SCALABLE VERY LOW BIT-RATE VIDEO COMPRESSION USING MOTION COMPENSATED 3-D WAVELET DECOMPOSITION

Jo Yew Tham Surendra Ranganath Ashraf A. Kassim

Department of Electrical Engineering, National University of Singapore
10 Kent Ridge Crescent, Singapore 119260

Abstract— ***In this paper, we present a scalable wavelet-based video compression algorithm for very low bit-rate applications. First the frames within a group of frames (GOF) are motion-compensated with respect to a reference frame using a three-parameter motion model. Second, they are decomposed into hierarchically oriented subbands using a separable three-dimensional wavelet decomposition framework. A new tri-zerotree (TRI-ZTR) data structure is then introduced to encode the wavelet coefficients in a very economical manner via successive approximation. Our experimental results showed that this algorithm not only produces perceptually good quality reconstructed sequences at high compression, but also generates a fully embedded compressed bit stream that supports both resolution and rate scalability.***

I. Introduction

ONEmajor drawback to the explosive growth of multimedia technology is the problem of handling the voluminous digital data produced by these applications. For instance, a typical NTSC color video frame, with 720 pixels x 480 lines, 8 bits/pixel per color, and 30 frames/second will require a transmission capacity of 248 Mbps. Without any compression, a compact disc (CD) with a storage capacity of about 650 Mbytes can store only approximately 20 seconds of NTSC video. On the other hand it can hold about one hour of a 200:1 compressed video.

In response to the above situation, many compression algorithms using different transformation techniques, coding and quantization strategies, and motion estimation/compensation approaches have been investigated. More traditional transforms such as the discrete cosine transform (DCT), together with the application of block-based motion compensation (MC), were employed successfully in MPEG-1 and 2 for motion picture compression. However, their inherent blocking artifacts become objectionable at higher compression ratios.

Over the past few years, the use of the wavelet transform for compression has been gaining wide popularity. The main advantages of wavelet-based methods lie in their energy compaction property and multiresolution decomposition capability. Nevertheless, 3-D subband coding schemes still introduce distortions in the form of lost high-frequency details and motion blurring. In this new tri-zerotree (TRI-ZTR) coding framework, we propose a scalable, detail-preserving video codec that operates well at both low and very low bit-rates.

Section II first looks into how interframe motions are exploited via a three-parameter motion model. Section III introduces a new TRI-ZTR data structure that supports both multirate and multiresolution scalabilities. Section IV addresses that issue of video scalability. Section V presents some simulation results, and Section VI concludes this paper.

II. Motion Estimation and Compensation

Exploiting temporal redundancies is one of the biggest issues in video compression. Here we are concerned with how to improve the correlations among the frames within a predetermined group of frames (GOF) of F frames. The most commonly used motion estimation technique is the block-matching algorithm which is employed in MPEG video coding. Others such as pel-recursive, optical flow, and Bayesian methods have also been applied. In our TRI-ZTR video coding scheme, we use a three-parameter motion model (see also [3]) to estimate the motion vector fields (MVFs) of each frame f_n with respect to a reference frame f_0, where n = 1, 2, ..., F-1.

A. Three-Parameter Motion Model for Estimating MVFs

As far as video coding is concerned, the overhead incurred in transmitting the MVFs needs to be minimized. With this in mind, we attempt to estimate both global and local motions simultaneously. First, an input frame is divided into B blocks of size N x N each. We then estimate the vector $\mathbf{a} = [a_1, a_2, a_3]^T$, associated with each block as follows:

$$u = a_1 x + a_2, \quad (1)$$

$$v = a_1 y + a_3, \quad (2)$$

where (x, y) and (u, v) denote the corresponding coordinates in the original and the projected (motion-estimated) frames, respectively. Parameter a_1 estimates the zooming factor whereas a_2 and a_3 are the translates along the x and y directions, respectively.

Unlike the two-parameter translational model, block-matching approach in this case is impractical as the number of search points becomes too large especially when sub-pixel accuracy is needed. Defining the Transformed Pixel Difference (TPD) as

$$\mathrm{TPD}[x,y,a] = I_n[u(x,y,a), v(x,y,a)] - I_0[x,y], \quad (3)$$

which is the difference between the corresponding pixels' intensities in frame n and the reference frame 0, the best we can do is to estimate the optimum a for each block B_N such that the distortion function, E,

$$E_N(a) = \sum_{x,y \in B_N}\sum \mathrm{TPD}^2[x,y,a] \quad (4)$$

is minimized. This creates a non-linear least squares optimization problem which can be solved iteratively using the Gauss-Newton (G-N) algorithm [4] as given by

$$a^{(k+1)} = a^{(k)} - 2[B(a^{(k)})]^{-1}[J(a^{(k)})]^T[f(a^{(k)})], \quad (5)$$

where superscript $k = 0,1,2,...,$ denotes the iteration number. Vector $[\mathbf{f}(\mathbf{a})]$ is an N^2 x 1 column vector, given by

$$[f(a^{(k)})] = \begin{bmatrix} \mathrm{TPD}\,(0,0,a^{(k)}) \\ \mathrm{TPD}\,(0,1,a^{(k)}) \\ \vdots \\ \mathrm{TPD}\,(N-1,N-1,a^{(k)}) \end{bmatrix}. \quad (6)$$

The matrix $\mathbf{J}$ is the Jacobian matrix which consists of the first order derivatives of the TPD function with respect to the motion parameters a_1, a_2, and a_3:

$$[J(a^{(k)})]^T = \begin{bmatrix} \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_1} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_1} \\ \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_2} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_2} \\ \frac{\partial TPD\,(0,0,a^{(k)})}{\partial a_3} & \cdots & \frac{\partial TPD\,(N-1,N-1,a^{(k)})}{\partial a_3} \end{bmatrix}. \quad (7)$$

Finally, matrix B in (5) is evaluated as a linear approximation of the 3 x 3 Hessian matrix, which consists of the second order derivatives of the objective function, $E(a^{(k)})$, with respect to the three motion parameters.

B. Frame Projection and Its Invertibility

Another important aspect of this scheme is inverse motion projection at the decoder. The problem of using simple bilinear filters for sub-pixel interpolation is poor reversibility. This means that we cannot fully recover the original frame by using only the inverse motion parameters and the projected frame. Motion discontinuities between blocks are also a main source of non-invertibility [5]. In our scheme, we employed a cubic spline interpolation filter which is computationally more efficient than the sinc interpolation filter used in [7]. We begin the iteration with $\mathbf{a} = [1,0,0]^T$.

Fig. 1 depicts the average prediction PSNR for the *Football* sequence as a function of the number of Gauss-Newton iterations and block size. As expected, better

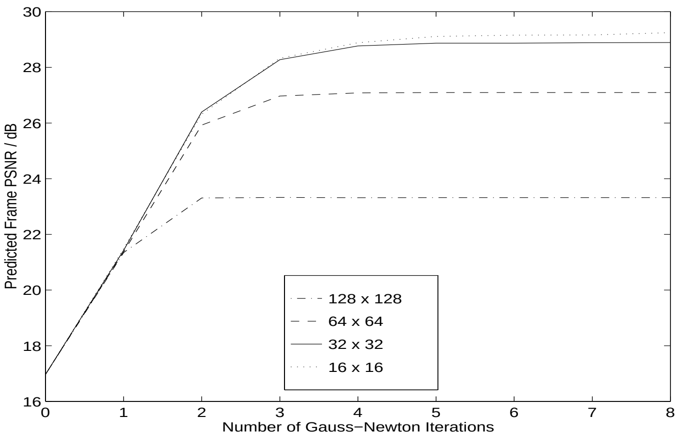

Fig. 1. Prediction PSNR of *Football* sequence using a three-parameter motion model for different number of Gauss-Newton iterations and block sizes.

prediction is achieved when using a smaller block size, and a larger number of iterations; however, such improvement is only marginal after a few iterations. Based on these observations we choose $k = 3$ or 4, and $N = 32$ or 64, depending on the motion content of the sequence.

III. Theory and Motivations for TRI-ZTR

Tri-Zerotree (TRI-ZTR) is a new extension of Shapiro's *Embedded Zerotrees of Wavelet coefficients* (EZW) generic data structure [6] for 2-D still image subband coding using wavelet transform. Several enhancement techniques to EZW which further improve both the objective and subjective quality are also reported in [8]. However in spite of the close similarity to EZW, TRI-ZTR provides a three-dimensional data structure needed for video coding. More importantly, it also supports multiresolution (both spatial and temporal) scalability in addition to multirate scalability in EZW.

A. 3-D Subband Framework for TRI-ZTR

First the incoming video frames are partitioned into distinct groups of frames (GOF) of size F. For simplicity we choose $F = 2^T$ where T is the number of octave-band temporal decomposition levels. After performing the motion estimation as described in Section II, the motion-compensated GOF is transformed using a separable 3-D wavelet decomposition framework (see Fig. 2).

We first perform a 1-D temporal decomposition using Daubechies 4-tap filter [2]. For the case of GOF = 2, the simple 2-tap Haar filter is used. Next, a separable 2-D octave bandwidth decomposition is applied along the rows and columns using biorthogonal spline wavelets [1] as we found that this bank of filters gives comparatively less ringing effect at lower bit-rates.

B. Subband Relationships and Formation of TRI-ZTR

An interesting characteristic of recursive subband decomposition is the formation of multiscale spatial orientation

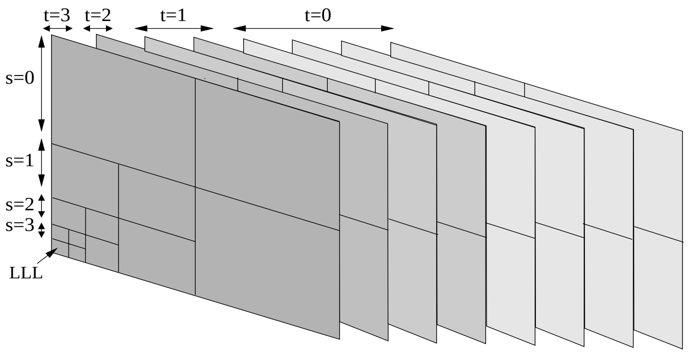

Fig. 2. Three-dimensional wavelet framework, with s = 3 and t = 3 levels of decomposition, for our TRI-ZTR scalable video compression system.

trees. Since the subbands are filtered and decimated versions of the same original images, a parent-child relationship can be defined. Let $c(x_s, t_n)$ denote a coefficient, $\mathbf{x}$ = (i, j), at scale s of temporal subband t_n (denoted as PARENT). Then its children nodes (denoted as CHILD) can be represented by

$$\text{CHILD}\{c(x_s, t_n)\} = \bigcup_{t \in [t_n, t_F)} c(x_{s+1}, t). \tag{8}$$

Note that this 3-D subband relationship also extends across the temporal scales (refer to Fig. 2).

Decomposing the motion-compensated GOF concentrates most of the signal energy into the LLL subband shown in Fig. 2. Such an energy compaction property motivates the formation of TRI-ZTR which forms the core compression data structure. The idea of a zerotree [6] is based on the hypothesis that if a wavelet coefficient at a coarse scale (both spatial and temporal) is smaller than a certain threshold, then *all* its children at finer scales are also *likely* to be insignificant with respect to the same threshold. Therefore a TRI-ZTR root is formed at a PARENT node if all its descendents are also insignificant. In addition a TRI-ZTR root must not also be a part of another previously formed TRI-ZTR at the same threshold. This concept provides a compact representation of predictably insignificant coefficients across both the spatial and temporal scales. Note that now only *one* TRI-ZTR symbol at a PARENT node $\{c(x_{s'}, t')\}$ is sufficient to represent $\sum_{t=t'}^{3} \sum_{s=s'}^{3} \{4^{s-s'}\}$ insignificant descendant coefficients (for the example in Fig. 2).

In order to be consistent with the idea of coding and transmitting coefficients with higher energy content first, we employ a predetermined scanning sequence in which coefficients at coarser scales are processed earlier than those in finer scales. The temporal scales are also assigned a higher priority than the spatial scales. This is based on the assumption of good interframe motion compensation which improves the temporal correlations. In other words we scan all the spatial scales of the lowest temporal subband from the coarsest to the finest scale first before the higher temporal subbands are further processed in a similar manner.

C. Progressive Video Coding via TRI-ZTR

In order to generate a scalable compressed bit stream, *layered/progressive coding* of the coefficients is employed. This approach is rather similar in spirit to bit-plane coding in which the selected coefficients, c_n, are encoded in stages - each stage adding another bit of precision to their magnitudes. Within each stage i, a *primary pass* and a *secondary pass* are performed alternately according to the inherent scanning sequence as outlined above. The process begins with an initial threshold, $T_0 = \max|c_n| - 1$, and its value is halved in every succeeding stage, such that $T_{i+1} = \frac{1}{2}T_i$.

A primary pass consists of three main steps. The first step is basically a thresholding process (a dominant pass in [6]) in which the magnitude of each coefficient is compared with the current threshold, T_i. It is considered *significant* if $|c_n| \geq T_i$ and *insignificant* otherwise. If c_n is significant, its sign (POS or NEG) is coded and its magnitude is transfered to a significant list which is initially empty. Then its value is set to zero to allow the formation of future TRI-ZTR's at lower thresholds. Else, c_n is checked as a potential candidate for a TRI-ZTR root. Once a TRI-ZTR is identified, all its descendents will be skipped in this current i^{th} pass as they are already predictably insignificant with respect to T_i. However if this fails, an isolated zero (IZ) symbol is produced instead. Note that this essentially creates a binary map whereby each coefficient is classified as either significant (POS or NEG) or insignificant (TRI-ZTR or IZ) with respect to T_i. These symbols are then entropy coded using an adaptive model arithmetic coder [9].

Nevertheless such a layered coding strategy only generates a bit-rate (distortion) scalable compressed bit stream. In order to achieve multiresolution scalability, the bit stream needs to be further partitioned into distinct *resolution blocks* in the second step. This is carried out by means of encoding appropriate resolution flag (RFG) symbols at each required spatial and temporal scales during the primary pass. More details about the issue of video scalability will be explained in Section IV. The third step rearranges all the significant coefficients found in all the i-1 previous primary passes, and the current i^{th} pass. Our principal aim here is to re-position the entries in the significant list according to their resolution blocks, without destroying the integrity of each block. This is done to reduce the cost of encoding the RFG symbols when we perform the secondary pass later. At the end of each primary pass, each significant coefficient is assigned a reconstruction value and is associated with an uncertainty interval equal to $|T_i|$. For simplicity, the center of the uncertainty interval [6] is chosen as the reconstruction value since this requires no additional information to be transmitted to the decoder. The main purpose of a secondary pass is to further refine the uncertainty interval, and hence the precisions, of all significant coefficients found so far. It assigns either a binary 0 or 1 depending on whether $|c_n|$ lies in the lower or upper half of the uncertainty interval, respectively. In this manner, their reconstruction values will successively approach their actual values as i increases. Similar to the primary pass, we will also encode appropriate RFG flags which define each required resolution block. In general these rounds of passes will continue until a certain condition (such as a bit-rate or distortion rate) is met.

D. Prioritization Protocol

Suppose the decoder receives a transform coefficient of value $|c_n|$, then the mean square error will decrease by $|c_n|^2/M$, where M is the image size. This concept motivates us to reorder the significant list in a decreasing order of magnitude for future secondary passes. However the following reordering/prioritization protocol (different from [6]) has to be adhered to so that the decoder can later duplicate the same process:

- **Temporal scale** - this assigns a higher priority to the temporally filtered subbands from the coarsest to the finest scales.
- **Spatial scale** - this orders the spatial scales from the coarsest resolution block according to the scanning sequence.
- **Reconstruction magnitude** - this sorts the coefficients within the same resolution block in a decreasing order of reconstruction magnitude.
- **Spatial position** - this preserves the integrity of each resolution block so as to synchronize with the decoder.

The main differences here are that we have assigned a higher priority to the temporal scales and sorting by reconstruction magnitudes while preserving the uniqueness of each resolution block. Using this protocol, the decoder can later perform the same reordering scheme *without* the need to transmit any explicit overhead.

IV. Issues of Video Scalability

The term *scalability* refers to the flexibility of the compressed bit stream to be manipulated in a manner that makes it possible to meet different decoder's requirements *after* the bit stream has been generated [7]. In other words, the compressed bit stream is made up of distinct resolution blocks of fully embedded bits (see Fig. 3) such that appropriate subsets can then be extracted to fulfil different display specifications. A very useful example is in video-on-demand where only **one** copy of a full resolution and full frame rate video is stored in the server. Different end users can then view the *same* video at their chosen spatial resolutions, bit-rates, and frame rates, depending on the available bandwidths and other constraints.

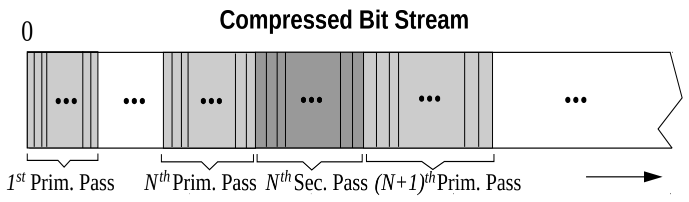

Fig. 3. A typical compressed bit stream which is made up of unique resolution blocks (marked by vertical solid lines).

Let us now consider a simple example illustrating how various subsets can be extracted to form new compressed bit streams with different display specifications. Suppose that we encode an input video sequence at T_f frames per second (fps) using a GOF = F, S spatial scales, $T = \log_2 F$ temporal scales, and an available bit budget of T_b bits. This encoding configuration allows the following video scalability features:

- **Bit-Rate (Distortion) Scalability -** refers to trading distortion for bit-rate at a particular display resolution and frame rate. In this scheme, one can actually choose any precise bit-rate of $t_b \leq T_b$ bits per second as the bit stream is fully multirate scalable.
- **Spatial Resolution Scalability -** refers to the display spatial resolution at any bit/frame rates. This feature is endowed by the hierarchical subband decomposition and the inherent scanning sequence which encodes progressively from the coarsest to the finest scale. In this example, it allows a maximum of $S+1$ different display spatial resolutions.
- **Frame Rate Scalability -** refers to the number of frames to be displayed per second. In this case, we can have a maximum of $T+1$ different possible frame rates (i.e., $(1/2)^k T_f$ fps, where k = 0, 1,...,T).

By scanning the bit stream from the beginning, we can now extract only the pertinent subsets and then generate a new compressed bit stream which can be further rescaled. Note also that by judiciously embedding the RFG symbols into the bit stream, we are able to minimize the overhead tremendously in order to achieve multiresolution scalability (compare this approach with [7] which uses high-overhead headers). In general, a larger F will give a lower level of distortion for the same bit-rate. However, a larger GOF will also require more memory buffers and incur more end-to-end delays.

V. Performance Evaluations

In this section, we present some simulation results of our video codec. Fig. 4 illustrates the luminance frame PSNR of the TRI-ZTR coding scheme on *Miss America* sequence using different GOF = 1, 2, 4 and 8 frames. Using an input CIF format (176 lines x 144 pixels) sequence with 10 fps, we can aim for a very low and precise constant bit-rate (CBR) at 15 kbps. Note that a GOF =

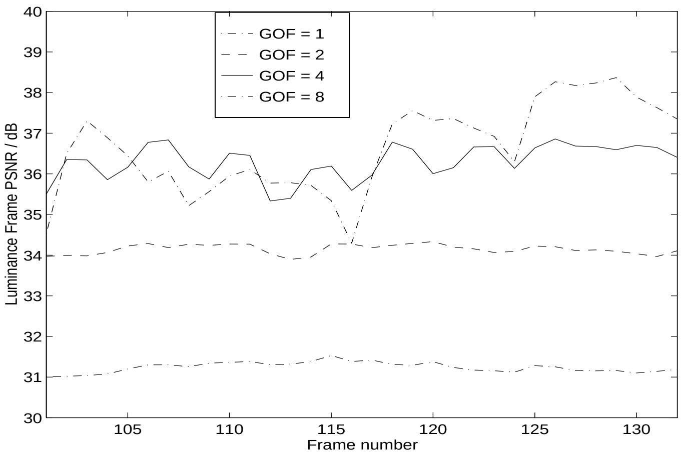

Fig. 4. Luminance frame PSNR of "Miss America" sequence using our TRI-ZTR coding system at 15 kbps, for different sizes of GOF.

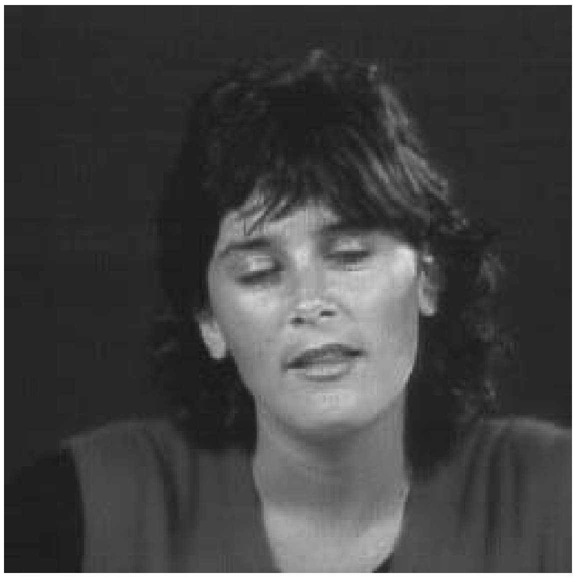

Fig. 5. Original frame 126 of *Miss America* sequence.

1 means only intraframe coding. Fig. 5 shows an original frame of *Miss America* sequence, while Fig. 6 depicts the same frame after a compression ratio of 250:1.

VI. Conclusions

We presented a scalable video compression codec for very low bit-rate applications. It was found that interframe motion can be exploited by means of a three-parameter motion model using block sizes of 32 x 32. The proposed TRI-ZTR data structure also worked very well with the three-dimensional hierarchical wavelet decomposition framework. By using progressive coding together with resolution blocks, a fully embedded and scalable bit stream that supports both multirate and multiresolution video scalability can be generated. This allows us to obtain many different possible combinations of precise bit-rates (CBR) regardless of the chosen spatial resolutions and frame rates.

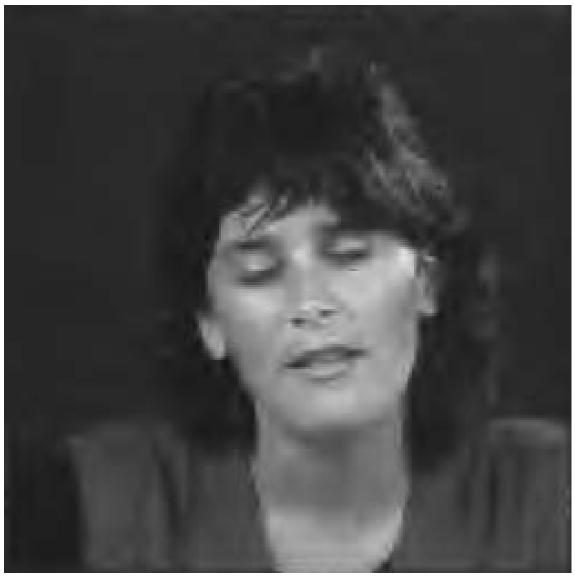

Fig. 6. Reconstructed frame 126 of *Miss America* sequence with a compression 250:1 times using a GOF = 4 frames.

References

[1] Antonini, M., Barlaud, M., Mathieu, P., and Daubechies, I., "Image coding using wavelet transform," in *IEEE Trans. on Image Processing*, vol. 1, pp. 205-220, Apr. 1992
[2] Daubechies, I., "Orthonormal bases of compactly supported wavelets," in *Comm. Pure Appl. Math.*, vol. 41, pp. 906-966, 1988.
[3] Höetter, M., "Differential estimation of the global motion parameters zoom and pan," in *Signal Process*, vol. 16, pp. 249-265, 1989.
[4] Murray, W., *Numerical Methods for Unconstrained Optimization*, London: Academic, pp. 29-42, 1972.
[5] Ohm, J.-R., "Three-dimensional subband coding with motion compensation," in *IEEE Trans. Image Proc.*, vol. 3, no. 5, pp. 559-571, Sept. 1994.
[6] Shapiro, J. M., "Embedded image coding using zerotrees of wavelet coefficients," in *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445-3462, 1993.
[7] Taubman. D., and Zakhor, A., "Multirate 3-D subband coding of video," in *IEEE Trans. Image Proc.*, vol. 3, no. 5, Sept. 1994.
[8] Tham, J. Y., "Detail preserving image compression using wavelet transform," *IEEE Region 10 Best Student Paper (UG Category) 1995*, IEEE Student Paper Book.
[9] Witten, I. H., Neal, R., Cleary, J. G., "Arithmetic coding for data compression," in *Commun. ACM*, vol. 30, pp. 859-861, July 1990.