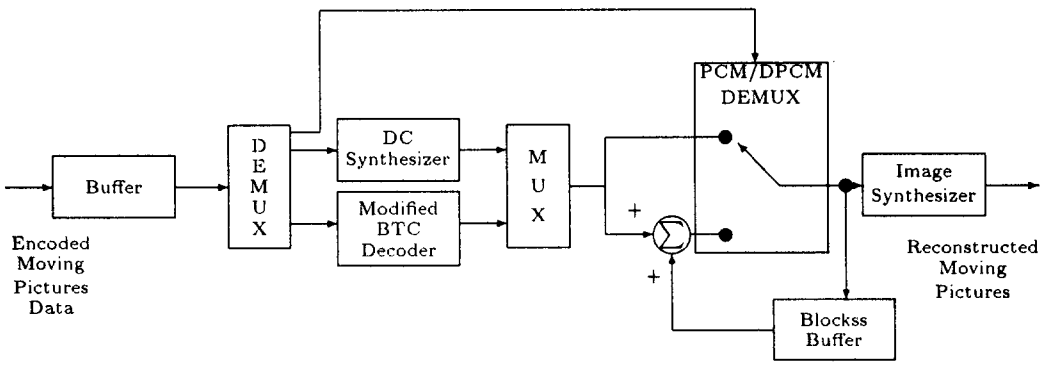


Fig. 9. The block diagram of the SBMPC decoder.

previous frame to those of the current frame. If the mean/MBTC control bit is set to '0', which indicates that this image block is represented just by a fixed (mean) value, then the mean value can be obtained from the buffer. The SBMPC decoder repeats the previous image block if the DPCM is used, and then adds the mean value to each pixel. Otherwise, if the mean/MBTC control bit is set to '1', the modified BTC decoder is applied to decode this image block, and if DPCM is used, the image block in the previous frame is added to the decoded image block. Finally, these decoded image blocks are synthesized into a picture frame through the Image Synthesizer and then displayed by the monitor.

## 4. Experimental results

Several experimental results are shown in this section. In these experiments, the input image size is $128 \times 128$, the pixel format is 15 bits/pixel and the R, G, B fields, respectively, occupy 5 bits. The SBMPC system is running on a PC/486-33 with the MS-DOS 5.0 operation system and a RAM disk driver. The distortion measurement function used in the following experiments is the peak signal-to-noise ratio (peak SNR) of the luminance component. Eq. (13) is used to perform the MIT sampling, and $TH_{P/D}$ and $R_{MIT}$ are, respectively, set to be $-1$ and 2000 in these experiments.

The following experiments use two B frames between each P frame pair and a logarithmic block-matching algorithm in which sums of magnitude differences are defined as the cost functions used by the adopted MPEG codec.[1] The search range for the motion estimation algorithm is $(-7.5)$–$(7.5)$ pixels for adjacent frame pairs and $(-15.5)$–$(15.5)$ pixels for other frame pairs. The quantizer's factor for P and B frames is twice that of I frames. The coding speed is 0.45 frame/s for MPEG encoding and 6.6 frame/s for decoding.

In order to evaluate the coding speed of the SBMPC system, two time intervals are measured. One is the gross coding time including the frame grab time, encoding time, transmission (saving) time, and the time interval for playing back to the display. The other is the net coding time including the encoding time and the transmission time only. In these experiments, the average gross coding time is 141 ms/frame and the average gross coding speed is about 7 frame/s. The average net coding time is 72 ms/frame and the average net coding speed is about 14 frame/s. The comparison of the coding speed of the SBMPC system and the MPEG coder is shown in Table 1. In this table, the time for converting RGB frames into YUV frames is included in measuring the MPEG coding speed, and it is based on the gross coding speed measurement.

[1] The MPEG coders are CMPEG V1.0 for the MPEG encoder and DMPEG V1.1 for the MPEG decoder, copyright by Stefan Eckart, Chair for Integrated Circuits, Technical University Munich, Germany, June 1993, and was posted on newsgroup: alt.binaries.pictures.utilities. Email: stefan@lis.e-technik.tu-muenchen.de.

In the first experiment, the SBMPC system encodes the Chinese girl video sequence. The $TH_{MIT}$s are, respectively, set to be 40, 80 and 15 for the R, G and B pixel planes. The compression ratios and the peak SNRs of the encoded video sequence are shown in Fig. 10(a) and 10(b), respectively. In this experiment, the average compression ratio is 13.263 and the average peak SNR is 29.749. Under the same compression ratio, the peak SNR of the video sequence encoded by the MPEG coder is 27.4376.

In the second experiment, the same video sequence is used except that the $TH_{MIT}$s are set to be 20, 40 and 7. The compression ratio and the peak SNR of the encoded video sequence are, respectively, shown in Fig. 11(a) and 11(b), the average compression ratio is 22.226, and the average peak SNR is 28.59. The peak SNR of the same video sequence encoded by the MPEG coder with the same compression ratio is 27.1901.

In the third and fourth experiments (see Figs. 12 and 13), the $TH_{MIT}$s are the same as those used in the first and second experiments, respectively, but the salesman video sequence is used instead of the Chinese girl one. In the third experiment, the average compression ratio is 18.157 and the average peak SNR is 29.157. The average compression ratio is 24.640 and the average peak SNR is 27.176 in the fourth experiment. Under the same compression ratios, the peak SNRs of the video sequences encoded by the MPEG coder are 29.4266 and 29.1786, respectively.

As shown in the above experimental results, the larger the $TH_{MIT}$, the better the picture quality and the lower the compression ratio. The other control parameter, $R_{MIT}$ (used in Eq. (13)), affects the picture quality and the compression ratio in the opposite way.

The final experiment shows the effects caused by adjusting the parameter $TH_{P/D}$. In this experiment, all the control parameters are the same as those used in the first experiment except that the $TH_{P/D}$ varies from $-3$ to 3. Table 2 shows the DPCM

Table 1
The measurements of coding speed of the SBMPC system and the MPEG coder

| Functions | Measurements |
|---|---|
| Net coding time of SBMPC | 72 ms/frame |
| Net coding speed of SBMPC | 14 frame/s |
| Gross coding time of SBMPC | 141 ms/frame |
| Gross coding speed of SBMPC | 7 frame/s |
| Encoding speed of MPEG coder | 0.45 frame/s |
| Decoding speed of MPEG coder | 6.6 frame/s |

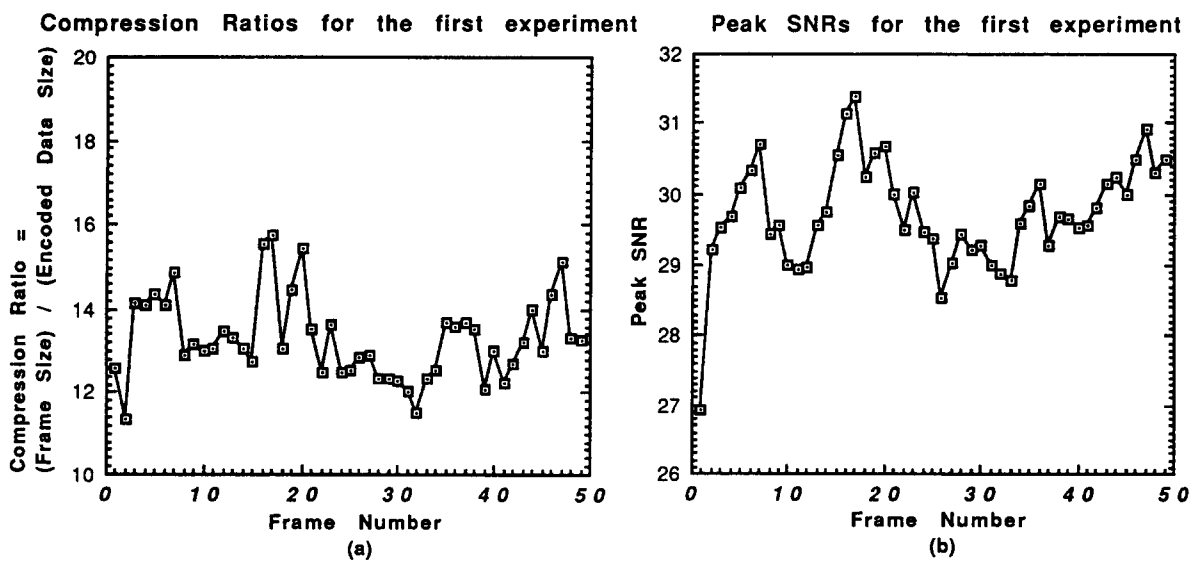


Fig. 10. (a) Compression ratio and (b) peak SNR of the first experiment for encoding the Chinese girl video sequence.

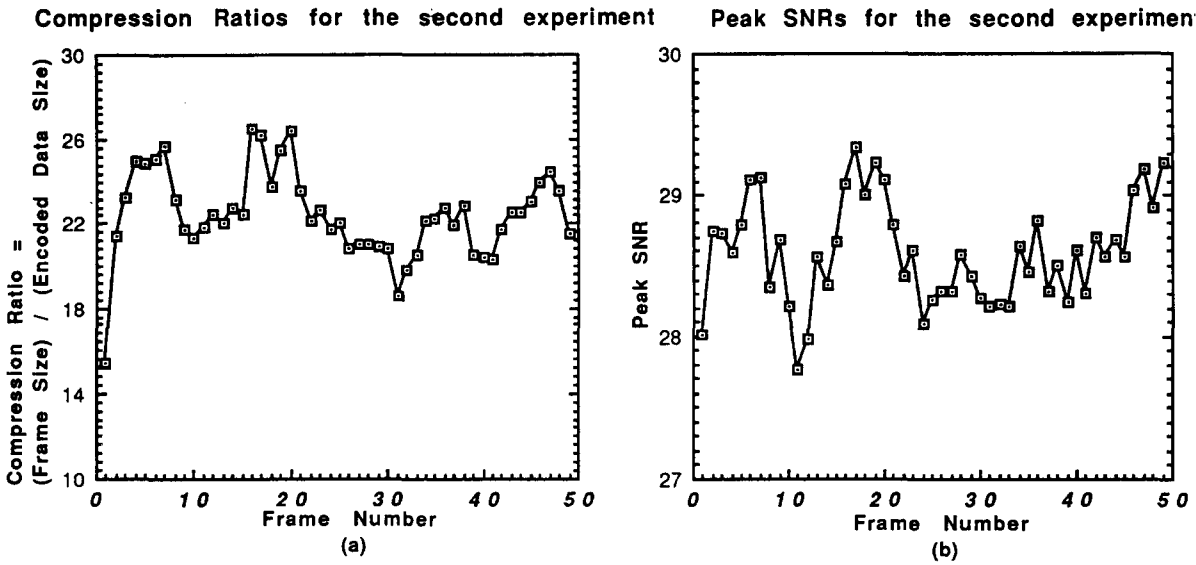


Fig. 11. (a) Compression ratio and (b) peak SNR of the second experiment for encoding the Chinese girl video sequence.

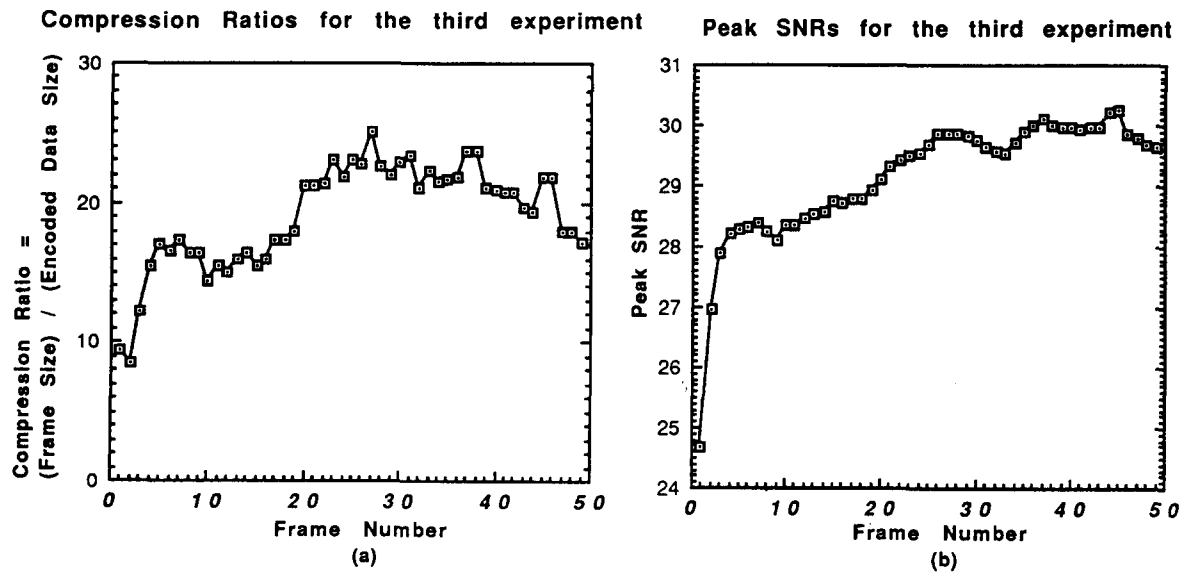


Fig. 12. (a) Compression ratio and (b) peak SNR of the third experiment for encoding the salesman video sequence.

utilities, the average compression ratios, and the average peak SNRs with respect to these distinct $TH_{P/D}$ values. As shown in Table 2, the larger the $TH_{P/D}$, the higher the DPCM utility. If the value of $TH_{P/D}$ is too large, the SBMPC system tends to use the DPCM technique regardless of the fast changes of image blocks; on the other hand, if the value of $TH_{P/D}$ is too small, the SBMPC system tends to use the PCM technique without taking the advantage of the high correlations between two similar image blocks. Both cases degrade the compression ratio and the image quality of the output data. In this experiment, the optimal value of $TH_{P/D}$ is $-1$, which is suitable for most cases.

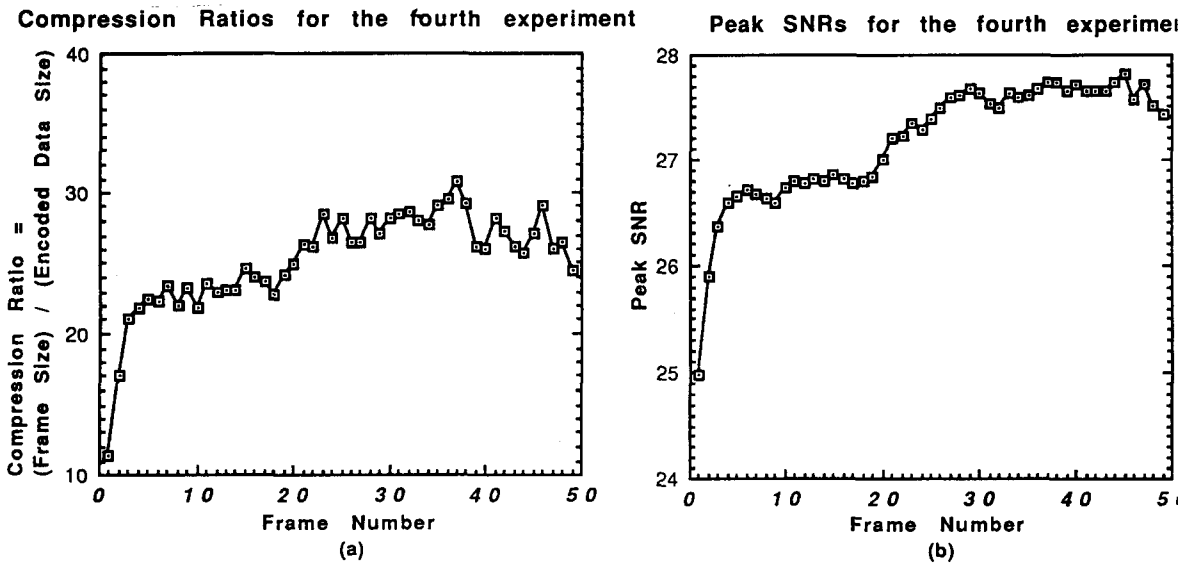


Fig. 13. (a) Compression ratio and (b) peak SNR of the fourth experiment for encoding the salesman video sequence.

Table 2
DPCM utilities, average compression ratios, and average peak SNRs at variations of the $TH_{P/D}$ value

| $TH_{P/D}$ | −3 | −2 | −1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|---|
| DPCM utility (%) | 20.22 | 27.798 | 72.47 | 97.57 | 98.868 | 99.320 | 99.532 |
| Compression ratio | 12.053 | 12.009 | 13.263 | 12.552 | 12.302 | 12.178 | 12.046 |
| Peak SNR | 29.069 | 29.345 | 29.749 | 29.589 | 29.381 | 29.272 | 29.182 |

## 5. Discussions and conclusions

This paper presents a newly proposed software-based moving picture coding system. This system uses the modified BTC and the MIT sampling techniques to achieve a high compression ratio and maintain good enough picture quality. As shown by the experimental results, given in the previous section, the coding speed of the SBMPC system can meet the real-time requirement of commercial video applications when the size of the compressing video is $128 \times 128$ and the system is run on a PC/486-33. Most of the video applications, such as videophone and videoconference, can reduce the amount of communication and storage costs, by using the SBMPC system, without adding any hardware costs.

When the computation power is increased in the future, the performance of the SBMPC system can be improved in several ways. First, entropy coding can be included to encode the data more compactly. Second, some more complex algorithms such as wavelet transform may be used to remove the correlation of the video data more precisely. Third, the frame rate can be further increased to obtain a more smooth video, and finally the frame size can be enlarged.

In order to decide whether the DPCM technique is used for encoding an image block or not, the decision function defined in Eq. (14) was used; however, this function was designed from the viewpoint of computational speed and thus is not perfect. The design of a better decision function without adding too much computational cost is still underway.

For encoding color video, the SBMPC system separates a picture frame into red, green and blue pixel planes, and then encodes and decodes each

pixel plane independently. The compression ratio may be increased by transforming the color pixels into another color domain, such as YUV, and this is also a future work.

## Acknowledgments

This work was supported by National Science Council of Republic of China under grants NSC-83-0425-E-002-140 and NSC-83-0408-E-002-004.

## References

[1] N. Ahmed, T. Natarajan and K. Rao, "Discrete cosine transform", *IEEE Trans. Comput.*, January 1987, pp. 90–93.

[2] CCITT Study Group XV, Description of Reference Model 8 (RM8), Working Party XV/4, Specialists Group on Coding for Visual Telephony, Doc. 525, June 1989.

[3] E.J. Delp and O.R. Mitchell, "Image compression using block truncation coding", *IEEE Trans. Comm.*, Vol. 27, 1979, pp. 1335–1342.

[4] "Draft revision of Reommendation H.261: Video codec for audio-visual services at $p \times 64$ kbits/s", *Signal Processing: Image Communication*, Vol. 2, No. 2, August 1990, pp. 221–239.

[5] P. Elias, "Predictive coding", *IRE Trans. Inform. Theory*, March 1955, pp. 16–33.

[6] MPEG Standard Draft ISO-IEC/JTC1 SC29 on 22, November 1991.

[7] R. Srinivasan and K.R. Rao, "Motion-compensated coder for video-conferencing", *IEEE Trans. Comm.*, Vol. 35, March 1987, pp. 397–404.