Synchronous backward error tracking algorithm for H.264 video coding

Ming-Kuang Tsai[a], Tien-Hsu Lee[a], Jong-Tzy Wang[*b], Pao-Chi Chang[a]
[a]Department of Communication Engineering, National Central University, Zhongli, Taiwan 320;
[b]Department of Electronic Engineering, Jin Wen Institute of Technology, Xindian, Taiwan 231

ABSTRACT

The objective of this paper is to develop a robust error-resilient algorithm, called the Synchronous Backward Error Tracking (SBET), to completely terminate the error propagation effects in the error-prone environment for H.264 video coding. The motivation is that if the state of the decoder is available to the encoder, i.e., the state of the encoder can synchronize to the state of the decoder, the effect of error propagation can be entirely terminated because all predictions are based on the same references. Therefore, we assume that a feedback channel is available and the encoder can be aware of the decoder's error concealment by any external means. The pixel-based Precise Backward Error Tracking (PBET) is modified and utilized to track the error locations and reconstruct the state of the decoder in the encoder. The proposed method only involves memory access, simple addition and multiplication operations for the error-contaminated pixels to achieve encoder-decoder synchronization. By observing simulation results, the rate-distortion performance of the proposed algorithm is always better than that of the conventional algorithms. Specifically, SBET outperforms PBET up to 1.21 dB under 3% slice error rate for the QCIF format *Foreman* sequence. In addition, instead of forced INTRA refreshing, the phenomenon of burst bit rate can be avoided.

Keywords: H.264, error tracking, error concealment, feedback channel, error resilience

1. INTRODUCTION

In the recent year the application of video communication has become popular. However, with the much higher compression efficiency, coded video data is prone to suffering from transmission errors. H.264 [1] is the newest video coding standard formulated by JVT (Joint Video Team of ITU-T Video Coding Experts Group [2] and ISO/IEC Moving Picture Experts Group) in 2001 and finalized in 2003. It utilizes complicated predictions in temporal and spatial domains to enhance the coding efficiency, such as the directional prediction in intra coding, variable-block size, multiple reference picture, and quarter-pixel-accurate motion compensation in inter coding. Besides, more efficient variable length code (VLC), in-loop de-blocking filter and no-mismatch integer transform are also included in the standard to enhance the video quality [3]. By using the above tools, H.264 has better performance than other existing standards. However, some of these techniques are inherently extremely sensitive to the transmission errors. An error occurred in the bit stream may propagate in both spatial and temporal domains and consequently cause serious quality degradation [4].

Due to the use of intra prediction, errors may spread from neighboring causal MBs to the inner sub-blocks of current MB. With the multiple-reference-frame prediction, variable-block-size motion estimation, and sub-pixel generation, the error may broadcast from preceding frames directly or indirectly and also propagate from the neighboring MBs. On the other hand, owing to the utilization of VLC, the erroneous compressed data usually cannot be correctly decoded until the next resynchronization point. Moreover, as a result of using de-blocking filter, errors may also diffuse from adjacent MBs, like the effect of motion compensation. Consequently, because of the out-of-synchronization between encoder and decoder states, the error may propagate in both spatial and temporal domains. Thus, the reconstructed video quality is deteriorated and the error-propagation continues.

To combat with the problem, H.264 provides many error resilient tools [5], such as the flexible macroblock order (FMO), redundant slice (RS), parameter sets, etc. FMO alters the way how pictures are partitioned into slices and MBs

[*] This work was supported by the National Science Council under Grant NSC 93-2213-E-228-006, Taiwan, R.O.C.

Image and Video Communications and Processing 2005, edited by Amir Said,
John G. Apostolopoulos, Proc. of SPIE-IS&T Electronic Imaging,
SPIE Vol. 5685 © 2005 SPIE and IS&T · 0277-786X/05/$15

by employing the concept of slice groups. Every slice group consists of one or more slices and a slice is a sequence of MBs. When using FMO, a picture can be split into many MB scanning patterns such as interleaved slices. RS allows the encoder to place, in addition to the coded MBs of the slice itself, one or more redundant representations of the same MBs into the same bit stream. The utilization of RS is that the redundant representation can be coded using different coding parameters, such as QP or reference frame index. The parameter set contains information that is expected to rarely change and offer the decoding of a large number of VCL NAL units. The sequence and picture parameter-set mechanism effectively decouples the transmission of infrequently changing information from the transmission of coded representation of the values of the samples in the video pictures.

Although H.264 provides many error resilient tools, they still can't terminate the error propagation entirely. To do that, several techniques, like automatic repeat request (ARQ) [6], reference frame selection (RPS) [7], and error tracking (ET) [8], are proposed. ARQ scheme retransmits the lost data to bring about extra delay and bit rate. RPS selects the non-error-contaminated frame to achieve synchronization between the encoder and decoder but reduces the coding efficiency. ET uses lost information to track the decoder behavior and stop error propagation by intra-block refresh in the encoder. Consequently, in this paper we'll propose a robust error resilient algorithm based on the ET. If the state of the encoder can be synchronous to that of the decoder, the error propagation effects can be completely terminated. Therefore, it is assumed that a feedback channel is available and the encoder can know the decoder's error concealment by certain external means. The pixel-based backward ET [9] is used to track the error locations and propagate the concealment error of the erroneous frame to the corresponding areas to reconstruct the state of the decoder in the encoder. Comparing with the full re-encoding approach, the proposed algorithm only involves memory access, simple multiplication and addition operations for the error-contaminated pixels and effectively recovers the visual quality.

This paper is organized as follows. The details of the proposed algorithm named Synchronous Backward Error Tracking (SBET) are described in the next section. Simulation results and comparisons of the proposed algorithm are given in Section 3, and at last, this research work is concluded.

2. SYNCHRONOUS BACKWARD ERROR TRACKING ALGORITHM

Once the feedback information about unsuccessfully decoded image blocks is received, the encoder can perform the corresponding error concealment for the erroneous frame and propagate the concealment error to the frames between the erroneous one and the current one by error tracking. Note that these frames already have been stored in the frame buffer for the multiple-reference-frame motion estimation. While performing error tracking, some information, such as motion vector, MB type, intra prediction mode, reference frame index, and de-blocking filter mode, needs to be additionally recorded. To accomplish the proposed method, it's necessary to first calculate the initial concealment errors and distribute them to the corresponding frames within the round-trip delay. In this manner, the state of encoder and decoder can synchronize and the error propagation effect can be entirely stopped.

Here the condition of ignoring the de-blocking filter in H.264 is considered first. The following formulas illustrate how to spread the concealment error to the succeeding frames [10]. $E_n(x,y)$ is the concealment error at (x,y) location in the n^{th} frame, $X_n(x,y)$ is the error-free data, and $\tilde{X}_n(x,y)$ is the data after compensating the concealment error.

$$E_n(x,y) = \tilde{X}_n(x,y) - X_n(x,y) \quad \Rightarrow \quad \tilde{X}_n(x,y) = X_n(x,y) + E_n(x,y) \tag{1}$$

The updated pixel value of frame *n* within the round-trip delay can be derived from the previous frame,

$$\begin{aligned} &\tilde{X}_n(x,y) \\ &= D_n(x,y) + \tilde{X}_{n-1}(x-\delta x, y-\delta y) \\ &= D_n(x,y) + X_{n-1}(x-\delta x, y-\delta y) + E_{n-1}(x-\delta x, y-\delta y) \\ &= X_n(x,y) + E_{n-1}(x-\delta x, y-\delta y) \end{aligned} \tag{2}$$

where $D_n(x,y)$ is the residual, $(\delta x, \delta y)$ is the motion vector at (x,y) location, $X_n(x,y)$ is the original pixel data stored in the frame buffer, and $\tilde{X}_n(x,y)$ is the updated pixel value after propagating the concealment error.

In the following, the in-loop de-blocking filter in H.264 is taken into account. $X'_n(x,y)$ is the error-free pixel data, $\tilde{X}_n(x,y)$ is the data that has passed through the motion compensation before de-blocking, and $\tilde{X}'_n(x,y)$ is the data after passing through the motion compensation and de-blocking.

$$E_n(x,y) = \tilde{X}'_n(x,y) - X'_n(x,y) \quad \Rightarrow \quad \tilde{X}'_n(x,y) = X'_n(x,y) + E_n(x,y) \tag{3}$$

The updated pixel value of frame *n* within the round-trip delay can be derived from the previous frame,

$$\begin{aligned}
&\tilde{X}'_n(x,y) \\
&= \tilde{X}_n(x,y) * W \\
&= \{D_n(x,y) + \tilde{X}'_{n-1}(x-\delta x, y-\delta y)\} * W \\
&= \{D_n(x,y) + X'_{n-1}(x-\delta x, y-\delta y) + E_{n-1}(x-\delta x, y-\delta y)\} * W \\
&= \{D_n(x,y) + X'_{n-1}(x-\delta x, y-\delta y)\} * W + \{E_{n-1}(x-\delta x, y-\delta y)\} * W \\
&= X'_n(x,y) + \{E_{n-1}(x-\delta x, y-\delta y)\} * W
\end{aligned} \tag{4}$$

where W is the weighting matrix of de-blocking filter, $X'_n(x,y)$ is the original pixel data stored in the frame buffer, and $\tilde{X}'_n(x,y)$ is the updated pixel value after spreading the concealment error.

From observing the formula (4), the updated pixel value is gained by adding the original data to the concealment error multiplied by weighting values of the employed de-blocking filter that need to be additionally recorded. That is, assuming a 3-tap de-blocking filter is utilized, it's necessary to find out the corresponding weighting matrix and concealment error. For example, A' = A + W x E_A will be changed to A' = A + (W_1 x E_A + W_2 x E_B + W_3 x E_C) due to a 3-tap filter. However, in order to decrease the computation, we can ignore the de-blocking effect by setting W_1 = 1, W_2 = W_3 = 0. Consequently, it's only necessary to track E_A without saving the de-blocking filter index. The required memory can also be diminished.

In summary, if feedback information about error location *k* in frame *n-d* is received before encoding frame *n*, the encoder tracks and propagates the error magnitudes via the following steps:

1) Extract the data of *k* in frame *n-d*. This is the error-free version.
2) For frame *n-d*, perform the error concealment and update de-blocking-related variables. And then, execute de-blocking again.
3) Initialize the concealment error by subtracting the results of the above two steps.
4) Perform the pixel-based backward error tracking (PBET) to find out the corresponding concealment error for frame *n-d+1* by utilizing motion vectors.
5) Modify the encoder frame buffer by adding the weighted concealment error. By the way, the concealment error of current frame is also obtained.
6) Repeat 4) and 5) for the following frames until frame *n-1*. Then the original coding procedure continues.

In step 2), some information such as the MB type, CBP, reference frame index, and motion vector should be recorded to execute de-blocking. These parameters also need to be revised after performing the error concealment. In step 4), we only need to check the neighbors for the corrupted MB instead of all MBs in the corrupted frame. In this way, the computation can be decreased. After accomplishing the above procedure, the frame buffer state of encoder and decoder will suppose to be the same. Synchronization between the encoder and the decoder will be achieved in the n^{th} frame if the frames within the round-trip delay (i.e., *d* frames) aren't further subject to transmission errors. Therefore, the error propagation effects will entirely terminate. The proposed method only involves memory access, simple addition and multiplication operations that are suitable for realistic implementation.

3. SIMULATION RESULTS

The parameters used in the simulations are listed as follows:

- H.264 test model software version: Joint Model (JM) 7.3
- Image format: QCIF (176 x 144)
- Encoded sequence type: I P P P…
- Frame rate: 7.5 frames per second
- Slice mode: fixed number of MB
- Number of MB in a slice: 11
- Constrained intra prediction: ON
- Error concealment in the decoder: ST, TR

The adopted error patterns are supplied by JVT with packet error rate 3% and 5%. Meanwhile, it is also under the assumption that the round-trip delay is two frames (267 ms) and one slice is packetized into one packet. The feedback information will inform the erroneous slice. ST means the error concealment of weighted pixel value averaging for intra frame and boundary matching based motion vector recovery for inter frame. TR means the error concealment of temporal replacement from the preceding frame.

In the comparison of R-D curves shown as Fig. 1, SBET performs better than PBET. Specifically, SBET outperforms PBET up to 1.21 dB under 3% slice error rate for the QCIF format *Foreman* sequence. When the bit rate is increased gradually, the phenomenon of burst bit rate due to the intra refresh becomes unobvious, thus the two curves are close to each other. While observing Fig. 2, the curve of SBET is close to that of PBET owing to more intra blocks are generated under higher motion. From Fig. 3, the two ET curves are far from error-free curve as a result of error rate increase. Moreover, the bended degree of SBET curve is more obvious than that of PBET because the worse reconstructed frame causes extra cost while encoding the next frame. In addition, in Fig. 4 we can find that when a better error concealment technique is utilized, the performance of SBET is also becoming better. Finally, the simplified version of SBET (i.e., SBET without consideration of de-blocking; SBET_noD) shows nearly the same performance as the original SBET in Fig. 5.

In the comparison of complexity, first we'll focus on the computation quantity of PBET and SBET. Here a frame with 176 x 144 image format and 4:2:0 sub-sampling is assumed. There are (16 x 16 + 8 x 8 + 8 x 8) = (16 x 16 x 3 / 2) pixels in a MB, (176 / 16) MBs in a row, (176 x 144 / 256) MBs in a frame. Every frame is referred to the previous one frame and the round-trip delay is also assumed to be two frames. In the PBET algorithm, 2 x 2 additions, 0 multiplication, and 1 decision are needed when every pixel is tracking for contamination. The total computation is listed in Table 1.

In SBET algorithm, it's assumed that the TR error concealment is used. One erroneous slice's address is sent back to the encoder via the feedback channel and the 3-tap de-blocking filter is utilized. For each pixel of erroneous frame, 1 addition is employed to initialize the concealment error, so that total (16 x 16 x 11) x 3 / 2 = 4,224 additions are required. For the succeeding frames, every updated pixel ($A' = A + (W_1 \times E_A + W_2 \times E_B + W_3 \times E_C)$) requires (2 x 3 + 3) additions and 3 multiplications. There are 3 x 176 / 16 x (16 x 16 x 3 / 2) = 12,672 pixels need to be tracked for the first propagated frame. The overall computation is shown in Table 2.

When $W_1 = 1$ and $W_2 = W_3 = 0$ are taken into account, i.e., SBET_noD, we only need to track E_A. The whole computation is shown in Table 3. From Table 1 and 2, the addition operation of SBET is 1.16 times of PBET. By comparing Table 2 and 3, the addition operation of SBET_noD is only 0.36 times of SBET and the multiplication operation of SBET_noD decreases to 0. The computation is greatly reduced. In addition, the performance of SBET and SBET_noD are close to each other.

Finally, the following discusses the memory requirement. We assume that five reference frames are utilized in the motion compensation and all encoded block size is 4 x 4. For PBET, sub-mode of intra prediction, reference frame index, and motion vectors of inter prediction are required to be stored as listed in Table 4. In SBET, in addition to the above, some de-blocking information like CBP, QP, and filter index are needed and the concealment error for each frame is also desirable. As shown in Table 5, the required memory of SBET is 7.4 times of PBET, and the main differences are the de-blocking information and concealment error. Noticeably, the demanded memory of SBET_noD listed in Table 6 is much smaller than SBET and more suitable for practical usage.

4. CONCLUSIONS

By knowing the state of decoder, the encoder could reconstruct the error propagation effect by employing the proposed SBET algorithm. Only memory access, simple addition and multiplication operations are involved. According to the simulation results, the rate-distortion performance of the proposed algorithm is always better than that of the conventional algorithms. In addition, without utilizing the forced intra refreshing, the phenomenon of burst bit rate could be avoided. In the future, if a better error concealment technique is utilized, a better performance of SBET is also expected.

REFERENCES

1. ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC, *Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification*, 2003.
2. Thomas Wiegand, "*H.26L Test Model Long-Term Number 9 (TML-9) draft*," document VCEG-N83 d1, ITU-T Video Coding Experts Group (VCEG) Meeting, 21 Dec. 2001.
3. T. Wiegand, G. J.Sullian, G. Bjontegaard, and A.Luthra, "*Overview of the H.264/AVC video coding standard*," IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 560-576, July 2003.
4. T. Stockhammer, M. Hannuksela, and T. Wiegand, "*H.264/AVC in wireless environments*," IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 657-673, July 2003.
5. S. Wenger, "*H.264/AVC over IP*," IEEE Trans. Circuit Syst. Video Technol., vol. 13, pp. 645-656, July 2003.
6. S. Lin, D.J. Costello, and M.J. Millier, "*Automatic repeat error control schemes*," IEEE Commun. Mag., vol. 22, pp. 5-17, 1984.
7. S. Fukunaga, T. Nakai, and H. Inoue, " *Error resilient video coding by dynamic replacing of reference pictures*," Proc. IEEE GLOBECOM, vol. 3, pp. 1503-1508, Nov. 1996.
8. E. Steinbach, N. Farber, and B. Girod," *Standard compatible extension of H.263 for robust video transmission in mobile environments*," IEEE Trans. Circuits Syst. Video Technol., vol. 7, pp. 872-881, Dec. 1997.
9. P.C. Chang and T.H. Lee, "*Precise and fast error tracking for error resilient transmission of H.263 video*," IEEE Trans. Circuits Syst. Video Technol., vol. 10, pp. 600-607, June 2000.
10. R. Vedantham and A. Nosratinia, "*Video error resilience through efficient shadowing of decoder*," 41st Annual Allerton Conference on Communications Control and Computing, Monticello, IL, Oct. 2003.

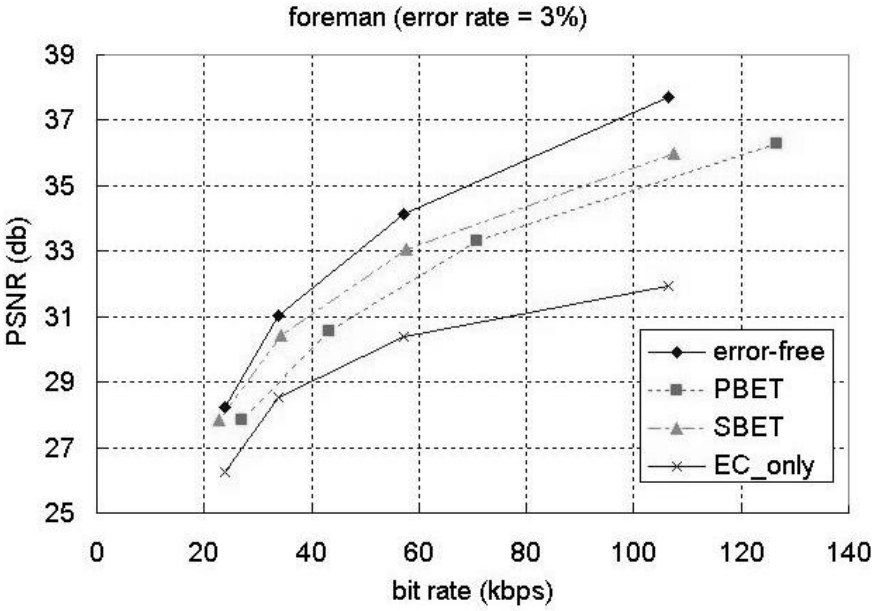

Figure 1: R-D curve comparison of PBET and SBET for *Foreman* under 3% error rate.

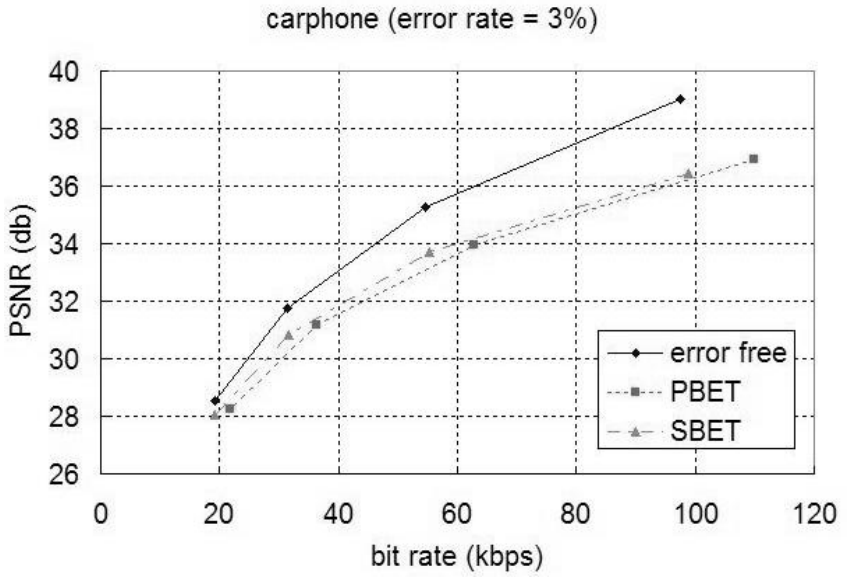

Figure 2: R-D curve comparison of PBET and SBET for *Carphone* under 3% error rate.

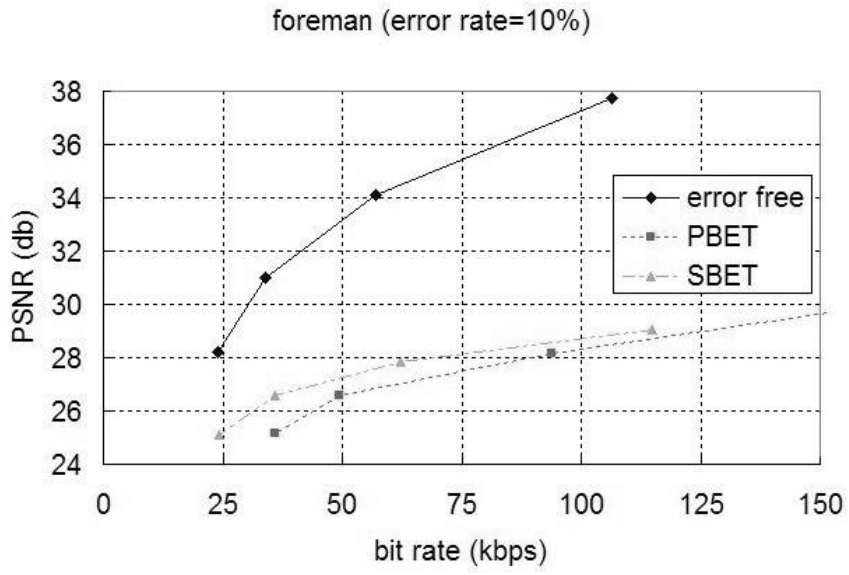

Figure 3: R-D curve comparison of PBET and SBET for *Foreman* under 10% error rate.

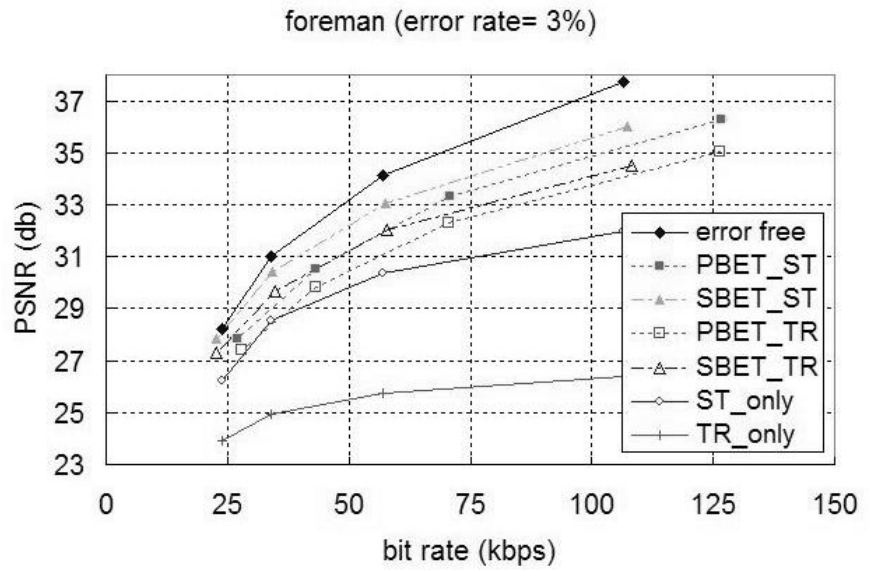

Figure 4: R-D curve comparison of PBET and SBET for different error concealment methods.

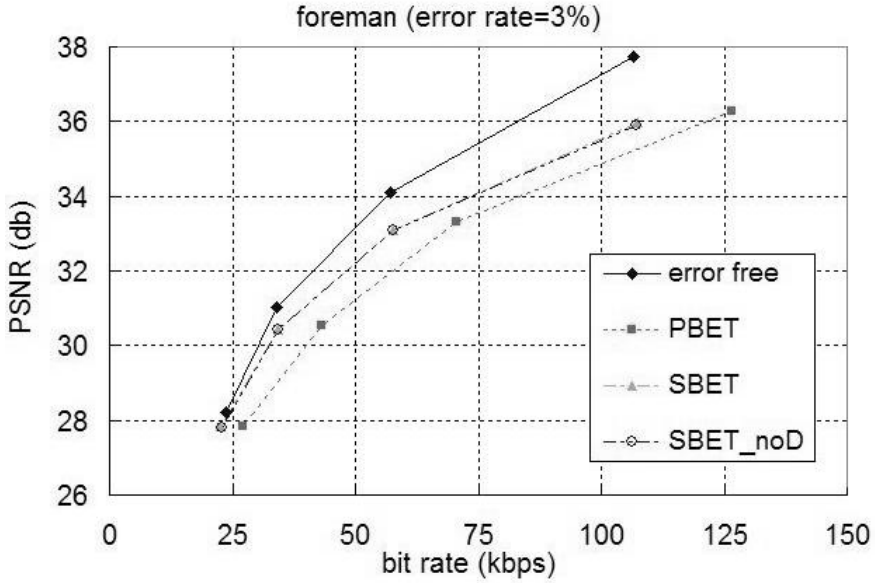

Figure 5: R-D curve comparison of SBET and it's low-complexity mode.

Table 1: Computation complexity of PBET.

Addition	(176 x 144) x (2 x 2) = 101,376
Multiplication	0
Decision	(176 x 144) = 25,344

Table 2: Computation complexity of SBET.

Addition	4224 + 12672 x (2 x 3 + 3) = 118,272
Multiplication	12672 x 3 = 38,016
Decision	3 x 176 / 16 x (16 x 16 x 3 / 2) = 12,672

Table 3: Computation complexity of SBET_noD.

Addition	4224 + 12672 x (2 x 1 + 1) = 42,240
Multiplication	0
Decision	(3 x 176 / 16 x (16 x 16 x 3 / 2) = 12,672

Table 4: Required memory for PBET.

MBtype	11 x 9 x 5
Intra prediction mode	(11 x 4) x (9 x 4) x 5
Reference frame index	(11 x 4) x (9 x 4) x 5
Motion vector	(11 x 4) x (9 x 4) x 5 x 2
Total	32,175 bytes

Table 5: Required memory for SBET.

MB type	11 x 9 x 5
Intra prediction mode	(11 x 4) x (9 x 4) x 5
Reference frame index	(11 x 4) x (9 x 4) x 5
Motion vector	(11 x 4) x (9 x 4) x 5 x 2
CBP	(11 x 4) x (9 x 4) x 5
QP	(11 x 4) x (9 x 4) x 5
Filter index	(176 x 144) x (3 / 2) x 5
Concealment error	(176 x 144) x (3 / 2) x 1
Total	268,191 bytes

Table 6: Required memory for SBET_noD.

MB type	11 x 9 x 5
Intra prediction mode	(11 x 4) x (9 x 4) x 5
Reference frame index	(11 x 4) x (9 x 4) x 5
Motion vector	(11 x 4) x (9 x 4) x 5 x 2
CBP	(11 x 4) x (9 x 4) x 5
QP	(11 x 4) x (9 x 4) x 5
Concealment error	(176 x 144) x (3 / 2) x 1
Total	78,111 bytes