Efficient Bit-Plane Coding Scheme for Fine Granular Scalable Video Coding

Seung-Hwan Kim, Yo-Sung Ho

Gwangju Institute of Science and Technology (GIST), 1 Oryong-dong, Buk-gu, Gwangju 500-712, Korea

Received 11 September 2006; accepted 27 November 2006

ABSTRACT: In this paper, we propose a new efficient bit-plane coding method for fine granular scalable (FGS) video coding. The general structure of the proposed bit-plane coding method is based on the traditional bit-plane coding scheme in MPEG-4 FGS. However, to enhance coding efficiency of bit-plane encoding, we apply an efficient probability estimation scheme through employing the binary arithmetic coding. For probability estimation, various context models are designed to take advantage of the characteristics of each bit-plane as well as the correlations of symbols among different bit-planes. Experimental results show that the proposed FGS coding scheme provides better coding performance, compared to the well-known FGS coding schemes in MPEG-4 FGS and JSVM. Int J Imaging Syst Technol, 16, 113–120, 2006; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/ima.20073

Key words: scalable video coding; H.264; bit-plane coding

I. INTRODUCTION

The fine granularity scalability (FGS) video coding technique, defined in the Amendment of MPEG-4 (Li, 2001), has become an attractive topic because it can provide scalable bit streams to different bandwidth channels. In MPEG-4 FGS, in order to prevent the drift problem, temporal prediction is limited to the base layer. However, it causes severely reduced coding performance relative to the nonscalable single layer video encoding. Hence, there has been a growing interest in FGS approaches that can enhance coding efficiency by allowing the temporal prediction in the enhancement layer (Peng and Chen, 2001; Sun et al., 2001; Wu et al., 2001; Reibman et al., 2003).

The progressive fine granularity scalable (PFGS) coding scheme (Wu et al., 2001) has a significant improvement over MPEG-4 FGS by introducing two prediction loops with different quality references. Moreover, macroblock-based PFGS (Sun et al., 2001) and its improved version (Huang et al., 2002) provide a good trade-off between coding efficiency improvement and drifting error reduction by optimally selecting the reference of the enhancement layer at the macroblock level. Generally, all these schemes mainly focus on motion estimation, motion compensation, and mode decision. Furthermore, it is required to have a specific coding scheme to control the drift problem in these approaches. Even though numerous scalable tools for drift control have been investigated, a wide acceptance in the market of prospective applications has never occurred (Ohm, 2005).

With the introduction of H.264/AVC, the most powerful and state-of-the-art video coding standard, the scalable extension of H.264/AVC was selected as the first working draft (Joint Video Team, 2005a). A reference encoder is described in the joint scalable video model (JSVM 0). To support fine granular SNR scalability, a progressive refinement (PR) slice is introduced to JSVM. A refinement signal that corresponds to a bisection of the quantization step size is represented in the PR slice. In addition, with an exception of the modified coding order, the CABAC (Marpe et al., 2003) entropy coding scheme, as specified in H.264/AVC, is reused in the PR slice. In JSVM, key picture uses the conventional closed loop motion compensation and the reference pictures for the key picture shall be base representation without FGS enhancement. On the contrary, the reference pictures of nonkey picture shall be base representation + FGS enhancement. Hence, if FGS layers are truncated, drift error is inevitable for the nonkey pictures. To limit the corresponding drift, temporal prediction in a key picture is limited to the base layer. Thus, the key picture is used as resynchronization point between encoder and decoder.

In this paper, we focus on the design of an efficient FGS coding scheme. After we briefly review the FGS coding schemes in MPEG-4 FGS and JSVM, we propose a new FGS coding scheme. The general structure of the proposed FGS coding scheme is based on the traditional bit-plane coding in MPEG-4 FGS. However, to enhance coding efficiency, a context-based binary arithmetic coding scheme is specially designed for enhancement layer coding. Experimental results show that the proposed FGS coding scheme outperforms the FGS schemes in MPEG-4 and JSVM.

The rest of this paper is organized as follows. We first briefly overview two FGS coding schemes in MEPG-4 FGS and JSVM in Section II. The new FGS coding scheme is then proposed in Section III. The performance of the proposed coding technique is compared to those of the existing FGS coding schemes in Section IV. The paper is concluded with some discussions in Section V.

Correspondence to: Seung-Hwan Kim; e-mail: kshkim@gist.ac.kr or Yo-Sung Ho; e-mail: hoyo@gist.ac.kr

Grant sponsors: MIC through RBRC at GIST and MOE through the BK21 project.

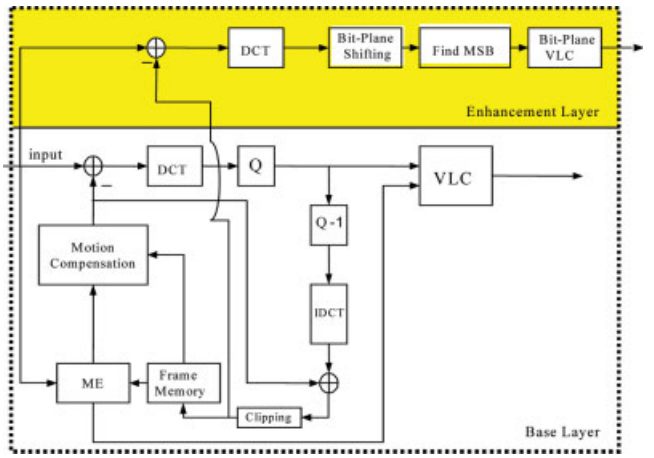

Figure 1. MPEG-4 FGS encoder. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

II. REVIEW OF PREVIOUS FGS CODING

In this section, we will discuss two kinds of well-known FGS coding techniques used in MPEG-4 FGS and JSVM, respectively.

A. MPEG-4 FGS. Figure 1 shows the encoder structure of the MPEG-4 FGS system. As shown in Figure 1, the FGS encoder consists of two parts: base layer and enhancement layer. In the base layer, the basic information of the input signal is coded in the same way as the traditional block-based coding method. In the enhancement layer, the residual signal that is not coded in the base layer is divided into 8×8 blocks and each block is discrete cosine transformed (DCT). All the 64 DCT coefficients in each block are bit-plane coded using four VLC tables (Li, 2001).

The bit-plane coding in MPEG-4 FGS (or Bit-plane VLC) considers each quantized DCT coefficient as a binary number of several bits instead of a decimal integer of a certain value (Li, 2001). For each DCT block, the 64 absolute values are zigzag ordered into an array. A bit-plane of the block is defined as an array of 64 bits, taken one from each absolute value of the DCT coefficients at the same significant position. For each bit-plane of each block, (RUN, EOP) symbols are formed and variable-length coded (''Bit-plane VLC'') to produce the output bitstream.

Bit-plane VLC is based on Huffman coding. Since the statistics of the first three bit planes (MSB, MSB-1, and MSB-2) are very different from each other and from the lower bit-planes, corresponding four VLC tables have been designed for MSB plane, MSB-1 plane, MSB-2 plane, and the other bit planes. The designed Bit-plane VLC tables reflect the statistical distribution of each bit plane. But still there is plenty of room for improvement. It still shares the fundamental disadvantage of Huffman coding that assigns a codeword containing an integral number of bits to each symbol. The usage of only fixed VLC tables does not allow an adaptation to the actual symbol statistics, which may vary over spatial and temporal as well as for different source material and coding conditions.

B. JSVM. To support fine granular SNR scalability, JSVM adopted so-called PR slices (Reichel et al. 2004; Ridge et al., 2004; Joint Video Team, 2005a,b; Schwarz et al., 2005a,b). To achieve the SNR scalability, the residues after the inter-layer prediction are coded using a hybrid approach of bit-plane and (Run, Level) coding. Specifically, the residues are firstly transformed by the 4×4 integer transform. Then the transform coefficients are successively quantized into multiple quality enhancement layers (also known as FGS layers) for the SNR scalability. For the property of fine granularity, the bit-planes in a FGS layer are coded using a cyclical block coding. Basically, the coding is partitioned into two passes, the significant and the refinement passes. The significant pass first encodes the insignificant coefficients that have values of zero in the subordinate layers. Then, the refinement pass refines the remaining significant coefficients with range from -1 to $+1$. During the significance pass, the transform blocks are coded in a cyclical and block-interleaved manner. On the other hand, the coding of the refinement pass is conducted in a suband-by-subband fashion. In cyclical block coding, for each cycle, the coding of a block is continued until a nonzero coefficient in zigzag order is coded. Particularly, the coding of each cycle in a block includes an EOB symbol, a Run index and a nonzero quantization level. The EOB symbol is coded prior to the other symbols for signaling whether there are nonzero coefficients to be coded in a cycle. In addition, the Run index, represented by several significance bits, is used for recording the location of a nonzero coefficient. To further reduce the bit rate, each symbol is coded by a context-adaptive binary arithmetic coder.

In Figure 2, the encoding structure of the PR slice is depicted. Each PR slice is regarded as a FGS layer which represents a refinement signal that corresponds to a bisection of the quantization step size ($\Delta = 6$). In JSVM, in order to enhance coding efficiency within an FGS slice, they first encode "more significant" coefficients. By arranging the bit stream in this way, the decoding process is biased so that a simple truncation is likely to retain those "more significant" coefficients, and therefore improve the reconstructed quality.

In Figure 2, each FGS layer is represented as a group of multiple bit-planes. However, these bit-planes are coded by a cyclic block coding instead of traditional bit-plane coding used in MPEG-4 FGS. The coding order of transform coefficient levels has been modified. Instead of scanning the transform coefficients macroblock by macroblock as it is done in the "normal" slices, transform coefficient blocks are scanned in several paths, and in each path only a few coding symbols for a transform coefficient block are coded. Therefore, quality of the SNR base layer can be improved in a fine granular way. With the exception of the modified coding order, the CABAC entropy coding is reused, as specified in H.264/MPEG4-AVC (Marpe et al., 2003).

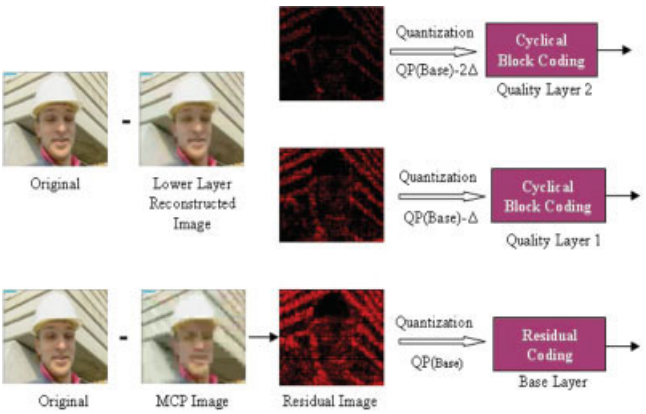

Figure 2. Encoding structure of progressive refinement slice. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

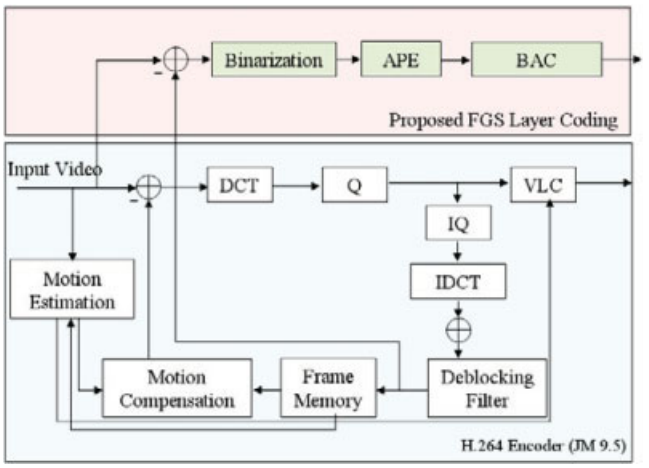

Figure 3. The basic structure of the proposed FGS encoder. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

III. PROPOSED FGS CODING SCHEME

In this Section, we further employ a context-adaptive bit-plane coding to improve the coding scheme of the enhancement layer. Our goal is to provide a generic bit-plane coding that delivers higher coding efficiency. Moreover, our scheme is designed not only for MPEG-4 FGS (Li, 2001), but also for the advanced FGS algorithms.

The main difference in coding schemes between previous FGS coding methods and proposed method are as follows.

1. Firstly, the proposed FGS replace the Bit-plane VLC in MPEG-4 FGS with context-adaptive binary arithmetic coding.
2. Secondly, the proposed context-adaptive binary arithmetic coding scheme is newly designed for bit-plane coding and different from CABAC in JSVM in terms of coding structure and used context information.

Figure 3 shows the basic structure of the proposed entropy coding. The proposed entropy coding is divided into two steps. The first step is the same as MPEG-4 FGS. The residual image, which is obtained by subtracting the reconstructed base image from the original image is first divided into 8×8 blocks and transformed (DCT). Then, those DCT coefficients are zigzag-scanned as represented by corresponding bit-planes (Li, 2001). In the second step, the bit-plane VLC (Li, 2001) is replaced by binary arithmetic coding with adaptive probability estimation to enhance coding efficiency of bit-plane coding. For adaptive probability estimation, a proper context model is selected to estimate the probability distribution of the current symbol. In the next step, a binary arithmetic coding engine generates a bitstream using the estimated probability distribution of the symbol.

A. Bit Classification and Bit-Plane Partition. In our proposed FGS coding scheme, the design of context model is most critical to the coding efficiency. To improve the efficiency of context reference, we propose a bit classification scheme and a bit-plane partition method for distinguishing the coefficient bits with different importance so that the context models can be designed by different sources of correlations. For the bit classification, we partition the coefficient bits into three types, including significant bit, refinement bit, and sign bit, as in Li (2001). Given our bit classification, the following further presents the context model for each type of bits. Particularly, in our design, we consider various context information to efficiently encode the significant bit and refinement bit.

B. MSB REACHED and sign bit. The proposed FGS coding scheme is based on the bit-plane coding, and the MSB plane of each 8×8 block can vary from block to block. Hence, before the MSB plane in each block is reached, we need to send a signal, msb_not_reached, which indicates whether MSB plane is reached or not. To signal msb_not_reached, we use the same coding scheme used in MPEG-4 FGS (Li, 2001). From the extensive experiments, we confirm that MPEG-4 FGS provided very good solution for the case. Since many 8×8 blocks have fewer bit planes than the maximum number of bit planes in a frame, there are many cases that MSB is not reached. Therefore, to signal msb_not_reached for every 8×8 block efficiently, we group the blocks in each macroblock and code the MB_msb_not_reached case in the macroblock unit. By combining FLC and VLC used in MPEG-4 FGS, we encode MB_msb_not_reached and msb_not_reached.

The sign bit records the sign of a coefficient. Statistical analysis reveals that the distribution of a transform coefficient is approximately symmetric with respect to zero, i.e., the sign bit averagely consumes one bit. Thus, we use a fixed probability model for the coding of sign bits.

C. Adaptive Probability Estimation for bit-plane coding. Binary arithmetic coding is based on the principle of recursive interval subdivision and an estimated probability of the given symbol is represented by its range. The given interval is subdivided into two subintervals, LPS and MPS. Depending on the observed binary decision, either identified as the LPS or the MPS, the corresponding subinterval is then chosen as the new current interval. Thus, if we know the probability of a given symbol to be zero, $P(0)$, the probability of the symbol to be one is automatically determined by $1 - P(0)$. Therefore, in the probability estimation step, only the estimation of $P(0)$ is necessary.

To efficiently estimate the probability of a given symbol to code, we apply three types of probability estimation techniques. The first type relies on local information, such as bit-plane level and frequency component, of a given symbol to code. The second type mainly depends on the complexity of a block where a current symbol is included. The complexity of a block is measured by the number of nonzero coefficients. In the third type, we use preceding symbols in the same bit-plane layer and in the higher bit-plane layer. In the following paragraphs, we are going to discuss in detail these three types of probability estimation techniques.

In bit-plane coding, we generally have different statistical distribution of binary symbols for each bit-plane level. To more accurately estimate $P(0)$ for a symbol, we basically use the local information. Local information includes bit-plane level (BPL), data level (DL), and scanning position (SP). Each of them is defined as follows.

- BPL (bit-plane level): the index of the bit-plane layer of the given symbol in terms of the most significant bit-plane level in each frame.
- DL (data level): the index of the bit-plane layer of the given symbol in terms of the most significant bit-plane level in each 8×8 block.

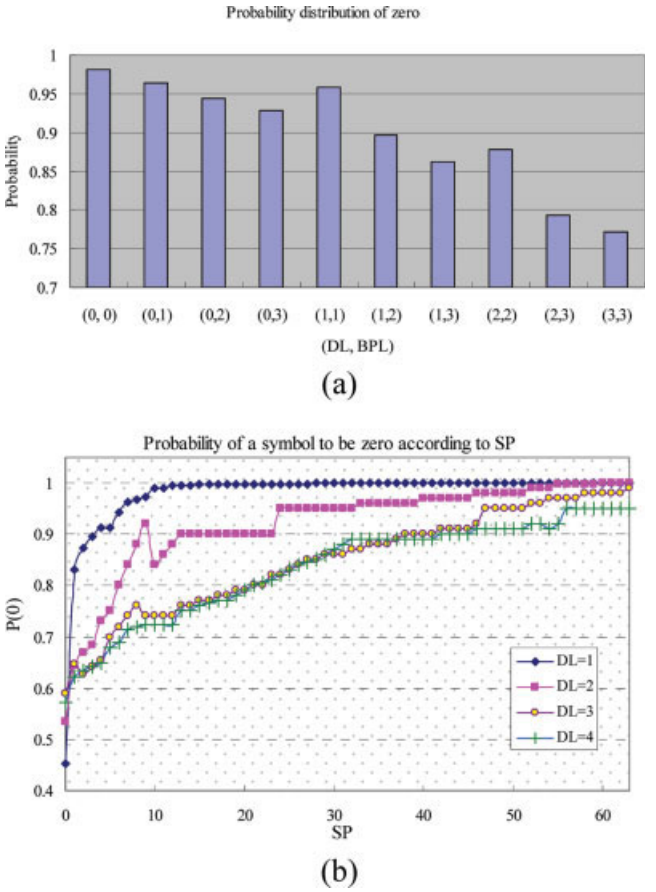

Figure 4. Probability distribution of zero. (a) Data level and bit-plane level. (b) Scanning position. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

- SP (scanning position): the order of zigzag scanning of the given symbol in each 8 × 8 block (0–63).

Since not every 8 × 8 DCT block has the same number of bit-planes, the MSB plane of each individual block (DL) can vary from block to block in terms of the MSB plane of each frame (BPL). Through extensive experiments on many video sequences, we have observed the probability distribution of binary data according to BPL, DL, and SP. In Figure 4, the actual probability distribution of zero according to local information is represented. From Figure 4, it can be observed that the probability distribution depends not only on SP but also on DL and BPL. Hence, if we use the local information together, we can estimate the probability distribution of the source data more precisely.

In the first DL (DL = 0), we can estimate, $P(0)$, probability of a given symbol to be zero by using local information. However, for DL larger than zero, we can also use the correlation between current bit-plane layer and the previously coded bit-plane layers.

Context modeling is based on the fact that a symbol which has appeared recently will appear in a near future, and so the more times that it appears the higher probability that it appears again (Fig. 5). We use context because the probability of a binary symbol appearing in a given position heavily depends on the symbols that have appeared before. When using the information of context we also have to take care of what is the context of the current symbol, and thus we take only the probability of the symbols which have appeared under the given context. In bit-plane coding, binary data are context sensible, that is, under a given context the same symbols tend to appear, and so we can get a more reliable probability. The goal of a context modeling is to get a skewed probability distribution, which hopefully gives high probability to the symbols which actually occur, and thus we can code them with fewer bits. For the sake of convenience in context modeling, we define several terminologies as follows.

C (BPL, DL, SP, i, j) represents a given binary symbol in a block located in (i, j) position in a frame. Each index i and j represents the horizontal and vertical index of an 8 × 8 block in a frame, respectively.

K is the index that represents whether nonzero symbol ("1") is already reached or not at a given SP in a block. If a given symbol is in a L-th DL, we calculate the index K as follows:

$$K = \begin{cases} 0 & \text{for } \sum_{\mathrm{DL}=0}^{L-1} C(\mathrm{BPL}, \mathrm{DL}, \mathrm{SP}, i, j) = 0 \\ 1 & \text{Otherwise} \end{cases} \tag{1}$$

N is the number of encoded nonzero coefficients in a block for the higher DLs of a given symbol to encode. Hence, N is also used for measuring the degree of complexity of a given block. If a given symbol is in a L-th DL, we calculate the index K as follows:

$$C(\mathrm{SP}) = \begin{cases} 0 & \text{for } \sum_{\mathrm{DL}=0}^{L-1} C(\mathrm{BPL}, \mathrm{DL}, \mathrm{SP}, i, j) = 0 \\ 1 & \text{Otherwise} \end{cases} \tag{2}$$

$$N = \sum_{\mathrm{SP}=0}^{63} C(\mathrm{SP})$$

We define the block as a complex block, which includes many nonzero coefficients. In the complex block, we have more nonzero symbols compared to a normal block. After calculating N, we classify the complex bit-plane into four cases according to the value of N.

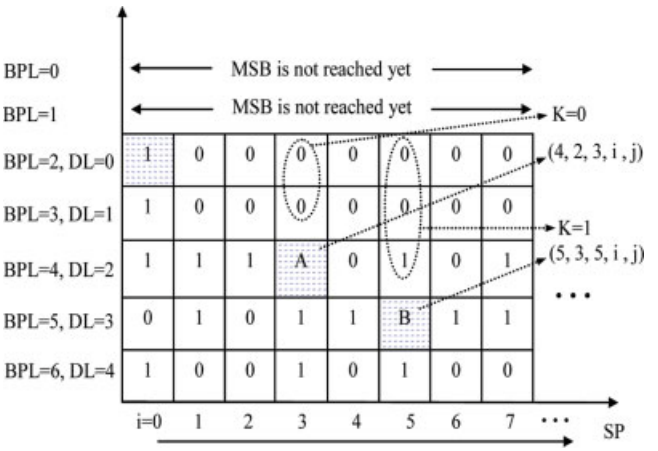

Figure 5. Contexts of a given symbol in a block. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

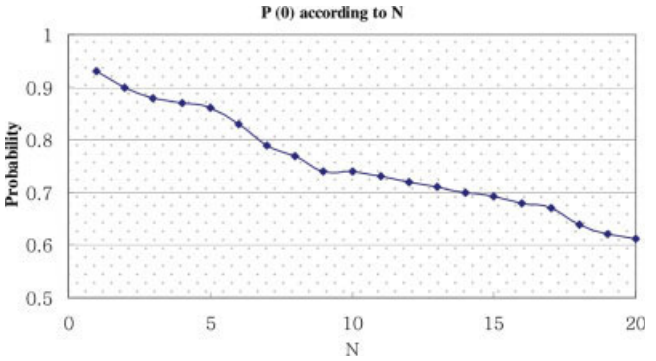

Figure 6. The various of $P(0)$ according to N. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

$$\begin{aligned} A \le N < A + \alpha \\ A + \alpha \le N < A + 2\alpha \\ A + 2\alpha \le N < A + 3\alpha \\ A + \alpha \le N \end{aligned} \tag{3}$$

Selection of A and α depends on the statistical distribution of the source data. For simplicity, in our experiments, we set A and α to the fixed values 5 and 3, respectively. The variation of $P(0)$ according to N is shown in Figure 6.

If a given symbol is neither in the first DL nor in a complex block, we use neighboring symbols in the same data level (Fig. 7). We then calculate context offset M as follows:

$$M = \sum_{m=1}^{2} C(\mathrm{BPL}, \mathrm{DL}, \mathrm{SP} - m, i, j) \tag{4}$$

Based on the context information K and M, we estimate the probability of the corresponding symbol. For some low frequency components ($\mathrm{SP} < 2$), we consider only K.

Basic procedure of probability estimation technique of the proposed entropy coding is represented in Figure 9. Binary arithmetic coding procedure in Figure 8 represents the binary arithmetic coding engine that generates a bitstream using $P(0)$ of the given symbol to code. To implement binary arithmetic coding, we followed the general rule for the arithmetic coding. At the start, the

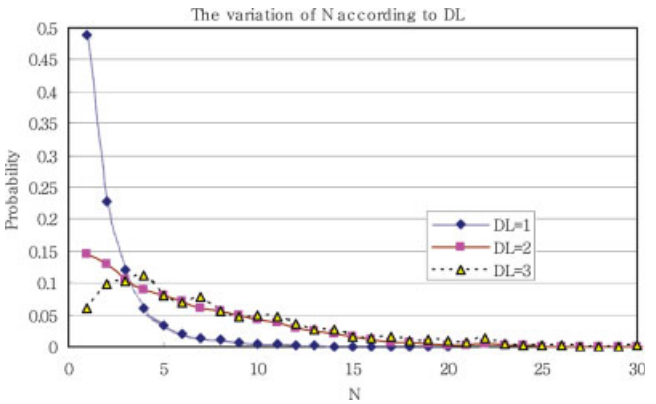

Figure 7. The various of N according to DL. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

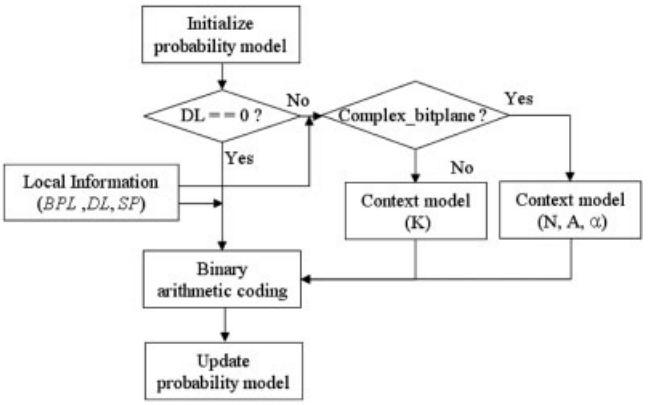

Figure 8. Basic procedure of probability estimation.

coding interval is initialized to the whole scale. We then project the estimated probability model of a new symbol to the initialized interval and obtain a new interval. The new interval for the symbol is repeatedly updated and calculated from the model scale of the new input symbols. In the meantime, we compare the new interval with the scale to determine whether there is any bit for transmission down the channel. These bits are due to the redundancy in the binary representation of lower and upper values.

D. Binary Arithmetic Coding. To implement binary arithmetic coding, we designed several steps. At the start, the coding interval is initialized to the whole scale $[0, T)$. We then project the estimated probability model of a new symbol to the initialized interval and obtain a new interval. The new interval for the symbol is repeatedly updated and calculated from the model scale of the new input symbols. In the meantime, we compare the new interval with the scale $[0, T)$ to determine whether there is any bit for transmission down the channel. These bits are due to the redundancy in the binary representation of lower and upper values. For example, if values of both lower and upper are less than half the $[0, T)$ range, then their most significant number in binary form is 0. Similarly, if both belong to the upper half range, their most significant number is 1. Hence we follow the general rule for the arithmetic coding:

In Figure 9, if the lower value is in the second quarter of the scale and the upper value is in the third quarter, then the range of

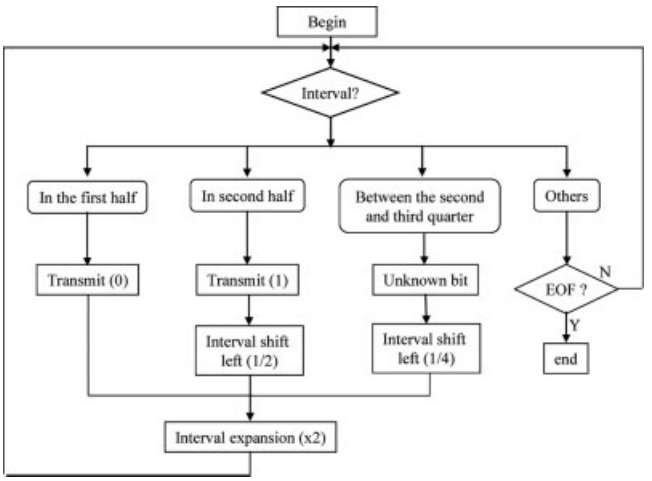

Figure 9. Basic structure of binary arithmetic coding.

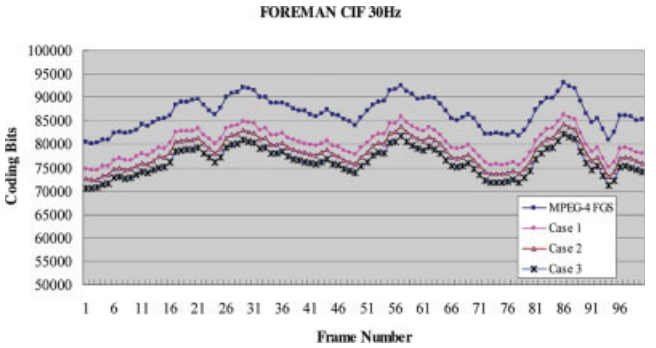

Figure 10. The number of average coding bits according to using each proposed probability estimation method. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

frequency is less than 0.5. In this case, the two most significant bits are different, 01 for the lower and 10 for the upper, but we can deduce some information from them. That is, in both cases, the second bit is the complementary bit to the first. Hence, we can find out the second bit, the previous bit is its complement. Therefore, if the second bit is 1, then the previous value should be 0, and we are in the second quarter, meaning the lower value. Then, the scaling and shifting for this case will be done only on the portion of the scale containing second and third quarters. Therefore, if the interval belongs to the second and third quarters of the scale, unknown bit is sent to a waiting buffer for later definition and the interval transformed as shown in Figure 9. If the interval belongs to more than two different quarters of the scale, the probability in the given interval is greater than 0.5. Hence, no bits are transmitted.

To verify the efficiency of the each proposed probability estimation schemes depicted in Figure 10, we compare the number of average coding bits for encoding four bit-planes in the enhancement layer by changing used probability estimation schemes for APE. In Case 1, only local information (BPL, DL, SP) is used for probability estimation process. In Case 2, local information and the context information obtained from neighboring symbols, (K, M), are used for probability estimation. In Case 3, all proposed methods depicted in Figure 8 are used for probability estimation. From the Figure 10, we can verify that each probability estimation technique provides additional coding efficiency.

IV. EXPERIMENTAL RESULTS

Through experimentation, we compared coding efficiency among three different FGS coding schemes: MPEG-4 FGS, JSVM FGS, and the proposed FGS. We have tested a large number of sequences with different motion and texture characteristics to represent various types of contents.

In the first experiment, we used the same base layer coding scheme (H.264: JM 9.5; Suehring) to evaluate coding efficiency between the proposed FGS and MPEG-4 FGS coding. Since we have encoded the base layer as the same quality, the overall PSNR values and subjective quality are exactly the same for both codecs. The only difference is the number of bits used to encode each bitplane in the enhancement layer. Table I represents the relative coding gain of the proposed FGS coding over MPEG-4 FGS in terms of the total number of bits. The results show that the proposed FGS coding is more efficient than MPEG-4 FGS in all cases, and up to 20% of bit savings can be achieved with the proposed FGS coding.

In the second experiment, we used compliance with H.264 for the base layer coding in JSVM (version 6; Joint Video Team, 2006) to evaluate coding efficiency between JSVM FGS and the proposed FGS. In the proposed FGS codec, no temporal prediction is performed for enhancement layers. In JSVM, the key picture utilizes the conventional closed loop motion compensation, and the reference pictures for the key picture shall be the base representation without FGS enhancement. On the contrary, the reference pictures of a nonkey picture shall be the base representation plus the FGS enhancement. Hence, to compare the performance of the FGS coding schemes without any temporal prediction in the enhancement layer, we apply only key pictures by selecting GOP = 1 (I P P P. . .). We then compared the rate-distortion performance between two FGS schemes. The results have been shown in Figure 11. It can be observed that the proposed FGS coding provides better rate distortion performance than the FGS coding scheme in JSVM.

Table I. Comparison of coding bit rate between CBACBP and H.264-FGS-VLC.

Sequences (CIF, 30 Hz)	Coded Bit-Planes	MPEG-4 FGS (kbps)	Proposed FGS (kbps)	Bit-Savings (%)
FOREMAN	Base + 2 bit-planes	317	290	8.51
	Base + 4 bit-planes	3,033	2,752	9.26
BUS	Base + 2 bit-planes	652	607	6.90
	Base + 4 bit-planes	4,809	4,459	7.27
CITY	Base + 2 bit-planes	508	467	8.07
	Base + 4 bit-planes	4,032	3,704	8.13
CREW	Base + 2 bit-planes	355	326	8.16
	Base + 4 bit-planes	2,840	2,596	8.59
HARBOUR	Base + 2 bit-planes	691	624	9.69
	Base + 4 bit-planes	4,723	4,245	10.12
ICE	Base + 2 bit-planes	201	184	8.45
	Base + 4 bit-planes	1,824	1,654	9.32
SOCCER	Base + 2 bit-planes	412	395	4.12
	Base + 4 bit-planes	3,830	3,655	4.56
MOBILE	Base + 2 bit-planes	845	751	11.12
	Base + 4 bit-planes	6,326	5,487	13.26
FOOTBALL	Base + 2 bit-planes	643	554	13.84
	Base + 4 bit-planes	5,107	4,350	14.82

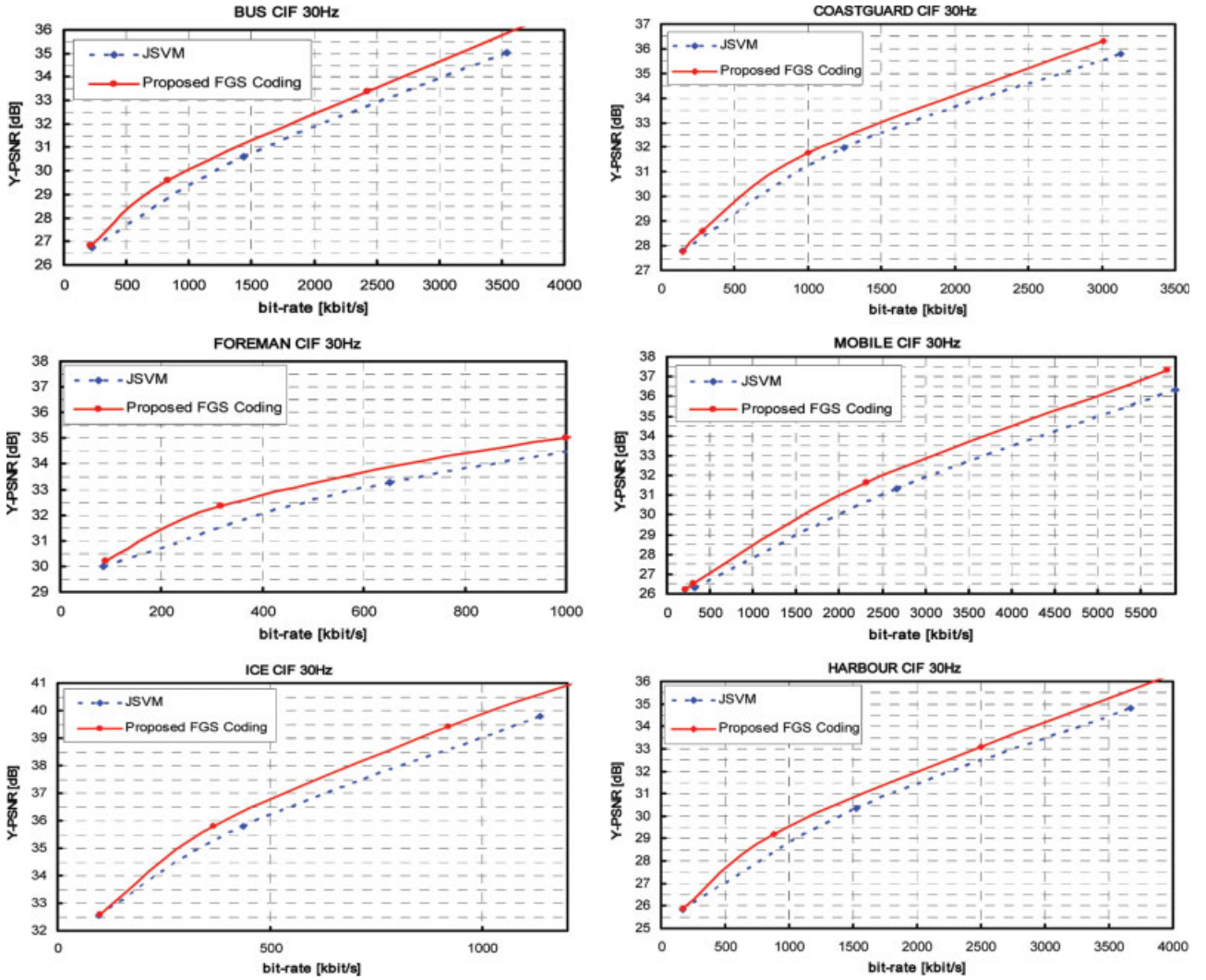

Figure 11. Comparison of R–D performance between JSVM and proposed FGS coding. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

V. CONCLUSIONS

In this paper, a new FGS coding scheme was presented. The basic structure of the proposed FGS coding is based on the traditional bit-plane coding in MPEG-4 FGS. However, to enhance coding efficiency of the bit-plane coding, we proposed a new efficient binary arithmetic coding with adaptive probability estimation. For adaptive probability estimation, various probability estimation models were designed to take advantage of the characteristics of each bit-plane and correlations of symbols among different bit-planes. Experimental results showed that the proposed FGS coding provides bit-savings up to 20% than MPEG-4 FGS, and even better rate-distortion performance than JSVM.

REFERENCES

H. Huang, C. Wang, and T. Chiang, A robust fine granularity-scalability using trellis-based predictive leak, IEEE Trans Circuits Syst Video Technol 12(6) (2002), 372–385.

Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, Joint scalable video model JSVM0, Joint Video Team, Document JVT-N021, January 2005a.

Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, Scalable video coding—Working draft 1, Joint Video Team, Document JVT-N020, January 2005b.

Joint scalable video model JSVM-6, Joint Video Team JVT-S202, Geneva, Switzerland, April 2006.

W. Li, Overview of fine granularity scalability in MPEG-4 video standard, IEEE Trans Circuits Syst Video Technol 11(3) (2001), 301–317.

D. Marpe, H. Schwarz, and T. Wiegand, Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard, IEEE Trans Circuits Syst Video Technol 13(7) (2003), 620–636.

J.-R. Ohm, Advances in scalable video coding, Proc IEEE 93 (2005), 42–56.

W.S. Peng and Y.K. Chen, Mode-adaptive fine granularity scalability, IEEE ICIP, Greece, October 2001.

A.R. Reibman, L. Bottou, and A. Basso, Scalable video coding with managed drift, IEEE Trans Circuits Syst Video Technol 13(2) (2003), 131–140.

J. Reichel, H. Schwarz, M. Wien, Joint scalable video model (JSVM) 2, ISO/IEC JTC1/WG11, m11244, Palma de Mallorca, Spain, October 2004.

J. Ridge, Y. Bao, M. Karczewicz, and X. Wang, FGS block enhancements for scalable video coding, ISO/IEC JTC1/SC29/WG11, M11509, 2004.

H. Schwarz, D. Marpe, and T. Wiegand, Basics concepts for supporting spatial and SNR scalability in the scalable H. 264/MPEG4-AVC extension, IEEE IWSSIP, Greece, 2005a.

H. Schwarz, T. Hinz, H. Kirchhoffer, D. Marpe, and T. Wiegand, Technical description of the HHI proposal for SVC CE1, IEEE IWSSIP 2005b.

K. Suehring, H. 264/AVC Reference Software, JM 9.5, http://bs.hhi.de/~suehring/tml/.

X. Sun, F. Wu, S. Li, W. Gao, and Y.-Q. Zhang, Macroblock-based progressive fine granularity scalable (PFGS) video coding with flexible temporal-SNR scalabilities, IEEE ICIP, Greece, October 2001.

F. Wu, S. Li, and Y.-Q. Zhang, A framework for efficient progressive fine granular scalable video coding, IEEE Trans Circuits Syst Video Technol 11(3) (2001) 332–344.