ARBITRARILY SHAPED VIDEO OBJECT CODING BY WAVELET

Guiwei Xing[1*], *Jin Li, Shipeng Li and Ya-qin Zhang*[2]

[1]Institute of Mathematical Science, Peking University, Beijing 100871, China
[2]Microsoft Research China, 5F Research, Sigma Ctr, 49 Zhichun Road, Haidian, Beijing 100080, China
Email: xinggw@yahoo.com, {jinl,spli,yzhang}@microsoft.com

ABSTRACT

Video object coding is one of the most important functionalities proposed by MPEG4. In this paper, we propose a new wavelet method to encode the texture of an arbitrarily shaped object, both for the still and for video object. The method uses the shape adaptive wavelet transform (SA-DWT) in MPEG4 still object coding, but with a computationally more efficient lifting implementation. The transformed object coefficients are then quantized and entropy encoded with a partial bitplane embedded coder, which greatly improves the coding efficiency. We denote the coding algorithm as video object wavelet (VOW) coder. Experimental results show that VOW significantly outperforms MPEG4 in still object coding, and achieves a comparable performance in video object coding in terms of PSNR. Moreover, the VOW decoded object looks better subjectively, with less annoying blocking artifacts than that of MPEG4.

1. INTRODUCTION

Video object coding is one of the most important features introduced by MPEG4. By compressing an arbitrarily shaped video object rather than a rectangular frame, MPEG4 makes it possible to manipulate and interact with the objects after they are created and compressed. The MPEG4 compressed file can thus be played with special effects and multiple script lines previously only available at expensive studios, or with low-quality cartoonish graphic.

It is essential to represent an arbitrarily shaped object efficiently. The compression of an arbitrarily shaped video object includes the compression of its shape, object motion and texture. Among them, the texture coding is the most important because texture information surpasses the other two. Though in MPEG4 [1][2], both wavelet and DCT schemes have been developed to encode the texture of a still arbitrarily shaped object, only the DCT scheme is applied for the coding of a moving object. For the DCT scheme, the arbitrarily shaped object is first constrained to a bounding box, and then segmented into blocks. Blocks entirely inside the object are encoded with the traditional method; blocks entirely outside the object are not encoded. For boundary blocks with partial object occupation, the available coding options are: the low pass extrapolation (LPE) DCT and △DC-SA-DCT for intraframe object; and the zero padding DCT and SA-DCT for interframe object in MPEG4 version 1 and 2, respectively. A wavelet scheme has also been proposed to encode the still arbitrarily shaped object in MPEG4. In the wavelet scheme, the object is not segmented into blocks, in stead, the entire object first undergoes a shape-adaptive wavelet transform (SA-DWT)[3], which is also called arbitrarily shaped wavelet transform with phase alignment (ASWP) [4]. The wavelet coefficients are then quantized and entropy encoded with a modified PEZW or ZTE scheme [2]. Comparing the three schemes for still object coding, LPE DCT is the simplest in implementation. However, it pads the block and generates more coefficients than the number of pixels in the original object. As a result, its coding performance suffers. △DC-SA-DCT generates exactly the same number of coefficients as that of the original; however, its implementation is more complex as it requires the implementation of a variable basis DCT. Moreover, the coefficients are not aligned after the vertical transform, so the subsequent horizontal transform is applied on coefficients with incoherent phases, which results in a poor transform performance. The still object coded with the two DCT schemes also suffers from the blocking artifact commonly visible in DCT coding. In comparison, SA-DWT not only generates exactly the same number of coefficients as that of the original, but also aligns the coefficients after the horizontal transform so that the subsequent vertical transform can be applied on coefficients with coherent phases. The objective coding performance of SA-DWT is thus better. The still object encoded by SA-DWT also looks better, because the wavelet transform scheme does not have the annoying blocking artifact visible in the DCT scheme.

Motivated by good subjective appearance of wavelet coded images, we target to further improve the wavelet coding performance of both still and moving objects in this paper. We implement SA-DWT with lifting scheme, and improve the object coefficient coding with a partial bitplane embedded coder. The scheme is denoted as the video object wavelet (VOW) coder. We also extend VOW to encode the prediction residue of an interframe object, which is not supported in the wavelet coding mode of MPEG4. Experimental results show that VOW significantly outperforms MPEG4 in still object coding, and is comparable to MPEG4 in video object coding. Since wavelet coded object has less blocking artifact, the VOW decoded object looks better subjectively than that of MPEG4.

2. VIDEO OBJECT WAVELET CODER

2.1 Framework

The framework of the proposed video object wavelet (VOW) coder can be shown in Figure 1. A moving arbitrarily shaped video object consists of shape, motion and texture. The coding begins with the object shape, which further guides the motion and residue coding, as only the motion and residue within the object needs to be encoded. Then, the object motion is estimated and the motion information is encoded. After that, the prediction

[*] This work was done while the author was an intern at Microsoft Research China.

residue of the object is encoded. The encoded shape, motion and texture are multiplexed and sent to the decoder. The reconstructed residue is then added back to the motion compensated object, and serves as a reference for the object in the next frame. VOW encodes the shape and motion of the object exactly the same way as MPEG4, and calculates the prediction residue exactly the same way as MPEG4. The contribution of VOW is the proposal of a new wavelet scheme to encode the still texture and the prediction residue of the object.

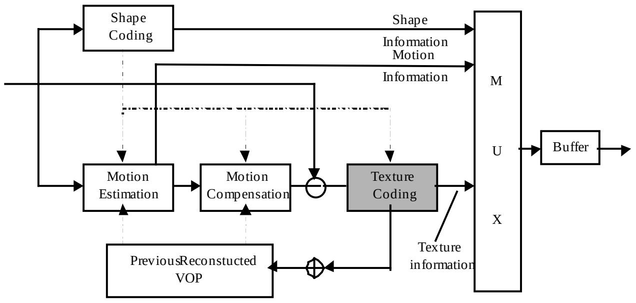

Figure 1 Framework of video object wavelet (VOW) coder.

2.2 Shape-Adaptive Discrete Wavelet Transform with Lifting Implementation

The texture of the intra frame object and the prediction residue of the inter frame object are decomposed by the shape adaptive wavelet transform so that the energy of the texture/residue can be compacted into a few wavelet coefficients. The pyramidal wavelet transform is implemented, which further decomposes only the horizontal and vertical low pass band of an upper scale. We implement a 4 level transform on video object of size 352x288. The contribution in this paper is that the SA-DWT is implemented with lifting wavelet so that the wavelet transform may be performed faster and with easier boundary extension. The biorthogonal 9-7 filter is used instead of the biorthogonal 9-3 filter specified by MPEG4 for its better transform efficiency. The detail implementation of a single scale SA-DWT is as follows:

Step 1. Row transform

The texture (for intra) or the prediction residue (for inter) of the arbitrarily shaped object is first horizontally filtered row by row. Each row intersects the object and forms one or more segments. A lifting wavelet with symmetrical boundary extension is applied on each segment independently. We always align the center of the low pass filter with even index *2i*, and align the center of the high pass filter with odd index *2i+1,* with index position calculated from the start of the row. The resultant coefficients are stored in the index *i* of the low and high pass bands respectively. The alignment operation ensures that when the vertical lifting wavelet is applied on the *i*th column of the low and high pass bands, it is applied on the filtered coefficients with coherent phases, which improves the transform efficiency. If there is no object pixel at *2i* or *2i+1,* there will be no corresponding coefficient at index *i* of the low or high pass bands. A mask of the transform coefficient is derived which indicates whether there is coefficient at a specific position of the output subband. In contrast with MPEG4, we implement the segment filter with lifting wavelet, which greatly reduce the implementation complexity and ease the frequent boundary extension used in segment filter.

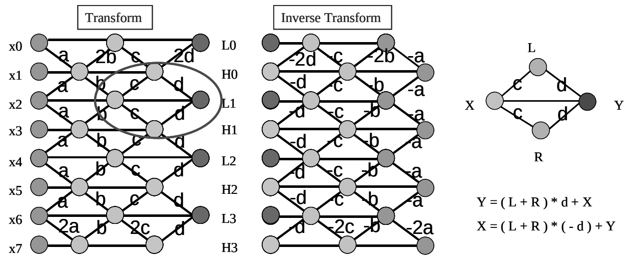

Figure 2 Forward and inverse lifting wavelet.

Suppose a segment of length 8 is fed into the lifting wavelet module, as shown in Figure 2. We further assume that the low pass wavelet filter is aligned with coefficient x_0. The biorthogonal 9-7 lifting wavelet can be depicted by a 4-stage lifting with coefficients $a=-1.586$, $b=-0.052$, $c=0.883$, and $d=0.444$. The wavelet coefficients are calculated as follows. First, a high pass stage is performed which updates the coefficients at odd indices $x_1, x_3, x_5,$È ; next, a low pass stage is performed which updates the coefficients at even indices $x_0, x_2, x_4,$È ; the third and fourth stages are a high and a low pass stage, which generate the final high and low pass coefficients, respectively. The entire forward lifting operation is shown in the left of Figure 2. A basic element of the lifting is circled with an oval, and is shown in the right side of Figure 2. It can be described with equation:

$$Y=(L+R)*d+X, \tag{1}$$

where X and Y are coefficients before and after the lifting, L and R are the neighbor coefficients, and d is the lifting parameter of the current stage. Because after the lifting operation, the old coefficient X is no longer needed, coefficients X and Y can share the same memory. Thus, the lifting operation does not need any additional memory during the transform stage. On average, the lifting implementation needs 2 multiplications and 4 additions per pixel, which is far less than the 4.5 multiplications and 7 additions per pixel required by the convolution wavelet implementation. Note the data is symmetrically extended at the boundary and the lifting structure itself is symmetrical, it is easily demonstrated that the intermediate and final lifting coefficients are all symmetrical. Therefore, we may implement symmetrical boundary extension by discarding the extended lifting structure and multiplies every branch pointing towards a boundary node by a factor of 2. The inverse of the lifting can be easily derived, because the basic operation (1) can be inverted straightforward as:

$$X=(L+R)*(-d)+Y. \tag{2}$$

The inverse biorthogonal 9-7 lifting filter is shown in the middle of Figure 2. For normalization, the generated low and high pass coefficients are multiplied by 1.140 and 0.887, which are the square root of energy of the low and high pass synthesis function, respectively.

Step 2. Column transform.

The low and high pass bands of the horizontal transform are further processed column by column. Each column intersects the object and forms one or more vertical segments. The lifting wavelet with symmetrical boundary extension is again applied on each vertical segment independently. We still align the low pass filter with even index *2i*, and the high pass filter with odd index

2i+1. The resultant coefficients are stored in the index *i* of the low and high pass vertical bands, respectively.

The number of wavelet coefficients after SA-DWT is exactly the same as the number of pixels in the video object. Since the phase of the wavelet filtering is always aligned, the transform efficiency is high. The spatial correlation and wavelet transform properties, such as locality property and self-similarity across sub-bands, are well preserved in the SA-DWT.

2.3 Scalar Quantization

After SA-DWT, the coefficients are quantized with a uniform scalar quantizer with a dead zone twice the step size, which can be formulated as:

$$\nu[m,n] = \left\lfloor \frac{|s[m,n]|}{\delta} \right\rfloor \quad \chi[m,n] = \begin{cases} 0 & +s \\ 1 & -s \end{cases}, \tag{3}$$

where *s(m,n)* is the transform coefficient, $\nu(m,n)$ and $\chi(m,n)$ are the magnitude and sign of the quantized coefficient, respectively. $\lfloor x \rfloor$ is the floor function returning the largest integer smaller than or equal to x. δ is the quantization step size, which is determined by ensuring the coefficient with the largest absolute value is quantized to 2^n-1. That is:

$$\delta = \frac{1}{2^n - 1} \max |s(m,n)| \tag{4}$$

where f_i enumerates through all wavelet coefficients, and *n* is the maximum coding bitplane, usually 16. The quantization step size is small to ensure sufficient precision for the embedded entropy coder. Unlike the DCT and wavelet mode of MPEG4, the coding quality of VOW is not controlled by the quantization step size δ. Instead, it is controlled by the subsequent embedded entropy coder. The functionality of the quantizer is just to convert the transform coefficients into integer.

2.4 Partial Bit-Plane Embedded Coding

The performance improvement of VOW is achieved by encoding the quantized coefficients through a partial bit-plane embedded entropy coder with context arithmetic coding. The developed partial bit-plane coder bears strong resemblance to the coder used in JPEG 2000 block coding[5]. It outperforms the PEZW and ZTE[2] used in MPEG4 still wavelet object coding mode, because it does not use the wavelet domain zerotree assumption which is frequently broken, especially in object boundaries and for the prediction residue. The implementation of the VOW partial bit-plane coder can be described as follows:

Let the absolute value of the quantized coefficient $\nu(m,n)$ be represented by a string of bits $b_{n-1}b_{n-2}\cdots b_0$, where b_{n-1} is the most significant bit (MSB), and b_0 is the least significant bit (LSB). Let bits b_j of the same significance level of all subband coefficients be grouped together and formed bitplane *j*. The coding always proceeds from a more significant bitplane to a less significant bitplane. We further classify each bitplane into three subgroups, which are encoded in three successive passes; hence the name ˆpartial bitplaneˆ coding is derived. The partial bitplane is determined by the significant status of the coefficient. For a specific coefficient at a certain bitplane, if all previous bits of the coefficient are encoded as zeros, the coefficient is called insignificant; otherwise, it is significant. During the coding, each coefficient may transit from the insignificance to significance, but the reverse transition never occurs. The first bitplane pass includes the bits of the coefficient that is still insignificant but has at least one significant coefficient in its immediate eight neighbors. We denote the pass as the predicted significance (PS) pass. The second pass, denoted as the refinement (REF) pass, includes the bits of the coefficients that are already significant. The third pass, the predicted insignificance (PN) pass, includes the bits of the rest coefficients, i.e., the one that is insignificant and all its eight neighbors are also insignificant. It is shown that the embedded coding achieves an optimal performance when symbols are encoded in the order of descending rate-distortion slope[6]. The separation of the bitplane into three passes ensures a gradual decay of the rate-distortion slope, and thus improves the coding efficiency. Within each partial bitplane pass, the subbands are scanned from the lowest resolution to the highest resolution. And at each resolution, the subbands are scanned with the order of LL, LH, HL and HH, where the first and second letters denote the vertical and horizontal filter in use, respectively. The partial bitplane scan can be illustrated in Figure 3. Note there is only a PN pass in the bitplane *n*, as all coefficients in bitplane *n* are insignificant, and there are no bits in the PS and REF passes. Since the partial bitplane coder is an embedded coder, the coding may be stopped at anytime whenever a desired bit rate is reached.

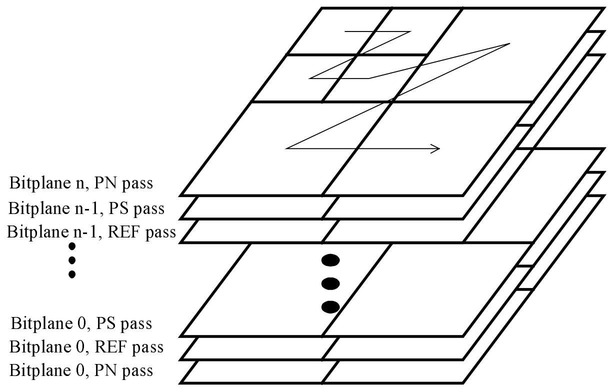

Figure 3 Partial bitplane embedded coding.

In the PS and PN passes, the bits are encoded with a zero coding mode. Whenever the coefficient transit from insignificance to significance, i.e., a HHis encoded in zero coding mode, the sign of the coefficient is encoded with a sign mode. The bits in the REF pass are encoded with a refinement mode. The zero coding, sign and refinement modes all use context arithmetic coding, with contexts determined by the significant statuses and signs of the neighbor coefficients. The context of the refinement mode is further determined by whether it is the first refinement pass of the coefficient. Lookup tables are used to reduce the number of contexts to a manageable level to alleviate the problem of context dilution. The lookup tables of zero coding, sign and refinement are designed exactly the same way as the lookup table used in JPEG 2000 block coding ([5], section IX-4). Since the lifting coefficient is of an object of arbitrarily shape, only coefficients within the object are scanned and encoded. If a neighbor coefficient outside the object is accessed for context evaluation, it is assumed to be insignificant.

3. EXPERIMENTAL RESULTS

Experiments have been performed to compare the proposed video object wavelet (VOW) coder with MPEG4 Verification Model 13.0 (VM 13.0). The test objects in use are the woman in Akiyo, boat in Coastguard (Coast), anchorpersons in News, and ship in Container (Cont). The sequences are in CIF resolution (352x288).

In the first experiment, we compare the performance of still object coding of VOW versus MPEG4. We encode the first frame of each object and compare VOW with the LPE-DCT and SA-DWT modes of MPEG4. The result is shown in Table 1. For each coding experiment, the total coding rate, the peak signal-to-noise ratio (PSNR) of the luminance component (Y) and the PSNR of the chrominance component (C) are listed. The total coding bit rate includes the coding bits for syntax, shape and the texture. The PSNR of the chrominance component is the average PSNR of components *Cr* and *Cb*. It is shown from Table 1 that VOW significantly outperforms both the DCT and the wavelet mode of MPEG4 still object coding. VOW outperforms the MPEG4 LPE-DCT mode by 0.94-1.74dB, with an average gain of 1.30dB. It also outperforms the MPEG4 still wavelet coding mode by 0.37-1.01dB, with an average of 0.65dB. Since the lifting implementation of VOW is equivalent to the convolution SA-DWT implementation of MPEG4, the result demonstrates that the partial bitplane embedded coder of VOW is superior to the ZTE or PEZW entropy coder of MPEG4.

Table 1 Intraframe object coding result

Seq	Bitrate	Y/C	VM (LPE)	VM (DWT)	VOW
Akiyo	26k	Y	34.40	35.14	35.51
		C	40.08	39.79	41.83
News	34k	Y	33.38	34.28	34.73
		C	40.71	40.51	42.58
Coast	10k	Y	31.02	31.72	32.25
		C	41.73	40.49	45.07
Cont	29k	Y	31.52	31.81	32.46
		C	36.56	35.03	36.47
Akiyo	46k	Y	39.03	39.47	40.48
		C	42.96	41.95	43.96
News	58k	Y	38.43	39.31	40.17
		C	42.71	42.32	44.48

In the second experiment, the entire test video objects are encoded. A total of 120 frames are encoded for each sequence, with the first frame encoded as I frame, and the rest encoded as P frames. In the comparison MPEG4 VM, LPE/zero padding DCT are used to encode the I and P frames, respectively. VOW encodes the shape and motion of the object exactly the same way as MPEG4, and calculates the prediction residue exactly the same way. A rate control strategy has been implemented to ensure that VOW spends the same number of bits for each frame as that of MPEG4. Since the entropy coder of VOW is an embedded coder, we may first run the MPEG4 coder, and then use the MPEG4 bitrate to determine the allocated bits for each frame of VOW. With everything else the same, the comparison demonstrates the performance difference of the VOW arbitrarily shaped texture coder versus that of the MPEG4. The results are shown in TABLE 2. VOW differs from MEPG4 for (0.24dB to +0.45dB, with an average gain of 0.04dB. Though the PSNR difference between VOW and MPEG4 is small, the subjective quality of VOW coded object surpasses that of MPEG4, as there is less blocking artifact.

TABLE 2 Video object coding result

Seq	Bit Rate	Y/C	MPEG4 VM	VOW
Akiyo	98kbps	Y	33.59	33.71
		C	39.49	41.13
News	120kbps	Y	32.50	32.95
		C	40.04	42.06
Coast	98kbps	Y	28.98	28.74
		C	40.71	44.54
Cont	120kpps	Y	29.42	29.26
		C	35.41	37.40

4. CONCLUSIONS

In this paper, we propose a video object wavelet (VOW) coder which improves the texture coding of arbitrary shape object, for both the still and moving objects. A lifting SA-DWT is implemented which speeds up the SA-DWT transform. We replace the ZTE or PEZW entropy coder with a better performance partial bitplane embedded coder. Experimental results show that VOW significantly outperforms the LPE-DCT and wavelet mode of MPEG4 still object coding. For video object coding, VOW achieves a comparable PSNR and a superior subjective quality compared to that of MPEG4, as there is less blocking artifact. The VOW coded still object is also quality scalable as the output bitstream can be truncated at an arbitrary point.

5. REFERENCES

[1] A. Kaup, (Object-based texture coding of moving video in MPEG4(, IEEE Trans. on Circuits and Systems for Video Technology, vol.9, pp.5-15, Feb. 1999.

[2] MPEG4 Video, (MPEG4 Video Verification Model Version 13.0(, ISO/IEC JTC1/SC29/WG11 N2687, March 1999

[3] S. Li and W. Li, {Shape adaptive discrete wavelet transform for arbitrarily shaped visual object coding(, Submitted to IEEE Trans. on Circuits and Systems for Video Technology.

[4] J. Li and S. Lei, ¡° Arbitrary shape wavelet transform with phase alignment¡±,1998 IEEE International Conf on Image Processing, vol. 3, pp. 683-687, Chicago, IL, Oct. 1998.

[5] JPEG VM ad-hoc, ¡°JPEG 2000 verification model 5 2 (technical description) , ISO/IEC JTC1/SC29/WG1N1422, Maui, Hawaii, Dec. 1999.

[6] J. Li and S. Lei, (An embedded still image coder with rate-distortion optimization(, IEEE Trans. On Image Processing, Vol. 8, No. 7, Jul. 1999, pp. 913-924.

[7] J. M. Shapiro, (Embedded image coding using zerotrees of wavelet coefficients ,(IEEE Trans. On Signal Processing, vol. 41, pp. 3445-3462, Dec. 1993