Research Article

Fast Implementation of the Subband Weighting for 3D Wavelet Coding

Guan-Ju Peng,^{1,2} Wen-Liang Hwang,¹ and Sao-Jie Chen²

¹*Institute of Information Science, Academia Sinica, Taipei 115, Taiwan*

²*Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan*

Correspondence should be addressed to Sao-Jie Chen, csj@cc.ntu.edu.tw

Received 21 February 2011; Accepted 11 April 2011

Academic Editors: S. Lee, A. M. Peinado, E. Salerno, and B. Yuan

Copyright © 2011 Guan-Ju Peng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Performing optimal bit-allocation with 3D wavelet coding methods is difficult, because energy is not conserved after applying the motion-compensated temporal filtering (MCTF) process and the spatial wavelet transform. The problem has been solved by extending the 2D wavelet coefficients weighting method with the consideration of the complicated pixel connectivity resulting from the lifting-based MCTF process. However, the computation of weighting factors demands much computing time and vast amount of memory. In this paper, we propose a method, which facilitates the property of sparse matrices, to compute the subband weighing factors efficiently. The proposed method is employed for the 5–3 temporal filter and the 9–7 temporal filter in the lifting-based MCTF process. Experiments on various video sequences show that the computing time and memory needed for the computation of weighting factors are dramatically reduced by applying the proposed approach.

1. Introduction

Scalable video coding, which is developed to support multiple resolutions in a single stream, is an excellent solution to the compression task of the video broadcasting over the heterogeneous network [1, 2]. Compared to H.264/SVC [3], which is recently developed based on the prevailing conventional close-loop codec H.264/AVC, the multiresolution property of 3D wavelet representation based on motion-compensated temporal filtering (MCTF) is a more natural solution to the scalability issue in video coding [4–7]. However, to compete with the great success of scalable coding methods based on H.264, the MCTF-based 3D wavelet video codec must be constantly improved.

For both of H.264/AVC and the wavelet-based encoder, the coding performance is greatly dependent on the technique used to solve the optimal bit-allocation problem. The discrete cosine transform (DCT), which is an orthogonal transform used in H.264/AVC, benefits the procedure of searching the optimal bit-allocation because the variances of the distortions caused by the quantization are equal in the pixel and the frequency domains. However, the wavelet

transforms used in the video/image coding (ex: 5–3 and 9–7 filters) are usually not orthogonal so the variances of the distortions in the pixel domain and the frequency domain are not the same. For image coding, the problem has been elegantly solved [8, 9] by assigning different weights to the subbands, resulting in equivalent reconstruction error variance and quantization error variance in the frequency domain. The method was directly extended for 3D wavelet coding previously [10], but the results are not satisfied because the energy difference between the pixel and frequency domains results not only from using a biorthogonal wavelet but also from the MCTF process, which imposes a different connectivity status (single-connected, multiple-connected, or unconnected) on each pixel during motion prediction. Thus, the direct extension of the methods in [8, 9] from 2D wavelet coefficients to 3D wavelet coefficients cannot solve the 3D wavelet optimal bit-allocation problem because they do not take account of the status of the pixel connectivity in the MCTF process.

To fix this problem, a method, which takes account of both the biorthogonal property of the wavelet and the pixel connectivity resulted from the motion compensation

FIGURE 1: An example of indexing the spatial and temporal decomposed subbands. The above figure shows the indices of the spatially decomposed subbands, in which the subband indexed by (x, y) represents the subband is the y -th after the x -th spatial decomposition. The below figure shows the indices of the temporally decomposed subbands, in which the subband indexed by (m, n) represents the subband is the n -th after the m -th spatial decomposition. Accordingly, any subband obtained by the MCTF can be indexed by (xy, mn) .

FIGURE 2: An example of the lifting structure used in the temporal decomposition. The subbands h , which contain high frequency coefficients, are first obtained after the prediction stage. Then, the subbands l , which contain the low frequency coefficients, are obtained after the update stage.

in the MCTF process, is proposed to equal the variances of the distortion in the pixel and frequency domain by assigning weighting to subbands and, therefore, solves the bit-allocation problem in 3D wavelet coding [11]. Although the results show the method is effective, lots of computing time and memory are needed during the computation of the weighting factors and it causes that the method becomes a heavy load of the encoding process. To make the computation of weighting factors more efficient, we must develop an approach to decrease the time and memory used

FIGURE 3: An example of MCTF motion estimation. The types of connectivity pixels in the example are single-connected, multiple-connected, and unconnected pixels. The corresponding prediction (P) and update (U) matrices are given in (5), where $P^{2i, 2i+1}[x, y] = 1$ indicates that the x -th pixel in frame f_{2i+1} is predicted by the y -th pixel in frame f_{2i} . Note that $U^{2i+1, 2i}(x, y) = 1$ means the x -th pixel in frame f_{2i} is updated by the y -th pixel in frame f_{2i+1} .

by the process. For this purpose, a special data structure and corresponding operations are proposed to save the time and memory during the computation of the subband weighting factors in this paper.

The remainder of this paper is organized as follows. In Section 2, we briefly review the MCTF-EZBC (embedded zero block coding) [12] wavelet coding scheme. In Section 3, we give a summary of how to derive the spatial and temporal weighting factors. In Section 4, we explain the proposed method used to save time and memory during the computation of the subband weighting factors. We also report the experiment results that compare the time and memory used by the traditional method and our method in Section 5. And finally the conclusion is given in Section 6.

2. A MCTF-EZBC Wavelet Video Coding Scheme

In a MCTF-EZBC wavelet video coding scheme, the video frames are first decomposed into multiple wavelet subbands by spatial and temporal wavelet transforms, then the quantization and entropy coding are sequentially applied to the wavelet subbands. According to the order of the spatial and the temporal decompositions, the wavelet coding schemes can be categorized into two categories: 2D+T (spatial filtering first) and T+2D (temporal filtering first). However, no matter 2D+T or T+2D scheme is applied, the spatial and the temporal filterings can be described independently.

The purpose of spatial filtering is separating low and high frequency coefficients from a video frame. The spatial filtering usually consists of multiple sequential 2D wavelet decompositions. In a 2D wavelet decomposition,

FIGURE 4: An example of the arraylist representation. The matrix is $P^{2i,2i+1}$ in (5). According to whether row (above) or column (below) is considered first, the arraylist representation can be categorized into row representation and column representation. The arraylists in the left are used to store the locations of nonzero coefficients, and the arraylists in the right are used to store the values of nonzero coefficients.

the input signal, which is represented by a N by N two-dimensional matrix, is decomposed into four $N/2$ by $N/2$ two-dimensional matrices, which are denoted by LL , HL , LH , and HH . For these subbands, the previous letter means that the subband contains the low (L) or high (H) frequency coefficients after the horizontal 1D wavelet transform, and the following letter means the subband contains the low (L) or high (H) frequency coefficients after the vertical 1D wavelet transform. After the decomposition, subband LL is sequentially taken as the input of the next level spatial decomposition.

If we let $\mathcal{H}_{k,0}$ and $\mathcal{H}_{k,1}$ denote the analyzing matrices in the k -th spatial decomposition, the corresponding wavelet subbands can be computed as

$$\begin{bmatrix} F_{k0} & F_{k1} \\ F_{k2} & F_{k3} \end{bmatrix} = \begin{bmatrix} \mathcal{H}_{k0} \\ \mathcal{H}_{k1} \end{bmatrix} F_{(k-1)0} \begin{bmatrix} \mathcal{H}_{k0}^T & \mathcal{H}_{k1}^T \end{bmatrix}, \quad (1)$$

where F_{k0} , F_{k1} , F_{k2} , and F_{k3} correspond to the LL , HL , LH , and HH subbands, respectively. If there is another spatial decomposition applied to the frame, the subband F_{k0} is further decomposed, by the $k+1$ -th analyzing matrices $\mathcal{H}_{k+1,0}$ and $\mathcal{H}_{k+1,1}$. According to this definition, the subband F_{00} is the frame before any spatial decomposition, and any wavelet subband after the spatial filtering can be indexed by xy , which represent the subband is the y -th subband after the x -th spatial decomposition. In Figure 1(a), the example shows the wavelet subbands with the indices after performing the spatial filtering consisting of two spatial decompositions.

To reconstruct $F_{(k-1)0}$ from the wavelet subbands F_{k0} , F_{k1} , F_{k2} , and F_{k3} , the synthetic matrices $\mathcal{G}_{k0}$ and $\mathcal{G}_{k1}$ are used in the synthetic procedure, which can be represented by several matrix operations as

$$F_{(k-1)0} = \begin{bmatrix} \mathcal{G}_{k0} & \mathcal{G}_{k1} \end{bmatrix} \begin{bmatrix} F_{k0} & F_{k1} \\ F_{k2} & F_{k3} \end{bmatrix} \begin{bmatrix} \mathcal{G}_{k0}^T \\ \mathcal{G}_{k1}^T \end{bmatrix}. \quad (2)$$

FIGURE 5: An example of adding two matrices with the arraylist representation. First, two arraylists $C_{\text{row,pos}}$ and $C_{\text{row,val}}$ are created by cloning $P_{\text{row,pos}}$ and $P_{\text{row,val}}$. For all $x \in \{1, 2, 3, \dots, N2\}$, we move the linklists $U_{\text{row,pos}}[x]$ and $U_{\text{row,val}}[x]$ to the tails of the linklists $C_{\text{row,pos}}[x]$ and $C_{\text{row,val}}[x]$, respectively. For any linklist in $C_{\text{row,pos}}$, if any two elements, which are denoted by $C_{\text{row,pos}}[x][m]$ and $C_{\text{row,pos}}[x][n]$, have the same value, the elements $C_{\text{row,pos}}[x][n]$ and $C_{\text{row,val}}[x][n]$ are removed (if $m < n$), then value of the element $C_{\text{row,val}}[x][m]$ is assigned as $C_{\text{row,val}}[x][m] + C_{\text{row,val}}[x][n]$. This procedure continues until no elements in a linklist have the same value. Then, the arraylists $C_{\text{row,pos}}$ and $C_{\text{row,val}}$ are the arraylist representation of $P + U$.

Compared to the spatial filtering, which takes a frame as its input, the temporal filtering takes multiple frames ($T+2D$) or the subbands with the same spatial indices ($2D+T$) as its input. The temporal filtering consists of multiple temporal decompositions. However, unlike the spatial filtering, which only needs several 2D wavelet transforms during the procedure, the temporal filtering needs multiple 1D wavelet transforms and motion estimation/compensation to specify the input coefficients of each wavelet transform. So, the temporal filtering is more complicated compared to the spatial filtering.

Usually, a temporal decomposition is implemented by the lifting structure, which consists of the predicting stage and the updating stage. As depicted in in Figure 2, the H-frames (denoted by h) are firstly obtained after the predicting stage, then the L-frames (denoted by l) are obtained after the updating stage. The lifting structure can be described by the matrix operations, in which a frame is represented by a one-dimensional matrix f . If each input frame of the temporal filtering has the size N by N and can be represented by a N by N two-dimensional matrix F , the one dimensional matrix f has the size N^2 and can be mapped from F by letting $f[i * N + j] = F[i, j]$.

The motion vectors are computed before the predicting and updating stages. We use the $P_m^{x,y}$, which is a two-dimensional N^2 by N^2 matrix, to denote the motion vectors

obtained by predicting the y -th frame from the x -th frame in the m -th temporal decomposition. Then, the predicting stage of the m -th temporal decomposition can be written as

$$h_m^{2i+1} = f_{m-1}^{2i+1} - \left(\mathcal{H}_m[2i] P_m^{2i,2i+1} f_{m-1}^{2i} + \mathcal{H}_m[2i+2] P_m^{2i+2,2i+1} f_{m-1}^{2i+2} \right), \quad (3)$$

where $\mathcal{H}_m$ is the corresponding coefficient of the wavelet filter. Since the index m represents the m -th lifting stage, the term f_{m-1}^i represents the i -th frame which is obtained after the $(m-1)$ -th lifting stage. If the 5-3 wavelet transform is used, the value of $\mathcal{H}_m$ in the predicting stage is -0.5 .

In the updating stage, the inverse motion vector matrix $U_m^{y,x}$, which has the same size as $P_m^{x,y}$ and can be calculated from the matrix $P_m^{x,y}$ [13], is used to compute the decomposed L-frame as

$$l_m^{2i} = f_{m-1}^{2i} + \left(\mathcal{H}_m[2i+1] U_m^{2i+1,2i} h_m^{2i+1} + \mathcal{H}_m[2i-1] U_m^{2i-1,2i} h_m^{2i-1} \right), \quad (4)$$

where $\mathcal{H}_m$ is the corresponding coefficient of the wavelet filter. If the 5-3 wavelet transform is used, the value of $\mathcal{H}_m$ in the updating stage is 1.

FIGURE 6: An example of adding two matrices with the arraylist representation. To compute $(PU)[x, y]$, we find the elements with the same value in the linklists connected to $P_{row,pos}[x]$ and $U_{column,pos}[y]$. If any two elements are found, the values of these two elements are multiplied. Then, the summation of the products is the value of $(PU)[x, y]$. To store the matrix (PU) , two empty arraylists $M_{row,pos}$ and $M_{row,val}$ are created. Two elements are inserted to the arraylists to store the coefficient $(PU)[x, y]$. The element which has the value y is inserted to the linklist connected to $M_{row,pos}[x]$. And the element which has the value $(PU)[x, y]$ is inserted to the linklist connected to $M_{row,val}[x]$. The arraylist representation of the matrix PU is stored in the arraylists $M_{row,pos}$ and $M_{row,val}$ after $(PU)[x, y]$ for all $x, y \in \{1, 2, 3, \dots, N^2\}$ are computed are inserted to the arraylists.

To understand how to represent the motion vectors with a matrix, an example is given in Figure 3. And these two matrices are constructed as follows:

$$P^{2i,2i+1} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad U^{2i+1,2i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (5)$$

After the m -th temporal decomposition, if another temporal decomposition is needed for the temporal filtering, the L-frames generated by the current temporal decomposition are taken as its input. So, we let f_m^i , which is the i -th frame of the $(m+1)$ -th temporal decomposition, as l_m^{2i} , which is the output $2i$ -th frame of the m -th temporal decomposition. Although the H-frames do not participate in the $(m+1)$ -th

temporal decomposition, we still arrange the indices to them by letting $f_m^{i+S} = h_m^{2i+1}$, in which S is the number of the input frames in the $(m+1)$ -th temporal decomposition. So, any frames obtained by the temporal filtering can be indexed by mn , which represents it is the n -th subband after the m -th temporal decomposition.

Usually, the temporal decompositions are sequentially performed until the output has only one L-frame. Since the decomposed frame can be synthesized, only the frames which cannot be synthesized are necessary to reconstruct all the frames. In Figure 1(b), an example, in which the size of the group of pictures (GOP) is 8, shows the indices of the frames that cannot be synthesized from the decomposed frames.

To recover the original frames, the synthesis is applied to the decomposed frames. The frame which is decomposed last is synthesized first, and vice versa. In the procedure of the synthesis, the inverse updating is firstly performed as

$$f_{m-1}^{2i} = l_m^{2i} - (g_m[2j+1]U_m^{2j+1,2i}h_m^{2j+1} + g_m[2j-1]U_m^{2j-1,2i}h_m^{2j-1}), \quad (6)$$

where $\mathcal{G}_m$ is the coefficient of the wavelet transform used in the inverse updating stage. If the temporal 5–3 filter is used, the value of $\mathcal{G}_m$ is -1 in the inverse updating stage. After the inverse updating stage, the inverse predicting stage is performed as

$$f_{m-1}^{2i+1} = h_m^{2i+1} + (\mathcal{G}_m[2j]P_m^{2j,2i+1}f_{m-1}^{2j} + \mathcal{G}_m[2j+2]P_m^{2j+2,2i+1}f_{m-1}^{2j+2}). \quad (7)$$

If the temporal 5–3 filter is used, the value of $\mathcal{G}_m$ is 0.5 in the inverse updating stage. For some wavelet filters, such as 9–7 filter, a temporal decomposition needs multiple level lifting structure and can be easily extended by cascading multiple one level lifting structures.

After the spatial and temporal filterings, the quantization and entropy coding are applied to these wavelet subbands. The coefficients of these subbands may be quantized by scalar or vector quantization, then the quantized coefficients are coded without loss by the entropy coder. The method is common in the still image compression standard, such as JPEG 2000 [14]. In the decoding process, the quantized coefficients are obtained by decoding the received bitstream, then the subbands are rescaled according to the quantization step used in the encoder. The advantage of separating the quantization and the entropy coding is that the quality of the reconstructed video can be predicted according to the quantization step.

The quantization and the entropy coding can also be combined with the bitplane coding method, such as the EZBC entropy coder [12]. In these methods, the rates allocated to the subbands are calculated first, then the entropy coder encodes the subbands with these rates. The advantage of the scheme is that the rates of the subbands can be any nonnegative integers and the performance of the bit-allocation can be improved accordingly.

No matter the quantization and entropy coding are combined or separated, the bit-allocation greatly affects the coding efficiency. However, because the energy in the pixel domain can be altered after application of the spatial wavelet transform, temporal wavelet transform, and motion estimation in the MCTF process, the bit-allocation is prevented from achieving optimum. To preserve the energy between the pixel and the wavelet domain, we derive the weighting factors of the decomposed subbands. We explain how to derive these weighting factors in the next section.

3. Subband Weighting

The weighting factor indicates how much a unit quantization power in the subband contributes to the overall distortion in the reconstructed GOP, that is, for the subband indexed by (xy, mn) , which means the subband is the spatial y -th subband after the x -th spatial decomposition, and the tem-

poral n -th subband after the m -th temporal decomposition (an example of the index method is depicted in Figure 1), a weighting $\gamma(xy, mn)$ is given to satisfy

$$D = \sum_{(xy, mn) \in \mathbf{S}} \gamma(xy, mn) \times D_{xy, mn}, \quad (8)$$

where D is the variance of the distortion in the pixel domain, $D_{xy, mn}$ is the variance of subband's distortion in the wavelet domain, and $\mathbf{S}$ is the set including the indices of the subbands which cannot be synthesized during the reconstruction. The weighting factor $\gamma(xy, mn)$ can be computed by

$$\gamma(xy, mn) = \alpha(xy) \times \beta(mn), \quad (9)$$

where $\alpha(xy)$ is the spatial weighting factor and $\beta(mn)$ is the temporal weighting factor with respect to the subband indexed by (xy, mn) .

The spatial weighting factors can be directly computed according to the error propagation model mentioned in [15]. However, to compute the temporal weighting factors, the effect of the motion compensation must also be considered, and the approach is mentioned in [11]. In the following paragraph, we explain how to derive the spatial and temporal weighting factors, respectively.

3.1. Spatial Subband Weighting. In this section, we review the method used to derive the weighting factors of spatial wavelet transforms in [15]. According to (2), the reconstructed subband $F_{(k-1)0}$ can be written as

$$F_{(k-1)0} = \mathcal{G}_{k0}F_{k0}\mathcal{G}_{k0}^T + \mathcal{G}_{k0}F_{k1}\mathcal{G}_{k1}^T + \mathcal{G}_{k1}F_{k2}\mathcal{G}_{k0}^T + \mathcal{G}_{k1}F_{k3}\mathcal{G}_{k1}^T. \quad (10)$$

Let Δ_F denote the error matrix resulting from the quantization of an image F ; according to (10), we have

$$\Delta_{F_{(k-1)0}} = \mathcal{G}_{k0}\Delta_{F_{k0}}\mathcal{G}_{k0}^T + \mathcal{G}_{k0}\Delta_{F_{k1}}\mathcal{G}_{k1}^T + \mathcal{G}_{k1}\Delta_{F_{k2}}\mathcal{G}_{k0}^T + \mathcal{G}_{k1}\Delta_{F_{k3}}\mathcal{G}_{k1}^T. \quad (11)$$

Using the property of Kronecker products,

$$V = AUB^T \implies v = (A \otimes B)u, \quad (12)$$

where v, u are column vectors constructed row-wise from the matrices V, U , respectively. Applying this identity to (11), we now have

$$\begin{aligned}\Delta_{c(k-1)0} &= (\mathcal{G}_{k0} \otimes \mathcal{G}_{k0})\Delta_{c_{k0}} + (\mathcal{G}_{k0} \otimes \mathcal{G}_{k1})\Delta_{c_{k1}} + (\mathcal{G}_{k1} \otimes \mathcal{G}_{k0})\Delta_{c_{k2}} \\ &\quad + (\mathcal{G}_{k1} \otimes \mathcal{G}_{k1})\Delta_{c_{k3}},\end{aligned}\quad (13)$$

where Δ_c is the column vector constructed row-wise from the matrices Δ_F . The reconstruction mean square error (MSE) of an image F is

$$\sigma_c^2 = \frac{1}{N^2} E\{\Delta_c^T \Delta_c\}. \quad (14)$$

The equation can be solved under the high bit rate assumption. At a high bit rate, it is assumed that the quantization errors of wavelet subbands are white and mutually uncorrelated [16]. So, the following identities for the vector representation of errors in subbands x and y are obtained:

$$\begin{aligned}E\{\Delta_x^T \Delta_x\} &= E\{\text{tr}(\Delta_x \Delta_x^T)\} = N^2 \sigma_x^2, \\ E\{\Delta_x^T \Delta_y\} &= E\{\text{tr}(\Delta_x \Delta_y^T)\} = 0, \quad \text{when } x \neq y,\end{aligned}\quad (15)$$

where σ_x^2 is the MSE of any element in the vector Δ_x . By substituting (13) into (14) and applying the identities in (15), we obtain

$$\begin{aligned}\sigma_{c(k-1)0}^2 &= \frac{1}{N^2} E\{\Delta_{c(k-1)0}^T \Delta_{c(k-1)0}\} = \frac{1}{N^2} E\{\text{tr}(\Delta_{c(k-1)0} \Delta_{c(k-1)0}^T)\} \\ &= \sigma_{c_{k0}}^2 \text{tr}((\mathcal{G}_{k0} \otimes \mathcal{G}_{k0})(\mathcal{G}_{k0} \otimes \mathcal{G}_{k0})^T) \\ &\quad + \sigma_{c_{k1}}^2 \text{tr}((\mathcal{G}_{k0} \otimes \mathcal{G}_{k1})(\mathcal{G}_{k0} \otimes \mathcal{G}_{k1})^T) \\ &\quad + \sigma_{c_{k2}}^2 \text{tr}((\mathcal{G}_{k1} \otimes \mathcal{G}_{k0})(\mathcal{G}_{k1} \otimes \mathcal{G}_{k0})^T) \\ &\quad + \sigma_{c_{k3}}^2 \text{tr}((\mathcal{G}_{k1} \otimes \mathcal{G}_{k1})(\mathcal{G}_{k1} \otimes \mathcal{G}_{k1})^T) \\ &= \alpha'(k0)\sigma_{c_{k0}}^2 + \alpha'(k1)\sigma_{c_{k1}}^2 + \alpha'(k2)\sigma_{c_{k2}}^2 + \alpha'(k3)\sigma_{c_{k3}}^2.\end{aligned}\quad (16)$$

The above derivation shows that the MSE measured before the wavelet transform is the weighted sum of subbands' MSE after the wavelet transform. Note that the weighting factors are determined only by the filters. According to the discussion in the previous section, the subband indexed by $(k0)$ is further decomposed in the $(k+1)$ -th spatial decomposition. So, the spatial weighting factors of a multilevel decomposition case can be solved by the recursive substitution. To summarize, the MSE of the original frame can be computed as

$$\begin{aligned}\sigma_{c00}^2 &= \alpha'(10)\sigma_{c_{10}}^2 + \alpha'(11)\sigma_{c_{11}}^2 + \alpha'(12)\sigma_{c_{12}}^2 + \alpha'(13)\sigma_{c_{13}}^2 \\ &= \alpha'(10)\{\alpha'(20)\sigma_{c_{20}}^2 + \alpha'(21)\sigma_{c_{21}}^2 + \alpha'(22)\sigma_{c_{22}}^2 + \alpha'(23)\sigma_{c_{23}}^2\} \\ &\quad + \alpha'(11)\sigma_{c_{11}}^2 + \alpha'(12)\sigma_{c_{12}}^2 + \alpha'(13)\sigma_{c_{13}}^2 \\ &= \left\{ \sum_{k=1}^s \sum_{i=1}^3 \left[\prod_{j=1}^{k-1} \alpha'(j0) \right] \alpha'(ki)\sigma_{c_{ki}}^2 \right\} + \left\{ \prod_{j=1}^s \alpha'(j0)\sigma_{c_{s0}}^2 \right\},\end{aligned}\quad (17)$$

in which s is the total number of the spatial wavelet transforms. So, the multilevel spatial weighting factor $\alpha(xy)$ can be computed from one-level spatial weighting factor $\alpha(xy)$ as

$$\alpha(xy) = \begin{cases} \prod_{j=1}^{x-1} \alpha'(j0)\alpha'(xy), & \text{for } x \in \{1, 2, 3, \dots, s\}, y \in \{1, 2, 3\}, \\ \prod_{j=1}^s \alpha'(j0), & \text{for } x = s, y = 0. \end{cases}\quad (18)$$

3.2. Temporal Subband Weighting. We follow the scheme proposed in [11] to explain how to derive and compute the temporal subband weighting factors. The temporal 5–3 filter, which is usually used in the wavelet video coding, is used to explain the procedure. And the derivation of the weighting factors with respect to the temporal 9–3 filter, which is the other filter usually used in the wavelet video coding, is detailed in [11].

We first explain how to compute the weighting factor $\beta'(mn)$, and the weighting factor β' is expected to satisfy

$$\sum_{i=0}^{S-1} \sigma_{f_{m-1}^i}^2 = \sum_{i=0}^{S-1} \beta'(mi)\sigma_{f_m^i}^2, \quad (19)$$

where S is the total number of the input subbands in the m -th temporal decomposition.

If the temporal 5–3 filter is used in the MCTF process, the filter coefficients $\mathcal{H}_m$ used in the predicting and the updating stages are -0.5 and 1 , respectively. To simplifying the notation, $\mathcal{H}_m P_m$ is denoted by P'_m , and $\mathcal{H}_m U_m$ is denoted by U'_m , then (3) and (4) can be written as

$$h_m^{2i+1} = f_{m-1}^{2i+1} - (P_m'^{2i,2i+1} f_{m-1}^{2i} + P_m'^{2i+2,2i+1} f_{m-1}^{2i+2}), \quad (20)$$

$$l_m^{2i} = f_{m-1}^{2i} + (U_m'^{2i+1,2i} h_m^{2i+1} + U_m'^{2i-1,2i} h_m^{2i-1}). \quad (21)$$

Let Δ_f represents the quantization error resulting from lossy source coding, and $f + \Delta_f$ denotes thereconstructed

frame. From (21), we can obtain the reconstruction error of the $2i$ -th frame as follows:

$$\Delta_{f_{m-1}^{2i}} = \Delta_{l_m^{2i}} - U_m'^{2i+1,2i} \Delta_{h_m^{2i-1}} - U_m'^{2i+1,2i} \Delta_{h_m^{2i+1}}. \quad (22)$$

Substituting (22) into (20) for $\Delta_{f_{m-1}^{2i}}$ and $\Delta_{f_{m-1}^{2i+2}}$, we obtain the following reconstruction error of the $(2i+1)$ -th frame:

$$\begin{aligned} \Delta_{f_{m-1}^{2i+1}} &= \Delta_{h_m^{2i+1}} + P_m'^{2i,2i+1} \Delta_{f_{m-1}^{2i}} + P_m'^{2i,2i+1} \Delta_{f_{m-1}^{2i+2}} \\ &= P_m'^{2i,2i+1} \Delta_{l_m^{2i}} + P_m'^{2i,2i+1} \Delta_{l_m^{2i+2}} - P_m'^{2i,2i+1} U_0^{2i,-} \Delta_{h_m^{2i-1}} \\ &\quad + \left(I - P_m'^{2i,2i+1} U_m'^{2i+1,2i} - P_m'^{2i,2i+1} U_m'^{2i+1,2i+2} \right) \Delta_{h_m^{2i+1}} \\ &\quad - P_m'^{2i,2i+1} U_0^{2i+2,+} \Delta_{h_m^{2i+3}}. \end{aligned} \quad (23)$$

Equations (22) and (24) represent a motion-dependent error propagation model for a one-level MCTF process using the 5–3 temporal wavelet filter. The reconstructed MSE of the $2i$ -th frame can be derived as follows: by (14),

$$\begin{aligned} \sigma_{f_{m-1}^{2i}}^2 &= \frac{1}{N^2} E \left\{ \Delta_{f_{m-1}^{2i}}^T \Delta_{f_{m-1}^{2i}} \right\} \\ &= \frac{1}{N^2} E \left\{ \text{tr} \left(\Delta_{l_m^{2i}} \Delta_{l_m^{2i}}^T \right) + \text{tr} \left(U_m'^{2i-1,2i} \Delta_{h_m^{2i-1}} \Delta_{h_m^{2i-1}}^T U_m'^{2i-1,2iT} \right) \right. \\ &\quad + \text{tr} \left(U_m'^{2i+1,2i} \Delta_{h_m^{2i+1}} \Delta_{h_m^{2i+1}}^T U_m'^{2i+1,2iT} \right) \\ &\quad - 2 \text{tr} \left(U_m'^{2i-1,2i} \Delta_{h_m^{2i-1}} \Delta_{l_m^{2i}}^T \right) \\ &\quad + 2 \text{tr} \left(U_m'^{2i+1,2i} \Delta_{h_m^{2i+1}} \Delta_{l_m^{2i}}^T U_m'^{2i-1,2iT} \right) \\ &\quad \left. - 2 \text{tr} \left(U_m'^{2i+1,2i} \Delta_{h_m^{2i+1}} \Delta_{h_m^{2i-1}}^T \right) \right\}. \end{aligned} \quad (24)$$

By defining $\mathcal{T}(\mathcal{A}) = \text{tr}(\mathcal{A} \cdot \mathcal{A}^T)$ and using the high bit rate assumption that derives the identities in (15), the last three cross-terms are zero and (24) becomes

$$\begin{aligned} \sigma_{f_{m-1}^{2i}}^2 &= \sigma_{l_m^{2i}}^2 + \mathcal{T} \left(U_m'^{2i+1,2i} \right) \sigma_{h_m^{2i-1}}^2 + \mathcal{T} \left(U_m'^{2i-1,2i} \right) \sigma_{h_m^{2i+1}}^2 \\ &= \omega \left(f_{m-1}^{2i} | l_m^{2i} \right) \sigma_{l_m^{2i}}^2 + \omega \left(f_{m-1}^{2i} | h_m^{2i-1} \right) \sigma_{h_m^{2i-1}}^2 + \omega \left(f_{m-1}^{2i} | h_m^{2i+1} \right) \sigma_{h_m^{2i+1}}^2 \end{aligned} \quad (25)$$

Following the same derivation, the reconstruction error of the $(2i+1)$ -th frame is

$$\begin{aligned} \sigma_{f_{m-1}^{2i+1}}^2 &= \mathcal{T} \left(P_m'^{2i+1,2i} \right) \sigma_{l_m^{2i}}^2 + \mathcal{T} \left(P_m'^{2i+1,2i+2} \right) \sigma_{l_m^{2i+2}}^2 \\ &\quad + \mathcal{T} \left(P_m'^{2i-1,2i} U_m'^{2i+1,2i} \right) \sigma_{h_m^{2i-1}}^2 \\ &\quad + \mathcal{T} \left(I - P_m'^{2i-1,2i} U_m'^{2i-1,2i} - P_m'^{2i+1,2i+2} U_m'^{2i+3,2i+2} \right) \sigma_{h_m^{2i+1}}^2 \\ &\quad + \mathcal{T} \left(P_m'^{2i+3,2i+2} U_m'^{2i+1,2i+2} \right) \sigma_{h_m^{2i+3}}^2 \\ &= \omega \left(f_{m-1}^{2i+1} | l_m^{2i} \right) \sigma_{l_m^{2i}}^2 + \omega \left(f_{m-1}^{2i+1} | l_m^{2i+2} \right) \sigma_{l_m^{2i+2}}^2 \\ &\quad + \omega \left(f_{m-1}^{2i+1} | h_m^{2i-1} \right) \sigma_{h_m^{2i-1}}^2 + \omega \left(f_{m-1}^{2i+1} | h_m^{2i+1} \right) \sigma_{h_m^{2i+1}}^2 \\ &\quad + \omega \left(f_{m-1}^{2i+1} | h_m^{2i+3} \right) \sigma_{h_m^{2i+3}}^2. \end{aligned} \quad (26)$$

By applying derivations similar to those in (25) and (26) to each input frame, we can relate the reconstruction errors before and after the temporal decomposition by a linear relation:

$$\begin{bmatrix} \sigma_{f_{m-1}^{2i-2}}^2 \\ \sigma_{f_{m-1}^{2i-1}}^2 \\ \sigma_{f_{m-1}^{2i}}^2 \\ \sigma_{f_{m-1}^{2i+1}}^2 \\ \sigma_{f_{m-1}^{2i+2}}^2 \\ \sigma_{f_{m-1}^{2i+3}}^2 \\ \sigma_{f_{m-1}^{2i+4}}^2 \\ \vdots \end{bmatrix} = \mathcal{W}^{m-m-1} \begin{bmatrix} \sigma_{h_m^{2i-3}}^2 \\ \sigma_{h_m^{2i-2}}^2 \\ \sigma_{h_m^{2i-1}}^2 \\ \sigma_{l_m^{2i}}^2 \\ \sigma_{h_m^{2i+1}}^2 \\ \sigma_{l_m^{2i+2}}^2 \\ \sigma_{h_m^{2i+3}}^2 \\ \vdots \end{bmatrix} = \mathcal{W}^{m-m-1} \mathcal{X}^{m-m-1} \begin{bmatrix} \sigma_{f_m^{2i-3}}^2 \\ \sigma_{f_m^{2i-2}}^2 \\ \sigma_{f_m^{2i-1}}^2 \\ \sigma_{f_m^{2i}}^2 \\ \sigma_{f_m^{2i+1}}^2 \\ \sigma_{f_m^{2i+2}}^2 \\ \sigma_{f_m^{2i+3}}^2 \\ \vdots \end{bmatrix}, \quad (27)$$

where

$\mathcal{W}^{m \rightarrow m-1}$

$$= \begin{bmatrix} \omega(f_{m-1}^{2i-2} | h_m^{2i-3}) & \omega(f_{m-1}^{2i-2} | l_m^{2i-2}) & \omega(f_{m-1}^{2i-2} | h_m^{2i-1}) & 0 & 0 & 0 & 0 & \dots \\ \omega(f_{m-1}^{2i-1} | h_m^{2i-3}) & \omega(f_{m-1}^{2i-1} | l_m^{2i-2}) & \omega(f_{m-1}^{2i-1} | h_m^{2i-1}) & \omega(f_{m-1}^{2i-1} | l_m^{2i}) & \omega(f_{m-1}^{2i-1} | h_m^{2i+1}) & 0 & 0 & \dots \\ 0 & 0 & \omega(f_{m-1}^{2i} | h_m^{2i-1}) & \omega(f_{m-1}^{2i} | l_m^{2i}) & \omega(f_{m-1}^{2i} | h_m^{2i+1}) & 0 & 0 & \dots \\ 0 & 0 & \omega(f_{m-1}^{2i+1} | h_m^{2i-1}) & \omega(f_{m-1}^{2i+1} | l_m^{2i}) & \omega(f_{m-1}^{2i+1} | h_m^{2i+1}) & \omega(f_{m-1}^{2i+1} | l_m^{2i+2}) & \omega(f_{m-1}^{2i+1} | h_m^{2i+3}) & \dots \\ 0 & 0 & 0 & 0 & \omega(f_{m-1}^{2i+2} | h_m^{2i+1}) & \omega(f_{m-1}^{2i+2} | l_m^{2i+2}) & \omega(f_{m-1}^{2i+2} | h_m^{2i+3}) & \dots \\ 0 & 0 & 0 & 0 & \omega(f_{m-1}^{2i+3} | h_m^{2i+1}) & \omega(f_{m-1}^{2i+3} | l_m^{2i+2}) & \omega(f_{m-1}^{2i+3} | h_m^{2i+3}) & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}, \quad (28)$$

$$\mathcal{X}^{m \rightarrow m-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & \dots \\ 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 & 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \vdots & \dots \end{bmatrix}. \quad (29)$$

The matrix $\mathcal{X}^{m \rightarrow m-1}$ is used to adjust the positions of the subbands. To measure the consequence of quantizing a subband after the temporal decomposition, we should aggregate the errors induced in all the subbands before the temporal decomposition. This is exactly the summation of the corresponding column of the matrix $\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1}$ for the subband. For example, the weighting factor of the reconstruction error resulting from subband f_m^{2i+1} is the summation of the values in the $2i+1$ column of $\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1}$ and denoted by $\beta'(m(2i+1))$. The following equation shows how to compute the weighting factor β' :

$$\begin{bmatrix} \beta'(m0) \\ \beta'(m1) \\ \beta'(m2) \\ \beta'(m3) \\ \beta'(m4) \\ \beta'(m5) \\ \beta'(m6) \\ \vdots \end{bmatrix} = \begin{bmatrix} \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q1} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q2} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q3} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q4} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q5} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q6} \\ \sum_q (\mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1})_{q7} \\ \vdots \end{bmatrix}. \quad (30)$$

If the temporal filtering consists of total t temporal decompositions, the variances of the distortions of the frames in the pixel domain can be computed as

$$\begin{bmatrix} \sigma_{f_0^0}^2 \\ \sigma_{f_0^1}^2 \\ \sigma_{f_0^2}^2 \\ \sigma_{f_0^3}^2 \\ \sigma_{f_0^4}^2 \\ \sigma_{f_0^5}^2 \\ \sigma_{f_0^6}^2 \\ \vdots \end{bmatrix} = \left(\prod_{m=1}^t \mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1} \right) \begin{bmatrix} \sigma_{f_t^0}^2 \\ \sigma_{f_t^1}^2 \\ \sigma_{f_t^2}^2 \\ \sigma_{f_t^3}^2 \\ \sigma_{f_t^4}^2 \\ \sigma_{f_t^5}^2 \\ \sigma_{f_t^6}^2 \\ \vdots \end{bmatrix} \quad (31)$$

$$= \left(\prod_{m=1}^t \mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1} \right) \begin{bmatrix} \sigma_{f_t^{(t0)}}^2 \\ \sigma_{f_t^{(t1)}}^2 \\ \sigma_{f_t^{((t-1)2)}}^2 \\ \sigma_{f_t^{((t-1)3)}}^2 \\ \sigma_{f_t^{((t-1)4)}}^2 \\ \sigma_{f_t^{((t-1)5)}}^2 \\ \sigma_{f_t^{((t-1)6)}}^2 \\ \vdots \end{bmatrix}.$$

Please note the index of subband f^i can be changed from i to (mn) , which represents the subband is obtained as the n -th subband after the m -th temporal decomposition (see Figure 1(b)). So, the temporal weighting factor $\beta(mn)$ used in (9) can be computed as

$$\beta(mn) = \sum_q \left(\prod_{m=1}^t \mathcal{W}^{m \rightarrow m-1} \mathcal{X}^{m \rightarrow m-1} \right)_{qn}. \quad (32)$$

Since the spatial and temporal weighting factors are both derived, the performance of the bit-allocation method

can be improved accordingly. In our previous work [11], we compared three bit-allocation methods, which consider spatial weighting factors, spatial-temporal weighting factors without motion vectors, and spatial-temporal weighting factors with motion vectors, respectively. And the results showed the bit-allocation method considering spatial-temporal weighting factors with motion vectors greatly outperforms the other two methods. However, the computation of the best weighting method costs lots of computing resources. So, the method, which reduces the needed resources, is proposed in the next section.

4. Fast Computation of the Weighting Factors

As our discussion in the previous section, (16) is used to compute the spatial weighting factors, and (24) and (25) are used to compute the temporal weighting factors. However, a lot of computing resources are needed to obtain these weighting factors. Let the size of an input frame is N by N . In (16), (24), and (25), if we use matrices to represent $\mathcal{G} \otimes \mathcal{G}$, P , and U , the size of each matrix is N^2 by N^2 . If m bytes are used to store a coefficient in a matrix, a matrix would take $N^2 \times N^2 \times m$ bytes in storage. For a CIF frame (of size 352×288), the matrix representation needs at least 39204 megabytes for only one matrix. Thus, using the matrix representation to compute the weighting factors is extremely inefficient since it needs to store many N^2 by N^2 matrices.

Another issue that handicap using the matrix representation for weighting factor is the computing time to perform matrix addition and multiplication. If we use the conventional matrix operations, which are defined as follows:

$$(AB)[x, y] = \sum_{i=1}^n A[x, i]B[i, y], \quad (33)$$

$$(A + B)[x, y] = A[x, y] + B[x, y], \quad (34)$$

the time complexity of the matrix multiplication in computing the weighting factors is bounded by $O(N^6)$. For a CIF frame, it costs approximately 10^{15} real number multiplications for performing one matrix multiplication. Fortunately, these matrices are usually very sparse, that is, only few coefficients in these matrices are not zero. By taking advantage of the property, we use an efficient data structure to store the required data in computing the weighting factors.

Thus, we propose to use arraylist representation. An arraylist consists of a one-dimensional array comprised of pointers to linklists. Figure 4 is an example to demonstrate the arraylist structure. In the proposed data structure, two arraylists, which, respectively, record the positions and values of the nonzero coefficients, are used to represent a sparse matrix of large size. In Figure 4, the matrix P is an abbreviation of the matrix $P^{2i, 2i+1}$. The arraylists $P_{\text{row, pos}}$ and $P_{\text{row, val}}$, respectively, record the positions and values of nonzero coefficients in the matrix P . Because the matrix P has 6 rows, both arraylists have 6 elements. For the arraylist $P_{\text{row, pos}}$, the x -th linklist links to the locations of nonzero elements at the x -th row of the matrix. The arraylist $P_{\text{row, val}}$ has a similar structure at that of $P_{\text{row, pos}}$, instead of linking

to locations of nonzero elements, $P_{\text{row, val}}$ links to the values of the nonzero elements. That if the position of the nonzero coefficient is recorded in $P_{\text{row, pos}}[x, y]$, we can deduce that the x -row and y -column element of the matrix P is a nonzero coefficient, and whose value is stored in $P_{\text{row, val}}[x, y]$.

If the matrix is sparse enough, the proposed arraylist data structure can greatly reduce the memory because only the nonzero locations and values are stored in the arraylists. If we assume that there are n nonzero coefficients in each row of an N^2 by N^2 matrix. To construct an arraylist, we need a one-dimensional array of N^2 elements, and N^2 linklists, one for each element in the arraylist. Each linklist has n elements. Because our structure uses two arraylists, it needs $2(n+1)N^2m$ bytes, where m is the number of bytes used to store a coefficient. Thus, our data structure has reduced to size to represent an N^2 by N^2 matrix from mN^4 bytes to $2(n+1)N^2m$ bytes. The proposed data structure in many examples can reduce the memory to store a of size $352^2 \times 288^2$ matrix from 39204 megabytes to about 1 megabytes.

After solving the matrix storage, we use the matrices in (5) to explain how to perform matrix addition and multiplication with the proposed data structure. To simplify the notation, P and U are used to represent $P^{2i, 2i+1}$ and $U^{2i+1, 2i}$ in (5), respectively. In matrix addition, we first create two new arraylists $C_{\text{row, pos}}$ and $C_{\text{row, val}}$ to store the result of the addition with

$$\begin{aligned} C_{\text{row, pos}} &= P_{\text{row, pos}}, \\ C_{\text{row, val}} &= P_{\text{row, val}}. \end{aligned} \quad (35)$$

For all $x \in \{1, 2, 3, \dots, N^2\}$, we move the linklists $U_{\text{row, pos}}[x]$ and $U_{\text{row, val}}[x]$ to the tails of the linklists $C_{\text{row, pos}}[x]$ and $C_{\text{row, val}}[x]$, respectively. Then, all linklists in $C_{\text{row, pos}}$ are examined to search if any two elements have the same value. If the two elements, denoted by $C_{\text{row, pos}}[x, m]$ and $C_{\text{row, pos}}[x, n]$ with $m < n$, have the same value, then the value of $C_{\text{row, val}}[x, m] + C_{\text{row, val}}[x, n]$ is assigned to the element $C_{\text{row, val}}[x, m]$. Finally, the element $C_{\text{row, val}}[x, n]$ is removed for the consistency of the data structure. If no elements of a linklist in $C_{\text{row, pos}}$ have the same value for all the rows, the matrix addition with respect to the proposed data structure is completed. The results of the matrix $P + U$ are stored in $C_{\text{row, pos}}$ and $C_{\text{row, val}}$. Figure 5 demonstrates the matrix addition by using the proposed data structure.

To obtain the matrix multiplication PU with the proposed data structure, the coefficients $(PU)[x, y]$ are needed for all $x, y \in \{1, 2, 3, \dots, N^2\}$. We create two empty arraylists $M_{\text{row, pos}}$ and $M_{\text{row, val}}$, each of which arraylists has N^2 linklists to store the result of matrix multiplication. To compute $(PU)[x, y]$, we examine the linklists $P_{\text{row, pos}}[x]$ and $U_{\text{column, pos}}[y]$ to search for the elements of the same value, then we multiply their corresponding values in $P_{\text{row, val}}[x]$ and $U_{\text{column, val}}[y]$. The value of $(PU)[x, y]$ is the summation of the products obtained by the process. To store the value of $(PU)[x, y]$, the values y and $(PU)[x, y]$ are appended to the tail of the x -th link lists of $M_{\text{row, pos}}$ and $M_{\text{row, val}}$, respectively. The result of performing matrix multiplication with the specified data structure is depicted in Figure 6.

FIGURE 7: The time consumed to compute the weighting factors. The x -axis identifies Scheme 1 (the traditional matrix representation), or Scheme 2 (the proposed arraylist representation) is applied to compute the weighting factors. The y -axis is the logarithm of the time consumed by the schemes. The size of the video sequence is 1/4QCIF (88×72), and the motion vectors is obtained from T+2D MCTF. Each GOP has 32 frames. The experiment is executed on the work station with double quadcore CPUs and total 12 GB memory. The consumed time is measured by subtracting the times stamp between the beginning and the end of the process that computes the weighting factors.

If we assume that a row of the matrix has n nonzero coefficients, to compute the coefficients of a row with respect to the matrix addition needs a matching algorithm between two sets with n elements, several ($< n$) real-number additions for the matched elements, several moving ($< n$) and deleting ($< n$) operations on the linklists. Compared to the time

complexity $O(N^4)$ of the conventional matrix addition, the time complexity $O(nN^2)$ of the our method is much more efficient, if the matrix is sparse enough. Meanwhile, for the matrix multiplication PU with the proposed data structure, for any $x, y \in \{1, 2, 3, \dots, N^2\}$, the value of $(PU)[x, y]$ is necessary. And $(PU)[x, y]$ can be computed by a matching

FIGURE 8: The amount of the memory used to compute the weighting factors. The x -axis identifies Scheme 1 or Scheme 2 is applied in the process, and the y -axis identifies the amount of the memory used during the process. The experiment is executed on the same equipments and settings as those used in Figure 7. The amount of memory used by the process is measured by averaging the monitored size of the memory used to compute each weighting factor.

algorithm between two sets with n elements, several ($< n$) real-number multiplications, and a summation of these real-number. Compared to the time complexity $O(N^6)$ of the conventional matrix addition, the time complexity $O(nN^4)$ of the proposed method is much lower if the matrix is sparse enough.

In addition to the matrix addition and multiplication, the operator $\mathcal{T}$ defined by $\mathcal{T}(\mathcal{A}) = \text{tr}(\mathcal{A} \cdot \mathcal{A}^T)$ is also

needed during the computation of the subband weighting factors. However, the value of $\mathcal{T}(\mathcal{A})$ (of size N^2 by N^2) can be computed by summing all squares of the coefficients in the matrix $\mathcal{A}$. For the conventional matrix representation, to compute the value of $\mathcal{T}(\mathcal{A})$ needs N^4 real number multiplications and about N^4 real number additions, and for the proposed data structure, it needs nN^2 real number multiplications and about nN^2 real number additions, where

n is the average number of nonzero coefficients in a row of the matrix.

Although the arraylist representation is more complicated than the matrix representation, the proposed method can reduce the memory as well as the computing time while the operated matrices are quite sparse. As a consequence, because most of the matrices used to compute the weighting factors are very sparse, the proposed arraylist representation is extremely suitable for the weighting factor computing process.

5. Experiment Results

We now demonstrate the efficacy of applying the arraylist representation to compute the weighting factors. In the experiment, we compare the time and memory used by the matrix representation and the arraylist representation during the computation of the weighting factors. The conventional matrix representation uses a two-dimensional array to store a matrix and the operations in (34) to calculate the addition and multiplication between two matrices. The arraylist representation uses two arraylists to store a matrix, and the operations are performed as the description in the previous section.

A T+2D video codec is used in the experiment. Because a GOP has 32 CIF frames, a five-level MCTF process using the 5–3 or the 9–7 temporal wavelet filter is applied to the GOP. Then, each frame of a video sequence is decomposed by applying a three-level spatial wavelet transform using either the 5–3 or the 9–7 wavelet filter. To search the motion vectors used in the temporal filtering, the motion estimation using a full-search with integer-pixel accuracy on the dyadic wavelet coefficients is applied. The block size is 4×4 , and the search range for both the horizontal and vertical dimensions is $[-16, 15]$.

After the MCTF process, the motion vectors and the wavelet filters used in the spatial and temporal filterings are used to compute the weighting factors. We demonstrate and compare the results on two different schemes. Scheme 1 computes the weighting factors with the conventional matrix representation. Scheme 2 computes the weighting factors with the proposed arraylist representation. The equipment used in the experiment is a work station with double quadcore CPUs and total 12 GB memory. Both 5–3 and 9–7 temporal filters are experimented for various $1/4$ QCIF (88×72) sequences.

Figure 7 compares the time consumed by Scheme 1 and Scheme 2. The consumed time is measured by subtracting the time stamp between the beginning and the end of computing the weighting factors. We observe that the time consumed by Scheme 2 is much less than that consumed by Scheme 1.

We also compare the amount of memory used by these two schemes. Figure 8 illustrates the amount of memory used by Scheme 2 is much less than that used by Scheme 1. The amount of used memory is obtained by averaging the monitored size of memory used to compute each weighting factor.

No matter the time or memory used to compute the weighting factors, the experimental results show the proposed arraylist representation greatly reduce them compared to the conventional matrix representation.

6. Conclusion

In this paper, we proposed the arraylist representation for a matrix and the proposed method reduces the time and memory consumed used by the computation of weighting factors. We employed the proposed method on a T+2D structure and show that the time and memory needed to compute the weighting factors are greatly reduced compared to the conventional matrix representation. In a future work, we will expand the method to deal with the motion vectors obtained from the subpixel motion estimation to support more wavelet encoders.

References

- [1] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. Van Der Schaar, J. Cornelis, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Processing: Image Communication*, vol. 19, no. 7, pp. 653–673, 2004.
- [2] Q. Zhang, Q. Guo, Q. Ni, W. Zhu, and Y.-Q. Zhang, "Sender-adaptive and receiver-driven layered multicast for scalable video over the internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 4, pp. 482–495, 2005.
- [3] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.
- [4] J.-R. Ohm, "Three-dimensional subband coding with motion compensation," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 559–571, 1994.
- [5] S.-J. Choi and J. W. Woods, "Motion-compensated 3-D subband coding of video," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 155–167, 1999.
- [6] T. Ruster, K. Hanke, and J.-R. Ohm, "Transition filtering and optimized quantization in interframe wavelet video coding," in *Proceedings of the SPIE Visual Communications and Image Processing (VCIP '03)*, vol. 5150, pp. 682–694, 2003.
- [7] P. Chen and J. W. Woods, "Bidirectional MC-EZBC with lifting implementation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 10, pp. 1183–1194, 2004.
- [8] B. Usevitch, "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 22–34, 2001.
- [9] D. S. Taubman and M. W. Marcellin, *JPEG2000*, Kluwer Academic, 2002.
- [10] R. Xiong, J. Xu, F. Wu, S. Li, and Y.-Q. Zhang, "Optimal subband rate allocation for spatial scalability in 3D wavelet video coding with motion aligned temporal filtering," in *Proceedings of the SPIE Visual Communications and Image Processing (VCIP '05)*, vol. 5960, pp. 381–392, 2005.
- [11] C.-C. Cheng, G.-J. Peng, and W.-L. Hwang, "Subband weighting with pixel connectivity for 3-D wavelet coding," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 52–62, 2009.
- [12] S. T. Hsiang and J. W. Woods, "Embedded image coding using zeroblocks of subband/wavelet coefficients and context

- modeling,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '00)*, pp. 662–665, May 2000.
- [13] J.-R. Ohm, M. Van Der Schaar, and J. W. Woods, “Interframe wavelet coding—motion picture representation for universal scalability,” *Signal Processing: Image Communication*, vol. 19, no. 9, pp. 877–908, 2004.
- [14] M. J. Gormish, D. Lee, and M. W. Marcellin, “JPEG 2000: overview, architecture, and applications,” in *Proceedings of the IEEE International Conference on Image Processing (ICIP '00)*, pp. 29–32, September 2000.
- [15] B. Usevitch, “Optimal bit allocation for biorthogonal wavelet coding,” in *Proceedings of the Data Compression Conference (DCC '96)*, pp. 387–395, 1996.
- [16] N. Jayant and P. Noll, *Digital Coding of Waveforms*, Prentice Hall, Englewood Cliffs, NJ, USA, 1984.