Three-Dimensional PAC Video Codec for Wireless Data Transmission

Kuei-Ann Wen, An-Yi Chen, Yin-Jin Lan, Jun-Lin Lin, and Chia-Huang Lin

***Abstract*—A newly derived algorithm called three-dimensional polynomial approximation coding (3DPAC) is being utilized in the applications of wireless video transmission. After combining the spatial and temporal domain, the video data is organized in a cube form, which has the features of processing parallelism and great compression ratio. Such a very low bit-rate video codec is able to support the wireless data transmission with a bit rate lower than 9.6 kbits/s. The system architecture will be presented and an experimental demonstration is also shown. For a video sequence of 176×144 frame size, PSNR up to 35 dB maybe obtained under 6.5 kbits/s transmission rate.**

***Index Terms*—Combined source coding and channel modulation, entropy coding, multimedia communication, variable bit rate coding, videocoding.**

I. Introduction

NOWADAYS, multimedia communication has become one of those most actively developed topics in both academic and industry. Under the fast development of mobile communication, home RF, and wireless LAN, most of the applications of multimedia and personal communication are going wireless. Limited spectral provided by various wireless systems can be from 11 Mbits/s down to 9.6 kbits/s. High compression ratio and low architectural complexity will be the key factors for video to be transmitted wirelessly. Being extended from the 2-D polynomial approximation coding in Lu's [1] and other related works, the 3-D polynomial approximation coding video codec (3DPAC VCodec), which is a DPCM system, has been designed and developed. For featuring a very high compression ratio and a very low structure complexity, the bit rate of this 3DPAC VCodec can be 9.6 kbits/s or even lower with the satisfactory subjective quality maintained, this system is highly feasible and applicable to wireless transmission.

The 3DPAC VCodec is illustrated in Fig. 1. The major blocks include "*coefficient generation*," "*cube prediction*," "*thresholding*," and "*residual coding*." The source (video data) first enter the block-coefficient generation to go through the polynomial approximation coding to generate the coefficients (its reverse process in the decoder part is called "*surface generation*"), which approximately represent the information of the data. In order to eliminate the redundancy owing to the temporal and spatial similarity and continuity of the video data, a so-called block "cube prediction" is designed (its reverse process in the decoder part is called "*cube compensation*"). Since the known datum is employed as the arguments in the processing of cube prediction, a "*cube buffer*" is required to store the previous coefficients. As depicted in Fig. 1(a), the symbol $\mathbf{A}$ stands for the original coefficients, $\mathbf{P}$ for the predicted ones, $\mathbf{D}$ is the difference between the original coefficients and the predicted coefficients, and $\mathbf{B}$ is the set of the coefficients to be preserved in the prediction buffer. In order to achieve even greater performance, there are two extra blocks, "thresholding" and "residual coding," designed before "*run length cube coding*" (RLCC). As to the "residual coding," it is a newly derived smoothing algorithm to deal with the blocky effect due to the processes of "quantization," "thresholding," and "inverse quantization." More detailed descriptions of this 3DPAC VCodec will be provided in the later sections.

Beside, without transform-based computation, a transpose buffer is not required, and thus hardware complexity can be lowered.

II. 3DPAC VCodec Algorithm

The procedures for both encoder and decoder can be depicted below.

Encoding process:

1) generate coefficients from the gray-level input of pixels;
2) derive the predicted coefficients from the previously known cubes;
3) subtract the exact reference coefficients from the predicted ones and then send the difference to quantization;
4) perform "thresholding" to obtain higher compression ratio;
5) Recover the quantized datum and send them to residual coding to increase PSNR and for the better subjective quality of video;
6) send the data out of thresholding to tje RLCC block to reduce the redundancy of datum;
7) send the final data to entropy coding and then to channel coding.

Decoding process:

1) get data from channel decoding and proceed entropy decoding;
2) execute inverse RLCC to get the quantized datum;
3) push the quantized datum through inverse quantization;
4) derive the predicted coefficients from the previously known cubes;

Manuscript received September 10, 1998; revised February 7, 2000. This paper was recommended by Guest Editor K. N. Ngan.

K.-A. Wen and C.-H. Lin are with the Institute of Electronics Engineering, National Chiao-Tung University, Hsin-Chu, Taiwan.

A.-Y. Chen and J.-L. Lin are with Philips Research East Asia—Taipei, Taipei, Taiwan.

Y.-J. Lan is with Hsinchu Science-Based Industrial Park, Taiwan, R.O.C.

Publisher Item Identifier S 1051-8215(00)08586-4.

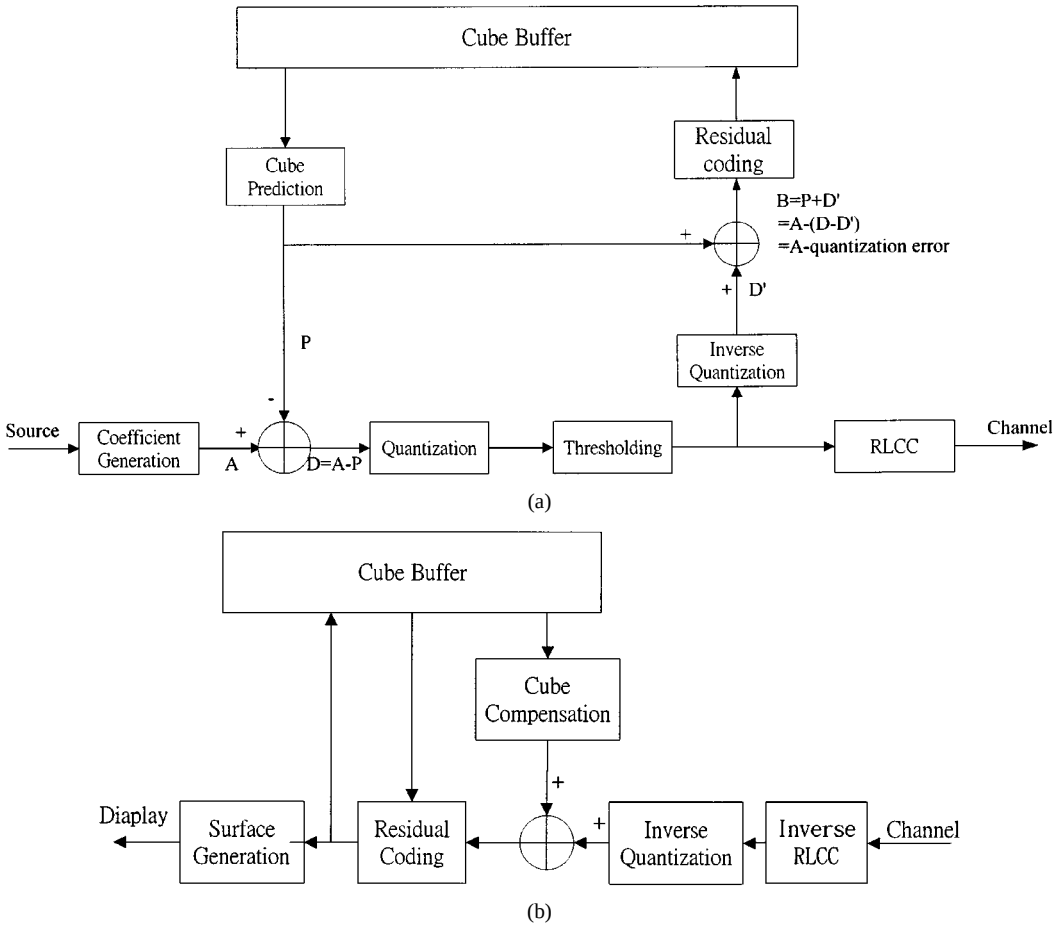


Fig. 1. 3DPAC VCodec. (a) Encoder. (b) Decoder

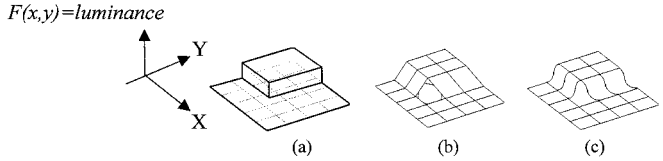


Fig. 2. Example of edges and corners. (a) Original. (b) First-order approximation. (c) Second-order approximation

5) add the predicted coefficients with the results of inverse quantization and then get the approximated coefficient that is close to the actual coefficients;
6) perform residual coding on the coefficients and then store them in the prediction buffer;
7) compute and send out the decoded pixels with proper synchronization to the display equipment.

A. Coefficient Generation

In view of the polynomial approximation techniques, a 3-D chart of x–y-luminance is used to demonstrate a 2-D image. In Fig. 2(a), which represents the original luminance of a certain frame, there are sharp edges and corners; while after being approximated with first-order and second-order polynomial equations, i.e., $f(x, y) = ax + by + c$ and $f(x, y) = ax^2 + by^2 + cxy + dx + ey + f$, as illustrated, it can be seen that with higher order approximation, the mathematical models will be more precise at edge and corner while less difference at smooth surface.

For video sequence, both spatial and time domains are mapped into the three coordinate axes, as shown in Figs. 3 and 4. The X-axis points go from left to the right direction and the Y-axis points are from upper to the lower direction to construct the 2-D spatial domain, and the Z axis points to the time direction, just like the scan order of the video sequence of frames. In our model, though the center of the cube is located at the origin (0, 0, 0) and each pixel has the unit distance from neighboring pixels, somehow they are not with the normal index system to describe their position. For example, in $4\times4\times4$ block size, the first pixel of the first frame has a coordinate of $(-1.5, -1.5, -1.5)$, and the last pixel of the last frame that of (1.5, 1.5, 1.5). That is, $X = \{-1.5, -0.5, 0.5, 1.5\} = Y = Z$. The reason for such a definition is to take the convenience and simplicity when it comes to computation, that is, we just perform shift-and-add instead of multiplication of the value of X, Y, Z coordinate.

If the gray level of the pixel (X, Y, Z) of a video sequence is to be represented perfectly with a 3-D polynomial function, it would require an infinite order function. In other words, the better the approximation is, the more terms are needed for the function. This model description will be set out with a second-order polynomial approximation:

$$\tilde{f}(X, Y, Z) = C_0 + C_1X + C_2Y + C_3Z + C_4X^2 + C_5Y^2 + C_6Z^2 + C_7XY + C_8YZ + C_9XZ$$
$$f_{X,Y,Z} = \tilde{f}(X, Y, Z) + \varepsilon \quad (1)$$

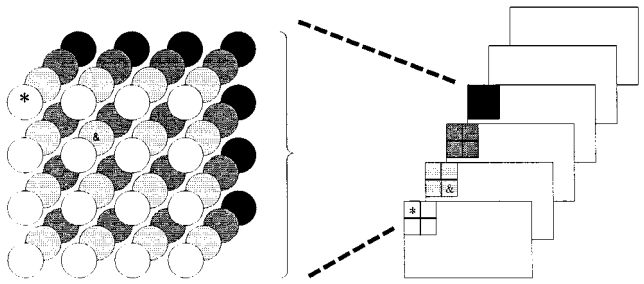


Fig. 3. Graphical illustration of the pixel locations in a cube.

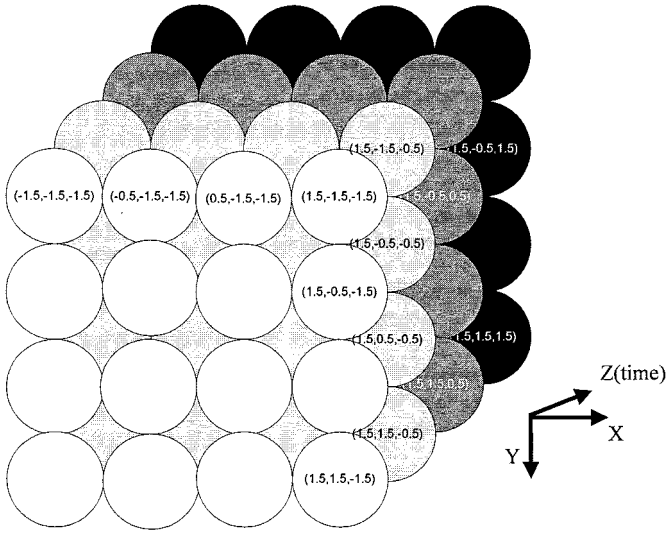


Fig. 4. Graphical illustration of the pixel coordinate in a cube.

where $f(X, Y, Z)$ is a constant and the mean square error (MSE) is

$$\text{MSE} = E\{\varepsilon^2\} = E\left\{\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right)^2\right\} \quad (2)$$

with equally transmitted probability

$$\text{MSE} = \frac{1}{N}\sum\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right)^2 \quad (3)$$

where N is the number of pixels in a block (64 in a $4 \times 4 \times 4$ block or 512 in an $8\times8\times8$ block). The summation is taken over all the indexes of X, Y, and Z as assigned previously, then the coefficients can be obtained in consideration of minimum MSE for C_0

$$\frac{\partial \text{MSE}}{\partial C_0} = \frac{-2}{N}\sum\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right) \quad (4)$$

because of the property of symmetry, the summation of those odd symmetric terms such as $\sum X$ and $\sum XY$ are zero. This simplifies (3) to be

$$\frac{\partial \text{MSE}}{\partial C_0} = \frac{-2}{N}\left(\sum f_{X,Y,Z} - C_0 N - C_4\sum X^2 - C_5\sum Y^2 - C_6\sum Z^2\right) \quad (5)$$

for the minimum MSE with respect to C_0, the left side of (4) should be zero, and then

$$\sum f_{X,Y,Z} - C_0 N - C_4\sum X^2 - C_5\sum Y^2 - C_6\sum Z^2 = 0. \quad (6)$$

Similarly, C1 will be processed and the result would be

$$\frac{\partial \text{MSE}}{\partial C_1} = \frac{-2}{N}\sum X\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right) \quad (7)$$

and

$$\sum X f_{X,Y,Z} - C_1\sum X^2 = 0. \quad (8)$$

Similarly, C_4 will be processed and the result would be

$$\frac{\partial \text{MSE}}{\partial C_4} = \frac{-2}{N}\sum X^2\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right) \quad (9)$$

and the odd symmetry terms such as $\sum X^3$ and $\sum X^2YZ$ are also zero and the equation is simplified to

$$\sum X^2 f_{X,Y,Z} - C_0\sum X^2 - C_4\sum X^4 - C_5\sum X^2Y^2 - C_6\sum X^2Z^2 = 0. \quad (10)$$

For C_7, there are more terms eliminated

$$\frac{\partial \text{MSE}}{\partial C_7} = \frac{-2}{N}\sum XY\left(f_{X,Y,Z} - \tilde{f}(X, Y, Z)\right) \quad (11)$$

and

$$\sum XY f_{X,Y,Z} - C_7\sum X^2Y^2 = 0. \quad (12)$$

To sum up, the group of equations with the coefficients mentioned above are collected here

$$\begin{cases} C_0 N + C_4\Sigma X^2 + C_5\Sigma Y^2 + C_6\Sigma Z^2 = \Sigma f_{X,Y,Z} \\ C_1\Sigma X^2 = \Sigma X f_{X,Y,Z} \\ C_2\Sigma Y^2 = \Sigma Y f_{X,Y,Z} \\ C_3\Sigma Z^2 = \Sigma Z f_{X,Y,Z} \\ C_0\Sigma X^2 + C_4\Sigma X^4 + C_5\Sigma X^2Y^2 + C_6\Sigma X^2Z^2 = \Sigma X^2 f_{X,Y,Z} \\ C_0\Sigma Y^2 + C_4\Sigma X^2Y^2 + C_5\Sigma X^4 + C_6\Sigma Y^2Z^2 = \Sigma Y^2 f_{X,Y,Z} \\ C_0\Sigma Z^2 + C_4\Sigma X^4Z^2 + C_5\Sigma Y^2Z^2 + C_6\Sigma X^4 = \Sigma Z^2 f_{X,Y,Z} \\ C_7\Sigma X^2Z^2 = \Sigma XY f_{X,Y,Z} \\ C_8\Sigma Y^2Z^2 = \Sigma YZ f_{X,Y,Z} \\ C_9\Sigma Z^2X^2 = \Sigma ZX f_{X,Y,Z} \end{cases} \quad (13)$$

With (13), the simultaneous equations can be derived in matrix form and the matrix element (cube) scan order will be from left to right (X-axis), then from up to down (Y-axis), and then from front to rear (Z-axis), as shown in (14), at the bottom of the next page. Therefore, with the example of $4 \times 4 \times 4$ cube and second-order polynomial approximation, the matrix named $F_{4*4*4,2}$ contains the approximated luminance information that would be generated with (13) and (14) (see (15), shown at the bottom of the page). The coefficient matrix $C_{4*4*4,2}$ is in the form of

$$C_{4*4*4,2} = [C_0 \quad C_1 \quad \cdots \quad C_9]^t. \quad (16)$$

Therefore, the mapping matrix $M_{4*4*4,2}$ is shown in (17), at the bottom of the page. The mapping relationship would be

$$F_{4*4*4,2} = M_{4*4*4,2}C_{4*4*4,2} \quad (18)$$

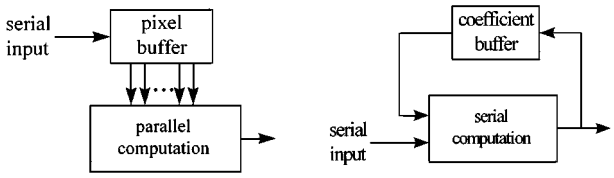


Fig. 5. Computation modes for 3DPAC VCodec. (a) Parallel mode. (b) Serial mode (PSO).

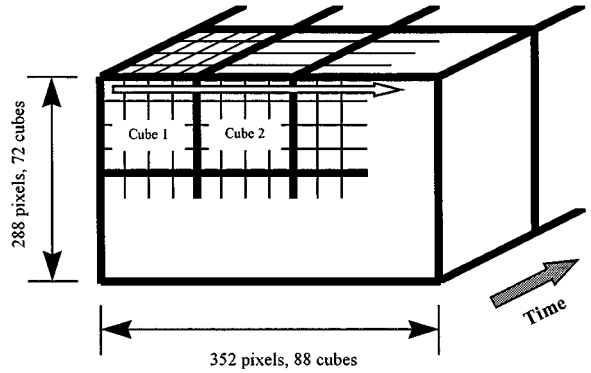


Fig. 6. Transition of the active cube.

Since $M_{4*4*4,\,2}$ is not a square matrix, it could be easier to deal with by the transpose and inverse calculation, and coefficient matrix $C_{4*4*4,\,2}$ will be generated after several stages, which are shown below

$$M^t_{4*4*4,\,2}F_{4*4*4,\,2} = M^t_{4*4*4,\,2}M_{4*4*4,\,2}C_{4*4*4,\,2} \tag{19}$$

$$C_{4*4*4,\,2} = (M^t_{4*4*4,\,2}M_{4*4*4,\,2})^{-1}M^t_{4*4*4,\,2}F_{4*4*4,\,2} \tag{20}$$

where

$$M^t_{4*4*4,\,2}F_{4*4*4,\,2} = \begin{bmatrix} \sum \tilde{f}(x,\,y,\,x) \\ \sum X\tilde{f}(x,\,y,\,x) \\ \vdots \\ \sum XZ\tilde{f}(x,\,y,\,x) \end{bmatrix} \tag{21}$$

$$M^t_{4*4*4,\,2}M_{4*4*4,\,2} = \begin{bmatrix} N & 0 & 0 & 0 & P & P & P & 0 & 0 & 0 \\ 0 & P & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P & 0 & 0 & 0 & 0 & 0 & 0 \\ P & 0 & 0 & 0 & Q & R & R & 0 & 0 & 0 \\ P & 0 & 0 & 0 & R & Q & R & 0 & 0 & 0 \\ P & 0 & 0 & 0 & R & R & Q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & R & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix} \begin{bmatrix} \sum 1 & 0 & 0 & 0 & \sum X^2 & \sum Y^2 & \sum Z^2 & 0 & 0 & 0 \\ 0 & \sum X^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sum Y^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sum Z^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ \sum X^2 & 0 & 0 & 0 & \sum X^4 & \sum X^2Y^2 & \sum X^2Z^2 & 0 & 0 & 0 \\ \sum Y^2 & 0 & 0 & 0 & \sum X^2Y^2 & \sum Y^4 & \sum Z^2Y^2 & 0 & 0 & 0 \\ \sum Z^2 & 0 & 0 & 0 & \sum X^2Z^2 & \sum Y^2Z^2 & \sum Z^4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sum X^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sum Y^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sum Z^2 \end{bmatrix} = \begin{bmatrix} \sum f_{X,Y,Z} \\ \sum Xf_{X,Y,Z} \\ \sum Yf_{X,Y,Z} \\ \sum Zf_{X,Y,Z} \\ \sum X^2f_{X,Y,Z} \\ \sum Y^2f_{X,Y,Z} \\ \sum Z^2f_{X,Y,Z} \\ \sum XYf_{X,Y,Z} \\ \sum YZf_{X,Y,Z} \\ \sum XZf_{X,Y,Z} \end{bmatrix} \tag{14}$$

$$F_{4*4*4,\,2} = [\,\tilde{f}(-1.5,\,-1.5,\,-1.5) \quad \tilde{f}(-0.5,\,-1.5,\,-1.5) \quad \cdots \quad \tilde{f}(1.5,\,1.5,\,1.5)\,]^t \tag{15}$$

$$M_{4*4*4,\,2} = \begin{array}{c} \begin{matrix} \mathbf{1} & \mathbf{X} & \mathbf{Y} & \mathbf{Z} & \mathbf{X^2} & \mathbf{Y^2} & \mathbf{Z^2} & \mathbf{XY} & \mathbf{YZ} & \mathbf{XZ} \end{matrix} \\ \begin{bmatrix} 1 & -1.5 & -1.5 & -1.5 & 2.25 & 2.25 & 2.25 & 2.25 & 2.25 & 2.25 \\ 1 & -0.5 & -1.5 & -1.5 & 0.25 & 2.25 & 2.25 & 0.75 & 2.25 & 0.75 \\ & & & & \vdots & & & & & \\ 1 & 1.5 & 1.5 & 1.5 & 2.25 & 2.25 & 2.25 & 2.25 & 2.25 & 2.25 \end{bmatrix} \end{array} \tag{17}$$

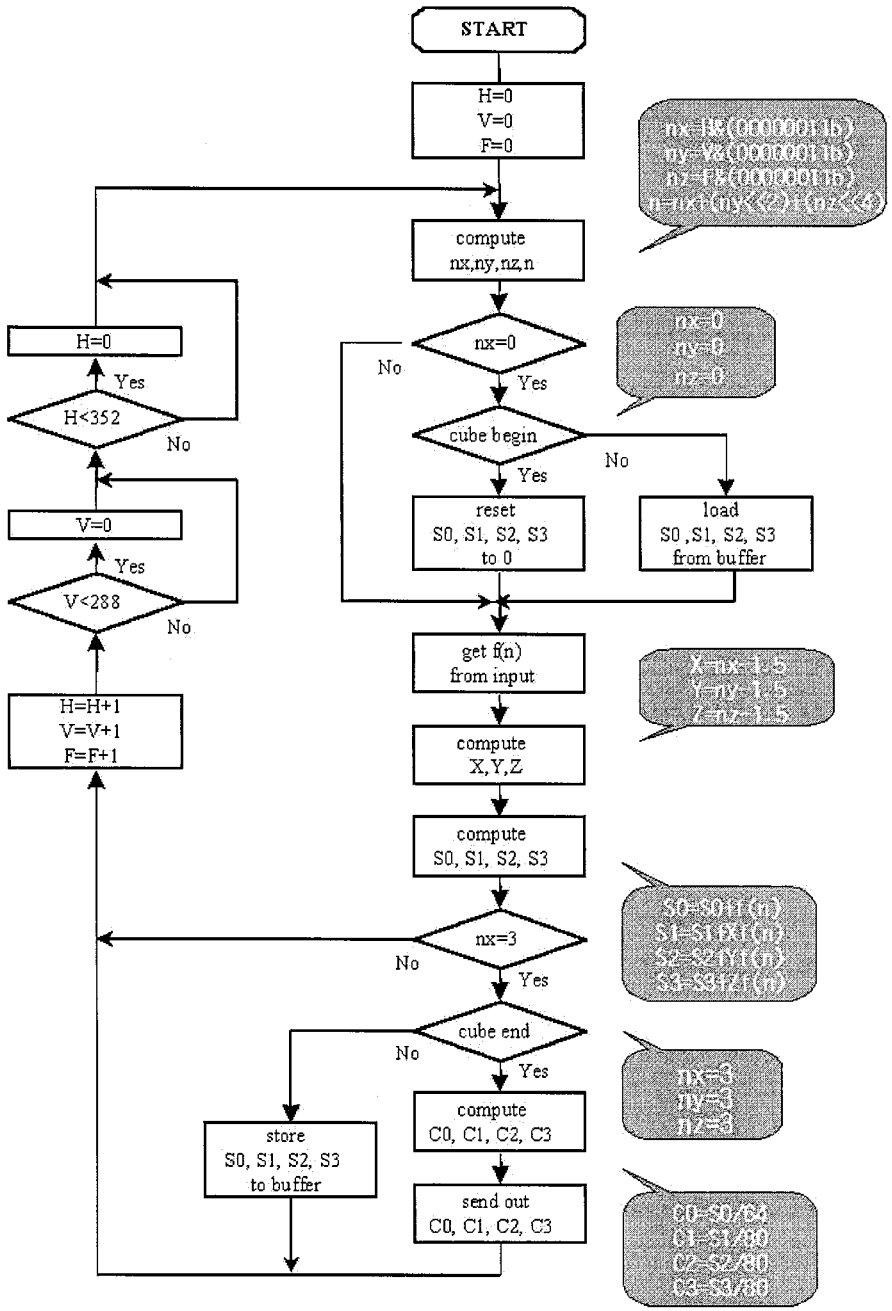


Fig. 7. Flowchart of the coefficient generation block

and

$$N = \sum 1, \quad P = \sum X^2 = \sum Y^2 = \sum Z^2$$
$$Q = \sum X^4 = \sum Y^4 = \sum Z^4$$
$$R = \sum X^2Y^2 = \sum Y^2Z^2 = \sum X^2Z^2.$$

After all the computations, those elements of $C_{4*4*4,\,2}$, that is, the coefficients of the approximated polynomial coding, will be quantized after a subtractive process and then be encoded by RLCC for video signal transmission. The decoder at the receiver would be another matrix operation processor to recover the F function to work with the video-sequence playback.

1) Computational Simplification: Computation simplification is highly and strictly demanded for wireless applications. To represent the gray level of every pixel in a $4 \times 4 \times 4$ cube in an efficient way, nth-order polynomial approximation has been treated and, experimentally, it has already been proved that the 1st order approximation may get the best tradeoff of error performance and simplicity [1, Appendix]; that is

$$\tilde{f}(X, Y, Z) = C_0 + C_1X + C_2Y + C_3Z. \tag{23}$$

To represent the gray levels of the 64 pixels in the cube, we just need to give the four polynomial coefficients C_0, C_1, C_2, and C_3, instead of the data of 64 pixels. Utilizing the cube size $4 \times 4 \times 4$ and first-order polynomial approximation, within this

cube-based block, the four coefficients will be generated with the matrix computation, as previously obtained in equation (19)

$$C = (M^t M)^{-1} M^{-t} F \tag{24}$$

where

C	coefficient matrix of four elements;
F	pixel matrix of 64 elements;
$(M^t M)^{-1} M^{-t}$	constant computation matrix.

With computation simplification, the specified index of M can be pre-computed and the matrix computation of C can be implemented with simple calculations as

$$C = \begin{bmatrix} \dfrac{\sum f_{X,Y,Z}}{64} \\ \dfrac{\sum X \times f_{X,Y,Z}}{80} \\ \dfrac{\sum Y \times f_{X,Y,Z}}{80} \\ \dfrac{\sum Z \times f_{X,Y,Z}}{80} \end{bmatrix} \tag{25}$$

where $X, Y, Z = (-1.5, -0.5, 0.5, 1.5)$ and $f_{X,Y,Z}$ is the pixel value in the corresponding position.

Further, the multiplication of 0.5 and division of 64 can be implemented by merely shifting the bit of coefficient, and the multiplication of 1.5 can be implemented by addition with multiplication of 0.5. Thus, in this system, only the division of 80 requires a more complicated mathematical process.

2) Buffer Cost Reduction: Unlike the transform-based computation, the previous summations do not make it necessary that all the pixels to have arrived to process. The traditional parallel mode of computation or transpose memory as demanded for the DCT-based system may not be required anymore. The PSO developed and applied in the experiment of this work release the need for a large buffer. The *buffer size reduction ratio* $\boldsymbol{B}$ is obtained as shown in (25a), at the bottom of the page.

With parallel computation, computation of coefficients will not be made until 64 pixels of a cube are fully collected, and it would cost the capacity of 64 pixels, or 512 bits if 8 bits for a pixel gray-level value for a single cube. However, in the PSO mode, computation starts as a pixel comes in and the uncompleted coefficients are then stored to the buffer for accumulation/summation. Hence, the computation of each cube will cost

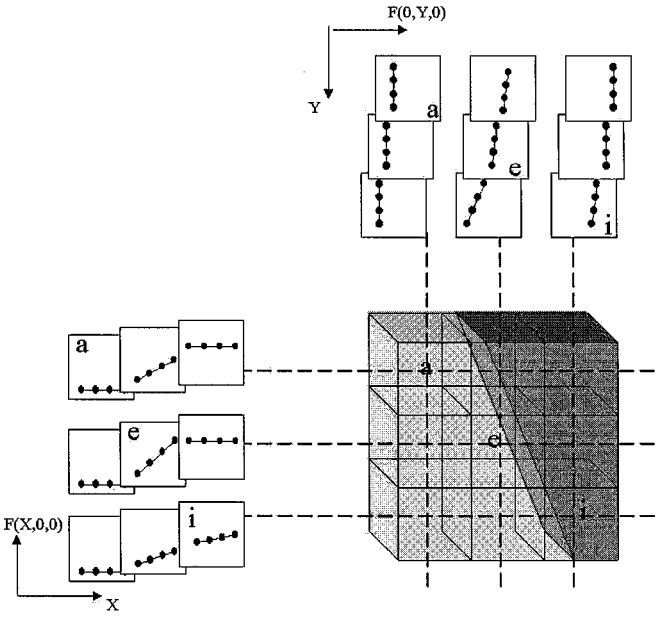


Fig. 8. Gray-level transition along three virtual axes.

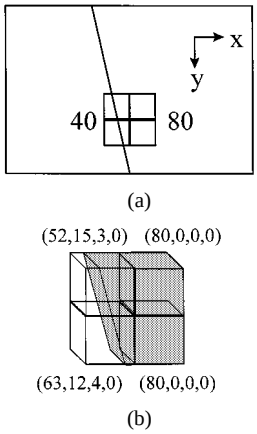


Fig. 9. Example of an edge within a frame. (a) Four cubes on the edge. (b) Coefficients of the four cubes.

the capacity of four coefficients. During the process, the maximum of S_i occurs when all the pixels are of gray level 255, that is, $\mathrm{Max}(S_i) = 255 \times 64 = 2^{14}$, each coefficient maximally needs 14 bits, and the buffer size required is

$$\text{Buffer size (bits)} = \frac{S}{W^2} \times N \times 14 = 56\frac{S}{W^2}. \tag{26}$$

$$B = Buffer\,size\,reduction\,ratio = \frac{parallel\,mode\,buffer\,size}{serial\,mode\,buffer\,size}$$
$$= \frac{frame\,size \times cube\,width \times bits\,required\,per\,pixel}{number\,of\,cubes\,with\,a\,frame \times number\,of\,coefficients\,per\,cube \times bits\,required\,per\,coefficient}$$

where

$$S = Frame\,size = Frame\,width \times Frame\,height$$
$$N = Number\,of\,Coefficients$$
$$W = Cube\,width. \tag{25a}$$

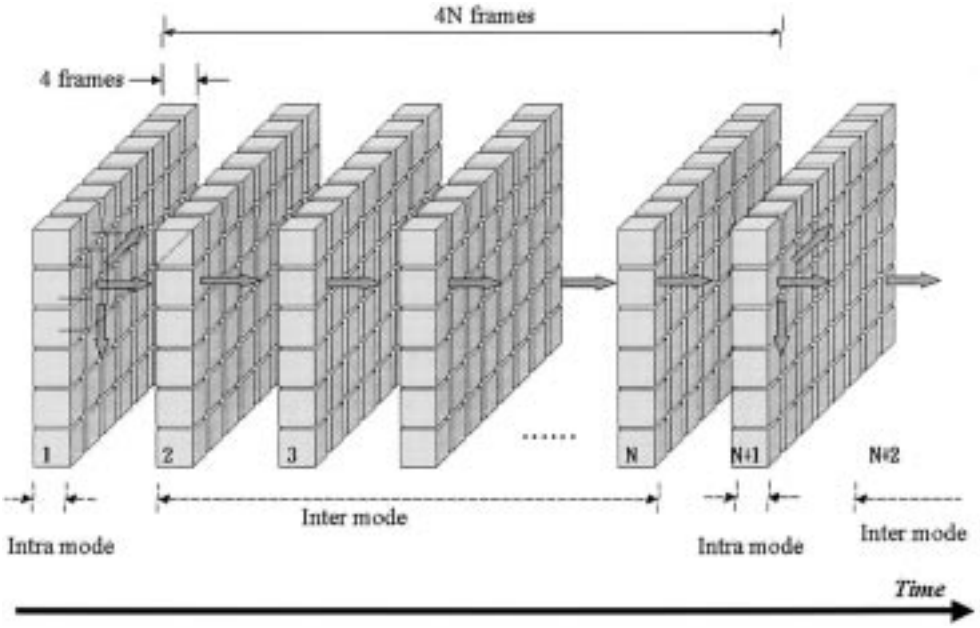


Fig. 10. The available prediction directions.

Therefore, for a 352×288 screen, the size of the coefficient buffer is about 354.8 kbits and that for a 176×144 screen is 88.7 kbits, and the buffer-size reduction ratio is

$$B = \frac{S \times W \times 8}{\frac{S}{W^2} \times N \times 14} = 0.57 \times \frac{W^3}{N} = 0.57 \times \frac{4^3}{4} = 9.14. \tag{27}$$

3) Flowchart of Coefficient Generation: However, the pixels come in the scan order, which is somewhat different from the order of the cubes. Fig. 6 illustrates the transition of the active cubes.

The flowchart of coefficient generation is shown in the Fig. 7. According to this flowchart, the following variables are computed in turn.

1) H, V, F: the position of the active pixel corresponding to the whole video sequence in each direction.
2) n_X, n_Y, n_Z: the position of the active pixel corresponding to the active cube in each direction.
3) n: the number of pixels that the active cube has received.
4) X, Y, Z: the coordinates of the active pixel in the active cube.
5) $S_i(n)$: the temporary recursive summation for each coefficient.
6) C_i: the 3DPAC coefficients which is generated after 64 pixels have arrived and the recursive summation is completed.

B. Cube Prediction

After the transformation of source data to coefficients in coefficient generation, many of the cubes are highly correlated with each other for the continuity owing to the existence of an edge crossing the adjacent cubes. The relationship between neighboring cubes has been studied and observed in order to derive algorithms and schemes to increase the compression ratio.

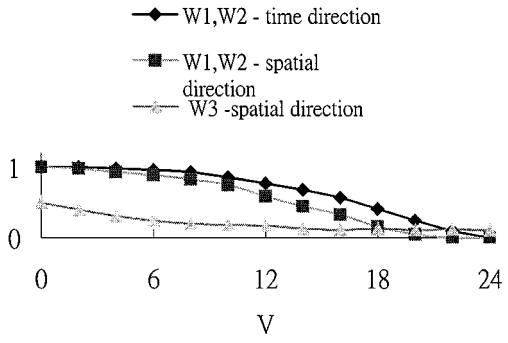


Fig. 11. Various W versus V.

1) Gray-Level Transition: To explain the algorithm of cube prediction, it will be started with the gray-level transition described in Fig. 8. For each of the $4\times4\times4$ cube, a 2-D coordinate is formed with horizontal axis indicate the four discrete pixels (X, 0, 0) in a $4\times4\times4$ cube and with vertical axis indicates the gray level, i.e.

$$f_{\text{transition},\,X} = C_0 + C_1X + C_2Y + C_3Z = C_0 + C_1X. \tag{28}$$

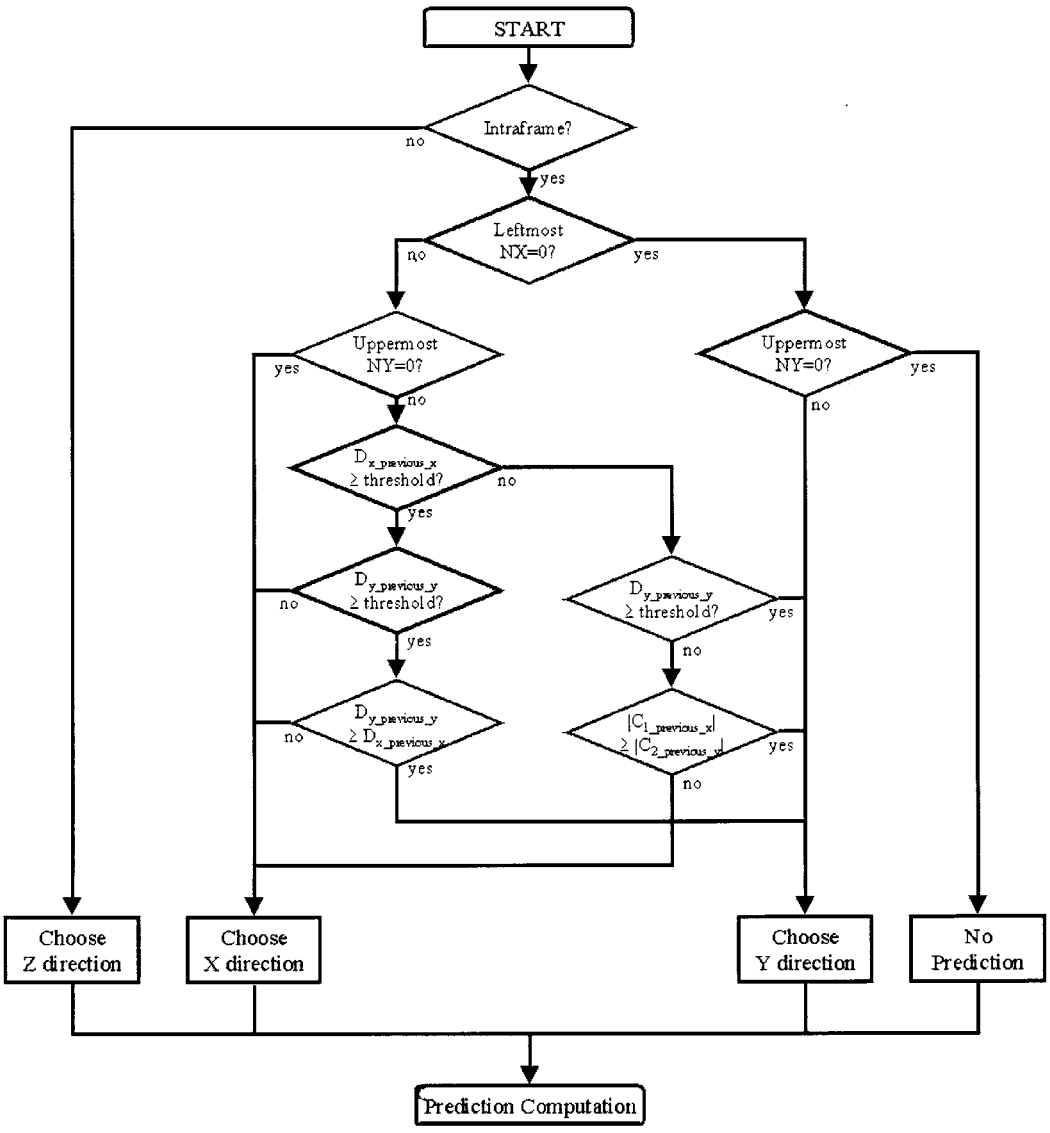
Similarly, we can get another set of 2-D coordinates with vertical denotes the four pixels (0, Y, 0) and with horizontal axis indicates the corresponding gray-value Y-axis at (0, -1.5, 0), (0, -0.5, 0), (0, 0.5, 0), (0, 1.5, 0) with

$$\begin{aligned} f_{\text{transition},\,Y} &= C_0 + C_1X + C_2Y + C_3Z \\ &= C_0 + C_2Y \end{aligned} \tag{29}$$

and at (0, 0, -1.5), (0, 0, -0.5), (0, 0, 0.5), (0, 0, 1.5) when the Z direction is taken into account

$$\begin{aligned} f_{\text{transition},\,Y} &= C_0 + C_1X + C_2Y + C_3Z \\ &= C_0 + C_3Z. \end{aligned} \tag{30}$$

The $f_{\text{transition}}$ values obtained by these three equations will be the basis to analyze gray-level transition between adjacent



(a)

Fig. 12. (a) Flowchart of the direction decision.

cubes according to what direction is considered. Additionally, the coefficient left, other than C_0, defined as the "dominant coefficient," is C_1 for the X, C_2 for the Y, and C_3 for the Z direction. It is because, as illustrated in Fig. 9(b), cubes with a higher gray level imply a larger value of C_1 and thus C_1 will dominate the gray-level change in the X direction.

For example, in Fig. 9, the edge is more vertical and less horizontal, and therefore the C_2's of the left two cubes are smaller and closer to zero because of less change. Abstractly, C_1's are much larger. The dominant coefficient scales the degree of change of a single cube in a specified direction, simply and clearly.

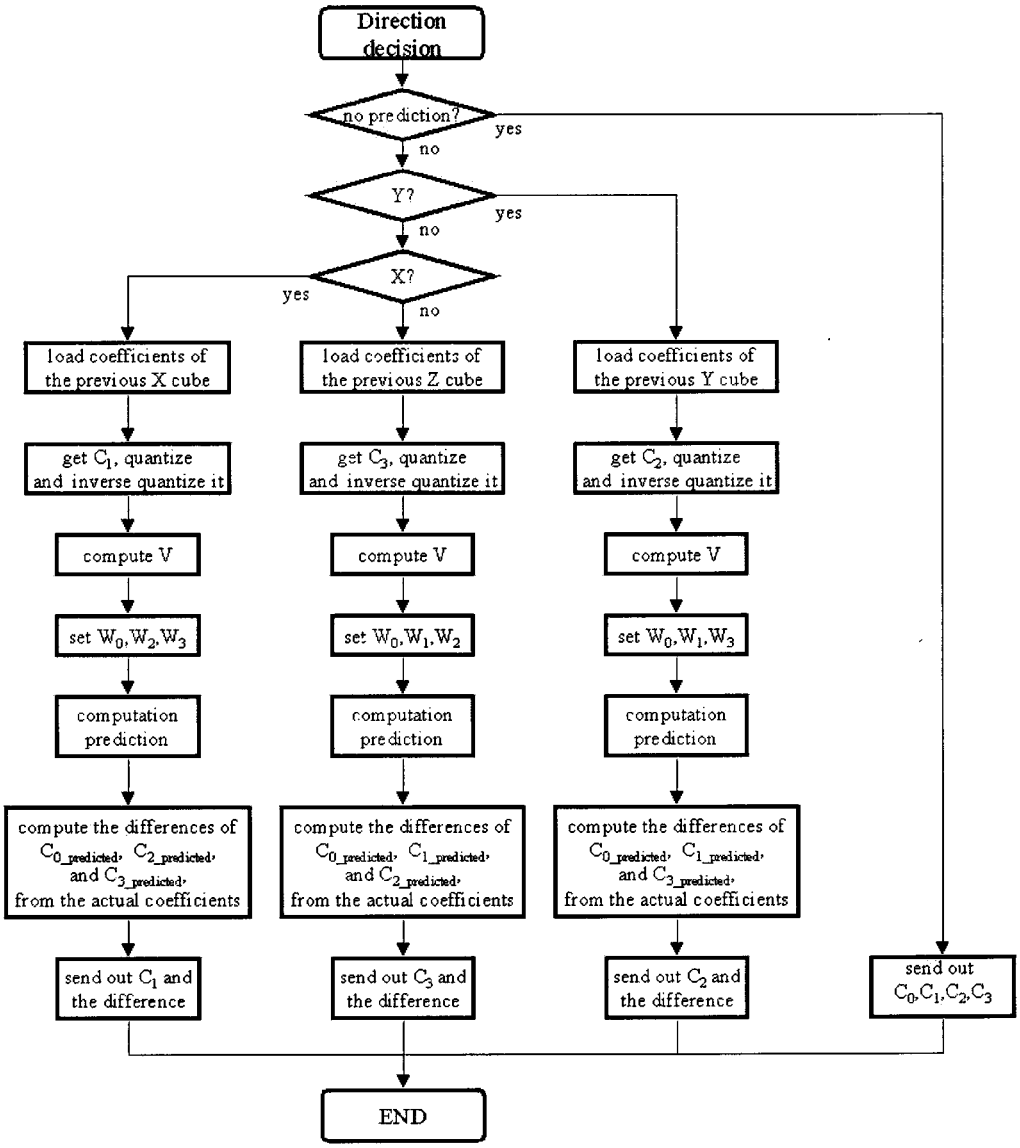
1) It is almost piecewisely linear, and the contrast of edge is visualized with the slope
2) At the virtual origin point in each cube, where X, Y, and Z are also zero, the recovered pixel is C_0, which also stands for the average value of 64 pixels in the cube.
3) The transition view values of Y direction seem to be more similar and more highly correlated. C_0 and C_2 are more similar. Such a feature is called "reflectivity," for the horizontally spatial continuity of the image is reflected at the tendency of lateral-view values at the relative direction.
4) Both in X and Y directions, the boundaries of neighboring cubes on the common face will be very close to each other. Such a feature is called "connectivity."

2) Cube-Prediction Algorithm: Fig. 10 provides the prediction model with intra and inter cubes being defined. Coefficients of intra-mode cubes can be predicted for intra-mode cubes with the following prediction process.

a) Direction Decision: Two parameters, D_x and D_y, are defined for a direction decision

$$D_X = \frac{|C_2|+1}{|C_1|+1} \tag{31}$$

$$D_y = \frac{|C_1|+1}{|C_2|+1}. \tag{32}$$



(b)

Fig. 12. *(Continued.)* (b) Flowchart of the cube prediction.

In the two equations above, the constant 1 is added to both divider and dividend in order to prevent the divider from zero. The larger D_x is, the more possibly the edge is extending in the X direction. Conversely, large D_y means the edge extends in the Y direction.

The direction decision is made by comparing the conditions of D_x and D_y in both the previous X and Y cubes as follows:

a) the first "wall" of cubes within a period, say N cubes or $4N$ frames, will be labeled as inter mode cubes, where no prediction is performed;
b) for the interframes, the prediction direction is forced to be in the Z direction;
c) for the intraframes, if one or both of $D_{x\_\text{previous}\_x}$ and $D_{y\_\text{previous}\_y}$ is larger than the lower limit (□2.5), choose the direction corresponding to the larger one;
d) for the intraframes, if both are smaller than the lower limit (□2.5), compare $|C_{1\_\text{previous}\_x}|$ and $|C_{2\_\text{previous}\_y}|$, and choose the direction corresponding to the smaller one.

b) Prediction: After the direction is decided, the prediction will be performed. The prediction equations are described in the following paragraphs.

The variance V is defined as the absolute sum of two dominant coefficients at the direction previously chosen

$$V = |C_{\text{dominant\_previous}}| + |C_{\text{dominant\_current}}|. \tag{33}$$

The larger V is, the less connectivity there would be between two neighboring cubes. Thus,

for the X direction:

$$\begin{aligned} V &= |C_{1\_\text{previous}}| + |C_{1\_\text{current}}| \\ C_{0\_\text{current\_predicted}} &= C_{0\_\text{previous}} + W_{10}(V) \\ &\quad \times (C_{1\_\text{previous}} + C_{1\_\text{current}}) \\ C_{2\_\text{current\_predicted}} &= W_{12}(V) \times C_{2\_\text{previous}} \\ C_{3\_\text{current\_predicted}} &= W_{13}(V) \times C_{3\_\text{previous}} \end{aligned} \tag{34}$$

for the Y direction

$$\begin{aligned} V &= |C_{2\_\text{previous}}| + |C_{2\_\text{current}}| \\ C_{0\_\text{current\_predicted}} &= C_{0\_\text{previous}} + W_{20}(V) \\ &\quad \times (C_{2\_\text{previous}} + C_{2\_\text{current}}) \\ C_{1\_\text{current\_predicted}} &= W_{21}(V) \times C_{1\_\text{previous}} \\ C_{3\_\text{current\_predicted}} &= W_{23}(V) \times C_{3\_\text{previous}} \end{aligned} \tag{35}$$

and for the Z direction

$$\begin{aligned} V &= |C_{3\_\text{previous}}| + |C_{3\_\text{current}}| \\ C_{0\_\text{current\_predicted}} &= C_{0\_\text{previous}} + W_{30}(V) \\ &\quad \times (C_{3\_\text{previous}} + C_{\_\text{current}}) \\ C_{1\_\text{current\_predicted}} &= W_{31}(V) \times C_{1\_\text{current}} \\ C_{2\_\text{current\_predicted}} &= W_{32}(V) \times C_{2\_\text{current}} \end{aligned} \tag{36}$$

where $W_{ij}(V)$ is a weighting function of V obtained with experimental calculation and statistic with many benchmark sequences, such as "Miss America" and "Miss Claire" [1]. Statistically, $W_{i0}(V)$ approximates to a constant 1.8, and others approximate to linear functions of V.

3) Flowcharts of Cube-Prediction Algorithm: Procedures for cube prediction are illustration by flowchart in Fig. 12.

C. Thresholding

Since coefficient prediction shrinks the data size and hugely centralizes the data at the value of zero and after quantization, coefficients of most cubes become small value. The major function of thresholding coding is to ignore such small values, as shown in Fig. 13. A cube with the quantization result of (0, 0, 0, 0) is called a zero cube here, while thresholding coding is to increase the number of zero cubes for the further channel coding.

After thresholding processing, the four quantized coefficients of a zero cube can be replaced with merely leading bits in variable-length cube coding. More zero cubes are produced and clustered, more redundant datum can be eliminated by VLCC, and the compression ratio can be even greatly increased.

A parameter Δ_T is defined as

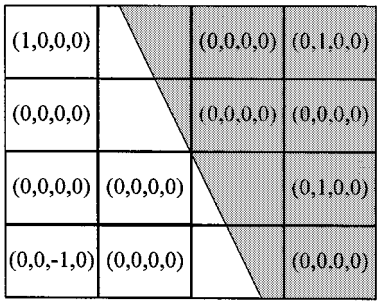
$$\Delta_T = \begin{cases} \sum_{i=0}^{2} (C_i - C_{i\_\text{predicted}}), & \text{for intraframes} \\ \sum_{i=0}^{3} (C_i - C_{i\_\text{predicted}}), & \text{for interframes.} \end{cases} \tag{37}$$

When Δ_T gets small enough, it can be replaced by zero. Therefore, a threshold for Δ_T has to be found and set with further calculation and statistics. Table I gives the simulation result, while Fig. 14 shows the performance of thresholding at different threshold values, and "missa x" means the Miss America sequence processed with quantization step size x. Thus, it is found that the reasonable threshold is set to one for further simulation and analysis.

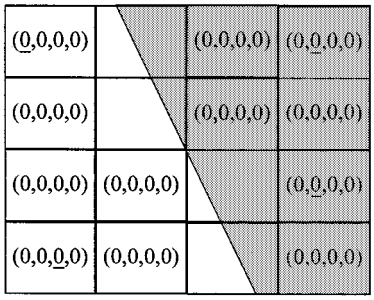
The flowchart of thresholding process is given in Fig. 15 with the example thresholding value of 1.

D. Residual Coding

Even if the thresholding process does lose some unimportant information, such an error, in fact, does not harm the subjective quality so much in the temporal domain because the human visual system is not as sensitive to thresholding noise in the temporal domain as the frame rate is reasonably high. However, in the spatial domain, the artifacts could corrupt the subjective quality at a certain degree. Hence, residual coding is derived and applied to smooth the blocky artifacts only in the spatial domain (see Fig. 16).



(a)

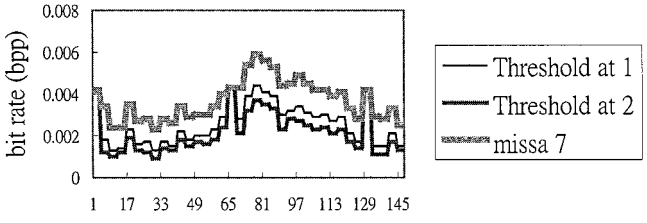


(b)

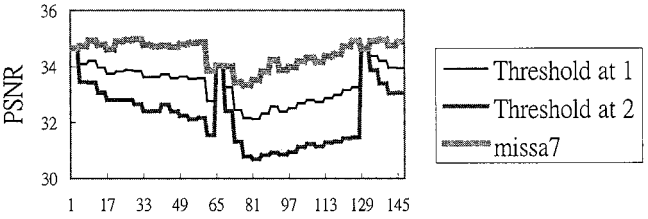
Fig. 13. Zero cubes for thresholding. (a) Original quantization result. (b) After thresholding is applied.

TABLE I
PSNR DROP BY THRESHOLDING

Threshold of S_T	1	2
PSNR drop	0.8 dB	2.7 dB



(a)



(b)

Fig. 14. The improvement of compression ratio by thresholding.

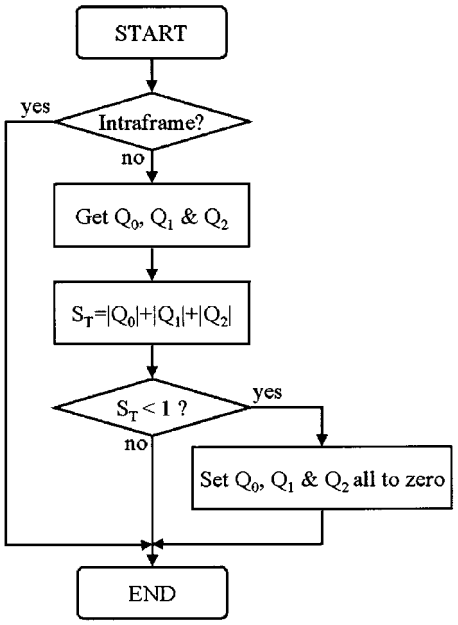


Fig. 15. Flowchart of thresholding.

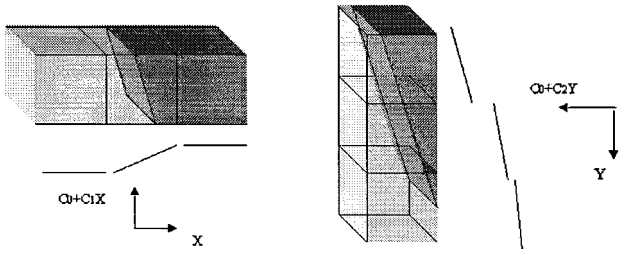


Fig. 16. Illustration of spatial continuity and gray-level transition.

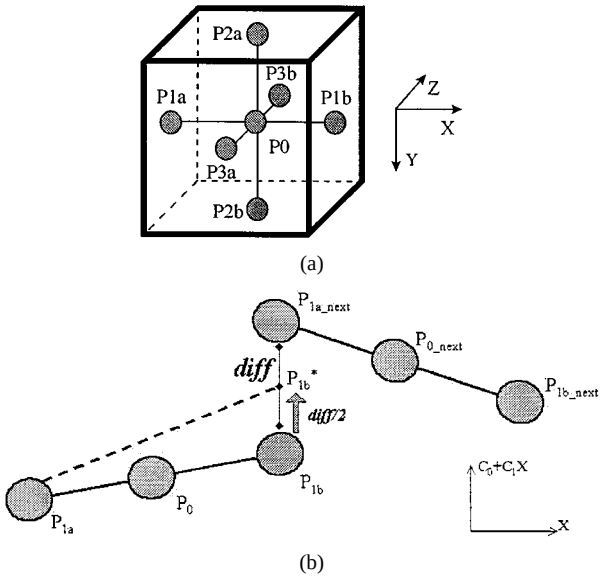


Fig. 17. Illustration of residual coding. (a) Virtual points (VPs). (b) Adjustment of VPs in X-axis.

Fig. 17 is an illustration of residual coding. At the two sides of the common face of two cubes, the gray-level transition graph may not be continuous or connected at all. That means that the image is not smooth at the adjacent border of these two cubes, and blocky artifacts exist. If the difference of the two ends of

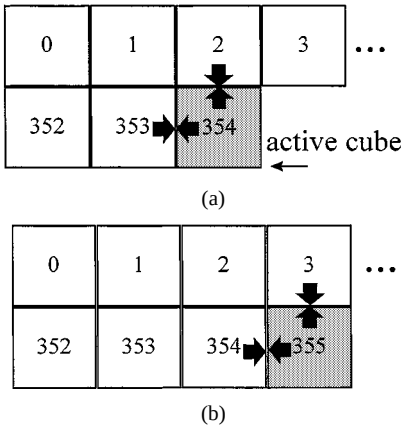


Fig. 18. Order of residual coding.

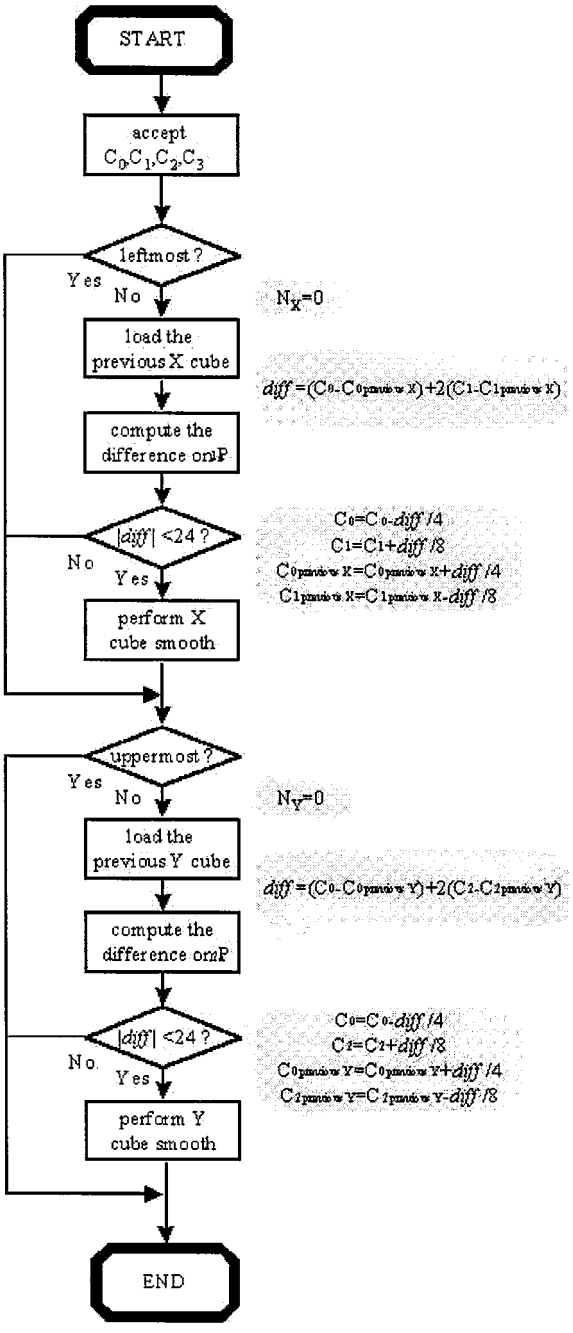


Fig. 19. Flowchart of residual coding.

each cube's gray-level transition can be approached and concatenated, the visual performance is improved.

The main idea of residual coding is that, in Fig. 17(a), those virtual points on the faces of a cube will have to be moved closer in gray-level transition with the neighboring cube in spatial domain, that is, X and Y directions. First the gray level of the two X virtual points of a cube is:

$$P_0 = C_0 \quad P_{1a} = C_0 - 2 \times C_1 \quad P_{1b} = C_0 + 2 \times C_1 \quad (38)$$

and the parameter *diff* is defined as

$$\begin{aligned} \text{diff} &= \mathbf{P}_{1a\_\text{next}} - \mathbf{P}_{1b} \\ &= (C_{0\_\text{next}} - 2 \times C_{1\_\text{next}}) - (C_0 + 2 \times C_1) \\ &= (C_{0\_\text{next}} - C_0) - 2 \times (C_{1\_\text{next}} + C_1). \end{aligned} \quad (39)$$

$P_{1a_{\text{next}}}$ is the VP at the position of P1a of the neighboring cube in the X direction. Therefore, in order to shorten the difference of the two edges, as shown in Fig. 17(b), the adjustment is applied to the two virtual points on the common face is $diff/2$, with the positive edge raised while the negative edge remained. Since C_i^* is the coefficients after adjustment, the computation in the *X*-axis is derived as follows:

$$\begin{aligned} P_{1b}^* &= P_{1a\_\text{next}} - \text{diff}/2 \\ &\Rightarrow P_{1b}^* = P_{1a\_\text{next}} - \text{diff} + \text{diff}/2 \\ &\Rightarrow P_{1b}^* = P_{1a\_\text{next}} - (P_{1a\_\text{next}} - P_{1b}) + \text{diff}/2 \\ &\Rightarrow P_{1b}^* = C_0 + 2C_1 + \text{diff}/2 \\ &\Rightarrow C_0^* + 2C_1^* = C_0 + 2C_1 + \text{diff}/2 \\ &\Rightarrow (C_0^* - C_0) + 2(C_1^* - C_1) = \text{diff}/2 \\ &\Rightarrow \delta_0 + 2 \times \delta_1 = \text{diff}/2, \qquad \text{for } \delta_i = C_i^* - C_i \end{aligned} \quad (40)$$

$$\begin{aligned} P_{1a}^* &= P_{1a} \\ &\Rightarrow C_0^* - 2C_1^* = C_0 - 2C_1 \\ &\Rightarrow \delta_0 - 2\delta_1 = 0, \qquad \text{for } \delta_i = C_i^* - C_i \end{aligned} \quad (41)$$

With the results of (40) and (41), it is obtained that $\square_0 = \text{diff}/4$ and $\square_1 = \text{diff}/8$. However, the case in Y direction is similar. Therefore, as both X and Y directions are considered, the adjustment should be

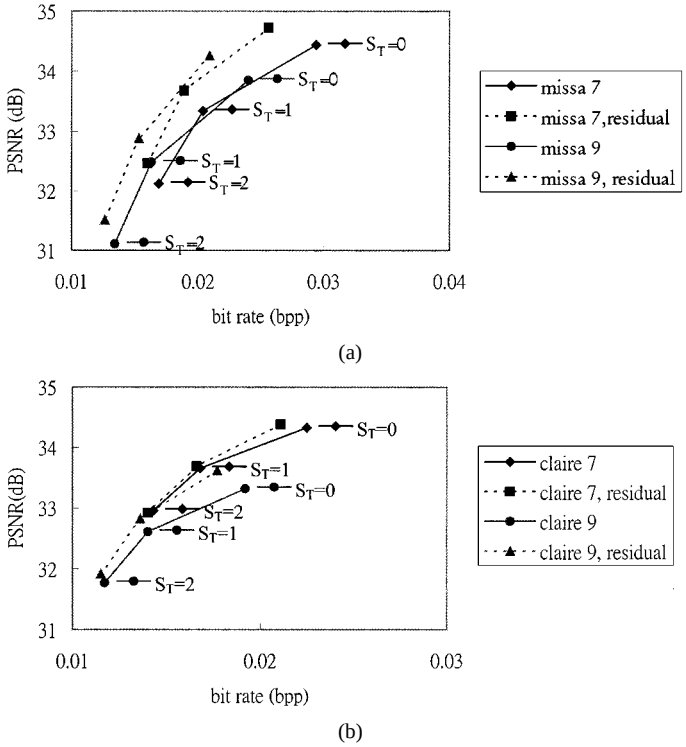
$$\begin{aligned} \delta_0 &= (\text{diff}_X/4 + \text{diff}_Y/4)/2 = \text{diff}_X/8 + \text{diff}_Y/8 \\ \delta_1 &= \text{diff}_X/8, \quad \delta_2 = \text{diff}_Y/8. \end{aligned} \quad (42)$$

If *diff* happens to be large enough, it either indicates that an error exists, or an edge is actually nearby. In order to avoid performing this procedure to where there has to be a great gray-value difference, the offset adjustments will be performed only when the absolute value of *diff* is equal to or less than a certain threshold L, named "residual threshold"

$$|\text{diff}| \le L \quad (43)$$

With the simulation results and analysis, the proper residual threshold is found to be 24.

Fig. 18 shows the order of residual coding. The previously utilized cubes are stored in the buffer, which is also used in cube prediction; therefore, no other buffer will be required in the system. When cube 354 comes, residual coding is performed on its upper (Y direction) and left faces (X direction). In Fig. 18(b), when cube 355 comes, the right faces of cube 354 are also smoothed. Therefore, residual coding of a cube works in different time. Fig. 19 shows the flowchart.



(a) (b)

Fig. 20. PSNR versus bit-rate curve. Dashed lines: residual coding effects. From top line down, the marks symbolize the thresholding of threshold S_T at 0, 1, and 2, respectively, for (a) "Miss America" and (b) "Claire."

TABLE II
THE TRANSMISSION RATE (KBITS/S)

Frame rate	10 frames/s		30 frames/s	
frame size	176*144	352*288	176*144	352*288
missa 7,0	6.5	26.0	19.5	77.9
missa 7,1	4.8	19.2	14.5	57.5
missa 7,2	4.1	16.2	12.2	48.7
missa 9,0	5.3	21.2	15.9	63.7
missa 9,1	3.9	15.5	11.7	46.5
missa 9,2	3.2	12.8	9.6	38.3
claire 7,0	5.3	21.4	16.1	64.2
claire 7,1	4.2	16.8	12.6	50.5
claire 7,2	3.8	14.2	10.7	42.6
claire 9,0	4.5	17.9	13.5	53.8
claire 9,1	3.5	13.8	10.4	41.4
claire 9,2	2.9	11.7	8.8	35.0

III. SIMULATION AND ANALYSIS

In the experimental work here for this system, the simulations are focused on the quantization, thresholding, residual coding

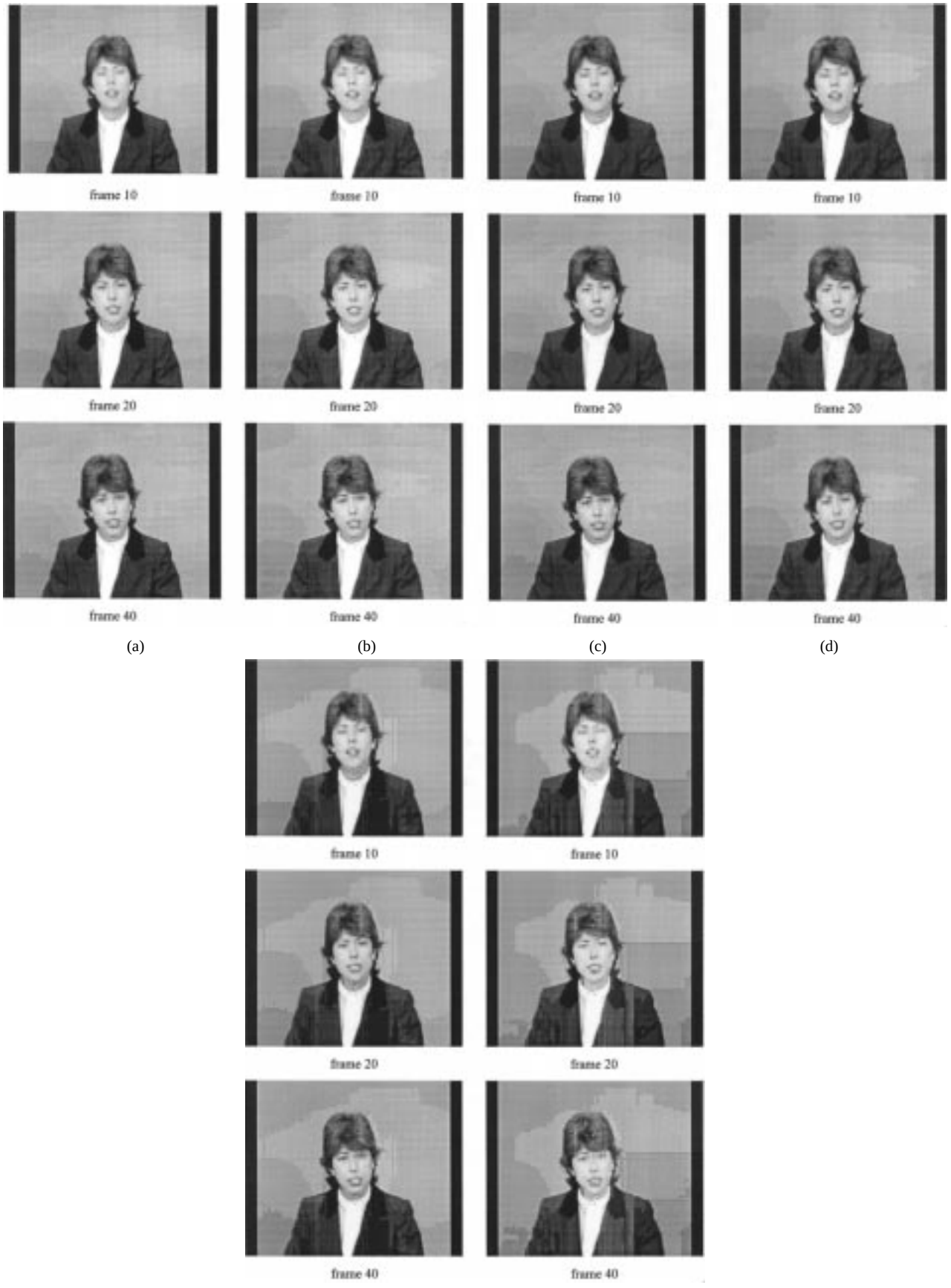


Fig. 21. Sample images: (a) Claire with quantization step 7 and $S_T = 0$. (b) Claire with quantization step 9 and $S_T = 0$. (c) Claire with quantization step 7 and $S_T = 1$. (d) Claire with quantization step 9 and $S_T = 1$. (e) Claire with quantization step 7 and $S_T = 2$. (f) Claire with quantization step 9 and $S_T = 2$.

TABLE III
EVALUATION OF THE MODELS

Cube Width	Order	No. of coefficients	CCDR	PSNR	Estimated Multiplications	Ψ
4	1	4	16	28.26	140	3.230
4	2	10	6.4	32.39	600	0.346
4	3	20	3.2	37.24	1200	0.099
8	1	4	128	21.43	1000	2.743
8	2	10	51.2	23.03	1600	0.737
8	3	20	25.6	25.58	2500	0.262

individually, and the performance of a combination. In most of the simulation processes, we frequently employ the following two estimation functions:

$$bit\,rate = \frac{the\,coded\,bits}{the\,number\,of\,pixels}(bit\,per\,pel,\ bpp) \tag{44}$$

$$\mathrm{PSNR} = 10 \times \log_{10}\left(\frac{\sum 255^2}{\sum error^2}\right)(\mathrm{dB}). \tag{45}$$

In case the subjective observation for the system performance is necessary, the selected frames from the original coefficients and the decoded coefficients will be displayed.

The final simulation result is drawn in Fig. 20. Both thresholding and residual coding were applied. Dots marked in one line are applied with thresholding of threshold (S_T) 0, 1, and 2, while quantization steps are equal to 7 and 9, respectively.

Thresholding makes the tradeoff between the compression ratio and video quality. It is found in Fig. 20 that the PSNR drops faster than the bit rate when the threshold increases from 0 to 2. Unless very low bit rate is demanded for a very narrow bandwidth, the thresholding at 2 is not recommended.

Residual coding not only has the ability of smoothing, but also improves the compression ratio. However, its performance is highly related to the nature of the video sequence. Its greatest contribution is the improvement of subjective quality. The samples of Claire video clips may provide some demonstration.

Finally, the required transmission rate of various configuration is listed in Table II, in which "missa x, y" means the Miss America images of quantization step x, and thresholding (S_T) at y. The configuration for a specified application should be chosen according to the provided channel bandwidth.

In Fig. 21, some example images processed with width-4 cube and first-order approximation are shown.The simulation time is 0.1 s/frame for encoding (352×288 pixels per frame), and 0.05 for decoding, with a Pentium 233-MHz CPU and 32 MB RAM. So this scheme achieves real-time decoding for a 20 frames/s sequence.

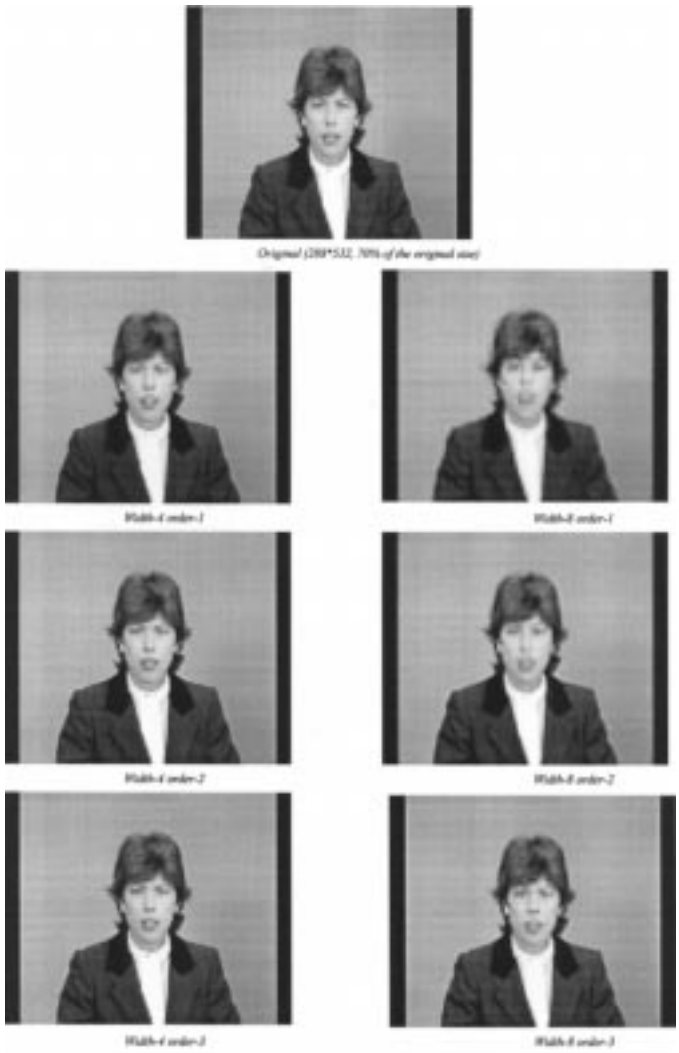


Fig. 22. Frame 1 of video sequence "Claire" generated from various width and order.

IV. CONCLUSION

3DPAC has been derived to obtain a high compression ratio with simple circuitry for wireless video transmission. To achieve

Fig. 23. Comparison with Frames 1, 3, 5, and 7 (50% of the original size, extracted from the video sequence "Claire").

the high compression ratio, the DPCM structure combined with the characteristics of 3DPAC has been derived, and also the lossy thresholding process. It offers beneficial tradeoff between the compression ratio and the quality when the threshold is 1.

To increase the PSNR, the function block residual coding is proposed. Residual coding improves the video quality by the way somewhat like the titter-totter. It raises or lowers the edges of cubes so as to approach the neighboring edges. Besides, since the coefficients are modified, the cube prediction will performs better and the compression ratio is also improved.

Low circuit cost, little storage buffer, and fast speed are predictable because almost every function block is designed making computation only on coefficients. Besides, thresholding and residual coding require only addition and comparison, which takes little implementation area.

APPENDIX

The reason why the $4 \times 4 \times 4$ cube and first-order polynomial approximation are chosen for 3DPAC VCodec has been described in [9], and is briefly illustrated below.

In Table III, there are six types of models with two different widths and three orders. The cube-to-coefficient data ratio (CCDR), PSNR, and computation complexity (in the view of estimated multiplication) are of our concern. The simulation statistics of PSNR and CCDR are also included.

Where Ψ is defined to be s judgement parameter which is a weighting function of CCDR, PSNR, and computation complexity, i.e.

$$\Psi(\mathrm{CCDR}, \mathrm{PSNR}, \mathit{Computation}) = \frac{\mathrm{CCDR} \times \mathrm{PSNR}}{\mathit{Computation}}. \tag{A.1}$$

It is shown mathematically that the models of width-4 and order-1 and of width-8 and order-1 have much higher Ψ-values than the others, yet the comparisons of each other's factors have clearly declared the performance of the models in several aspects of tradeoff consideration.

Furthermore, we have also made comparison of subjective performance on two of the models. Width-4-order–1 and width-8-order–1 both have outstanding Ψ-values and fairly good compression ratio; somehow we have to show the tradeoff with the visual quality. Several series of video sequence and frames are shown in Figs. 22 and 23.

REFERENCES

[1] C. Y. Lu, A. Y. Chen, and K. A. Wen, "Polynomial approximation coding for progressive image transmission," *J. Vis. Commun. Image Repres.*, vol. 8, no. 3, pp. 317–324, June 1997.
[2] M. Kocher and R. Leonardi, "Adaptive region growing technique using polynomial functions for image approximation," *Signal Process.*, vol. 11, pp. 47–60, 1986.
[3] E. Karabassis and E. Spetsakis, "An analysis of image interpolation, differentiation, and reduction using local polynomial fits," *Graph. Models Image Process.*, vol. 57, no. 3, pp. 183–196, May 1995.
[4] M. Eden, M. Unser, and R. Leonardi, "Polynomial representation of pictures," *Signal Process.*, vol. 10, pp. 385–393, 1986.
[5] L.-W. Chan, "The interpolation property of fuzzy polynomial approximation," in *Proc. Int. Symp. Speech, Image Processing and Neural Networks*, Hong Kong, Apr. 1994, pp. 449–452.
[6] L. Corte-Real and A. P. Alves, "Vector quantization of image sequences using variable size and variable shape blocks," *Electron. Lett.*, vol. 26, pp. 1483–1484, 1994.
[7] M. Chan, Y. Yu, and A. Constantinides, "Variable size block matching motion compensation with applications to video coding," *Proc. Inst. Elect. Eng.*, pt. I, vol. 137, pp. 205–212, 1990.
[8] H.-M. Hang and B. Haskell, "Interpolative vector quantization of color images," *IEEE Trans. Pattern. Anal. Machine Intell.*, vol. PAMI-2, pp. 165–168, 1980.
[9] P. Cherriman, "Packet video communications," Ph.D. dissertation, Dept. of Electron., Univ. of Southampton, Southampton, U.K., 1996.
[10] L. Hanzo, "Bandwidth-efficient wireless multimedia communications," *Proc. IEEE*, vol. 86, July 1998.

Kuei-Ann Wen was born in Keelung, Taiwan, R.O.C., in 1961. She received the B.E.E., M.E.E., and Ph.D. degrees from the Department of Electrical and Computer Engineering, National Cheng-Kung University, Taiwan, R.O.C., in 1983, 1985 and 1988, respectively.

She is currently a Professor in the Department of Electrical Engineering, National Chiao-Tung University, Taiwan, R.O.C., and is also involved in several research projects from Wireless Communication Consortium (http://wireless.eic.nctu.edu.tw) and Academic Center of Excellence (http://www.eic.nctu.edu.tw/ace/), under the supervision of the Department of Higher Education, Council of Academic Reviewal and Evaluation of R.O.C. Her research interests are in the areas of wireless communication system, RF circuit design, and video signal processing and transmission.

An-Yi Chen was born in Tainan, Taiwan, R.O.C., in 1971. He received the B.S. degree from the Department of Electrical Communication Engineering, National Chiao-Tung University, Taiwan, R.O.C., in 1993, the M.S. degree from the Department of Electrical and Computer Engineering, State University of New York at Buffalo, in 1995, and the Ph.D. degree from the Institute of Electronics, National Chiao-Tung University, in 2000.

He is currently a Research Engineer of the Information Appliance System Group, Philips Research East Asia–Taipei, Taipei, Taiwan. His research interests are in wireless video transmission, video signal processing, and circuit implementation.

Yin-Jan Lan was born in Tao-Yung, Taiwan, R.O.C., in 1973. He received the B.S. degree from the Department of Electrical Engineering, National Tsing-Hua University, in 1995, and the M.S. degree from the Institute of Electronics, National Chiao-Tung University, Taiwan, R.O.C., in 1997.

He is currently an Engineer in Hsinchu Science-Based Industrial Park, Taiwan, R.O.C.

Jun-Lin Lin was born in Hsinchu, Taiwan, R.O.C., in 1974. He received the B.S. degree from the Department of Electronics Engineering and the M.S. degree from the Institute of Electronics, both at National Chiao-Tung University, Taiwan, R.O.C., in 1997 and 1999, respectively.

He is currently a Research Engineer in the Information Appliance System Group, Philips Research East Asia–Taipei, Taipei, Taiwan. His research interests are video compression and SoC design.

Chia-Huang Lin was born in Taipei, Taiwan, R.O.C., in 1976. He received the B.S. degree rom the Department of Electronics Engineering and the M.S. degree from the Institute of Electronics, both at National Chiao-Tung University, Taiwan, R.O.C., in 1998 and 2000, respectively.

Currently he is serving his obligatory military service. He will then join Spring Soft in Hsinchu Science-Based Industrial Park, Taiwan, R.O.C.