Scalable Wavelet Video Coding Using Aliasing-Reduced Hierarchical Motion Compensation

Xuguang Yang, *Member, IEEE*, and Kannan Ramchandran, *Member, IEEE*

***Abstract*—We describe a spatially scalable video coding framework in which motion correspondences between successive video frames are exploited in the wavelet transform domain. The basic motivation for our coder is that motion fields are typically smooth and, therefore, can be efficiently captured through a multiresolutional framework. A wavelet decomposition is applied to each video frame and the coefficients at each level are predicted from the coarser level through backward motion compensation. To remove the aliasing effects caused by downsampling in the transform, a special interpolation filter is designed with the weighted aliasing energy as part of the optimization goal, and motion estimation is carried out with lowpass filtering and interpolation in the estimation loop. Further, to achieve robust motion estimation against quantization noise, we propose a novel backward/forward hybrid motion compensation scheme, and a tree structured dynamic programming algorithm to optimize the backward/forward mode choices. A novel adaptive quantization scheme is applied to code the motion predicted residue wavelet coefficients. Experimental results reveal 0.3–2-dB increase in coded PSNR at low bit rates over the state-of-the-art H.263 standard with all enhancement modes enabled, and similar improvements over MPEG-2 at high bit rates, with a considerable improvement in subjective reconstruction quality, while simultaneously supporting a scalable representation.**

I. Introduction

During the last decade, the discrete wavelet transform (DWT) has gained much popularity in image coding. A primary reason behind this trend is the DWT's improved ability to efficiently capture the space and frequency characteristics of typical images. State-of-the-art wavelet image coders typically owe their success to accurately modeling the statistical distribution of wavelet coefficients and exploiting their higher order dependencies. For tutorials on the wavelet transform and its applications to image coding, refer to [1]–[6].

Recently, there has also been active research in trying to apply the DWT to video coding, for which a critical problem is the estimation of motion fields. There are two major classes of motion estimation algorithms, namely, the *forward* and *backward* approaches. In a *forward* approach, the encoder performs motion estimation and transmits the estimated motion parameters explicitly to the decoder. In a *backward* approach, both the encoder and the decoder perform motion estimation on previously decoded data, therefore no motion information needs to be sent. The wavelet decomposition is most suitable for backward motion estimation, because it provides a multiresolution representation of the video frames. By coding each frame in a coarse-to-fine fashion in the wavelet domain, and exploiting the correlations among motion fields across the multiresolution pyramid using a backward approach, one can expect to build a highly efficient video coder.

Another motivation for wavelet-based video coding is to support scalability—partial decoding of the entire sequence at various quality levels. Scalability is a strongly desired property for many multimedia video applications, but is very hard to achieve in conventional motion compensated video coding. A primary reason is the so-called "drift" problem [7], [8]. "Drift" occurs between the encoder and decoder due to the availability of different resolution previous frames for motion compensation. In a hierarchical backward motion compensation framework, this problem is avoided, because the encoder and decoder always have the reconstructed previous frame for motion compensation.

Despite these significant potential advantages, practical implementation of an efficient wavelet-based video coder is complicated. A major difficulty is in motion estimation, and is caused by aliasing artifacts from downsampling operations [14], [15]. The aliasing problem can be better understood in the frequency domain, in which translational motion shows up as linear phase modulation. During downsampling, an aliasing term with a different phase shift is added to the original signal spectrum, which destroys the original linear phase relationship. Therefore, direct motion estimation from wavelet coefficients is in general infeasible.

In [12], Uz and Vetterli proposed a scalable video coder using a three-dimensional (3-D) spatio–temporal Laplacian pyramid, with motion vectors differentially estimated and coded for each level. The coding efficiency is affected by the over-complete Laplacian representation and the redundancy in the multiresolutional motion fields. Later, Naveen and Woods [13] considered both backward and forward motion compensation in the wavelet domain, in which the backward approach involves using the previous two video frames. They also discussed the alternative of using bandpass pyramid versus lowpass pyramid structures. Their backward motion field is defined in the previous frame, which however has to be extrapolated to the current frame for motion prediction. In [14], Tsunashima *et al.* discussed the aliasing problem, and presented a coder that performs motion estimation in the intensity domain (full resolution), and

Manuscript received March 26, 1998; revised November 5, 1999. This work was supported in part by the National Science Foundation under Award NSF MIP-97-03181 (CAREER), the Army Research Office under the Young Investigator Award DAAH04-96-1-0342, and the Office of Naval Research under the Young Investigator Award N00014-97-1-0864. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Rashid Ansari.

X. Yang is with the Imaging Technologies Department, Hewlett-Packard Laboratories, Palo Alto, CA 94304 USA (e-mail: yangx@ifp.uiuc.edu).

K. Ramchandran is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720 USA (e-mail: kannanr@eecs.berkeley.edu).

Publisher Item Identifier S 1057-7149(00)03861-6.

interpolates the lower resolution levels for motion compensation. However, scalability is impossible in this approach because of the need for full-resolution motion fields. In [15], Nosratinia and Orchard presented a purely backward coding framework, in which the aliasing problem is alleviated by matching the lower resolution current frame with the unsampled previous frame at the same resolution. In this approach, the current frame is still aliased, and the estimation accuracy could suffer because the two frames go through different operations for motion estimation, i.e., the current frame is downsampled once more and suffers one more level of aliasing than the previous frame.

In this paper, we present a new video coding framework that uses hierarchical motion estimation in the wavelet transform domain. In our coder, the aliasing problem is tackled from two angles. First, at each level of the wavelet pyramid, we use an interpolated and filtered coarser level for backward motion estimation. Second, more significantly, a specially designed interpolation filter is used to reduce the aliasing energy. An optimization criterion is derived for the interpolation filter design that takes into account the statistical properties of the input video signal. Applying this interpolation filter, we then build a basic backward coding system, in which the quantized video frames at coarser levels are used in the motion estimation operation, thus avoiding the need to send any motion information. Finally, the motion compensated error signal is coded using a novel adaptive estimation-quantization (EQ) coder [6]. Experimental results confirm and quantify the benefits of our interpolation filter, over alternatives like having no interpolation, or using the standard filter-bank synthesis filter in place of our optimized interpolation filter.

The purely backward motion estimation scheme performs reasonably well at moderate to high bit rates. At very low bit rates, however, the backward instability problem begins to manifest itself. Backward adaptive frameworks, as are well known, become very fragile in the face of excess quantization noise, and this can prove detrimental to accurate motion estimation in our case. This effect is magnified by the motion prediction feedback loop, and becomes very serious at low bit rates, when the quantization operation is very lossy. This backward instability problem has not been addressed in previous backward adaptive works. We propose a novel hybrid backward/forward motion compensation technique, in which we judiciously choose to send forward motion information estimated from the current resolution level when the backward motion vector fails to be very accurate. In this case, we trade off the accuracy of motion prediction with the amount of motion information we need to send. The backward/forward binary decisions at all spatial locations and resolution levels are undertaken to attain rate-distortion (R-D) optimality. To this end, we borrow the zerotree data structure from the celebrated embedded zerotree wavelet (EZW) image coder [3], inspired by the striking similarity between our backward/forward binary decision field and the so-called significance map of wavelet coefficients, and invoke a dynamic programming algorithm to optimize the binary decision tree. The backward instability problem is then solved by adding more "protection" against quantization noise, and the resulting coder demonstrates remarkable improvement over the basic purely backward coder, especially at low bit rates.

Our proposed coder in its full resolution operation achieves a 0.3–2 dB improvement in PSNR over both MPEG-2 standard implementation at high bit rates and that of H.263 at low bit rates. In addition, it also has the following salient features.

- It is spatially scalable. The coder achieves scalability through its inherent multiresolutional design. Our motion estimation/compensation and backward/forward mode optimization involve only coarser levels of both the current and reference frames. Motion compensated residue coding is a simple coefficient scan estimation-quantization process, and is interleaved with motion compensation at each level. Therefore, to decode the current frame at a coarser resolution, only the reconstructed previous frame at that same resolution is needed. As a result, the "drift" problem does not exist in our coder [7], [8].
- In contrast to conventional block-DCT based coders, our coder is free from blocky artifacts. There are two reasons to account for this. First, our backward motion estimation makes it feasible to employ a dense motion field, which is more flexible in motion representation. Second, the motion compensated residue signal is quantized in the wavelet transform domain, as opposed to the block-DCT domain. Both stages are relaxed from the rigid block constraints and therefore the overall coder is free from blocky artifacts.
- The backward/forward hybrid mode for motion compensation in our coder provides an adaptive bit allocation strategy between motion representation and residue coding. Conventional coders are severely restricted in this capability as they typically partition the motion field into a fixed number of piecewise (usually blockwise) constant motion subfields, and send forward motion parameters for each subfield. The proposed coder, however, can adaptively regulate its motion bits investment according to the motion complexity in each frame, by adjusting the relative frequency with which forward modes are used. As a consequence, our coder is capable of superior performance for a wider range of input sequences and desired bit rates, without its inherent structure needing to be modified.

The remainder of the paper are organized as follows. In Section II, we first analyze the aliasing problem in performing motion estimation in the wavelet domain, and derive a suitable optimization criterion for our interpolation filter design. Section III then presents our basic video coding framework in purely backward mode. In Section IV, we propose as a solution to the backward instability problem our hybrid backward/forward motion compensation scheme, and describe in detail the dynamic programming optimization algorithm. Section V discusses the computational complexity issues. Experimental results are given in Section VI. Finally, some conclusions and discussions for future research are given in Section VII.

II. Basic Derivation

We start by examining the possibility of estimating motion in the wavelet domain through appropriate processing of transformed signals. Consider the generic subband decomposition

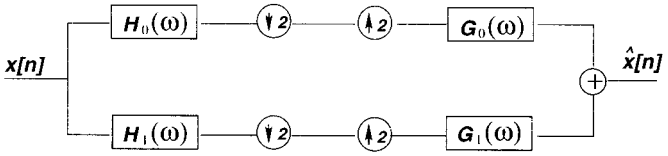


Fig. 1. Generic subband filtering system.

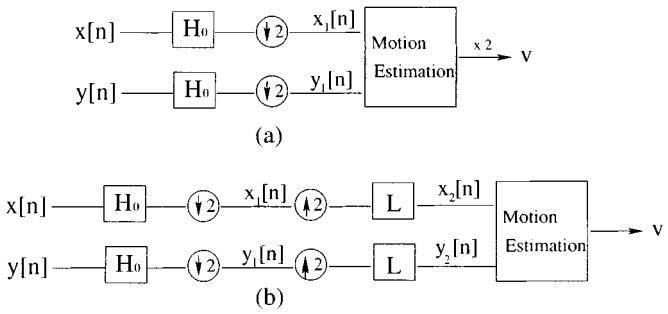


Fig. 2. Two different schemes for estimating the motion vector v from the lowpass signals in the wavelet domain.

system in Fig. 1. For simplicity, let us denote the current and reference video frames by the one-dimensional (1-D) notations $x[n]$, $y[n]$, respectively. The discussion here can be generalized trivially to two dimensions through separable construction of two-dimensional (2-D) filters. Assume that there is a motion v between $x[n]$ and $y[n]$, $y[n] = x[n-v]$. Further, let $x_1[n]$ and $y_1[n]$ denote their transformed signals in the lowpass band of the wavelet decomposition (see Fig. 2). Our coder operates in a coarse-to-fine fashion, in which $x_1[n]$ and $y_1[n]$ are coded first. The purpose here is to find a close estimate of the motion vector v from the coded $x_1[n]$ and $y_1[n]$. (We do not consider quantization effects for the time being.)

Straightforwardly, one could perform motion estimation directly between $x_1[n]$ and $y_1[n]$ to obtain a motion vector v_0, and scale it by 2 to get v, as shown in Fig. 2(a). However, the aliasing noise that enter $x_1[n]$ and $y_1[n]$ at downsampling make this direct estimation inaccurate. Let us illustrate this in the spectral domain. We have,[1]

$$X_1(\omega) = \frac{1}{2}\left(H_0\left(\frac{\omega}{2}\right)X\left(\frac{\omega}{2}\right) + H_0\left(\frac{\omega}{2}+\pi\right)X\left(\frac{\omega}{2}+\pi\right)\right) \tag{1}$$

and

$$Y_1(\omega) = \frac{1}{2}\left(H_0\left(\frac{\omega}{2}\right)Y\left(\frac{\omega}{2}\right) + H_0\left(\frac{\omega}{2}+\pi\right)Y\left(\frac{\omega}{2}+\pi\right)\right). \tag{2}$$

Substitution of $Y(\omega) = X(\omega)e^{-j\omega v}$ into (2) leads to

$$\begin{aligned} Y_1(\omega) = \frac{1}{2}\Big(&H_0\left(\frac{\omega}{2}\right)X\left(\frac{\omega}{2}\right)e^{-j(\omega/2)v} \\ &+ H_0\left(\frac{\omega}{2}+\pi\right)X\left(\frac{\omega}{2}+\pi\right)e^{-j((\omega/2)+\pi)v}\Big). \end{aligned} \tag{3}$$

Both $X_1(\omega)$ and $Y_1(\omega)$ contain two parts: The original signal filtered by $H_0(\omega/2)$ and the aliasing signal filtered by $H_0((\omega/2)+\pi)$. Now, consider the ideal motion vector $v_0 = v/2$. Subtracting (1) $\cdot e^{-j\omega(v/2)}$ from (3), we can see that for this candidate motion vector, the signal parts in $X_1(\omega)$ and $Y_1(\omega)$ are exactly matched, but the aliasing parts are not. In other words, even for perfect estimation, we still have a nonnegligible matching error coming from the aliasing noise. Therefore this method can not be very accurate.

A. Aliasing Reduction Using an Interpolation Filter

To improve the motion estimation accuracy, let us upsample $x_1[n]$, $y_1[n]$ by 2 and filter them by an (usually lowpass) interpolation filter $L(\omega)$ to reduce some aliasing noise. We then estimate v from the interpolated signals $x_2[n]$, $y_2[n]$, as shown in Fig. 2(b). This approach is expected to perform better, since the detrimental aliasing noise are reduced. However, the choice of the interpolation filter $L(\omega)$ is of crucial importance here. While some filters may do well in removing the aliasing noise, they may also simultaneously weaken or distort the signal components in $x_2[n]$ and $y_2[n]$, and therefore destroy their inherent motion relationships. This is undesirable since our motion estimation is based on these signal components, and requires that they are preserved faithfully. Therefore, in the design of $L(\omega)$, we have two optimization goals: one is *aliasing reduction*, the other is *signal preservation*. An ideal $L(\omega)$ should seek a reasonable balance between these two conflicting optimization goals.

Now, let us find a quantitative representation of our two optimization purposes. First, we have

$$X_2(\omega) = \tfrac{1}{2}(H_0(\omega)X(\omega) + H_0(\omega+\pi)X(\omega+\pi))L(\omega). \tag{4}$$

Our signal preservation criterion, denoted S, requires that the first part of $X_2(\omega)$ equals $X(\omega)$. We use the mean squared error as a measure of deviation from this ideal

$$S = \int_{-\pi}^{\pi} \left|\left(1 - \tfrac{1}{2}H_0(\omega)L(\omega)\right)X(\omega)\right|^2 d\omega. \tag{5}$$

On the other hand, our aliasing reduction criterion, denoted T, says that the second part of $X_2(\omega)$ should be zero. Therefore

$$T = \int_{-\pi}^{\pi} |H_0(\omega)L(\omega+\pi)X(\omega)|^2 d\omega. \tag{6}$$

Combining these two minimization criteria with a weighting coefficient μ, we have the following cost function

$$\begin{aligned} F(L;\mu) &= S + \mu T \\ &= \int_{-\pi}^{\pi} (|2 - H_0(\omega)L(\omega)|^2 \\ &\quad + \mu|H_0(\omega)L(\omega+\pi)|^2)|X(\omega)|^2 d\omega. \end{aligned} \tag{7}$$

For each given input signal $x[n]$, minimizing (7) gives the optimal interpolation filter $L(\omega)$. However, the problem of (7) is that it involves the input spectrum $X(\omega)$, which is often hard to evaluate in practice. Here, let us utilize a stochastic input model; we assume that $x[n]$ is a wide sense stationary (WSS) random process, and seek to minimize the expectation of (7). In this case

[1]Throughout this paper, we use capital letters to denote the Fourier Transforms of the corresponding time signals

we simply substitute $|X(\omega)|^2$ in (7) by its expectation, the input power spectral density $S_x(\omega)$. Our cost function now is

$$\begin{aligned}\boldsymbol{E}_x(F(L;\,\mu)) &= \boldsymbol{E}_x(S+\mu T)\\ &= \int_{-\pi}^{\pi}(|2-H_0(\omega)L(\omega)|^2\\ &\quad +\mu|H_0(\omega)L(\omega+\pi)|^2)S_x(\omega)\,d\omega. \quad (8)\end{aligned}$$

Here, the weighting coefficient μ represents a tradeoff between our two optimization purposes. Obviously, a larger μ favors aliasing reduction, while a smaller μ favors signal preservation. In our implementation, we find μ by training on the first few frames of the sequence. For a selected range of μ values, and a corresponding set of optimal interpolation filters, we apply our backward coding scheme as described in Section III to choose the μ that gives the best performance. This μ^* is then sent as side information.[2]

B. Optimal Solution

Our optimization criterion in (8) is a quadratic form with respect to $L(\omega)$. The optimal filter $L^*(\omega)$ can be obtained by setting its variation to zero

$$L^*(\omega)=\frac{2S_x(\omega)H_0^*(\omega)}{S_x(\omega)|H_0(\omega)|^2+\mu S_x(\omega+\pi)|H_0(\omega+\pi)|^2}. \quad (9)$$

This solution is an infinite impulse response (IIR) filter. In practice, however, FIR filters are often preferred. The optimal FIR solution to (8) under a given length constraint can also be found in closed-form. Let us first rewrite our optimization criterion in (8) in the time-domain. We define the convolution matrix $\boldsymbol{C}(h)$ for a FIR filter h as shown in (10) at the bottom of the page.

Using $\boldsymbol{C}(h)$, filtering any sequence $x[n]$ by $h[n]$ is $\boldsymbol{C}(h)x$. Equation (8) is then rewritten as

$$\begin{aligned}F(l,\,\mu) &= [2\delta-\boldsymbol{C}(h_0)l]'\boldsymbol{R_x}[2\delta-\boldsymbol{C}(h_0)l]\\ &\quad +\mu l'\boldsymbol{C}'(h_0^+)\boldsymbol{R_x^+}\boldsymbol{C}(h_0^+)l \quad (11)\end{aligned}$$

where $\boldsymbol{R_x}$ is the correlation matrix of the input $x[n]$ and $\delta[n]$ is the unit impulse sequence. $h_0^+[n]=(-1)^n h_0[n]$, and $(\boldsymbol{R_x^+})_{ij}=(-1)^{i+j}(\boldsymbol{R_x})_{ij}$. Setting to zero the derivative with respect to l, we get

$$l^*=[\boldsymbol{C}'(h_0)\boldsymbol{R_x}\boldsymbol{C}(h_0)+\mu\boldsymbol{C}'(h_0^+)\boldsymbol{R_x^+}\boldsymbol{C}(h_0^+)]^{-1}2\boldsymbol{C}'(h_0)\boldsymbol{R_x}\delta. \quad (12)$$

The above optimization has been carried out to design an interpolation filter $L(\omega)$ for the well known Daubechies 9-7 filter bank [16], in which $H_0(\omega)$ and $G_0(\omega)$ are both symmetric filters, having lengths 9 and 7, respectively. The input signal is modeled as a WSS AR(1) process, with a high correlation coefficient $\rho=0.95$ [17]. In this case, $(R_x)_{ij}=\rho^{|i-j|}$ and $S_x(\omega)=(1-\rho^2)/(|1-\rho e^{-j\omega}|^2)$. Fig. 3 compares the magnitude frequency responses of $H_0(\omega)$, $G_0(\omega)$, the optimal IIR filter $L^*(\omega)$, and the optimized FIR filters of lengths five, seven, nine, and 11, respectively. Table I lists their filter taps. It can be seen that for lengths larger than nine, the advantage obtainable by using longer filters is very small. To keep a low computational complexity, we choose to use the optimized FIR filter of length nine.

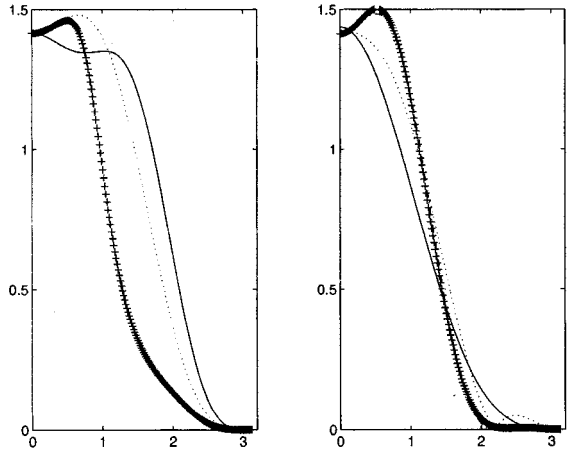


Fig. 3. Various filter responses in our design of $L(\omega)$, for Daubechies 9-7 filter bank and AR(1) input process with $\rho=0.95$. On the left, the solid curve is $H_0(\omega)$, "..." is $G_0(\omega)$, and "+" is the optimal IIR filter given by (9). On the right, the solid, "...," "-.," and "+" curves are our optimized FIR filters at lengths five, seven, nine, and 11, respectively. Note that the last two curves are almost overlapped.

III. Basic System Structure

Fig. 5 illustrates the basic system structure of our video coder using the optimized interpolation filter $L(\omega)$. Here a three level wavelet transform is performed on each video frame, resulting in 10 subbands, as shown in Fig. 4. Coding starts from the lowest

[2]Further, the following loose analytical guide is used in the training: our μ should result in an $L^*(\omega)$ that gives smaller values for both S and T than $G_0(\omega)$, the original synthesis lowpass filter. The motivation is that since $G_0(\omega)$ is part of the perfect reconstruction filter bank, it should be in general a reasonably good choice for $L(\omega)$. By enforcing smaller values on both criteria, we guarantee an improvement over $G_0(\omega)$.

$$\boldsymbol{C}(h)=\begin{pmatrix}\cdot & \cdot & & \cdot & \cdot & \cdot\\ \cdot & \cdot & & \cdot & \cdot & \cdot\\ \cdot & \cdot & & \cdot & \cdot & \cdot\\ h[N-1] & h[N-2] & \cdots & h[0] & 0 & 0\\ 0 & h[N-1] & h[N-2] & \cdots & h[0] & 0\\ 0 & 0 & h[N-1] & h[N-2] & \cdots & h[0]\\ \cdot & \cdot & \cdot & \cdot & & \cdot\\ \cdot & \cdot & \cdot & \cdot & & \cdot\\ \cdot & \cdot & \cdot & \cdot & & \cdot\end{pmatrix} \quad (10)$$

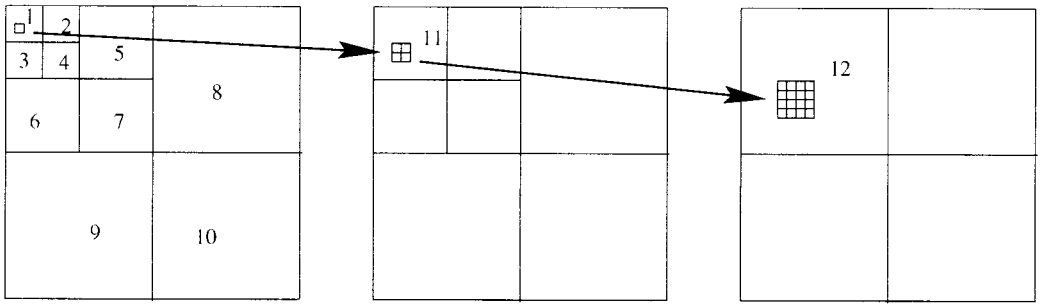


Fig. 4. The three-level wavelet transform and the parent–children relationships defined in our quadtree structure. Each node represents a spatial area of a fixed size in the LL band of that level. The four nodes that cover the same spatial area in the next LL band are defined as its children nodes.

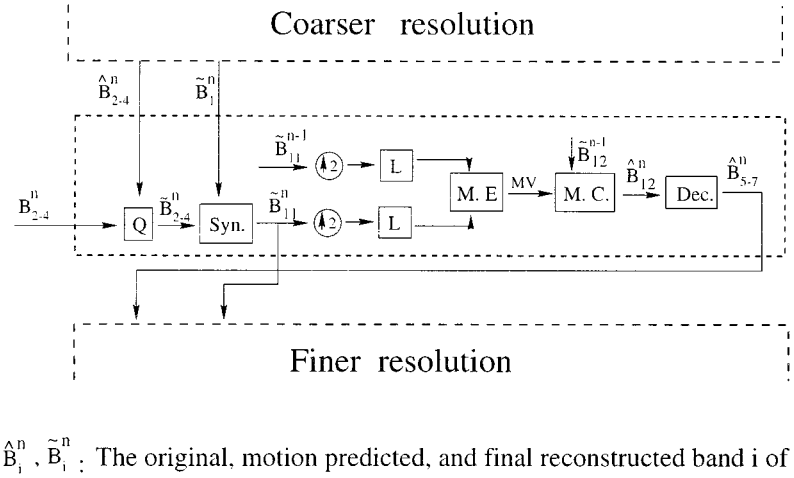


Fig. 5. The block diagram of our basic video coding system. For a labeling of subbands, refer to Fig. 4. The middle level shows how a motion predicted band 12 is found, and decomposed to code the prediction errors in bands 5–7. The coder iterates among quantization, synthesis, motion estimation (M. E), motion compensation (M. C), and decomposition.

TABLE I
COEFFICIENTS OF THE VARIOUS FILTERS IN OUR INTERPOLATION FILTER DESIGN. ONLY THE SECOND HALF OF THE COEFFICIENTS ARE LISTED AS THE FILTERS ARE ODD SYMMETRIC

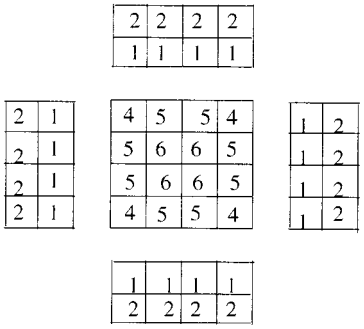
h_0	0.8527	0.3774	-0.1106	-0.0238	0.0378	
g_0	0.7885	0.4181	-0.0407	-0.0645		
l^*	0.6167	0.4269	0.0909	-0.0727	-0.0594	-0.0074
l^5	0.5521	0.3596	0.0827			
l^7	0.6005	0.4044	0.0568	-0.0493		
l^9	0.6151	0.4247	0.0889	-0.0713	-0.0438	
l^{11}	0.6141	0.4251	0.0924	-0.0656	-0.0469	-0.0067

resolution level and successively operates on each finer resolution. At each level, both motion estimation and compensation are performed on the LL band.[3]

Band 1 (see Fig. 4), is the first to be coded. Because of its small size, we simply use frame difference coding. Then at each resolution level, the following iterative steps are performed.

1) Interpolate and filter by $L(\omega)$ the reconstructed lowpass bands of both the previous and current frames.
2) Perform motion estimation on these interpolated and filtered lowpass bands, to obtain the motion vectors for this level. Note that as this step is purely backward, no motion information needs to be transmitted. Moreover, the backward nature makes it possible to use a dense motion field. Ideally, one would prefer pixelwise motion vectors, which can be estimated through such advanced motion analysis techniques as proposed in [18]–[22]. In our coder, we trade off motion accuracy with computational complexity by using block matching with a very small block size of 4×4.
3) Apply this set of motion vectors on the (reconstructed) LL band at the next level of the previous frame, to obtain the motion predicted next LL band, and decompose this band to obtain the predicted highpass bands at the current resolution. To reduce the spurious high frequency coefficients in the decomposition caused by blocky artifacts, we use overlapped block motion compensation (OBMC) [23], which uses the same small block size of 4×4 as in motion estimation, and a weighting window derived by averaging the 8×8 OBMC window in H.263 [11]. Fig. 6 shows the values of our weighting window.
4) Quantize and entropy code the prediction differences in the high bands. Here we use a modified version of the adaptive estimation-quantization (EQ) still image coder

[3]This is in contrast to [15], in which motion fields are estimated from the basebands, but are used to directly compensate the high frequency bands. Since quantization noise is usually high frequency, we believe that baseband compensation is more robust against quantization noise.



Relative weighting of the contribution of the motion vectors of the center, top, bottom, left, right blocks to the prediction of pixel values in the center block

Fig. 6. The weighting window used in our 4×4 overlapped block motion compensation. All numbers are normalized by $1/8$.

[6], whose basic idea is to model wavelet coefficients as generalized Gaussian distribution (GGD) fields, with slowly varying variances estimated from local neighborhood, and apply optimal adaptive quantization/entropy coding according to the estimated variance. The EQ coder is best credited for its fast speed, primarily due to the fact that the optimal GGD quantization parameters at different variances and bitrates are computed off-line and stored as lookup tables, therefore real-time coding consists of a simple and fast table look-up using the estimated variance. In our framework, we use the EQ coder for residue coding. Our statistical studies on motion predicted wavelet coefficient errors reveal that the GGD model still fits very well. In addition, the EQ coder can be modified in a way that each residue frequency band is coded completely independently of other bands, with little loss in coding efficiency, making it particularly suitable for our hierarchical motion compensation framework.

5) Run one stage of the synthesis operation on the reconstructed lowpass and highpass bands to form the reconstructed baseband of the next finer resolution level.

The entire process iterates until the multiresolution pyramid is exhausted. Fig. 7 illustrates graphically the basic operations involved between the current level and its coarser level.

IV. Backward/Forward Hybrid Motion Compensation

The coder as described above is purely backward. It performs reasonably well at moderate to high bit rates. Experiments have revealed a degraded performance at low bit rates and very complicated motion. The reason is that our backward motion estimation uses quantized coarser resolution frames, therefore its accuracy is dependent on the reconstruction quality of coarser frames. At low bitrates, increased quantization noise has a negative effect on the backward motion estimation accuracy. This exacerbates the reconstruction error, which further degrades the motion estimation accuracy and reconstruction quality at the next resolution level, and so on. In other words, in a backward framework, the mutual dependency between motion estimation and residual quantization forms a "positive feedback loop." In order to further attack this instability problem, we propose a novel backward/forward hybrid motion compensation strategy. The basic idea is to judiciously use forward information to help enhance the performance of purely backward motion prediction. In this case, we have a tradeoff between the accuracy of motion prediction, and the amount of motion information to send. The tradeoff is resolved in a R-D sense using a Lagrangian cost function [24] defined as

$$L(\lambda_m) = D + \lambda_m R. \tag{13}$$

Note that the distortion here is the motion compensated error energy, not the final coded distortion. Therefore we have chosen to use a Lagrangian parameter λ_m that is different from the parameter λ used in residue coding [6]. The choice of λ_m will be discussed shortly.

Our backward/forward mode selections are made in a tree structured fashion using dynamic programming, and a description follows.

A. Zerotrees of Mode Selections

We first construct a quadtree with each node representing a certain spatial area at a certain level of the multiresolution pyramid, and parent–children relationships among the tree nodes according to their spatial locations, as illustrated in Fig. 4. Each node of the tree can possibly be compensated by two motion vectors: 1) a backward motion vector estimated from its parent node, which corresponds to the same spatial location in the lower resolution frame and 2) a forward motion vector estimated directly from the same node. Usually, the backward motion vector is a good approximation to the forward motion vector, therefore a much smaller search range can be used for the forward motion estimation. Furthermore, in the case when a forward decision is made, the forward motion vector is differentially coded based on the backward motion vector. The backward/forward binary decisions at all the tree nodes are jointly optimized using the R-D criterion (13).

Two properties of the binary decision fields are observed in our experiments. They are both consequences of the typically high smoothness of motion fields, and are utilized to greatly facilitate our optimization. First, it is observed that backward decisions are highly dominant. In other words, with high probability the backward motion vector gives a sufficiently good prediction. Second, mode selections demonstrate highly tree-structured correlations, i.e., a backward decision at a certain node is most likely followed by backward selections at all its descendant nodes. These properties are similar to those of the significance maps of wavelet coefficients in still image coding. Therefore, we are motivated to borrow the zerotree concept from [3], and introduce for each tree node, the following three symbols regarding the mode choices of each node and all its descendant nodes (see Fig. 8).

- **ZR**: Node and all its descendants are determined to use backward mode.
- **IZ**: Node uses backward mode, but some of its descendants use forward mode.
- **NZ**: Node uses forward mode. (Nothing is assumed about its descendants).

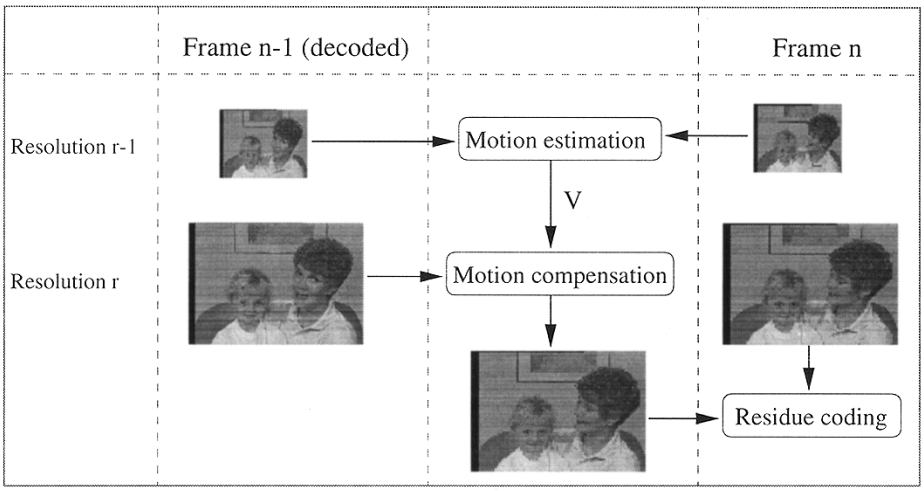


Fig. 7. The basic operations of our system in backward mode. The lower resolution signals are first interpolated and filtered for backward motion estimation. The estimated motion vectors are applied to the previous finer resolution signal, to find the motion predicted finer resolution, which is then decomposed to code the high-frequency bands.

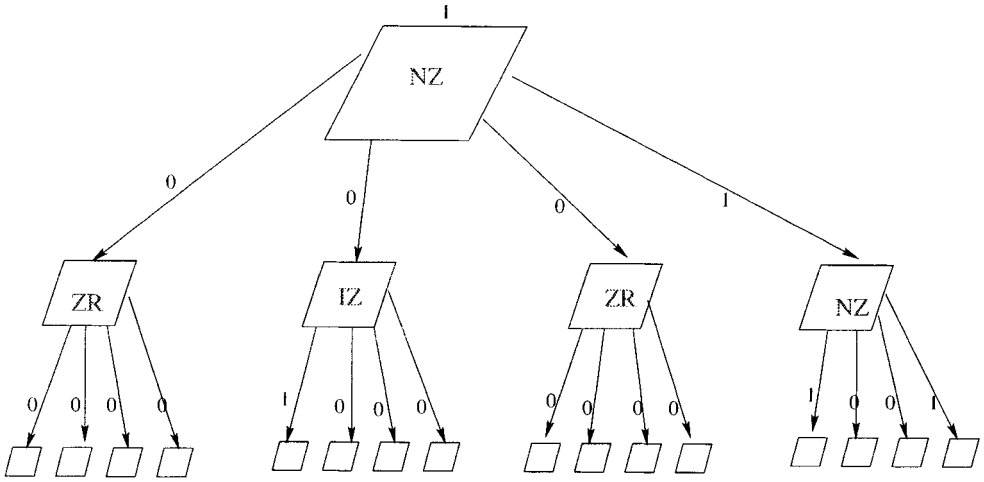


Fig. 8. The definition of tree symbols ZR, IZ, and NZ in our coder. A "0" represents a backward mode and a "1" represents a forward mode.

B. Mode Optimization

The algorithm starts by initializing all the tree nodes to backward mode, or the root nodes to ZR. When the coder is operating at level k, all the modes in that level are optimized. Ideally, to achieve global optimality, the mode at any node should be optimized jointly with its ancestor nodes. However, our backward motion estimation is based on reconstructed coarser signals, which again depend on the mode choices of ancestor nodes. Each different combination of ancestor modes would result in different reconstructed coarser signals, therefore requiring a different backward motion estimation. As is commonly known, motion estimation is the most computationally intensive part in video coding; thus, such a joint optimization with each iteration containing a motion estimation procedure would cost extraordinarily high complexity. Moreover, our experiments confirm that the advantage of this joint optimization is marginal over an alternative greedy algorithm, in which once a mode is chosen for a certain node, that selection always remains valid (i.e., it is never changed) in descendant mode optimization, and therefore backward motion estimation needs only to be performed once for each resolution level. To maintain a moderate complexity, we choose to implement the greedy algorithm. As a consequence, the only possible tree symbol change at an ancestor node is from ZR to IZ, corresponding to the case that the ancestor node uses backward mode, and a node at the current level switches from its initial backward mode to a forward mode.

Mode selection at each tree node is performed as comparing the R-D Lagrangian costs associated with both the backward and forward modes and selecting the one with the smaller cost. The Lagrangian cost for the backward mode is simply equal to the backward motion compensation error energy, because there is no motion information needing to be sent. To compute the Lagrangian cost for the forward mode, extra bits for sending the forward motion vector and the modified tree symbols for the ancestor nodes have to be counted. The optimization is implemented as a fast and efficient bottom-up dynamic programming strategy, in which for each node in the tree, the maximum reduction in Lagrangian cost obtainable by optimizing the subtree rooted at that node and up to level k is computed and stored. The computation is cumulative in the sense that for each inter-

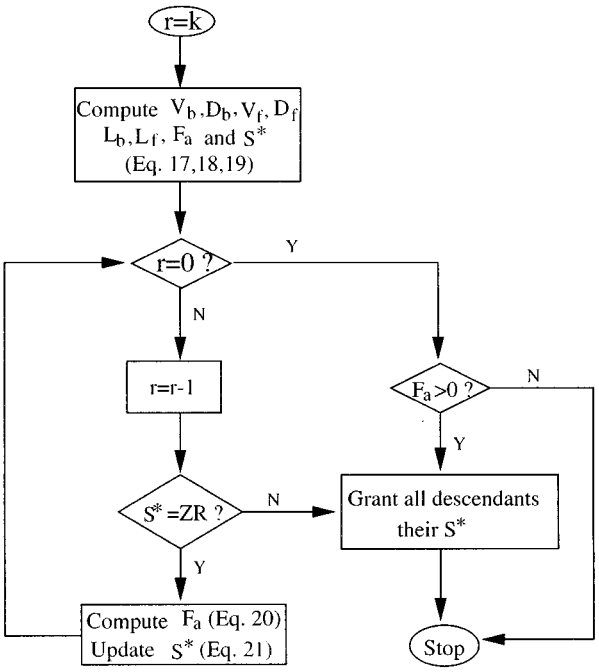


Fig. 9. The flowchart of our dynamic programming algorithm in optimizing backward/forward mode selections. All nodes are initialized to backward mode and their aggregated Lagrangian gains are initialized to 0. For each resolution level r, the algorithm is repeated for all nodes in that level.

mediate node, the maximum gain is the summation of those of its four children nodes, subtracted by the possible extra cost of transmitting the updated tree symbol at that node. The algorithm traces back from level k toward the root node, at which point a final decision is made.

In what follows, let us denote by n_i^l the ith node at level l with a tree symbol $S(n_i^l)$. For each n_i^l, we denote its backward motion vector by $v_b(n_i^l)$ and forward motion vector by $v_f(n_i^l)$, and their associated motion compensated error energies by $D_b(n_i^l)$ and $D_f(n_i^l)$, respectively. Furthermore, $C(n_i^l)$ represents the set of all its children nodes at level $l+1$. We start by defining the "aggregated" Lagrangian gain $F_a(n_i^l)$ at n_i^l as the reduction in total Lagrangian cost [compared to the default choice of $S(n_i^l) = \text{ZR}$] in the subtree rooted at n_i^l when an optimal $S(n_i^l)$ is used, under the assumption that all its descendant nodes up to level k are already optimized.

C. Dynamic Programming Algorithm

This subsection explains in detail our tree pruning algorithm using dynamic programming. The algorithm is plotted in Fig. 9.

1) Initializing the Current Level: At the start, we perform for each node n_i^k at the current level k, a backward motion estimation on the interpolated, filtered coarser signals, to obtain $v_b(n_i^k)$, $D_b(n_i^k)$, and a forward motion estimation on the current level, to obtain $v_f(n_i^k)$, $D_f(n_i^k)$, respectively. We then compute both the backward and forward Lagrangian costs for this node, $L_b(n_i^k)$ and $L_f(n_i^k)$ as

$$\begin{aligned} L_b(n_i^k) &= D_b(n_i^k) + \lambda_m l(\text{ZR}) \\ L_f(n_i^k) &= D_f(n_i^k) + \lambda_m [l(\text{NZ}) + 4l(\text{ZR}) \\ &\quad + l(v_f(n_i^k) - v_b(n_i^k))]. \end{aligned} \tag{14}$$

Here $l(.)$ denotes the codeword length for sending the enclosed (tree or motion) symbol. Note that in the forward case, in addition to differentially coding the forward motion vector, we need to send l(NZ) bits for the current node to indicate the forward mode choice, and l(ZR) bits for each of its four children, which, together with their descendants, are still defaulted to be in backward mode.

The initial aggregated Lagrangian gain $F_a(n_i^k)$ is computed by taking the difference of the backward and forward Lagrangian costs and thresholding by zero. The reason for zero-thresholding is that $F_a(n_i^k)$ is defined as the maximum Lagrangian gain one can achieve by considering a mode change at n_i^k. The gain is zero if $S(n_i^k)$ remains to be ZR. Therefore, the optimal gain $F_a(n_i^k)$ must be always greater than or equal to zero

$$F_a(n_i^k) = \max\left\{0,\, L_b(n_i^k) - L_f(n_i^k)\right\}. \tag{15}$$

Finally, we set the optimal tree symbol at level k according to whether $F_a(n_i^k)$ is larger than 0

$$S^*(n_i^k) = \begin{cases} \text{NZ}, & \text{if } F_a(n_i^k) > 0 \\ \text{ZR}, & \text{otherwise.} \end{cases} \tag{16}$$

2) Tracing Back the Tree: At the second stage, we trace back from level k toward the root node. For each intermediate node n_i^l along the tree path, we first examine whether $S(n_i^l)$ is ZR. If it is not ZR, we can immediately grant all its descendants up to level k their optimal tree symbols. If $S(n_i^l) = \text{ZR}$, however, we compute the aggregated Lagrangian gain $F_a(n_i^l)$ as the sum of those from all its children nodes, with the increased cost of sending the IZ symbol deducted, and thresholded by 0

$$F_a(n_i^l) = \max\left\{0,\, \sum_{N \in C(n_i^l)} F_a(N) - \lambda_m(l(\text{IZ}) - l(\text{ZR}))\right\}. \tag{17}$$

The optimal symbol for n_i^l is set as

$$S^*(n_i^l) = \begin{cases} \text{IZ}, & \text{if } F_a(n_i^l) > 0 \\ \text{ZR}, & \text{otherwise.} \end{cases} \tag{18}$$

3) Updating the Tree: Eventually when we reach the top level, each node n_i^0 at this level is a root node. Here the final decision is made according to whether $F_a(n_i^0)$ is greater than 0

if $F_a(n_i^0) > 0$ then
  grant n_i^0 and all its descendants their optimal symbols.
otherwise
  n_i^0 and all its descendants remain in their default (backward) states.

A toy example illustrating the optimization algorithm is shown in Fig. 10.

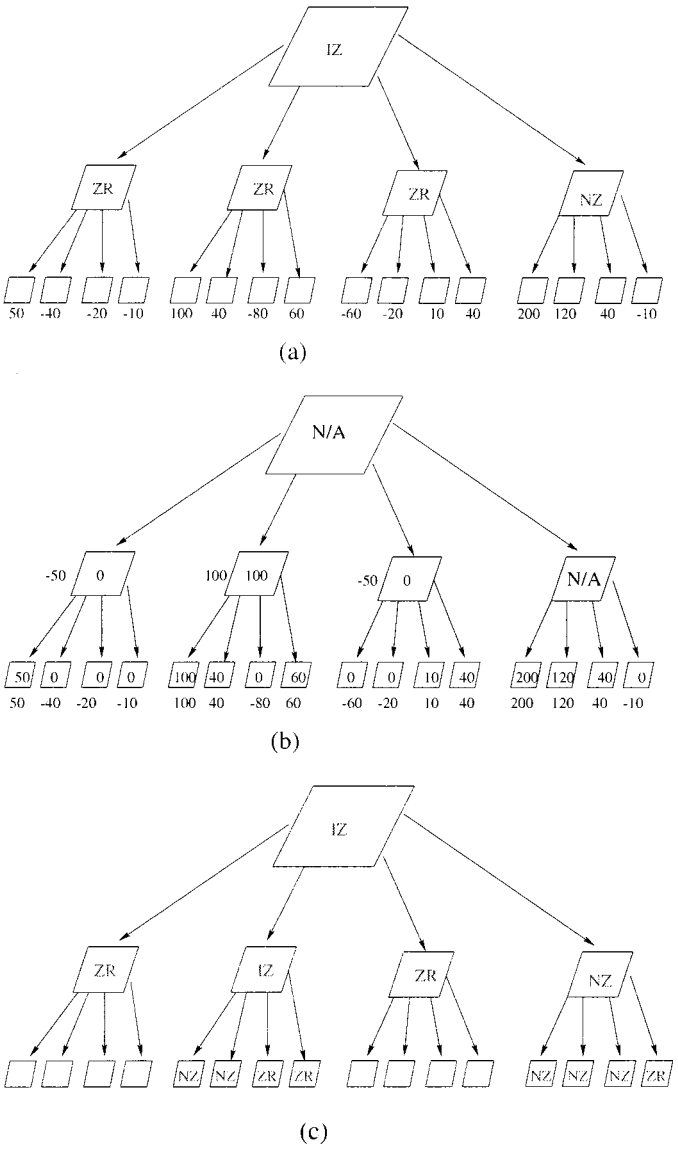


Fig. 10. A toy example of our mode optimization algorithm. Here the bottom level is the current coding level k. The numbers inside and outside each box represent the aggregated Lagrangian gain in that node, before and after zero thresholding, respectively. Letters ZR, NZ, and IZ are tree symbols, N/A stands for "not applicable": (a) the initial tree status when the backward and forward motion estimation have been performed, (b) the dynamic programming process to compute aggregated Lagrangian gain at each node, here the term $\lambda_m(l(\mathrm{IZ}) - l(\mathrm{ZR}))$ is assumed to be 100, and (c) the final optimized tree status.

D. Choice of λ_m

We now address the issues regarding how to choose a suitable λ_m for our quadtree optimization on mode selections. As has been pointed out, λ_m is different from the λ used in motion compensated residue coding, because the distortion here is the motion predicted error energy, not the final reconstructed error energy [28]. While minimizing the latter subject to a constraint on total bit rate is our ultimate coding goal, direct usage of this quantity would require a joint optimization of motion estimation and residue coding—a computationally infeasible task. Fortunately, there is a simple and efficient way to find a connection between these two Lagrangian parameters, as explained here.

TABLE II
AVERAGE VALUES OF RAW DIFFERENCE ENERGY, AND BACKWARD MOTION COMPENSATED ERROR ENERGY BY DIRECT ESTIMATION, INTERPOLATED ESTIMATION WITH $G_0(\omega)$, AND INTERPOLATED ESTIMATION WITH $L(\omega)$, RESPECTIVELY, ON EACH RESOLUTION LEVEL OF 100 FOOTBALL FRAMES

Resolution level	Raw diff.	Direct	$G_0(\omega)$	$L(\omega)$
1	2265.726	1100.2792	824.1337	658.6520
2	1003.985	469.7042	399.4755	305.0414
3	170.1889	101.0626	76.9035	62.2696

Let us denote by $D'(v)$ the motion predicted error energy associated with a certain candidate motion vector v, and $D(v)$ the final reconstruction error energy after residue coding. Further, $R_m(v)$ and $R_r(v)$ denote the number of bits spent on coding v and its residue signal, respectively. We seek to minimize $L(v, \lambda) = D(v) + \lambda(R_m(v) + R_r(v))$. However, since $D(v)$ and $R_r(v)$ are not directly available in motion estimation, we attempt to use an alternative cost function, $L'(v, \lambda_m) = D_m(v) + \lambda_m R_m(v)$ that gives us the same optimal v. To this end, we introduce the notion of "Lagrangian compression ratio" $C_l(\lambda, s)$ for residue coding, defined as a function of the input signal s, and the Lagrangian parameter λ

$$C_l(\lambda, s) = \frac{\|s\|^2}{\|s - \tilde{s}\|^2 + \lambda R(\tilde{s})}. \tag{19}$$

Where $\tilde{s}$ is the quantized signal and $R(\tilde{s})$ the number of bits to code s. Initially, before residue coding, we spend 0 bits on s, with a distortion of $\|s\|^2$, or a Lagrangian of $\|s\|^2$. After residue coding, we spend $R(\tilde{s})$ bits, and achieve a distortion of $\|s-\tilde{s}\|^2$. Therefore the above definition measures the Lagrangian gain due to residue coding.

An interesting and intuitively reasonable phenomenon we have observed through our experiments on motion compensated residue signals is that, the above defined Lagrangian compression ratio $C_l(\lambda, s)$ is almost solely a function of λ. Given a certain λ, $C_l(\lambda, s)$ achieves a constant value within a large range of residue signals s. Therefore we decide to ignore the dependency of $C_l(\lambda, s)$ on s and write it as $C_l(\lambda)$. This observation is the key to establishing a tie between λ_m and λ.

Using $C_l(\lambda)$, and substituting $D_m(v) = C_l(\lambda)(D(v) + \lambda R_r(v))$, we can rewrite $L(v, \lambda)$ as

$$L(v, \lambda) = L'(v, \lambda C_l(\lambda))/C_l(\lambda). \tag{20}$$

Now it is seen that $L(v, \lambda)$ is proportional to $L'(v, \lambda C_l(\lambda))$, Therefore to minimize $L(v, \lambda)$, we can equivalently minimize $L'(v, \lambda_m)$, with the substitution that $\lambda_m = \lambda C_l(\lambda)$. In our coder, we find $C_l(\lambda)$ by training, and send it as side information.

V. COMPUTATIONAL COMPLEXITY

This section addresses the computational complexity issues in our coder. A casual look at our previous description may give the impression of highly intensive computation, because of the multiple motion estimation processes needed. However, great computational savings can be achieved by taking advantage of the striking similarities between motion vectors in successive

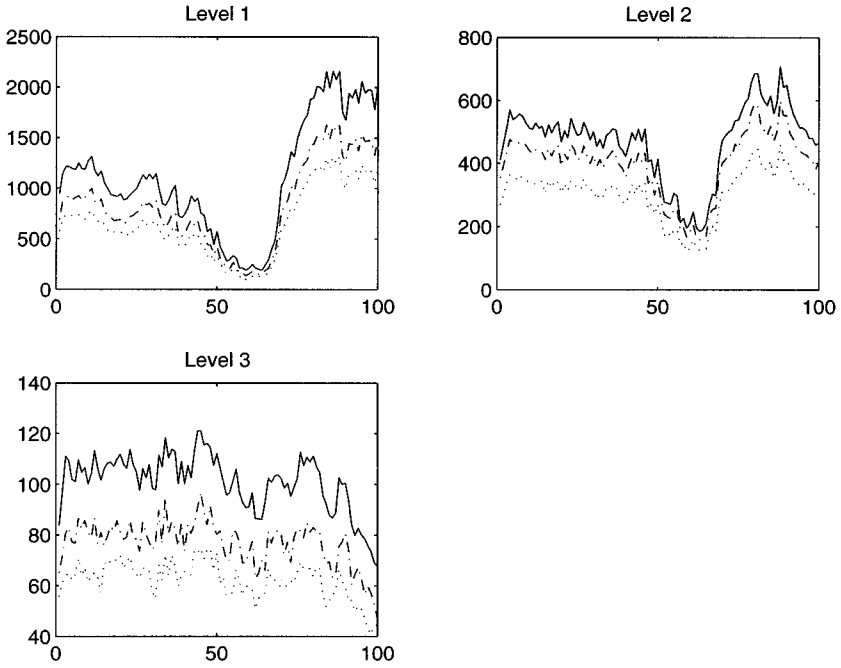


Fig. 11. The backward motion compensated error energy on 100 frames of the football sequence at three resolution levels. The three curves (solid, "-.," and "...") correspond to three different motion estimation schemes: direct estimation, interpolated estimation with $G_0(\omega)$, and interpolated estimation with $L(\omega)$, respectively.

resolution levels, and between the backward and forward motion vectors. Usually, the lower resolution motion vector can be used as a good initial estimate for its finer resolution counterpart, and so does the backward motion vector for its forward one. Therefore, typically, a much smaller neighborhood search is sufficient in our motion estimation. When log-scale search range sizes are used across resolution levels, the total computational complexity of our entire hierarchical backward motion estimation is equal to that of the well known three-step search method for block matching [25]. Our forward motion estimation does incur additional computation over this estimate. The increment is proportional to the square of the ratio between forward search range and backward search range, which is typically 20%–30%.

Another source of concern is the quadtree optimization algorithm in our backward/forward hybrid mode. As mentioned earlier, the optimization is greatly simplified by the use of a greedy search algorithm, in which the modes at each level are optimized when it is coded, and remain unchanged in the optimization of finer levels. The optimization at each level is implemented as dynamic programming, which involves accumulating the aggregated Lagrangian gain from that level up to the root level. At each node, only a simple summation over its four children nodes is needed. The total computation for the optimization is therefore of O(N) complexity (N is the total number of pixels), which is negligible compared to the O(N^2) complexity of motion estimation.

While the proposed coder saves complexity at the encoder, it requires an increase in decoder complexity, because the decoder has to repeat the backward motion estimation. This is a chief disadvantage of our coder. Nowadays, an increasing number

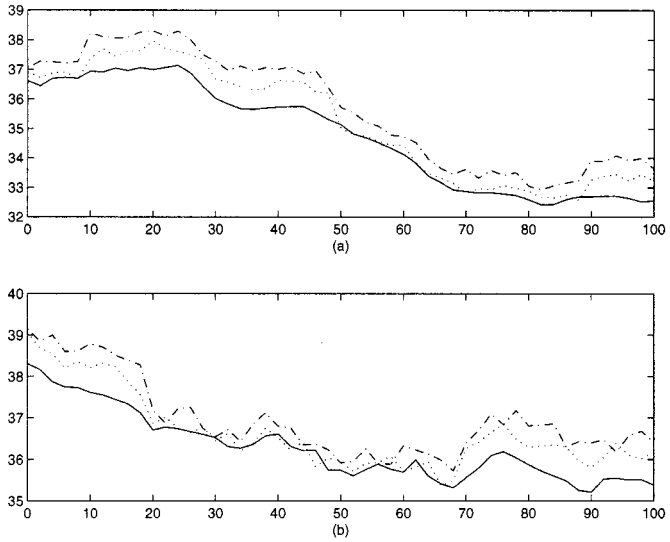


Fig. 12. Coding results. Final coded PSNR for luminance versus frame number. Here the solid, dotted, and dashed lines represent the results from H.263 with full option, our coder with the usage of $G_0(\omega)$, and our coder with the usage of $L(\omega)$, respectively. (a) MaD sequence at 48 kb/s and 15 frames/s and (b) Missa sequence at 24 kb/s and 15 frames/s.

of multimedia applications support symmetric complexity systems, with common software or hardware task modules installed in both the encoder and decoder (e.g., those for motion estimation). Improving the design of such modules will accelerate the computation at both ends. Our coder is better suited for such applications. We also envision that with the inevitable growth in computing power (the so-called Moore's law phenomenon), increased performance for future coding systems will depend on the ability to have more powerful decoders than exist today.

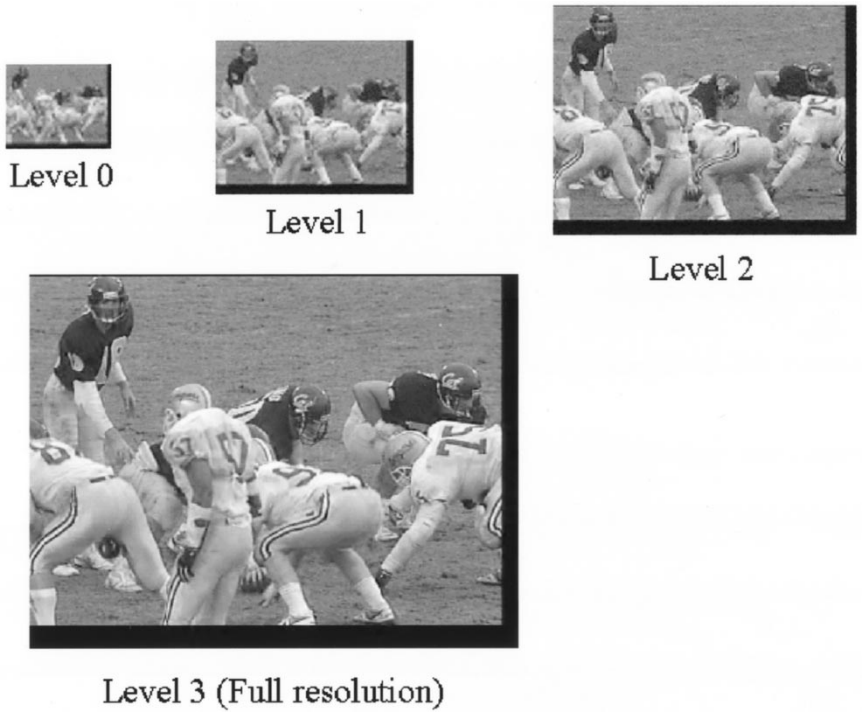


Fig. 13. Scalable decoding of a typical football frame at four resolution levels. The coding bit rate for full resolution is 0.5 Mb/s.

TABLE III
COMPARISON ON THE PSNR VALUES BETWEEN OUR CODER AND H.263

Sequence	bit rate (Kb/s)	frame rate	H.263	$G_0(\omega)$	$L(\omega)$
MaD	48	15	34.7673	35.1374	35.6359
Missa	24	15	36.2876	36.6164	36.9116

TABLE IV
COMPARISON ON THE PSNR VALUES BETWEEN OUR CODER AND MPEG-2 ON FOOTBALL AND 30 frames/s

bit rate (Mb/s)	MPEG-2	The proposed coder
0.5	25.1350	27.1676
2	30.0920	31.5567
7	37.1125	37.7186
10	40.5998	40.9770

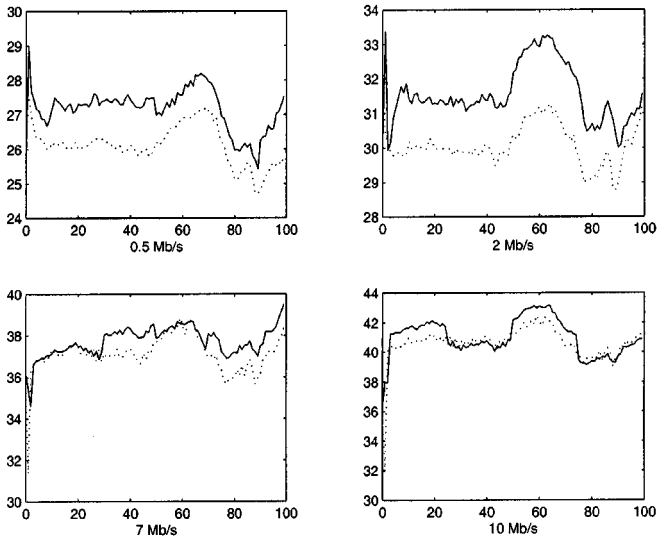


Fig. 14. Coding results on football. Final coded PSNR for luminance versus frame number at 30 frames/s, and different bit rates of 0.5, 2, 7, and 10 Mb/s. The solid and dotted lines represent the results from our proposed coder and MPEG-2 respectively.

VI. CODING RESULTS

The proposed video coder was implemented in software and tested on standard video testing sequences. The first experiment justified the utility of our optimized interpolation filter $L(\omega)$ in improving the performance of backward motion estimation. Here we compared the backward motion compensation error energy on each resolution level of the "football" sequence using three different motion estimation schemes: direct estimation, interpolated estimation using the synthesis lowpass filter $G_0(\omega)$, and interpolated estimation using $L(\omega)$. Fig. 11 gives the results on the first 100 frames of football. Table II lists the average values for raw difference energy and motion compensated error energy. As can be seen, interpolation by $L(\omega)$ gives us 30–40%

TABLE V
AVERAGE BITS/FRAME AND AVERAGE CODED PSNR VALUES OF OUR SCALABLE CODER ON FOOTBALL AT FOUR RESOLUTION LEVELS. THE CODING BIT RATE IS 0.5 Mb/s

Resolution level	Average bits/frame	Average PSNR
0	1781	18.8233
1	6024	21.0574
2	12265	24.5305
3	16703	28.5205

Fig. 15. Comparison of final coded subjective quality. On the left are the reconstructed frames from the standard video coders. On the right are the same reconstructed frames from our proposed coder. Top: MaD, compared with H.263 at 48 b/s. Middle: Missa, with H.263, 24 Kb/s. Bottom: football, with MPEG-2, 2 Mb/s.

less error energy than direct estimation, and 20–25% less than interpolation by $G_0(\omega)$.

Other experiments compare our coding efficiency with popular video coding standard simulation models, H.263, and MPEG-2 for low and high bit rates, respectively. In each test, a total of 100 frames of the original sequence were coded. Figs. 12 and 13 give the comparison results, in which the final coded PSNR for luminance is plotted against the frame number.

The first comparison addresses H.263. Here the input sequences were in CIF (352×288) resolution. Fig. 12(a) shows the results on "mother and daughter (MaD)" sequence coded at a fixed bit rate of 48 kb/s, and a frame rate of 15 frames/s, and Table III lists the numerical values. The solid curves represent the performance of H.263, while the dotted and dashed curves represent those from our proposed coder, with the interpolation filter chosen as $G_0(\omega)$ for the former, and the optimized filter $L(\omega)$ for the latter. Note that in generating the results of H.263, we have chosen to turn on all of the enhancing modes (except the PB-frame mode), i.e., the unrestricted motion vector mode, the advanced prediction mode, and the syntax-based arithmetic coding mode, in order to fully demonstrate the power of our coder. As can be seen, when the optimized filter $L(\omega)$ is used, our coder generally achieves 0.5–1.5 dB (average 0.87 dB) improvement in PSNR over H.263. Moreover, the advantage of using $L(\omega)$ over $G_0(\omega)$ can be clearly seen, as there is an average of 0.5 dB difference between the dotted and dashed curves. Fig. 12(b) shows the results on "Miss America (Missa)" sequence at 24 kb/s and 15 frames/s. Again we achieved 0.63

dB gain in average PSNR when using $L(\omega)$ and 0.33 dB gain when using $G_0(\omega)$.

The second comparison (Fig. 14) was with MPEG-2 on "football" sequence (352×240), at 30 frames/s, and various bit rates of 0.5, 2, 7, and 10 Mb/s. The solid curves are for our proposed coder (with $L(\omega)$), and the dotted curves for MPEG-2. Table IV gives their numeric PSNR results. As can be seen, our coder always outperforms MPEG-2, with however a much larger gain at lower bit rates, which can be attributed to the fact the savings in motion bits by backward motion compensation in our coder constitute a larger portion at lower bit rates.

As mentioned earlier, a scalable decoding is possible in our coder. Table V demonstrates the scalable feature for football at 0.5 Mb/s, in which the average bits/frame and the average PSNR values at the four resolution levels are listed.[4] Fig. 13 shows a typical reconstructed frame at different resolution levels.

Finally, Fig. 15 gives the comparison in subjective reconstruction quality corresponding to the above tests. On the left are the decoded frames from the standards, on the right are the same decoded frames from our coder. As is expected, our coder suffers from less blocky artifacts and hence leads to more subjectively pleasing reconstructed pictures. However, as is well known with wavelet coders, our reconstructed frames are contaminated by "mosquito" noise. We are currently researching techniques to reduce this noise through the use of shorter filters, which however result in less energy compaction—this tradeoff will be part of our future study.

VII. Conclusions and Future Research

We have introduced a multiresolutional video coding system based on performing motion estimation directly in the wavelet transform domain. Our coder alleviates the aliasing problem in motion estimation by upsampling and filtering the coarser signals using a specially designed interpolation filter. Further, in order to attack the instability problem caused by quantization noise inherent in a purely backward coder, we designed a novel backward/forward hybrid motion compensation framework using zerotree coding and dynamic programming. Experimental results showed up to 2 dB improvement in PSNR over the H.263 and MPEG-2 video coding standards, with also much better subjective quality. In addition, our coder is spatially scalable and allows for robust operation over a wider range of video sequences and bit rates. Ringing effects as a result of wavelet transform coding is a major problem that causes deterioration in subjective coding quality. We propose to address this problem in future research, as well as the issues related to object uncovering, occlusion, and camera zooming, etc.

References

[1] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[2] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[4]Here the PSNR values for each resolution level are computed with regard to the full-resolution original frame, in order to demonstrate the successive quality improvement nature of scalable coding. In other words, the PSNR of 18.8233 dB for level 0 is not that between the reconstructed level 0 and the original level 0, but the reconstructed level 0 and the original *full frame*.

[3] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3345–3463, Dec. 1993.

[4] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[5] Z. Xiong, K. Ramchandran, and M. T. Orchard, "Space-frequency quantization for wavelet image coding," *IEEE Trans. Image Processing*, vol. 6, pp. 677–693, May 1997.

[6] S. M. Lopresto, K. Ramchandran, and M. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1997.

[7] O. Werner, "Drift analysis and drift reduction for multiresolution hybrid video coding," EURASIP J. Image Commun., to be published.

[8] R. Mathew and J. F. Arnold, "Layered coding using bitstream decomposition with drift correction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 882–891, Dec. 1997.

[9] D. LeGall, "MPEG: A video compression standard for multimedia applications," *Commun. ACM*, vol. 34, pp. 46–58, Apr. 1991.

[10] "Video Codec of Audiovisual Services at kbits/s,", vol. XV-R 37-E, Aug. 1990.

[11] International Telecommun. Union and Telecommun. Standard. Sector, Study Group 15, "Draft Recommendation H.263,", Leidshendam, Doc. LBC-95-, Apr. 1995.

[12] K. Metin Uz and M. Vetterli, "Interpolative multiresolution coding of advanced television with compatible subchannels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 86–99, Mar. 1991.

[13] T. Naveen and J. Woods, "Motion compensated multiresolution transmission of high definition video," *IEEE Trans. Circuits Syst. Video Technol.*, Feb. 1994.

[14] K. Tsunashima, J. B. Stampleman, and V. M. Bove, "A scalable motion-compensated subband image coder," *IEEE Trans. Commun.*, vol. 42, pp. 1894–1901, Apr. 1994.

[15] A. Nosratinia and M. Orchard, "Multi-resolution backward video coding," in *IEEE Int. Conf. Image Processing*, vol. 2, Washington, DC, Oct. 1995, pp. 563–566.

[16] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," *IEEE Trans. Image Processing*, vol. 1, pp. 205–221, Apr. 1992.

[17] W. Mauersberger, "Generalize correlation model for designing 2-dimensional image coders," *Electron. Lett.*, vol. 15, pp. 664–665, 1979.

[18] H. Musmann, P. Pirsch, and H. Grallert, "Advances in picture coding," *Proc. IEEE*, Apr. 1985.

[19] S. N. Efstratiadis and A. K. Katsaggelos, "A model-based pel-recursive motion estimation algorithm," in *Proc. ICASSP*, vol. 4, Apr. 1990, pp. 1973–1976.

[20] Y. Nakaya and H. Harashima, "An iterative motion estimation method using triangular patches for motion compensation," *Proc. SPIE Visual Communication Image Processing*, vol. 1605, pp. 546–557, Nov. 1991.

[21] C. L. Huang and C. Y. Hsu, "A new motion compensation method for image sequence coding using hierarchical grid interpolation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, pp. 42–52, Feb. 1994.

[22] H. H. Nagel, "On the estimation of optical flow: Relations between different approaches and some new results," *Artif. Intell.*, vol. 33, pp. 299–324, 1987.

[23] M. Orchard and G. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach.," *IEEE Trans. Image Processing*, vol. 3, pp. 693–699, Sept. 1994.

[24] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.

[25] T. Koga *et al.*, "Motion compensating interframe coding for video conferencing," in *Proc. Nat. Telecommun. Conf.*, New Orleans, LA, Nov. 29, 1981, pp. G5.3.1–G5.3.5.

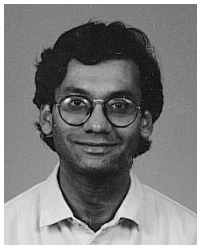
[26] A. Netravali and J. Robbins, "Motion compensated television coding—Part I," *Bell Syst. Tech. J.*, vol. 58, pp. 631–670, Mar. 1979.

[27] A. Gersho and R. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1995.

[28] S. Choi and J. Woods, "Three-dimensional subband/wavelet coding of video with motion compensation," in *Proc. SPIE VCIP97*, San Jose, CA, Feb. 1997.

Xuguang Yang (S'96–M'98) was born in Hunan, China, in 1971. He received the B.S. and M.S. degrees from the Department of Space Physics, Wuhan University, Wuhan, China, in 1989 and 1992, respectively. He received the M.C.S. degree from the Department of Computer Science and the Ph.D. degree from the Department of Electrical and Computer Engineering from the University of Illinois at Urbana-Champaign, both in May, 1998.

From January 1995 to May 1998, he was a Research Assistant with the Image Formation and Processing Group, Beckman Institute, University of Illinois at Urbana-Champaign. During the Summer of 1996, he was with PictureTel Inc., Boston, MA. He is currently a Member of Technical Staff in the Imaging Technologies Department, Hewlett-Packard Laboratories, Palo Alto, CA. His main research interests are in image and video processing, wavelets and filter banks, optimization theory, and combination algorithms.



Kannan Ramchandran (S'93–M'93) received the B.S. degree from the City College of New York, and the M.S. and Ph.D. degrees from Columbia University, New York, all in electrical engineering, in 1982, 1984, and 1993, respectively.

From 1984 to 1990, he was a Member of Technical Staff at AT&T Bell Labs, working in telecommunications R&D. From 1990 to 1993, he was a Graduate Research Assistant with the Center for Telecommunications Research at Columbia University. From 1993 to 1999, he was on the faculty of the Electrical and Computer Engineering Department, University of Illinois, Urbana-Champaign, and was affiliated with the Beckman Institute and the Coordinated Science Laboratory. Since the Fall of 1999, he has been an Associate Professor in the Electrical Engineering and Computer Science Department, University of California, Berkeley. His research interests include image and video compression and communications, multirate signal processing and wavelets, fast algorithms for signal and image processing, and unified algorithms for signal processing, communications, and networking.

Dr. Ramchandran was the recipient of the 1993 Elaihu I. Jury Award at Columbia University for the best doctoral dissertation in the area of systems, signal processing, or communications. He received an NSF Research Initiation Award in 1994, an Army Research Office Young Investigator Award in 1996, an NSF CAREER Award in 1997, and an Office of Naval Research Young Investigator Award in 1997. He was the co–recipient of the 1996 Senior Best Paper Award from the IEEE Signal Processing Society. He is a Member of the IEEE IMDSP Technical Committee, and serves as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING.