Proc. VIIth Digital Image Computing: Techniques and Applications, Sun C., Talbot H., Ourselin S. and Adriaansen T. (Eds.), 10-12 Dec. 2003, Sydney

ECC Video: An Active Approach for Error Resilience

Bing Du, Anthony Maeder and Miles Moody

School of Electrical and Electronic Systems Engineering
Queensland University of Technology, Brisbane, Australia

Abstract. Video coding standards have incorporated various error resilience and error concealment tools, yet data losses during video transmission in mobile systems still pose big challenges. Here an approach for error resilience in the application layer is proposed, employing SEC (second error control) with ECC (error correction coding) and some simulation results are presented.

1 Introduction

An inherent problem with video communication is that compressed video using current video coding standards is very vulnerable to channel noise, especially in mobile or wireless environments. Current video coding standards including MPEG-1 [9], MPEG-2 [10], MPEG-4 [1], H.261 [11] and H.263 [7] employ DCT, motion estimation and compensation and variable length coding techniques to exploit spatial and temporal redundancy in natural video sequences, to achieve desired compression efficiency. While these techniques achieve good compression ratios, the resulting compressed bitstream is highly sensitive to bit errors. For instance, a single bit error in the compressed bitstream may propagate until the next synchronization point in the bitstream, which will greatly degrade the quality of the decoded video. Diverse error resilience and error concealment techniques have thus been proposed [4,13]. Compatibility of any new approach with current video coding standards is desirable to maintain interoperability.

In the MPEG-4 standard, five error resilience coding tools have been incorporated. *Resynchronisation* or packetization attempts to stop error propagation after an error or errors have been detected, by inserting resynchronisation markers into the bitstream. Data between the synchronization point prior to the error and the first point where synchronization is re-established, is usually discarded. Different from resynchronisation, *Data Partitioning* (DP) is an error concealment tool which is realised by separating the motion and macroblock header information from the texture information. If the texture information is lost, this approach utilizes the motion information to conceal these errors. *Reversible VLC* (Variable Length Code) is designed such that a bitstream can be instantaneously decoded in both forward and reverse directions. A part of a bitstream, which cannot be decoded in the forward direction due to the presence of errors, can often be decoded in the backward direction, recovering some information which would otherwise have been discarded. However, RVLC is only applied to TCOEF (Texture Coefficient) coding at this stage. *Adaptive Intra Refresh* (AIR) is a technique using the intra refresh method to provide error resilience. In AIR mode, the moving part of the frame is encoded frequently in the Intra mode. Since corruption of the bitstream is more visibly noticeable in the moving parts of a video sequence, AIR will help give quick recovery from any errors. *NEWPRED* mode is another error resilience tool in which the reference picture for inter-frame coding is replaced adaptively according to upstream messages from the decoder.

Obviously AIR will increase the encoded bitstream rate significantly, while NEWPRED needs upstream messaging from the decoder, which may not be practical in some situations, especially in a multi-party video communication system. While Resynchronisation, Data Partitioning and Reversible VLC do give protection to the encoded bitstream, they also have their disadvantages. Firstly, they are passive as the loss of information caused by discarding bits in error is unrecoverable, so with the inter-frame error propagation effect the reconstructed video quality rapidly degrades to unrecognizable if no other measure is taken. Secondly, while bringing error resilience, they also introduce vulnerability. If the errors happen to be within markers (including resynchronisation marker, DC marker or motion marker), the decoder will lose synchronization, meaning a whole packet or even several packets must be discarded. Finally, there is an associated loss of efficiency with this scheme. In our simulation with video sequences Salesman and Akiyo, with the packet size set to 600 bits, packetization combining DP and RVLC results in a bitrate increase of over 9.9% in the final bitstream.

Generally, reducing packet size may further increase the robustness of the encoded bitstream, at the cost of coding efficiency. However, there is a limit on the effectiveness of improving the robustness by reducing the packet size due to the reasons stated above. So in some extreme channel conditions, an acceptable quality of video communications becomes impossible to achieve by simply employing the error resilience tools in MPEG-4. An example is that when the packet is so small that it contains only one macroblock, the bitstream will be more vulnerable than not using packetization, as the markers will take a higher percentage of the bitstream. Obviously other tools are needed.

2 Second Error Control (SEC)

Regarding the disadvantages of the current error resilience tools, we can ask whether we have to accept the error bits in the final video bitstream? If the answer is yes, then the current available error resilience techniques are the only choices, meaning we will have to accept the poor quality of real-time video transmission associated with these techniques. If the answer is no, we have to find some mechanism to correct these errors before final video decoding, so some kind of error control in the application layer is needed. To apply second error control (SEC) in the application layer after the first error control in the data link layer sounds unrealistic because of the huge overhead associated with the usual error control. Obviously employing ARQ in SEC is not realistic after the first error control, which probably has used up all the time limit allowed for retransmission with ARQ. A better option is to use error correction coding (ECC). Now another question arises, is there an effective error correction code with extremely high coding efficiency? With an overhead increase of more or less 9.9% in the final bitstream for error resilience, can we do something better? Considering the capability that PCC (punctured convolutional code) [3,12] can exhibit, the answer to the question may be yes. If ECC is applied to the application layer to correct those bits in error, that means we are going to take a SEC approach for real-time applications. Obviously SEC is an active error protection approach in the sense that it can recover the corrupted bitstream by correcting the errors in the bitstream. Does it work? Is PCC efficient enough? Before these questions can be answered, we must understand the fundamentals of PCC.

3 Punctured Convolutional Coding (PCC)

Convolutional coding [2] with its optimum Viterbi decoding, is an effective forward error correction (FEC) technique to improve the capacity of a channel: it is commonly used in channel coding. However there is no reason why it cannot be used with source coding. Convolutional codes are usually described using two parameters: the code rate and the constraint length. The code rate, *k/n*, is expressed as a ratio of the number of bits of the convolutional encoder (*k*) to the number of channel symbols output by the convolutional encoder (*n*) in a given encoder cycle. The constraint length parameter, *K*, denotes the "length" of the convolutional encoder, i.e. how many *k*-bit stages are available to feed the combinatorial logic that produces the output symbols. Closely related to *K* is the parameter, *m*, which indicates how many encoder cycles an input bit is retained and used for encoding after it first appears at the input to the convolutional encoder. The *m* parameter can be thought of as the memory length of the encoder. Increase of *K* or *m* usually improves the performance of convolutional codes.

A punctured convolutional code (PCC) is a high rate code obtained by the periodic elimination of specific code symbols from the output of a low rate encoder. The performance of punctured convolutional codes is degraded compared with the original codes, however the degradation is rather gentle as the coding rate increases from ½ to 7/8 and even higher. From existing work [3] it can be seen that:

1. For the same rate punctured codes, the coding gain increases by 0.2 – 0.5dB with the increase of the constraint length *K* by 1.
2. Although the coding gain of punctured codes decreases as the coding rate increases, the coding gain is still high even for the high rate punctured codes. For example, a 13/14 code provides a coding gain larger than 3dB if *K* >= 7.

4 Error Correction Coded (ECC) Video

In this work a new approach is proposed, in which the encoded bitstream is not protected by resynchronisation, data partitioning and RVLC; instead it is protected with ECC, which is realized with PCC in this work. After each video frame is encoded, the bitstream is segmented into segments and each segment is further encoded using ECC. There are three reasons for choosing convolutional coding. First, convolutional coding is more suited to mobile channels: enhanced with interleaving, it is very good at coping with bursty errors and packet loss. Second, it is easier to adapt the rate of error correction coding with PCC according to the residue error conditions. Here the residue error refers to the errors left to the application layer by the network after first error control takes place in data link layer. Third, when punctured, convolutional codes can achieve high coding rates while still retaining good error correction capability.

A generalised video transmission system employing ECC is shown in Figure 1. The interleaving operation provided in the system is to cope with bursty errors. The interleaving principle is to spread the bursty error into a wide range, by reordering the encoded video bitstream data which has gone through compression and the ECC procedure, and so to make it easier for the convolutional decoder to correct errors in the bitstream. More detail on interleaving to cope with bursty errors is found in [14].

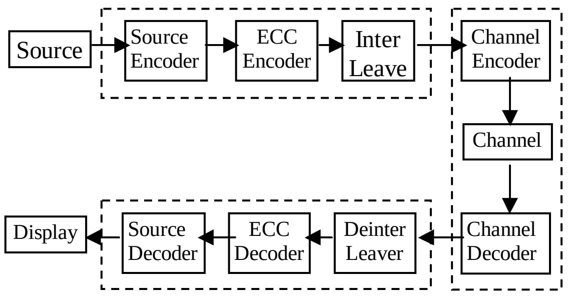

Figure 1: Video Communications System with ECC and Interleaving

From Figure 1 it is clear that here ECC enhanced with interleaving is not part of channel coding, although PCC has been widely used as a channel coding scheme in the data link layer of many other communication systems. Instead it is part of source coding for error resilience purposes: more precisely it is a SEC approach in addition to the first error control (conventional error control) in the data link layer. Obviously the operation of ECC on the compressed video bitstream is different from the ordinary forward error correction (FEC) technique commonly employed in the data link layer of the network, though in principle they take similar roles in correcting the errors in the bitstream. First, FEC is usually employed as a channel coding mechanism to improve the capacity of a channel and often combined with ARQ (automatic repeat request). From the point of view of the layered structure of telecommunication networks, FEC usually exists in the data link layer while ECC on an encoded video bitstream is part of the application layer, therefore is considered as part of the source data by FEC. Second, the design and choice of FEC usually depends on the channel conditions and the associated ARQ mechanism, while the design and choice of ECC depends on the capability of the network to combat the errors in the telecommunication channels. Third, FEC works on the original errors existing in the unfavorable communication channel, while ECC works on the residue error left in the source data by the network. In another words, FEC belongs to first error control while ECC belongs to second error control.

5 Simulation Results

To evaluate the effectiveness of the proposed algorithm, two video sequences are used: Akiyo with relatively slow motion and Salesman with fast movement. For each, the PSNR (Peak Signal-to-Noise Ratio) of the resulting video is compared for the bitstream protected with ECC, against the bitstream protected with packetization. The experiments are conducted based on the following conditions.

1. 50 picture frames of each video sequence are encoded with the first frame coded as I frame followed by P frame without rate control.
2. Packet size of both video sequences is 600 bits when packetization is used.
3. When ECC is employed, the ½ rate base convolutional code (561, 752) is chosen which has a constraint length of K=9. This base code is punctured to rate 11/12, which means that for every 11 bits in the encoded bitstream using the MPEG standard, another bit is added after convolutional encoding.

4. Convolutional encoded bitstream is decoded using the soft decision Viterbi decoding algorithm with trellis depth of 21x*K*.
5. Data partitioning and RVLC are employed with packetization, but not with ECC.
6. The same quantization parameters are used in all experiments, which means that correctly decoded bitstreams should have the same visual quality for the same video sequence in an error free environment.
7. In each test, the encoded bitstream is randomly error corrupted with Gaussian noise before being decoded, with BER (Bit Error Rate) set to 10^{-3} and 10^{-4} respectively.
8. After the corrupted bitstreams are decoded, the erroneous motion vectors and texture information are replaced by 0. This means that when motion vectors are not available, motion compensation is implemented by re-using the motion vectors from the same position in the previous frame. When the texture information is not available, the block is reconstructed using texture information from blocks referenced by the motion vectors.

To better express our results, we adopt the following notation. By ECC(11/12) we mean the ECC scheme is used at the application layer with ECC rate set to 11/12. By Packetization(600), we mean packetization is used with a packet size of 600 pixels. The simulation experiments reported in this work are mainly focused on the performance of ECC in random error situations, as the performance of ECC in bursty error and loss has been proved (for which more details can be found in [5]). The simulation results obtained (PSNR versus frame number) with random errors by averaging results from 100 individual tests, are shown in Figures 2 and 4. Figures 3 and 5 are zoomed in versions of these, to make the viewing of the results easier and clearer. Coding rate comparisons between ECC and packetization are shown in Tables 1 and 2. The advantage when using ECC with soft decision Viterbi decoding instead of packetization is clearly seen. ECC video with soft decision Viterbi decoding delivers decent reconstructed video even when the BER of the final video bitstream reaches 10^{-3}, with PSNR which is less than 1db lower compared with transmission-error free situation. In contrast, packetization cannot deliver a decent reconstructed video output while decoding the corrupted video bitstream even when the BER of the final video bitstream only reaches 10^{-4}.

When the BER of the final bitstream reaches 10^{-4}, ECC(11/12) delivers a video output with PSNR which is nearly the same as the transmission-error free situation for video sequences Salesman; while for Akiyo it delivers a video output with PSNR which is absolutely the same as the transmission-error free situation (see Figure 3, where the PSNR line for BER of 10^{-4} is coincident with the PSNR line for transmission-error free). In 99 out of 100 tests, the ECC corrects all of the transmission errors in the bitstream, leaving only one test with 5 bits in error for Salesman and 3 bits in error for Akiyo for the 50 frames after ECC decoding.

Here another question can be raised, how can a bitstream with transmission-errors still deliver a video output with PSNR which is the same as the transmission-error free situation? The answer is that when the bits in error in the bitstream correspond to the reconstructed picture areas, where there is no movement in the content of the picture, a basic error concealment operation (i.e. copying the corresponding area from the previous frame) will conceal the error effect completely.

With further negligible increase of ECC rate, our experiments also reveal that when ECC(9/11) is used, the ECC operation corrects all the transmission errors in the bitstreams of both sequences Akiyo and Salesman for all the 100 tests when the BER of the final bitstream is set to 10^{-4}, delivering transmission-error free reconstructed video output. Based on 100 tests, our experiments also reveal that ECC with coding rate of 7/8 will correct all the transmission errors in the bitstream of these two video sequences, when the BER of the final video bitstream reaches 10^{-3}. In these cases, the PSNRs of the video outputs are identical to the PSNRs of video sequences transmitted in an error free situation and there is no point to depict separate PSNR plots. The 11/12 puncturing rate of the convolutional code results in a 9.2% increase of final bit rate and the 9/10 puncturing rate of the convolutional code results in an 11.21% increase of the final bit rate, which does not seem very efficient. However from Tables 1 and 2 we can see that without employing RVLC, the bit number for the final bitstream using ECC(11/12) is still less than the number of bits for the final bitstream based on Packetization(600). It should be noted that the video decoding process is compatible with the current MPEG-4 standard, as it operates after compression and before decoding of the bitstream.

It is noteworthy that a similar approach has been introduced in the H.263 [7] video coding standard. In Annex H of H.263, forward error correction (FEC) for coded video signal is realized using block code BCH (511, 493). This allows for 492 bits of the coded data to be appended with 2 bits of framing information and 18 bits of parity information to form a FEC frame. The FEC coding allows the correction of single bit error in each FEC frame and the detection of two bit errors for an approximately 4% increase in bit rate. The FEC mechanism of Annex H is designed for ISDN, which is an isochronous, very low error rate network. No doubt the FEC capability of correcting errors is very limited compared with ECC video in harsher environments. First, the number of error bits ECC can correct is not limited to one in a chunk of 492 bits of the coded data. Second, ECC has the capability to cope with bursty errors and packet loss while FEC does not have these capabilities. Third, FEC is not so flexible for residue channel error conditions, while ECC can be adaptive to these.

6 ECC Video Design Considerations

The ECC operation based on segments results in a decoding delay of one segment, as the decoder needs to decode the punctured convolutional code first based on the segment. To reduce the effect of this decoding delay, the segment should not be set too big. The packetization (resynchronization) approach also introduces a decoding delay of one packet, so similarly the packet size should not be too long, to reduce the effect of decoding delay if packetization/resynchronization is employed.

To employ ECC with soft-decision Viterbi decoding, the network needs to deliver soft-decision output to application layer [6]. This can be accomplished by monitoring the difference between the survivor path and the path that has the next best metric in the channel decoding process, if channel coding is also achieved with convolutional coding. By monitoring the metric distance between different paths, the channel decoder produces reliability information assigned to each decoded bit.

The fundamental difference between SEC approaches for error resilience and traditional schemes represented by resynchronisation/packetization, DP and RVLC is that SEC applies before video decoding, while traditional approaches apply after video decoding. The SEC scheme fixes errors in the video bitstream before video decoding, while the traditional approach accepts the errors before video decoding and tries to hide or "repair" the error effects after video decoding via some error concealment techniques.

These different approaches toward error resilience create another feature which distinguishes ECC from resynchronisation/packetization. The performance of ECC mainly depends on the residue error conditions and the capability of ECC to correct error bits in the bitstream, if the ECC code is properly designed based on the residue error conditions and does not depend on the contents of the video sequence. This has been proven by simulation results, where ECC(7/8) corrects all transmission errors when the residue error is random and the BER of the video bitstream is less than 10^{-3}. However, the performance of packetization not only depends on the packet size and the residue error conditions, but also depends on the content of the particular video sequence, because basically the packetization approach relies heavily on the employment of techniques for error concealment. That is the main reason why we have not conducted experiments with other video sequences except the video sequences Akiyo and Salesman.

The SEC approach has provided a fresh view on both error control and error resilience coding. Traditionally implemented at the data link layer, error control is now not only a technique to improve the channel capacity at that layer but is also useful as an active error resilience tool implemented in the application layer. It combines several aspects of network operations into a generic and extended framework, which we can still call "Error Control", but now the term has a broader meaning. Under this broader meaning, source coding, channel coding and error resilience are not separate operations but different aspects of a functionally integrated error resilient real-time video delivery.

7 Conclusion and Future Work

In this paper, the passive conventional MPEG-4 error resilience tools are challenged by a simpler, more efficient and effective error resilience approach with second error correction, which is realized with error correction coding. In this work, ECC is implemented using punctured convolutional code. Based on the simulation results from this and previous work [6], ECC is much more beneficial than the passive error resilience tools in MPEG-4 to combat random or bursty errors in the final bitstream.

This work has proved the success of the SEC approach by simulation. In principle each individual technique in SEC is not a revolutionary innovation, but to apply ECC in the application layer is a new concept. Simply employing a technique commonly used in the area of channel coding in the data link layer now to the application layer, the results obtained by simulation are encouraging. At least we should have a reason to re-examine the error resilience tools in MPEG-4 and other video coding standards. Are these tools in the standards optimised for practical utilization? Are they the best tools in terms of effectiveness and efficiency? Adding redundancy in the video bitstream at the application layer to realize error resilience in different ways (resynchronisation/packetization versus ECC) results in different performance outcomes. The results obtained in this work are only by simulation. The performance of these two different approaches towards error resilience need to be tested and compared in a field implementation in the real world. Future work includes extending the SEC scheme to other real-time applications. We hope this work can open a new direction to the video coding and transmission community in the field of error resilience coding.

For ECC video, the coding efficiency can be further improved if a dynamic ECC is designed and implemented for video communication in mobile environments, where channel conditions and therefore the residue error conditions vary, with which the ECC rate can follow the change of residue error conditions. The distribution of error control between first error control and second error control needs to be optimized. The distribution of the available bandwidth of radio channels for source coding, first error control and SEC needs to be optimized as well. A generic rate control algorithm based on these optimum distributions will be more effective and efficient. More effective first error control schemes including FEC and ARQ at the data link layer need to be further investigated, taking second error control at the application layer into consideration.

To cope with wide range of residue error conditions, the optimum puncturing patterns of good base convolutional codes need to be further explored. At this stage the reported highest punctured code rate for base code (171,133) is 16/17 [8], while the highest punctured code rate for base code (561,752) is 13/14 [3]. The puncturing pattern of higher code rate of 14/15 15/16, 16/17 17/18, 18/19, etc. needs to be found for base code (561,752) or other good base codes including those with constraint length longer than 9, as higher rate codes will make ECC video more efficient in favorable channel conditions.

This work was supported by the Commonwealth of Australia under the Cooperative Research Centres program.

References

1. ISO/IEC 14496-2, "*Information Technology – Coding of Audio-Visual Objects: Visual*", 2001.
2. A. J. Viterbi, "Convolutional Codes and Their Performance in Communication Systems", *IEEE Trans. on Comm. Technology*, Vol. COM-19, No. 5, October 1971.
3. Y. Yasuda, K. Kashiki and Y. Hirata, "High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding", *IEEE Trans. on Comm.*, Vol. Com-32, No. 3, March 1984.
4. Yao Wang, Stephan Wenger, Jiangtao Wen and Aggelos K. Katsaggelos, "Error resilient video coding techniques", *IEEE Signal Processing Magazine*, Vol. 17, No.4, July 2000.
5. B. Du and M. Ghanbari, "ECC video in bursty errors and packet loss*", Proc.PictureCoding Symposium* (*PCS 2003)*, Saint-Malo, France, April 2003.
6. J. Hagenauer and P. Hoher, "A Viterbi algorithm with soft-decision output and its applications", *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, Dallas, TX, Nov. 1989.
7. ITU-T H.263, *"Video coding for low bit rate communication"*, 1998.
8. Y. Yasuda, Y. Hirata, K. Nakamura and S. Otani, "Development of variable-rate Viterbi decoder and its performance characteristic", *Proc. 6th Int. Conf. Digital Satellite Comm*., Phoenix, AZ, September 1983.
9. ISO/IEC 11172-2 (MPEG-1), "*Information technology-coding of moving picture and associated audio for digital storage media at up to about 1.5 mbit/s: Part 2 video*", Aug. 1993.
10. ISO/IEC: 13818 (MPEG-2). "*Information technology – Generic Coding of Moving Pictures and Associated Audio Information*".
11. ITU-T (CCITT) Rec. H.261: "*Video Codec for Audiovisual Services at p$\times$64 kbit/s"*., March 1993.
12. D. Haccoun and G. Begin, "High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding", *IEEE Trans. Comm.*, Vol. 37, No. 11, November 1989.
13. Y. Wang and Qin-Fan Zhu, "Error Control and Concealment for Video Communication: A Review", *Proc. of the IEEE*, Vol. 86, No. 5, May 1998.
14. G. David Forney, JR., "Bursty-Correcting Codes for the Classic Bursty Channel", *IEEE Trans. on Comm. Tech*, Vol. COM-19, No. 5, October 1971.

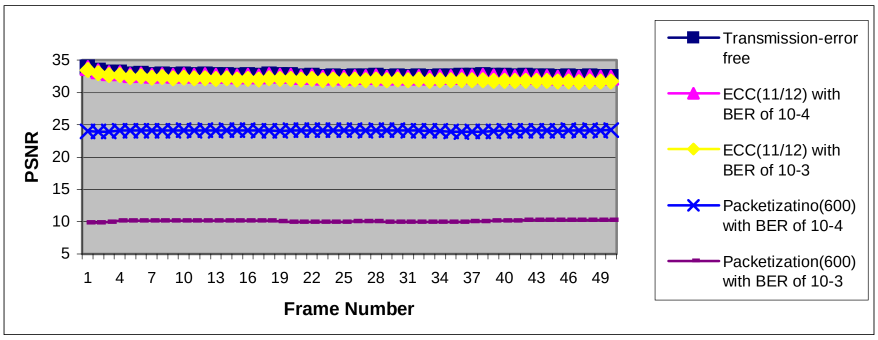

Figure 2: PSNR for Salesman with random errors.

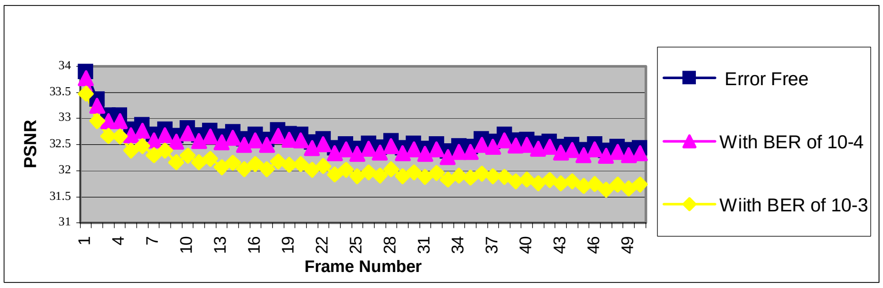

Figure 3: PSNR for Salesman with random errors (zoom in)

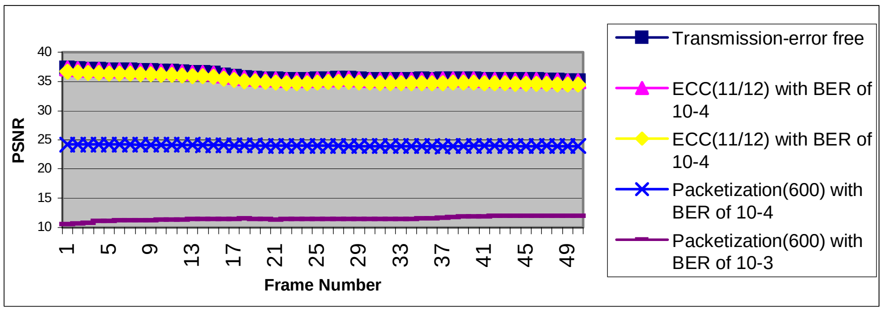

Figure 4: PSNR for Akiyo with random errors

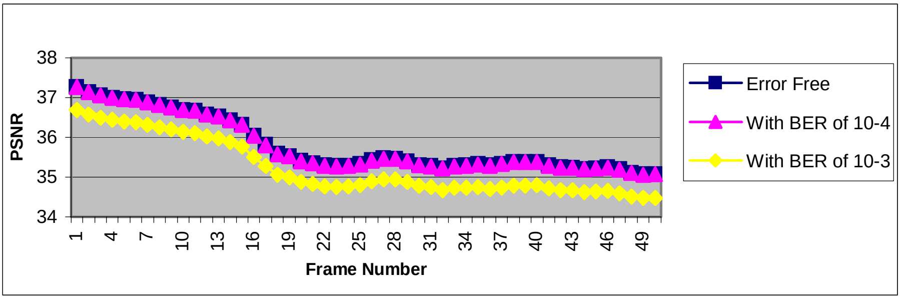

Figure 5: PSNR for Akiyo with random errors (zoom in)

Table 1 Bitrate Comparison between Packetization and ECC with Akiyo

Frame No	ECC (11/12)	Packetization (600)
0	46792	50352
1	528	488
2	664	616
3	664	616
4	1120	1112
5	1048	1040
6	1112	1088
7	1296	1272
8	1536	1552
9	1728	1736
10	1624	1632
11	1448	1472
12	2064	2024
13	2016	1976
14	2880	2912
15	4168	4208
16	5760	5840
17	6984	7000
18	7752	7896
19	7616	7704
20	6824	6880
21	6296	6352
22	5224	5232
23	4344	4368
24	4296	4312
25	3176	3184
26	3096	3104
27	3808	3848
28	4808	4816
29	5496	5512
30	6008	6048
31	6056	6112
32	5160	5176
33	4320	4344
34	4664	4688
35	4976	5008
36	5216	5256
37	5080	5128
38	5280	5272
39	5352	5408
40	4832	4840
41	4808	4872
42	4296	4328
43	3736	3752
44	3784	3784
45	4384	4416
46	4760	4760
47	4984	5040
48	5072	5096
49	4480	4504
Total	243416	247976
Average	4868.32	4959.52

Table 2 Bitrate Comparison between Packetization and ECC with Salesman

Frame Nor	ECC(11/12)	Packetization (600)
0	77896	81168
1	8600	8744
2	14176	14120
3	16320	16256
4	15592	15464
5	12536	12568
6	8888	8872
7	8488	8520
8	9280	9312
9	9272	9328
10	10272	10344
11	9480	9480
12	8472	8456
13	9432	9496
14	10120	10088
15	10200	10080
16	8688	8640
17	9880	9920
18	14472	14320
19	16744	16624
20	14232	14224
21	10712	10904
22	10536	10696
23	9048	9032
24	6440	6440
25	5720	5712
26	7032	7032
27	7976	8024
28	6848	6840
29	5096	5080
30	5928	5912
31	6824	6848
32	7496	7624
33	8280	8416
34	11040	11120
35	13872	13848
36	16000	15920
37	14216	14088
38	14888	14880
39	13384	13416
40	11096	11120
41	11352	11328
42	10728	10648
43	10216	10248
44	10856	10840
45	11488	11600
46	9464	9520
47	8968	8976
48	8360	8432
49	7784	7856
Total	584688	588424
Average	11693.76	11768.48