Multimedia Systems (1993) 1:110–119

Multimedia Systems

Real-time software-based video coder for multimedia communication systems

Ho-Chao Huang, Jau-Hsiung Huang, and Ja-Ling Wu

Communications & Multimedia Laboratory, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

Received May, 1993/Accepted July, 1993

Abstract. A novel software-based video compression algorithm, the Popular Video Coder (PVC), is presented in this paper, and a video phone system, the Popular Phone, is also implemented based on the PVC. The PVC simplifies the traditional video coder by removing the transform and the motion estimation parts and modifies the quantizer and entropy coder. Two novel coding algorithms, the adaptive quantizer and the modified windowed Huffman-like coder, are used in the PVC to encode the video data with a quality picture at a high compression ratio. The video quality of the proposed coder is as good as that of the MPEG coder when the input is a low-resolution and slow-motion video, and the computational complexity of the PVC is much lower than that of the Motion Picture Expert Group (MPEG). Since no compression hardware is needed for the PVC to encode and decode video data, the cost and complexity of developing multimedia applications, such as video phone and multimedia e-mail systems, can be greatly reduced. Furthermore, some networking issues, such as error control and flow control, are discussed in connection with applying the PVC to implement the Popular Phone.

Key words: Communication – Multimedia system – Software-based video compression – Video data compression – Video phone/conference

1 Introduction

Video has become one of the most important media in multimedia applications. The efficient compression of the video data is one of the most important functions in such applications. Most existing video coding techniques, such as those of the H.261 [1], RM8 [2] and MPEG [3], have been designed to compress data by reducing both the spatial correlation through the transform coding, such as the discrete cosine transform (DCT) [4], and the temporal correlation through the Differential Pulse Code Modulation (DPCM) [5] and motion compensation (MC) [6] techniques. The major problems of MC/DCT-based techniques are the high computation complexity, the block effect, and the poor performance in dealing with sharp edges. Due to high computation complexity of the coder, we are not able to realize those coding techniques for real-time applications with software. As a result, the cost of such video applications is high and the implementation and maintenance are difficult.

Recently, the realization of the video coder in software to reduce both the cost and the complexity of multimedia systems has been widely discussed. The real-time software-based moving picture coding (SBMPC) system [7] has been proposed to solve this problem. The SBMPC system uses the modified block truncation code (BTC) and the multiresolution-in-time (MIT) sampling techniques to achieve the high compression ratio and real-time requirements. However, the picture quality of SBMPC system is not good enough for commercial uses.

In this paper, a software-based video coding system, called the Popular Video Coder (PVC), is proposed. The PVC combines the DPCM technique, the adaptive quantizer and the modified windowed Huffman-like algorithm (which is a modified version of the windowed Huffman algorithm proposed in [8]) to compress the video data. By using these techniques, the PVC can compress the video data swiftly and efficiently. The range of the compression ratio of the PVC is from several to hundreds. In our experiments, the average compression ratio, peak signal-to-noise ratio (SNR) and the average frame rate are approximately 34, 30 dB and 29 frame/s respectively. The resolution of each video frame is 128 × 128 with color. When the input is a low-resolution and slow-motion video, the picture quality of the PVC is even better than that of the MPEG and therefore is good enough for commercial uses. The coding speed of both the encoding and the decoding of the PVC is mush faster than those of the MPEG and is also fast enough to meet the real-time constraint of commercial video applications. Moreover, no block, ripple, and blurring effects are produced by the PVC.

Correspondence to: H.C. Huang

When applying the PVC to multimedia communication applications, such as video phone and video conference, some networking issues must be considered. In this paper, we propose some methods of error control and flow control on the network transmission by using the PVC. We will also demonstrate the video phone system, the Popular Phone, developed in our laboratory using the PVC as the video coder.

This paper is organized as follows. Section 2 reviews the traditional MC/DCT-based video coders and lists the problems of such coders. The windowed Huffman algorithm is also reviewed in this section. Section 3 describes the proposed coder, the PVC, including the adaptive quantizer and the modified windowed Huffman-like algorithm. Section 4 demonstrates the experimental results of the PVC and the comparisons between the PVC and the MPEG coder. The networking issues and functions of the PVC are discussed in Sect. 5. In the same section, the Popular Phone is presented as an example of applying the PVC to multimedia communication systems. Finally, some discussions and conclusions are presented in Sect. 6.

2 Preliminary reviews

In order to make the materials presented in this paper complete, the MC/DCT-based video coders and the windowed Huffman algorithm [8] are briefly reviewed in this section.

2.1 The MC/DCT-based video coder

The basic idea of MC/DCT-based video coders is to reduce the bit rate by using the source coding to remove the spatial and temporal redundancies, and using the entropy coding to remove the codeword redundancies. A typical source coder usually consists of a predictor and a transform coder. The prediction coder removes the temporal redundancies, and the transform coder removes the spatial redundancies. A typical entropy coder usually contains a run-length coder and a Huffman coder. The run-length coder encodes the zeros in succession of the quantized transform coefficients, and the Huffman coder represents the run-length coded tuples in the most compact form. The block diagram of H.261 encoder is shown in Fig. 1.

Although MC/DCT-based video coders have the capability of encoding moving pictures with relatively high compression ratio and picture quality, the following problems of this coding scheme remain.

1. *High computation complexity is required.* This is the major problem of MC/DCT-based video coders. Enormous operations are needed in order to perform the functions of the motion estimation and the DCT. Even though fast algorithms of MC and DCT are used, the processing is still too slow for real-time multimedia applications using software based MC/DCT video coders. However, the hardware solution of the video coder has some disadvantages:
   a. The cost of these compression chips is very high.
   b. The development of device drivers is time-consuming and not trivial.

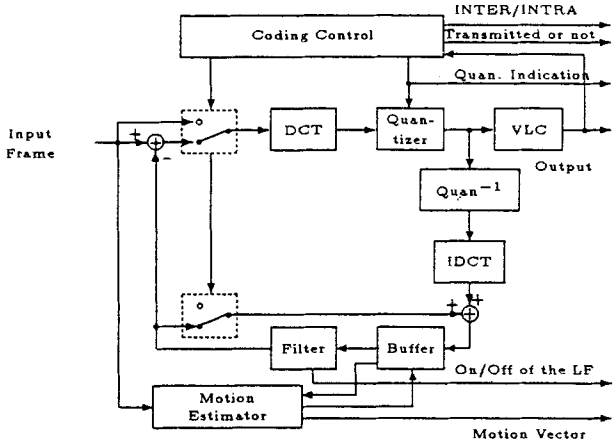


Fig. 1. Block diagram of the CCITT H.261 encoder

   c. It is inconvenient to install, maintain, and upgrade video applications by using sophisticated compression chips.

2. *Block effect exists.* MC/DCT-based video coders divide the input image into 8×8 image blocks and encode these image blocks independently. The distortion caused by the quantizer for each image block is discontinuous and independent (the block effect). Under high compression ratio, the block effect of output pictures will make viewers uncomfortable.

3. *Ripple and blurring effects exist.* The basic idea of the transform coding is to analyze the distribution of a signal, such as luminances of pixels in a picture, through transformation, and then to quantize the transformed coefficients. For a signal containing sharp edges, some values of high frequency coefficients cannot be ignored and the distortions caused by the quantizer will give the reconstructed signal obvious ripple effects at the plate regions and blurring effects at the sharp edges. At a high compression ratio, these effects will degrade the quality of the reconstructed video notably.

2.2 The windowed Huffman algorithm

The original idea of this algorithm was proposed in [9, 10] and a similar coding scheme applied to arithmetic coding was proposed in [11]. The code tree of the windowed Huffman algorithm is constructed by adjusting the tree when the weight of a node is increased or decreased. The weight of a leaf node in the windowed Huffman algorithm is the number of occurrences of the associated symbol in the limited past history. In the windowed Huffman algorithm, a window with size n is used to store the most recently n encoded symbols. When a symbol is encoded it is added into the window, then the weight of the corresponding node is increased, and the *Update_Increase* procedure, which is similar to the *Update* procedures of algorithm V [10] and algorithm FGK [12], is performed to adjust the windowed Huffman tree. When the window is full, the oldest symbol in the window is removed, the weight of the node corresponding to the removed symbol is decreased, and the *Update_Decrease* procedure, which is similar to the *Update* procedure of residual Huffman algorithm [13], is performed

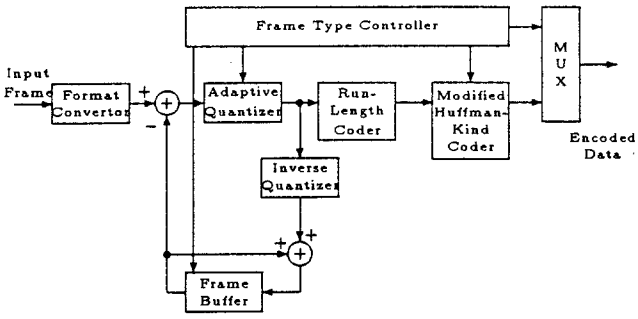


Fig. 2. The block diagram of the PVC encoder

to readjust the code tree. If the size of the window is selected properly, the coding efficiency of the windowed Huffman algorithm is the best among all static and dynamic Huffman algorithms.

Although the performance of the windowed Huffman algorithm is the best, the processing speed of the algorithm is not high enough to meet the real-time constraint of video applications. Therefore, in order to apply the windowed Huffman algorithm to those applications, some modifications to speed up the coding process is needed. Therefore, the modified windowed Huffman-like algorithm is proposed in this paper.

3 The PVC

The block diagram of the PVC encoder is shown in Fig. 2. In the PVC, the input video sequence is organized as groups of frames (GOFs). There are four frame types within a GOF, which will be described in Sect. 3.2, and they are controlled by the *frame type controller.* The type of frame affects the functions of the *adaptive quantizer,* the *frame buffer* and the *modified windowed Huffman-like coder.*

In order to meet the real-time constraint of video applications, the PVC discards the DCT coder and the motion compensator, which are widely used in MC/DCT-based coders. And then, in order to speed up the coding process and increase the coding efficiency further, a fast version of dynamic Huffman coder, called the modified windowed Huffman-like coder, is proposed as part of the entropy coder in the PVC. The adaptive quantizer of the PVC controls the bit rate of the output data and the picture quality of the reconstructed image, and the behavior of the adaptive quantizer is controlled by the frame type controller. Because the PVC encodes the input image pixel by pixel instead of block by block, no block effect will be induced. Moreover, since the PVC quantizes the image pixel in the time domain instead of the frequency domain, no blurring or ripple effects will be caused. The flow of the coding process follows.

For color image input, its pixel components are first converted from the RGB domain to the YUV domain, and then the Y, U and V components of the image are encoded independently. The Y, U and V components of the input image are subtracted from the reconstructed previous frame stored in the frame buffer, then quantized by the adaptive quantizer,

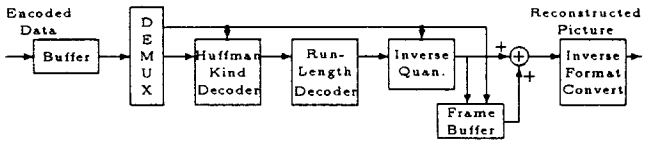


Fig. 3. The block diagram of the PVC decoder

and encoded by the run-length encoder and the modified windowed Huffman-like coder. Meanwhile, the quantized Y, U and V components are dequantized (reconstructed) and stored in frame buffer, to be used as the reconstructed previous frame in encoding the next frame. The decoding process of the PVC is similar to the reverse process of the encoder and its block diagram is shown in Fig. 3. The following subsections will depict the function blocks of the PVC in detail.

3.1 Format convertor

The *format convertor* converts the RGB components of pixels to YUV components. After the format conversion, the U and V components are then 4 to 1 subsampled. The functions of format conversion are:

$$Y(x) = 0.299 \times R(x) + 0.587 \times G(x) + 0.114 \times B(x) \quad (1)$$

$$U(x) = (B(x) - Y(x)) \times 0.493/1.74 \quad (2)$$

$$V(x) = (R(x) - Y(x)) \times 0.877/2.46 \quad ((3)$$

where $R(x), G(x), B(x), Y(x), U(x)$ and $V(x)$ are the values of red, green, blue, Y, U, and V components of pixel x, respectively.

3.2 Frame type controller

The frame type controller defines the organization of the GOF, sets the type of each frame, and controls the parameters of adaptive quantizer, frame buffer and modified windowed Huffman-like coder. There are four types of frames within a GOF: the Refresh Frame (R-Frame), the Fine Frame (F-Frame), the Medium Frame (M-Frame) and the Coarse Frame (C-Frame) types. The use of the R-Frame is just for the error control of the communicated video, and it is only used at the beginning of a GOF. Excepting the R-Frame, the F-Frame has the smallest compression ratio but possesses the best picture quality. The most gain of the compression ratio comes from the C-Frame, but the picture quality of the C-Frame is not good enough. The picture quality and compression ratio of the M-Frame are in between those of the F-Frame and the C-Frame. The use of the M-Frame is the tradeoff between the high picture quality and the high compression ratio. Their descriptions follow.

Refresh Frame. In the R-Frame, the frame buffer is cleared, the Huffman table of the modified window Huffman-like coder is reset, and the quantization steps of the adaptive quantizer are set similarly to those in the Fine Frame. The type of the first frame in a GOF is always set as the R-Frame.

Fig. 4. A typical frame type sequence for the case of the vector $\boldsymbol{FT}$ is equal to (12,6,3)

Fine Frame. In the F-Frame, the quantization steps are the smallest of all four frame types. Therefore, the picture quality of the F-Frame is the best of all.

Medium Frame. In the M-Frame, the quantization steps are larger than those in the F-Frame.

Coarse Frame. The quantization steps used in the C-Frame are the largest, and the picture quality of the C-Frame is the worst.

The parameters to control the frame type controller can be written as the vector FT: (GL, FL, ML), where GL indicates the length of GOF, FL sets the frame distance of two F-Better-Frames (where an F-Better-Frame is defined as an F-Frame or an R-Frame), and ML sets the frame distance of two M-Better-Frames (where an M-Better-Frame is defined as an M-Frame or an F-Frame or an R-Frame). The number of C-Frames between any two other type frames is equal to $ML - 1$. A typical organization of a GOF is shown in Fig. 4, and the vector FT is chosen to be (12,6,3).

3.3 Adaptive quantizer

The adaptive quantizer quantizes the Y, U, and V components output by the DPCM. The quantizer is controlled by two parameters: the step size S and the dead zone size Z. The function of the adaptive auantizer is:

$$Q(x) = \begin{cases} \left\lfloor \frac{x+S-Z-1}{S} \right\rfloor , & \text{if } x \geq Z \\ 0 , & \text{if } |x| < Z \\ \left\lfloor \frac{x-S+Z+1}{S} \right\rfloor , & \text{if } x \leq -Z \end{cases} .$$

The inverse quantizer reconstructs the signal $\hat{x}$ from the quantized value $Q(x)$. Figure 5 presents the function that combines those of the adaptive quantizer and the inverse quantizer, with $S = 3$ and $Z = 4$.

The determination of parameters of the adaptive quantizer depends on the component to be quantized. Therefore, the parameters that control the adaptive quantizer can be written as the vector $\boldsymbol{AQ}$: (YS, US, VS, YZ, UZ, VZ), where YS, US, and VS are the quantization step sizes for quantizing Y, U, and V components, respectively, and YZ, UZ, and VZ are the dead zone sizes for quantizing Y, U and V components respectively.

The adaptive quantizer is the key component of the PVC. Both S and Z are important control parameters. S affects the quality of all encoded frames, and the compression ratios of R-Frames and F-Frames. Z affects the compression ratios of all frames significantly. The larger the value of Z, the lower bit rate will be obtained. However, for a large Z value, the moving objects will retain visible residues on the succeeding frames which will degrade the quality of these frames.

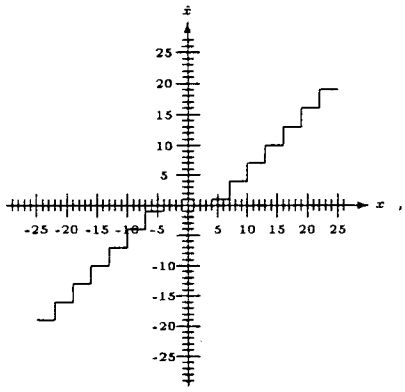


Fig. 5. The function which combines those of adaptive quantizer and inverse quantizer, with $Z = 4$ and $S = 3$

3.4 The run-length encoder

The run-length encoder represents the quantized values as the run-length tuples $RL(Z, V)$, where Z indicates the length of zeros in succession, and this zero run is followed by a quantized value V.

3.5 The modified windowed Huffman-like coder

After the quantized signal is encoded as the run-length tuples, these tuples are then represented in more compact form by using Huffman code [14]. Usually, the probability distribution of each symbol is time variant; hence, the concept of the windowed Huffman algorithm [8] is applied. In order to speed up the coding process to meet the real-time constraint, the modified windowed Huffman-like coder modifies the *Initial* and the *Update* procedures of [8], which are used to construct and adjust the Huffman code tree, respectively.

The maximum run-length used in run-length encoder is 15 so that 4 bits are needed for representing each run-length. The range of a quantized value in adaptive quantizer is from -7 to 8 so that 4 bits are needed to represent it, too. In the PVC, the two fields of a run-length tuple $RL(Z, V)$ are combined as an 8-bit symbol. That is, the alphabet size of modified windowed Huffman-like encoder is 256. In order to simplify the encoding procedure, the concept of the length-limited Huffman code [15, 16] is used. In order to speed up the process of code tree reconstruction, a simplified version of the *Construction* procedure is adopted in the PVC. The maximum code length of encoded symbols is limited to 12 bits.

The encoding process of modified windowed Huffman-like encoder is briefly described as follows. Before encoding each symbol, the encoder first checks whether it is time to reconstruct the Huffman code tree. If it is, the encoder calls the *Construct_CodeTree* procedure. In the PVC, the Huffman code tree is reconstructed for every three frames. After the reconstruction process, the next input symbol is encoded by

the *Encode* procedure and the code table is adjusted by *Update* procedure. The *Update* procedures proposed in [8] and [13] can be used to adjust the Huffman code tree, but in view of the computation speed, the *Update* procedure in modified windowed Huffman-like encoder removes the part of code tree adjustment, and increases the weight of the encoded symbol only.

The *Construct_CodeTree* procedure first sorts symbols by their probabilities, assigns an initial code length l_a to each symbol a with probability P_a using the formula $\lceil \log_2 1/P_a \rceil$, then adjusts the code lengths so that $\sum_a 2^{-l_a} = 1$, and finally, assigns a codeword with the adjusted code length to each symbol. The codeword constructed by the *Construct_CodeTree* procedure is more like a Shannon-Fano code than a Huffman code. The code efficiency of the modified windowed Huffman-like code is worse than that of windowed Huffman code, but the speed of tree construction is faster.

4 Experimental results

This section shows several experiment results. In these experiments, the resolution of each image is 128 × 128, the pixel format is 15 bits/pixel where the R, G, and B fields occupy 5 bits each. The AQ vectors are fixed as (3, 1, 1, 0, 0, 0), (2, 1, 1, 0, 0, 0), (2, 2, 2, 2, 0, 0) and (2, 1, 1, 2, 1, 1) in the R-Frame, the F-Frame, the M-Frame and the C-Frame, respectively. We run the PVC on a PC/486-33 under MS-DOS 5.0. The distortion-measurement function used in the following experiments is the weighted sum of peak-to-peak signal-to-noise ratio (PSNR) of the RGB components. The distortion function is defined as:

$$PSNR = 10 \times \log_{10} \frac{\sum 31^2}{\sum 0.299 \times (\widehat{R} - R)^2 + 0.587 \times (\widehat{G} - G)^2 + 0.114 \times (\widehat{B} - B)^2} \quad (5)$$

In order to evaluate the encoding speed of the PVC, two time intervals are measured. One is the *gross encoding time* including the frame grabbing time, encoding time, and the time interval for playback to the display. The other is the *net coding time* including the encoding time only. In these experiments, the average *net encoding time* is 34.617 ms/frame, which results in an average *net encoding speed* at about 29 frame/s. The average gross encoding time is 58.926 ms/frame, which results in an average *gross encoding speed* at about 17 frame/s. Two time intervals are also evaluated for the decoding process of the PVC. The average *net decoding time* is 23.383 ms/frame with an average *net decoding speed* at approximately 43 frame/s. The average *gross decoding time* is 33.672 ms/frame with an average *gross decoding speed* at approximately 30 frame/s. These results show that the coding speed of the PVC is well-suited to real-time video applications. The performance measurements of PVC are shown in Table 1.

The first three experiments encode the salesman video sequence with the PVC. In these experiments, the $\boldsymbol{FT}$ vectors are chosen to be (256,27,9), (256,3,1) and (256,60,20). The compression ratios and PSNRs of these experiments are shown in Fig. 6a and b. The compression ratio of the C-Frame is the highest of all, and that of the M-Frame is higher than that of F-Frame.

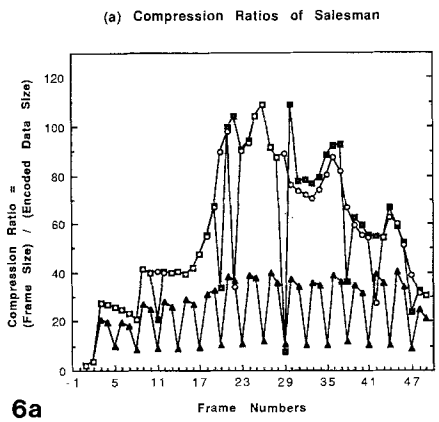
Table 1. The measurements of the PVC, the MPEG coder and the popular phone

Functions	Measurements
Net encoding time of the PVC	34.617 ms/frame
Net encoding speed of the PVC	29 frame/s
Gross encoding time of the PVC	58.926 ms/frame
Gross encoding speed of the PVC	17 frame/s
Net decoding time of the PVC	23.383 ms/frame
Net decoding speed of the PVC	43 frame/s
Gross decoding time of the PVC	33.672 ms/frame
Gross decoding speed of the PVC	30 frame/s
Encoding speed of the MPEG coder	0.45 frame/s
Decoding speed of the MPEG coder	6.6 frame/s
Video-for-windows (Enc 160×120)	2.2 frame/s
Video-for-windows (Dec 160×120)	14 frame/s
Frame rate of the popular phone	6–10 frame/s

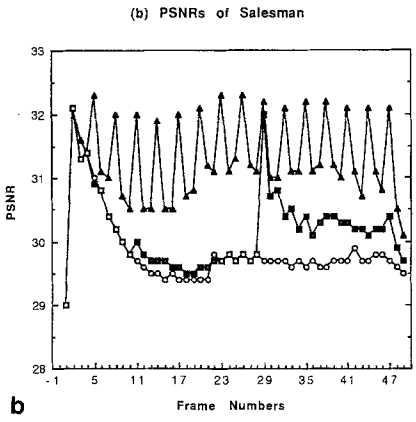
The second three experiments encode the Chinese girl video sequence. The $\boldsymbol{FT}$ vectors are the same as in the corresponding previous three experiments, and the compression ratios and PSNRs of the second set of experiments are shown in Fig. 7a and b respectively.

The third set of experiments evaluates the PSNRs and the average compression ratios of the PVC by changing the value of $\boldsymbol{FL}$, where the values of RL and ML are set at 256 and $\boldsymbol{FL}/3$ respectively. In these experiments, two average compression ratios are evaluated. One is the average compression ratio (ACR) of the entire video sequence, and the other is the average communication compression ratio (ACCR) of the video sequence excepting that of the refresh frame. In communication applications, the refresh frames are only used to initialize the PVC and to recover from transmission errors. They only affect the setup time of the PVC, and will not add any traffic load to the network in normal status. Therefore, for communication applications, the evaluation of the ACCR is more meaningful than the evaluation of the ACR of the PVC. Figure 8 shows the PSNRs, ACRs and ACCRs of encoding the salesman and the Chinese girl sequences. These figures show that the larger the $\boldsymbol{FL}$ is, the lower the average PSNR and the higher the ACCR would be.

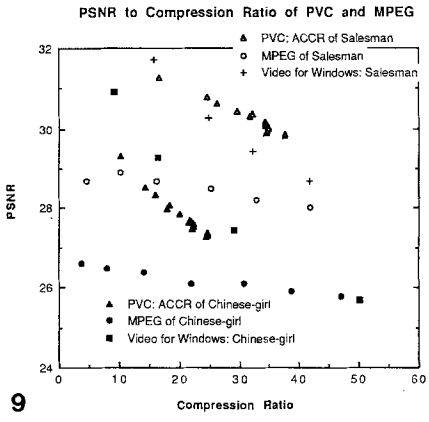
The fourth experiment compares the PSNR to that of the MPEG coder at the same compression ratio. As the experimental results shown in Fig. 9, the PSNRs of the PVC are even higher than those of the MPEG coder for the salesman and Chinese girl sequences. The experimental results for encoding the table tennis and the flower garden video sequences is shown in Fig. 10. In this experiment, the MPEG coder performs better than the PVC, because the table tennis sequence



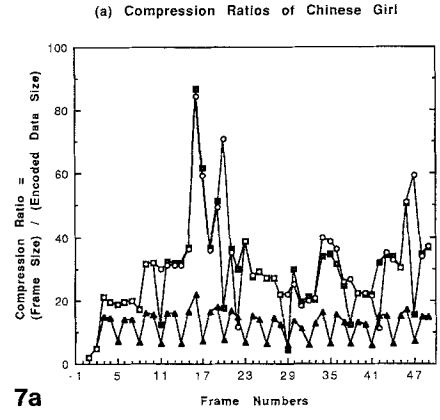
6a



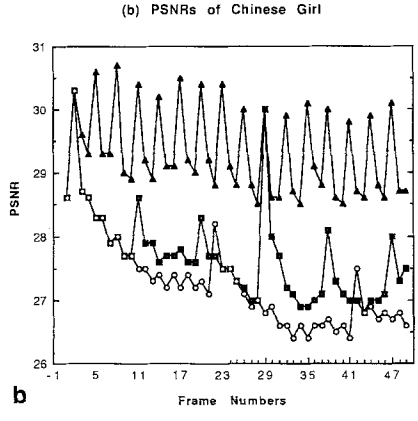
b



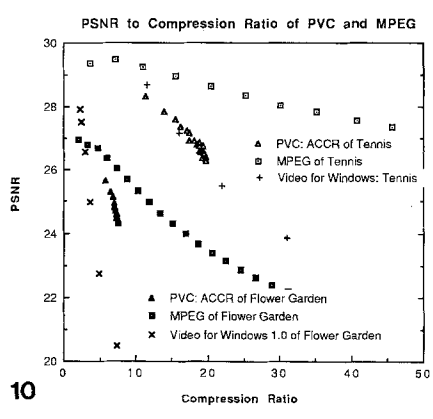
9



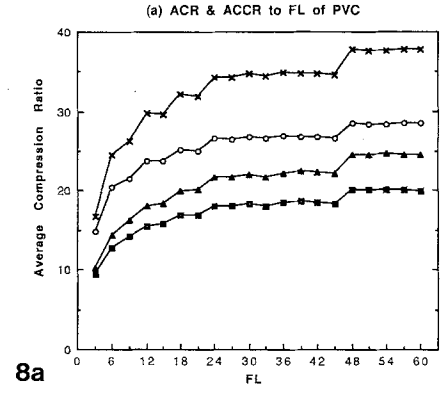
7a



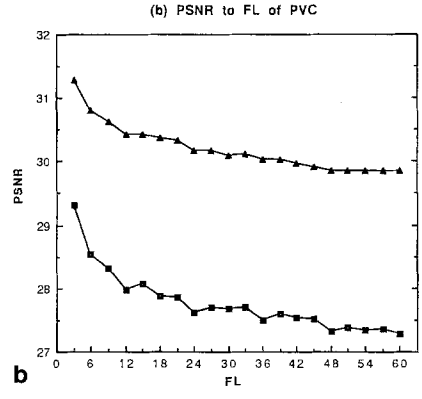
b



10



8a



b

Fig. 6a,b. Data from the first three experiments: **a** compression ratios; **b** PSNRs. –■– FT = (256,27,9); –▲– FT = (256,3,1); –○– FT = (256,60,20)

Fig. 7a,b. Data from the second set of experiments: **a** compression ratios; **b** PSNRs. –■– FT = (256,27,9); –▲– FT = (256,3,1); –○– FT = (256,60,20)

Fig. 8a,b. The salesman and the Chinese girl video sequences: **a** average compression ratios and average communication compression ratios; –■– ACR of Chinese girl; –▲– ACCR of Chinese girl; –○– ACR of salesman; –×– ACCR of salesman; **b** peak signal-to-noise ratios. –■– Chinese girl; –▲– Salesman

Fig. 9. The peak signal-to-noise ratios compared to the compression ratios of the salesman and the Chinese girl video sequences

Fig. 10. The peak signal-to-noise ratios compared to the compression ratios of the table tennis and the flower garden video sequences

contains fast motion, zoom out, and two scene changes, and the flower garden sequence contains fast panning with which the PVC cannot deal well.

In this experiment, the MPEG coder[1] uses two B-frames between each P-frame pair and uses a logarithmic block-matching algorithm in which the sum of the magnitude differences are used as the cost function. 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

[1] The MPEG coders are CMPEG V1.0 for MPEG encoder and DMPEG V1.1 for MPEG decoder, copyright by Stefan Eckart, June 1993, and was posted on newsgroup: alt.binaries.pictures.utilities. E-mail: stefan@lis.e-technik.tu-muenchen.de

11

12

Fig. 11. Comparisons of the PVC, the MPEG coder, and Video for Windows in sub-sequences of: **a** the salesman sequence; **b** the Chinese girl sequence

Fig. 12. Overview of the Popular Phone: **a** the left window is the remote video, which is compressed by the PVC; **b** the right window is the local video, which is not compressed

that of the I frames. The coding speed is 0.45 frames/s for MPEG encoding and 6.6 frames/s for decoding. It is true that MPEG coder is designed for compressing high-resolution, high-quality video data so that it may not perform better than the PVC under the 128 × 128 resolution. Further, the coding standard of video phone/conferencing systems is the H.261 instead of the MPEG standard because the bidirectional prediction of the MPEG standard may introduce a longer frame delay than the H.261 standard will do. However, in a simulation system, we can ignore the frame delays so that the performance of MPEG coding is close to that of H.261. That is why we compare the PVC with the MPEG coder, the general video coding standard, instead of the H.261 standard.

The final experiment compares the PVC with the Video for Windows V1.0 package. The coding speed of the Video for Windows for 160 × 120 DIB frames is 2.2 and 14 frames/s for encoding and decoding respectively. In this experiment, no audio data and key frames are included in the output file or the AVI file of Video for Windows, and the Microsoft Video 1 coding algorithm is used. The experimental results are shown in Figs. 9 and 10 and Table 1.

The original salesman video sequence and the sequences encoded by the PVC, the MPEG coder, and Video for Windows are shown in Fig. 11a, and the sequences of the Chinese girl are shown in Fig. 11b. The ACCRs for the PVC, the MPEG coder, and Video for Windows are 34 and 20 respectively in a and b.

5 PVC for a video phone system: the popular phone

The PVC is suitable for symmetric, real-time telephony because both the encoder and the decoder are fast. The PVC is also suitable for asymmetric broadcast use because its decoding speed is very fast. In this section, we apply the PVC to implement a video phone system, the Popular Phone, on personal computers over a 16 Mb/s token ring network. For such a system, video compression is definitely required since video data will consume a large amount of bandwidth. By using the PVC, the amount of the transmitted video data generated by the Popular Phone can be reduced by 10 to 30 times, which makes such a system feasible.

5.1 Control flow and protocols of the popular phone

Before describing the Popular Phone, we first show its monitor display as illustrated in Fig. 12 (see color plates). The right window contains the local video and the left window contains the remote video.

When starting up the Popular Phone, the system initializes the network interfaces and sets up the connection first. After the initial process, the Popular Phone always detects whether there are remote data waiting to be decoded before encoding the local data. If there are remote video data in the receiving buffer, the PVC decodes the video and the decoded video frame is displayed on the monitor. When the remote video frame is decoded, the error control is done (this procedure will be described in Sect. 5.2).

When all the remote data in the receiving buffer are processed, the Popular Phone enables the PVC encoder to encode the local video frame, which will be sent to the remote computer. At this point, the flow control is performed (its details will be explained in Sect. 5.3).

In the Popular Phone, by using the PVC, the traffic load (including video and audio data) on the token ring network is reduced from approximately 25% to approximately 3% and the frame rate is increased from approximately 6 frame/s to approximately 10 frame/s for two users on the network. Judging from the traffic load after compression, the Popular Phone system should be able to support more than thirty users concurrently without serious network delay.

The packet format used in the Popular Phone is shown in Fig. 13a, where each packet size is 512 bytes long with a type field indicating whether the packet contains video or

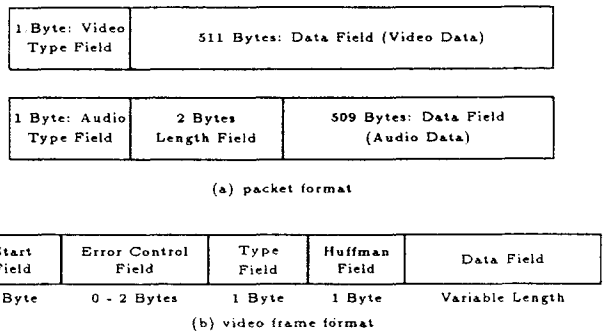


Fig. 13. Formats of: **a** the packet; **b** the video frame

audio data. For video packets, the remaining 511 bytes contain video data. Note that the encoded data of a video frame may occupy several video packets or only a portion of a packet, depending on the compression ratio. For example, an F-Frame may require several video packets to carry its data while a C-Frame may only need a portion of a packet. For audio packets, a two-byte field indicates the actual audio data size contained in the current packet, and the audio data are followed.

Each video frame is an independent entity and its format is shown in Fig. 13b. First, a one-byte start field is used to indicate the start of a new frame, and is also used for error detection. The second field is the error control field, which will be explained later. The third field indicates the type of the current frame, and the type value is used by the decoder of the receiver to adjust its quantizer. The fourth field is the Huffman field to indicate whether the decoder needs to rebuild its Huffman code table. The final field is the data field, containing all the encoded video data. The overhead of packet and video formats is less than 1% in most cases and it is not counted in the experimental results.

Lastly, we discuss the network protocols employed by the Popular Phone. The Popular Phone is built upon UDP/IP instead of TCP/IP. The reason why UDP is chosen is to reduce the protocol processing overhead since it is a connectionless transport protocol. However, UDP is not able to detect and recover from packet lost due to transmission error or buffer overflow; hence, it is necessary to include error control mechanism in the upper layer which will be described in the following subsection.

5.2 Error control of the Popular Phone

The error control of the Popular Phone is performed by the PVC and the packet-lost situations can be detected in the following ways:

1. Detecting the error within the error frame. Because a variable-length code is used in the PVC, a packet-lost situation is very likely to mismatch the decoded run length with the original run length sent. Such errors can be detected when matching the start field of the next frame.

2. Detecting the error when rebuilding the Huffman tree. If the errors are not detected in the first case, they will mismatch the accumulated frequencies of symbols of the decoding with those on the encoding. Most of such errors will make the reconstructed Huffman code trees of the encoder and the decoder inconsistent. Most of such inconsistency will lead to a mismatch of run length between the data sent and data received, which can be detected in the first case when coding further video frames.

After the errors have been detected, the receiver discards each video frame with error and sends an *UNACK* signal in the error control field of the sending packets by piggybacking. The receiver will discard all the following frames until a Refresh Frame is received. When the sender gets the *UNACK* signal in the received packets, it will reset its Huffman code table and frame type controller and send a Refresh Frame to the remote decoder.

In order to recover from the possible loss of the *UNACK* signal, the receiver sends the *UNACK* signals continuously in all succeeding video frames. The sender recognizes the first *UNACK* signal and discards the *UNACK* signals in succession for ten frames. However, if the Refresh Frame is lost, the *UNACK* will continue to appear after ten frames, so that the loss of Refresh Frame can be detected by the sender, and a new Refresh Frame is sent.

Although most of the errors can be detected by these methods, it is still possible that an error may go undetected by the receiver. To resolve this problem, the PVC will periodically force both the encoder of the sender and the decoder of the receiver to reset their Huffman code tables by sending Refresh Frames. By so doing, all propagated errors will be cleared.

5.3 Flow and rate control of the Popular Phone

In most cases, flow control is not needed in the Popular Phone. Nevertheless, if one computer of a communicating party is much faster than another, then the flow must be controlled to avoid the faster computer saturating the slower one.

In the Popular Phone, the flow can be controlled within local station without exchanging control packets. First of all, the Popular Phone evaluates the frame rates of the sending and receiving video frames. If the sending frame rate is much higher than the receiving frame rate, which means the sending station is faster than the receiving station, then the PVC will be suppressed in order to lighten the decoding load on the receiving station. Otherwise, the PVC will perform at its full speed.

The rate control of the Popular Phone monitores the output bit rate and adjusts the vector $\boldsymbol{FT}$: (GL, FL, ML). When the bit rate is too high, the Popular Phone increases the values of FL and ML so that the compression ratio will be increased. Otherwise, if the bit rate is too low, the Popular Phone decreases them.

6 Conclusions, discussion, and future work

This paper presents a software-based moving picture coding system: the PVC. This coder simplifies the MC/DCT-based

video coders to meet the real-time constraint of video applications. The PVC also modifies the quantizer and the Huffman coder to achieve a high compression ratio with good picture quality. As shown by experimental results, when the input data is a low-resolution and slow-motion video, the video quality of the PVC is as good as that of the MPEG video coder and there are no block, blurring, and ripple effects induced by the PVC. The coding speed of the PVC is much faster than that of the MPEG coder and the video quality of the PVC is good enough for commercial video applications, such as video phone, video conference, and multimedia e-mail systems. By using the PVC as the video coder, both the cost and the complexity of developing, maintaining, and installing video applications can be greatly reduced.

In our laboratory, the PVC has been successfully embedded into the Popular Phone system. By using the PVC, the traffic load on the token ring can be greatly reduced and the frame rate of the video is increased. In the Popular Phone system, the error control and flow control are also well developed. Applying the PVC to other systems, such as video conferencing and multimedia mail systems, is currently in process.

In the PVC, the quantized data are encoded by the run-length encoder in the scan line sequence. The data sequence can be rearranged in another form before being encoded as run-length tuples, such as the zigzag scan or the linear quadtree representation. The quadtree representation has been tested in the PVC, but the improvement is minor. A better way to rearrange the quantized signal without spending too much time is still being sought.

The DPCM technique used in the PVC removes the temporal redundancies. The spatial redundancies can be removed by applying DPCM or other techniques. However, based on the observation in our experiments, the use of DPCM for removing spatial redundancies will destroy the zero runs of quantized data. Searching for a better way to efficiently remove the spatial redundancies is one of our future works.

When the computation power of the PC is increased in the future, the performance of the PVC can be improved in several ways. First, the windowed Huffman algorithm [8] can be completely applied to encode the data more compactly. Second, a simple and fast motion estimation algorithm can be used to remove more of the temporal redundancies. Third, fast transform algorithms can be used to remove the spatial redundancies. Finally, the frame rate can be increased and the frame size can be enlarged.

Acknowledgments. This work is supported by the National Science Council, Taiwan, R.O.C., under grants no. NSC82-0408-E-002-232 and no. NSC82-0408-E-002-234.

References

1. Liou M (1991) Overview of the px64 kbit/s video coding standard. Commun ACM 34(4):59–63
2. CCITT Study Group XV (1989) Description of reference model 8 (RM8). Working Party XV/4, Specialists Group on Coding for Visual Telephony, Doc. 525
3. Le Gall D (1991) MPEG – A video compression standard for multimedia applications. Commun ACM 34(4):46–58
4. Rao KR, Yip P (1990) Discrete cosine transform. New York: Academic Press
5. Elias P (1955) Predictive coding. IRE Trans Information Theory 16–33
6. Wang Q, Clarke RJ (1992) Motion estimation and compensation for image sequence coding. Signal processing. Image Commun 4:161–174
7. Huang HC, Wu JL (1993) Real-time software-based moving picture coding (SBMPC) system. Proc Data Compression Conference, Snowbird, Utah, p 480
8. Huang HC, Wu JL (1993) Windowed Huffman coding algorithm with size adaptation. IEE Proc I, 140:109–113
9. Faller N (1973) An adaptive system for data compression. Record of the 7th Asilomar Conference on Circuit, System, and Computers, pp 593–597
10. Vitter JS (1986) Dynamic Huffman coding. ACM Trans Math Software 15:158–167
11. Chanbari M (1991) Arithmetic coding with limited past history. Electrics Lett 20:1157–1159
12. Knuth DE (1985) Dynamic Huffman coding. J Algorithm 6:163–180
13. Huang HC, Wu JL (1991) Design and analysis of residual Huffman algorithm. Proc of the National Computer Symposium. Taipei, Taiwan, R.O.C., pp 200–205
14. Huffman DA (1952) A method for the construction of minimum-redundancy codes. Proc. IRE 40:1098–1101
15. Larmore, LL (1987) Height-restricted optimal binary trees. SIAM J Comput 16:1115–1123
16. Larmore LL, Hirschberg, DS (1990) A fast algorithm for optimal length-limited Huffman codes. ACM 37:464–473

HO-CHAO HUANG received the degree of B.S. from the Department of Computer Science and Information Engineering from the National Taiwan University in 1990. He is now a Ph.D. candidate in the Department of Computer Science and Information Engineering at the National Taiwan University, Taipei, Taiwan. His research interests include variable length codes, software-based video data compression, model-based image coding, computer networks, computer graphics, graphics user interfaces and multimedia systems.

JAU-HSIUNG HUANG received the degree of B.S. in electrical engineering from the National Taiwan University in 1981, and the degrees of M.S. and Ph.D. in computer science from the University of California, Los Angeles, in 1985 and 1988, respectively. He is a Professor of the Department of Computer Science and Information Engineering at the National Taiwan University, Taipei, Taiwan. He joined the faculty at National Taiwan University in 1988. His research interests include design and performance evaluation of high speed networks, multimedia systems, and parallel and distributed systems.

JA-LING WU received the degree of B.S. in electronics engineering from the Tamkang University, Tamshoei, Taiwan, in 1979, and the degrees of M.S. and Ph.D. in electrical engineering from Tatung Institute of Technology in 1981 and 1986, respectively. From 1986 to 1987 he was an Associate Professor of the Electrical Engineering Department at Tatung Institute of Technology, Taipei, Taiwan. Since 1987 he has been with the Department of Computer Science and Information Engineering, National Taiwan University, where he is presently a Professor. Dr. Wu was the recipient of the 1989 Outstanding Youth Medal of the Republic of China and the Outstanding Research Award sponsored by the National Science Council, from 1987 to 1992. Dr. Wu has published more than 100 technique and conference papers. His research interests include neural networks, VLSI signal processing, parallel processing, image coding, algorithm design for DSP, data compression, and multimedia systems.