INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Error Detection and Error Concealment for MPEG2 Over Communication Networks

Vasilis Papadakis

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montréal, Québec, Canada

March 1998

ABSTRACT

Error Detection and Error Concealment
for MPEG2 Over Communication Networks

Vasilis Papadakis

Video will be an important application for communication networks. During transmission errors may occur. For compressed video, that means degradation of the picture quality. Different error control strategies have been proposed to compensate for the problem of transmission errors. In this work we concentrate on combining Error Detection together with Error Concealment. Two schemes based on that strategy were tested. The first uses redundancies which are still present in the compressed video signal. The second one relies on the redundancies that the syntax of compressed bitstream provides. The performance of those two schemes were compared against a general error correcting scheme.

Keywords: Video Transmission, Communication Networks, Error Detection, Error Concealment.

To my father and mother,
my aunt and uncle

ACKNOWLEDGEMENTS

This is the right place to acknowledge all the people that directly or indirectly contributed in the successful completion of this work. First I would like to thank my advisors Dr. Lynch and Dr. Le-Ngoc for proposing the research topic, providing a continuous support and directing this work. I would also like to thank all the professors with whom I have interacted during my studies at Concordia University. Last but not least, I would like to thank my family. Their love, trust and support throughout all these years made the impossible possible. I am grateful to you.

$Τα\ παντα\ ρει.$

Everything changes.

(Heracleitos)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

MPEG	Moving Picture Experts Group
BER	Bit Error Rate
VLC	Variable Length Code
MSE	Mean Squared Error
PSNR	Peak Signal to Noise Ratio
MAD	Mean Absolute Difference
HVS	Human Visual System
FLC	Fixed Length Code
DCT	Discrete Cosine Transform
F(u,v)	DCT Transform of a 2-D signal
GOP	Group Of Pictures
MB	Macroblock
HDTV	High Definition TV
DVD	Digital Versatile Disk
PES	Packetized Elementary Stream
PTS	Presentation Timestamp
DTS	Decoding Timestamp
TS	Transport Stream
FEC	Forward Error Correction
RS	Reed Solomon
DAVIC	Digital Audiovisual Council
ETSI	European Telecommunications Standards Institute
EC	Error Concealment
ED	Error Detection
CEDEC	Combined Error Detection & Error Concealment
BMA	Boundary Matching Algorithm

SBEC	Syntax Based Error Concealment
BE	Bandwidth Expansion
b(x;n,p)	Binomial Distribution
RVLC	Reversible Variable Length Code

Chapter 1

Introduction

1.1 Transmission of Compressed Video Signals

When compared to audio or text information, video signals require a huge amount of resources in order to store or transmit. For example, for a one second (1 sec.) sequence at 30 frames/sec frame rate of $704 * 480$ size for the luminant part of the frames and subsampled in both spatial directions for the color components, the required disk space for storage would be 14.5 Mbytes. Despite the increase of storage capacity and the development of broadband networks, compression techniques are needed. Video compression allows the reduction of the above number to approximate 1 Mbyte, assuming compression ratio $15 : 1$.

Modern image and video compression techniques offer the possibility to store or transmit the vast amount of data necessary to represent digital images and video in an efficient way. The video compression algorithms developed by the Moving Pictures Expert Group (MPEG) [1] have developed into important and successful video coding standards worldwide. An increasing number of MPEG1 and MPEG2 VLSI chip-sets and products are becoming available on the market [2]. This has led to a wide range of applications, such as video on demand, digital TV/HDTV, terrestrial and satellite broadcasting, multimedia, image/video database services.

Communication networks, like wireless, Internet, and ATM are not perfect and may introduce errors. Two types of errors exist. Cell loss and data corruption [3]. In packet switched networks, cell loss due to network congestion is an important source of transmission error. When several sources transmit at their peak rates simultaneously, the buffer space at some switches may be inadequate. The congestion at those switches will lead to cell loss due to buffer overflow. Thus in cell loss many bits often from the same spatial-temporal area of the video are lost. The large amount of bits lost often results in large areas of the image being lost. The impact of cell loss may become more significant as the cell size becomes larger.

In data corruption, noise in the communication channel may introduce bit

Figure 1.1: Frames with bit error

Figure 1.2: Spatial effect of bit error

Figure 1.3: Temporal effect of bit error

errors. This may also lead to large areas of the image being corrupted if the decompressor loses synchronization of the Variable Length Codes (VLC's).

In both cell loss and data corruption, errors may also propagate in the temporal direction because of the motion compensated predictive interframe coding. Figure 1.1 shows frame 0 from the sequence "Football" (compression ratio 15:1) decompressed after channel transmission, with Bit Error Rate (BER) equal to $4 * 10^{-5}$ and $6 * 10^{-6}$. A single bit error caused severe degradation in half a line of macroblocks in the first frame, while in the second there is more than one bit error present which leads to greater degradation of the image, due to the higher BER. Figure 1.2 and 1.2 show the impact of a bit error in the temporal direction. The first figure shows frame 12, where a bit error occurred, while the second figure shows the impact of that bit error after 3 frames, at frame 15.

1.2 Problem Statement

This thesis is concerned with transmission of MPEG2 compressed video over communication networks. To compensate with the possible transmission errors, a strategy of combining Error Detection (ED) together with Error Concealment (EC) is applied. ED provides the location of the errors in the compressed bitstream, while EC alleviates the effect of errors from the human viewer.

A number of different experiments were conducted to evaluate this strategy. The steps can be summarized as follows. First the compressed bitstream is encoded for ED, and then random white noise is added. Two schemes based on the above strategy were used to recovered the video sequence. The performance of those schemes was compared against a general error correcting method and a simple case where no effort is made to compensate with transmission errors.

For the performance of the examined schemes, both subjective and objective figures of merit were used. Images resulting from different experiments are presented

and are left to the reader for a subjective comparison. PSNR tables give objective measure of goodness for the performance of the different schemes. Also in this work the computational complexity was used to evaluate the examined schemes.

1.3 Figures of Merit

To evaluate the performance of the examined schemes a number of different figures of merit can be used. First subjective and objective measures of goodness, where the performance of the scheme is grated depending on the quality of the recovered video sequence. Second the bandwidth expansion set by each method. This is the amount of the overhead (extra information) added to the data prior the transmission. Third the computational complexity, which depends on the complexity of the algorithm used. The computational complexity introduces also a delay which can be critical for some applications.

Video fidelity assessment is necessary for measuring picture quality and for rating the performance. There are two types of criteria that can be used for evaluation of video quality: subjective and objective. The subjective measures [4, 5] use rating scales such as goodness scales and impairment scales. Although subjective evaluations are the most reliable measures because the user is a human observer, they are costly and time consuming.

Objective measures are performed using mathematical equations. Mean squared error (MSE), peak signal to noise ratio (PSNR), mean absolute difference (MAD) are among the common objective measures. PSNR [5], which is a variation of the MSE was used as an objective measure of goodness in this work. It is defined as

$$PSNR = 10log_{10}\frac{255^2}{MSE}$$

where MSE denotes the mean square of the differences between an original and a reconstructed image. Although objective measures are more practical to use they

may not provide a good measure of the distortion of images [6], because they do not take into account the properties of the Human Visual System (HVS).

The bandwidth expansion, defined as

$$BE = \frac{codewordsize}{datablocksize}$$

is the amount of extra information added to the data prior the transmission. There is a trade off between the BE and the performance of each method. The more bits are added, usually lead to better performance, but also increase the cost associated with the examined method.

Due to the nature of some targeted applications, the need for fast methods to compensate with transmission errors is necessary. Therefore the computational complexity of any candidate scheme and the required delay is an important factor for its evaluation.

1.4 Organization of the Thesis

This thesis is organized as follows:

Chapter 2 covers background material related to the work presented here. Here the most important aspects of the MPEG2 [1] video compression standard are presented. Also, this part introduces the idea of video transmission and the MPEG2 transport stream.

Chapter 3 introduces the main idea behind this thesis. First a FEC scheme which is currently used for the purpose of video transmission is presented, with an illustrative performance example. Next the concept of error concealment as an alternative way to compensate from transmission errors is discussed. The different error concealment categories are presented in detail. Finally the idea of combining error detection with error concealment is introduced.

Chapters 4 and 5 represent the contribution of this thesis. Each of them presents a different scheme for video transmission, based on the idea of coupling

error detection and error concealment. Each one is divided into three parts. First the different modules that the schemes consist of are presented in detail. This is followed by experimental results, including subjective and objective evaluations, which illustrate the performance of the scheme. Finally comes the comparison of the performance of the scheme with a FEC scheme, together with discussion about the schemes performance.

Chapter 6 concludes the thesis by summarizing the proposed schemes and discussing their performance. This chapter also includes future work.

The block diagrams of the programs are given in Appendix A. All simulations were done using the MPEG2 encoder/decoder [7], version 1.2, on a SPARC5 and an Ultra station.

Chapter 2

MPEG2 Compression Standard

Advances in digital video technology in the 1980s have made it possible to use digital video compression for a variety of telecommunication applications: teleconferencing, digital broadcast, video telephony and many others. In 1988 the International Standard Organization (ISO) undertook an effort to develop a standard for video and associated audio on digital storage media. The effort is better known by the name of the expert group that started it: Moving Picture Experts Group (MPEG) [7], currently part of the ISO-IEC/JTC1/SC2/WG11. The MPEG activities cover more than video compression since the compression of the associated audio and the issue of the audio visual synchronization can not work independently of the video compression.

The material covered in this chapter includes the following parts. First in Section 2.1, the MPEG2 video compression algorithm is examined. Sub-Section 2.1.1 gives the basic functions of a MPEG2 video encoder, while in 2.1.2 a quick overview of a MPEG2 decoder is given. The different components of the MPEG2 bitstream are examined in 2.1.3. The MPEG specification is intended to be generic. In order to put bounds on the many parameters for real time applications a system of profiles and levels is defined. These profiles and levels are discussed in 2.1.4. Finally in Section 2.2 a brief introduction to the MPEG2 Transport stream, which is designed for transmission applications, is given.

2.1 The MPEG2 Video Standard

For further study good starting points can be found in [2, 8]. MPEG2 [1] is the outcome of the second phase of the work of MPEG. The original goal of MPEG was to define a generic standard that could be applied to a wide range of applications. Some of the requirements were Random Access, Fast Forward/Reverse Searches, Reverse Playback, Audio Visual Synchronization, Robustness to Errors, Coding/Decoding

delay, Editability, Compatibility with MPEG1. MPEG2 is a lossy video compression scheme. It relies on two basic techniques: block based motion compensation for the reduction of the temporal redundancy and transform domain (DCT) based compression for the reduction of spatial redundancy.

2.1.1 Video Encoder

The MPEG standards do not define the encoding process. They only specify the syntax of the coded bitstream and the decoding process. Based on these requirements Figure 2.1 shows the functions needed by a typical MPEG2 encoder.

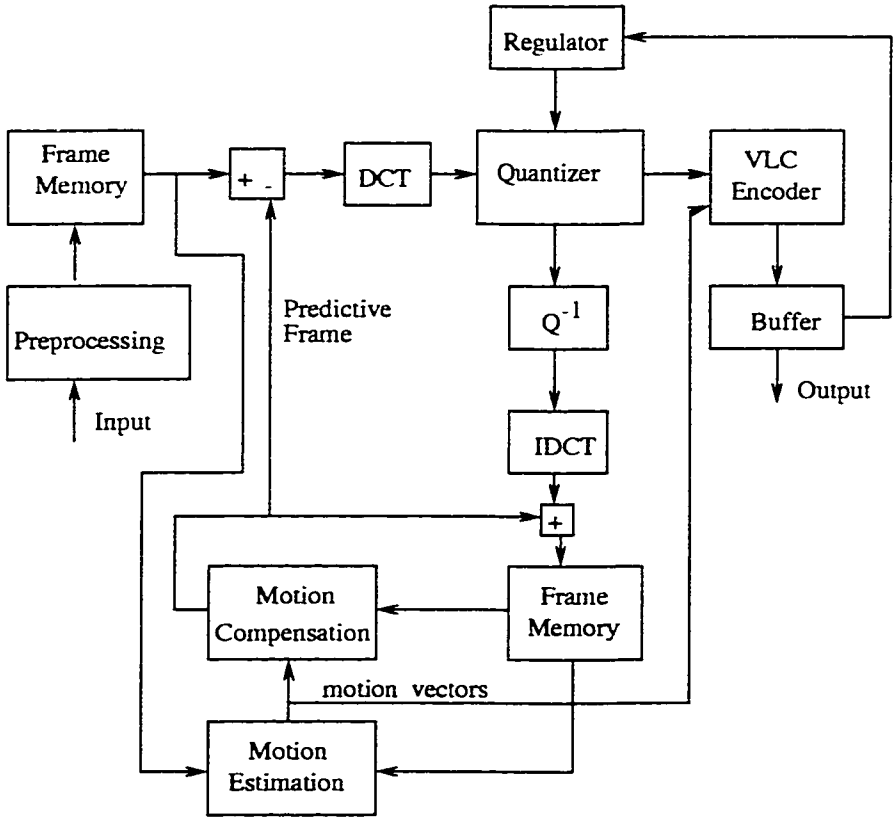


Figure 2.1: Block diagram of an MPEG2 encoder

Preprocessing

The encoding process usually begins with some preprocessing. This includes color conversion to YCbCr, format translation (interlace to progressive), prefiltering, and subsampling. None of the above operations is specified in the standard.

DCT

Discrete Cosine Transform (DCT) based implementations are used in most image and video coding standards [9] because of their high decorrelation performance and the fact that fast DCT algorithms suitable for real-time implementations are available. The purpose of the DCT is to convert the data into another form which can more easily be reduced.

At the input of the encoder, source sequences are grouped into $8 * 8$ blocks and are input to the DCT. The following equation is the mathematical definition of the $8 * 8$ forward DCT which is used in MPEG2.

$$F(u,v) = \frac{2}{N} C(u)C(v)[\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos[(2x+1)\frac{u\pi}{2N}]\cos[(2y+1)\frac{v\pi}{2N}]] \quad (2.1)$$

with u,v,x,y=0,1,2,...N-1

where x,y are spatial coordinates in the sample domain u,v are coordinates in the transform domain.

$$C(u) = C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for u,v=0} \\ 1 & \text{otherwise} \end{cases}$$

Each $8 * 8$ block is a 64-point discrete signal which is a function of the two spatial dimensions x and y. The DCT takes such a signal as an input and decomposes it into 64 orthogonal basis signals. Each contains one of the unique two dimensional "spatial frequencies" which comprise the input signal spectrum. The output of the DCT is a set of 64 DCT coefficients whose values are uniquely determined

by the particular 64 point input signal. The coefficient with zero frequency in both dimensions is called "DC-coefficient" and the 63 remaining coefficients are called "AC-coefficients". Because sample values typically vary slowly from point to point across an image, in the DCT processing lies the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. For a typical source sequence most of the spatial frequencies have zero or near zero amplitude and need not be coded. It is important to remember that the DCT, by itself, is a lossless, invertable linear transformation. The only loss or inaccuracy introduced by the DCT is due to the finite precision of the arithmetic used to implement it.

Quantization

The quantization stage comes after the DCT transformation stage. Quantization reduces the possible values for the DCT coefficients, reducing the required number of bits. This comes from observations showing that numerical precision of the DCT coefficients may be reduced without affecting image quality significantly.

Quantization takes into consideration the impact of this transformation to the human vision. Thus, each coefficient is weighted according to its impact on the human eye. Practically, high-frequency coefficients are more coarsely quantized than low-frequency ones.

Entropy Coding

The final compression stage starts with the serialization of the quantized DCT coefficients and attempts to exploit any redundancy left. The way the serialization is done affects the final compression. The DCT coefficients are rearranged in a zig-zag manner as shown in Figure 2.2. The scanning starts from the coefficient with the lowest frequency (DC coefficient) and follows the zig-zag pattern until it reaches

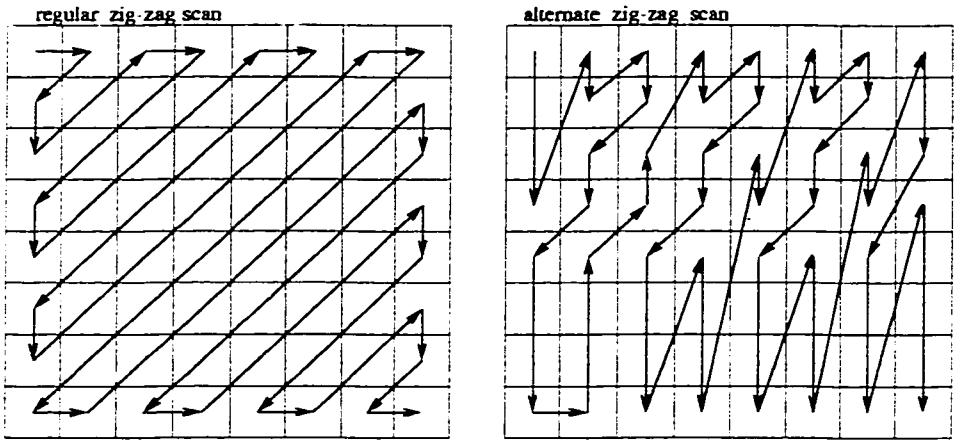


Figure 2.2: Zig-Zag patterns (regular and alternate)

the last coefficient. In MPEG2 there is an alternate scan pattern that is more efficient for interlace video signals. The sequence of coefficients is then coded using a Variable Length Code (VLC). The way the VLC allocates code lengths depends on the probability that they are expected to occur. Note that the use of VLC, codes although it provides coding efficiency, makes the compressed bitstream fragile to errors.

Picture Types

MPEG2 supports three types of pictures: I-frames, P-frames, and B-frames. Intraframes (I-frames) are compressed using intraframe coding, without the need to reference to another picture. Temporal redundancy is not taken into account. I-frames provide the random access, but achieve only low compression. Predicted frames (P-frames) are coded with reference to the nearest previously coded picture (either I or P frame), using a motion-compensated prediction mechanism. The coding process here exploits both spatial and temporal redundancies. The compression for P-frames is better than for I-frames. Bidirectional predicted frames (B-frames) which provide the highest degree of compression use both previous and future I or

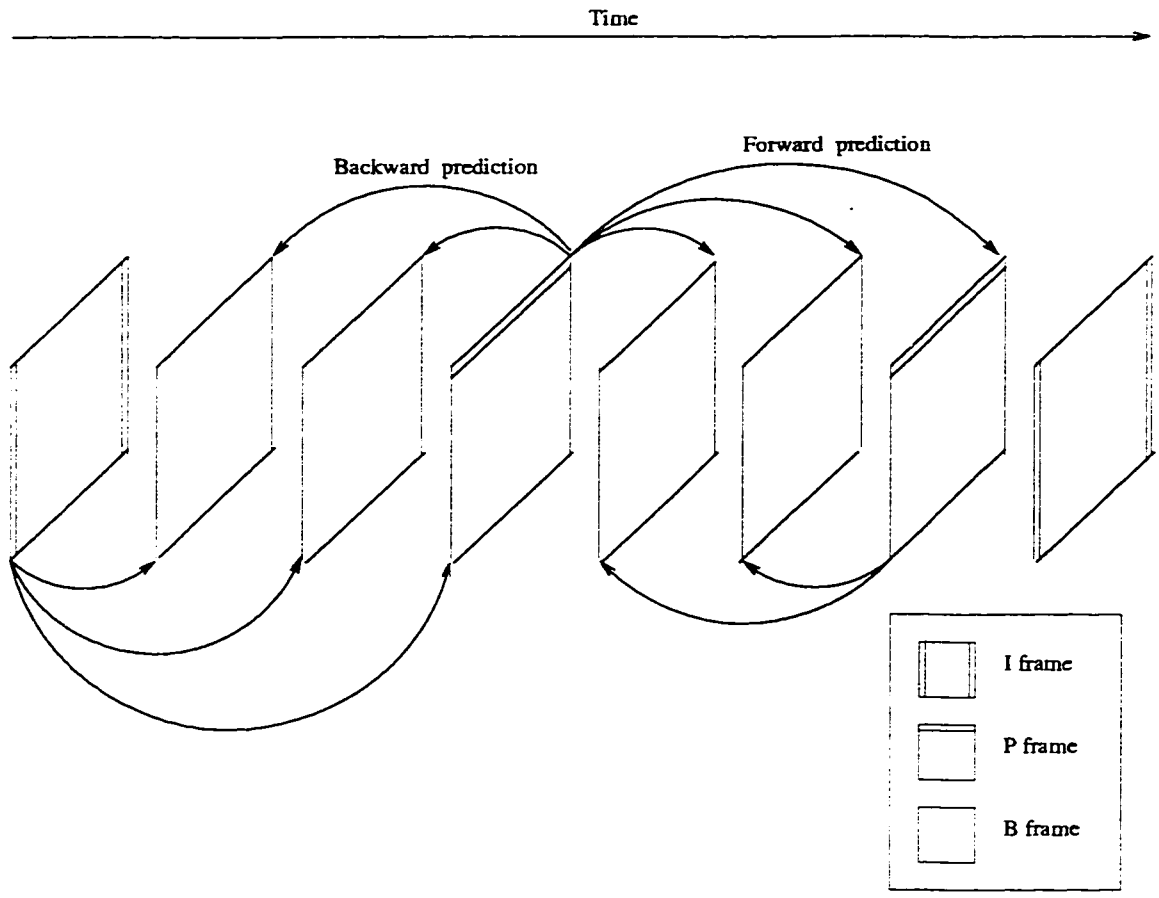


Figure 2.3: Example of inter dependence among I, P, and B frames in a video sequence

P frames as reference for motion estimation and compensation. Because they reference both past and future frames the coder has to reorder the pictures so that each B-frame is produced after all the frames it references. This introduces a reordering delay which depends on the interval between consecutive B-frames. A typical MPEG2 sequence is shown in Figure 2.3. The I-frame is coded first, then the next P-frame, and then the interpolated B-frames between the two. The process repeats with the next P-frame and B-frames.

Predicted frames, although they provide higher compression ratio, introduce temporal error propagation (see Section 1.1). That means that an error at an I frame will be carried on in the following P and B frames.

Motion Estimation and Compensation

Image compression techniques rely on two principles: the reduction of the statistical redundancies in the data and the exploitation of the human visual system. In video coding, the statistical redundancies can be categorized either as spatial or temporal. Coding techniques which reduce the spatial correlation are referred to as intraframe coding, whereas those which reduce the temporal correlation are called interframe techniques.

For the purpose of reducing temporal redundancies, motion estimation techniques [10, 11] have been successfully applied. They belong in the class of nonlinear predictive coding techniques. In a first stage the displacement of objects between successive frames is estimated (motion estimation). A number of different motion estimation techniques have been proposed in the literature [11, 12]. They can be divided into four main groups.

1. Gradient techniques
2. Pel-recursive techniques
3. Block matching techniques
4. Frequency domain techniques

Block matching techniques are based on the minimization of a disparity measure. They are the most common in compression applications.

The resulting motion information is exploited in an efficient interframe predictive coding (motion compensation). Motion compensated prediction assumes that locally the current picture is a translation of the picture at some previous time. Locally means that the amplitude and the direction of the displacement need not be the same everywhere in the picture.

Prediction Modes

In MPEG2, the picture sequence can be a collection of frame pictures or a collection of field pictures. Three classes of prediction are supported [1], frame prediction for frame pictures and field prediction for field or frame pictures.

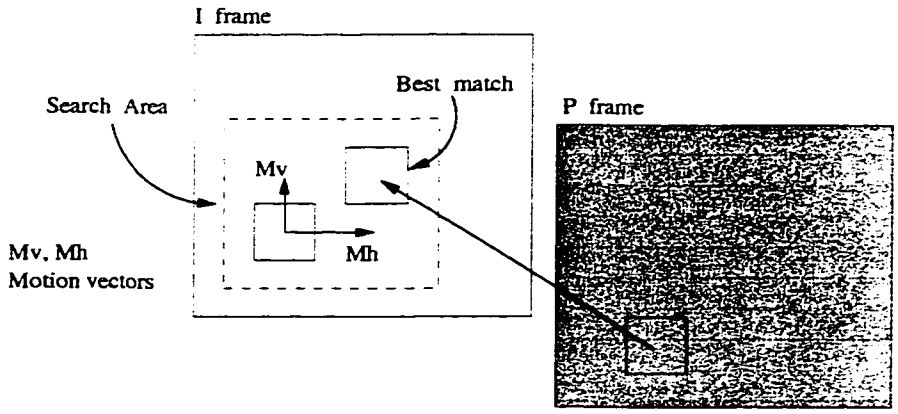


Figure 2.4: Forward motion compensation

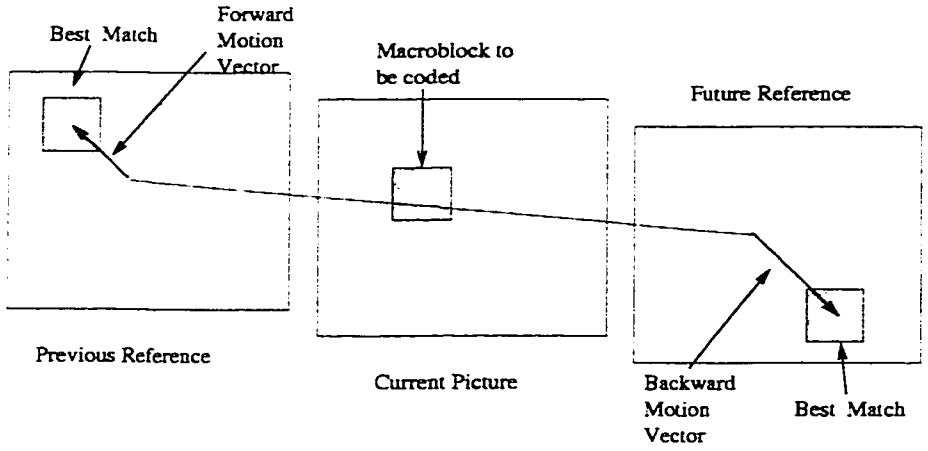


Figure 2.5: Bidirectional motion compensation

In frame prediction, predictions are made for each frame from reference frames. For each P frame macroblock the motion estimator finds the motion vector of the macroblock that best matches its characteristics in a search area in the past frame (see Figure 2.4). The two macroblocks are subtracted and their difference is DCT coded. This is referred to as interframe predictive coding.

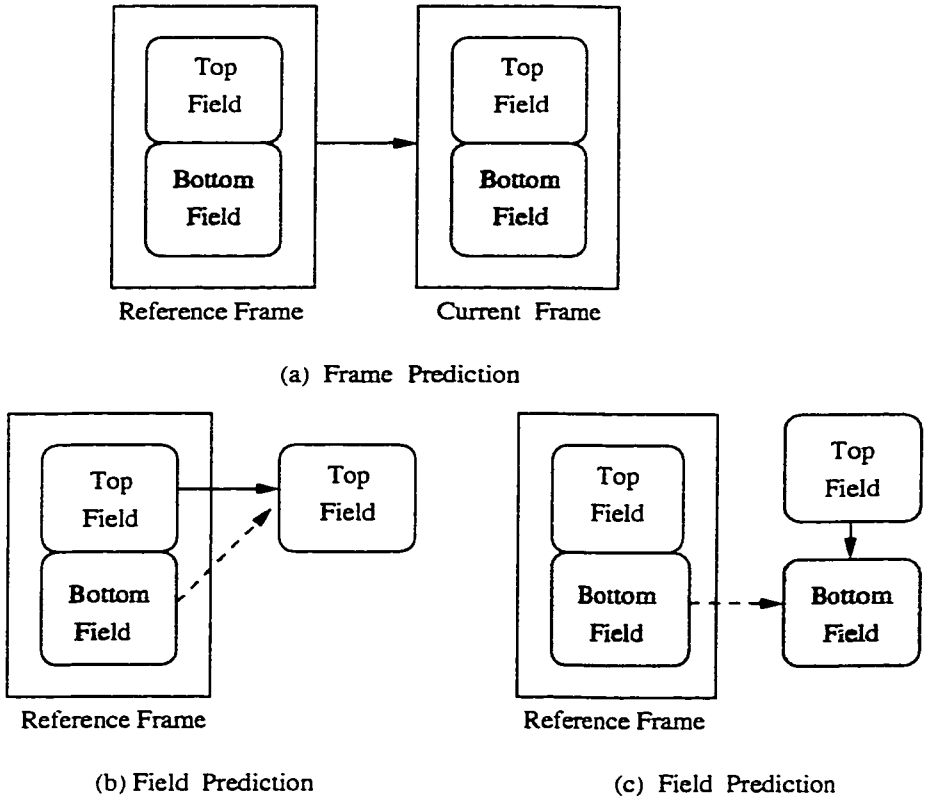


Figure 2.6: Examples of field prediction

For B frames, the motion estimation process is performed twice, once for the nearest past I or P frame and once for the nearest future I or P frame. The encoder can form a prediction error macroblock from the two macroblocks (see Figure 2.5). This is referred to as interframe interpolative coding. The prediction error is then coded using the block based DCT. The DCT coefficients of the prediction errors, together with the motion vectors, are multiplexed and coded using VLC codes.

In field prediction for field pictures, a prediction for a field picture is based on one or more previously decoded fields. In field prediction for frame pictures, the frame is divided into two fields (top and bottom). Field prediction is then carried out in each field. Figure 2.6 shows examples of field prediction for both frame and field pictures.

MPEG2 provides two additional motion compensation modes to efficiently

explore temporal redundancies between frames, the 16*8 motion compensation and the Dual-prime motion compensation.

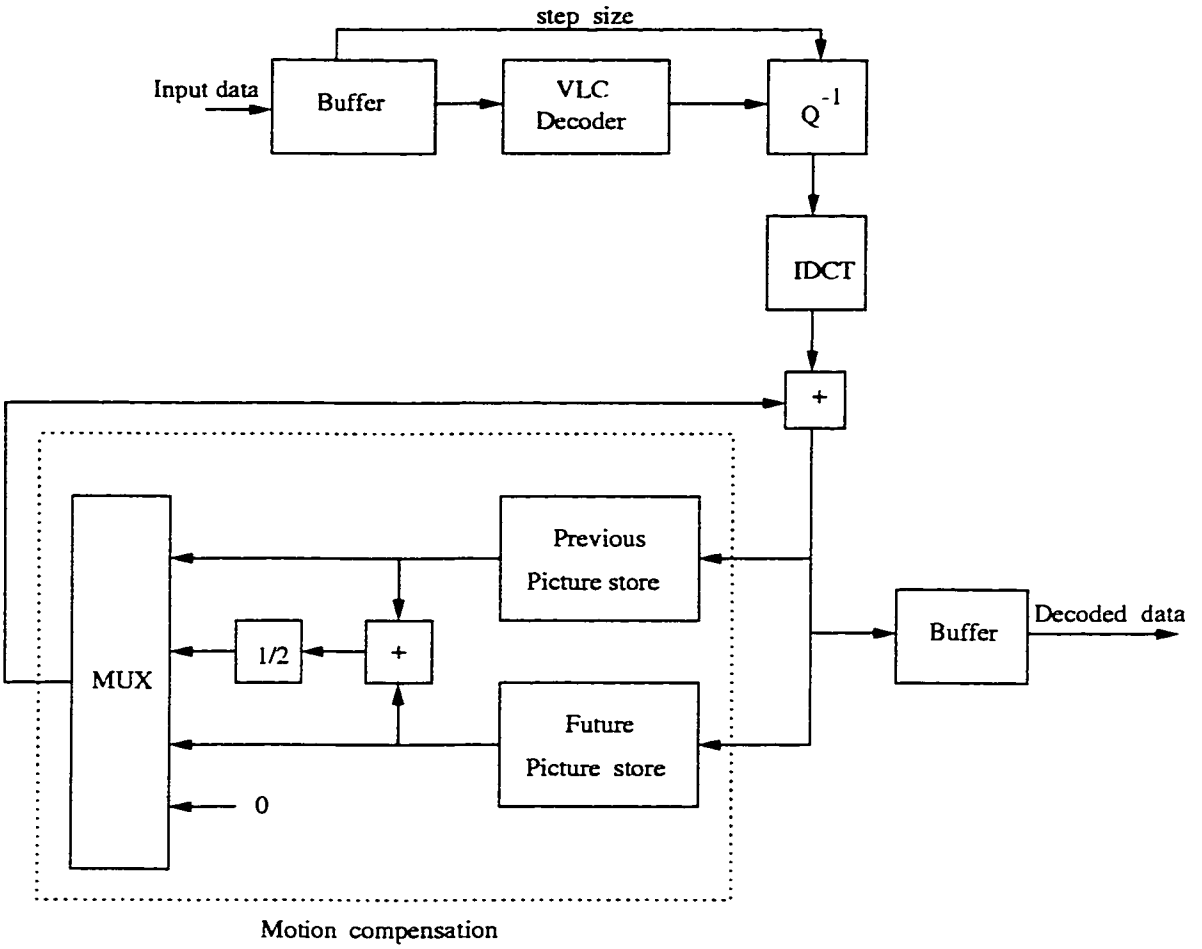


Figure 2.7: Block diagram of an MPEG2 decoder

2.1.2 Video Decoder

The MPEG standard defines the decoding process, not the decoder. There are many ways to implement a decoder and the standard does not recommend a particular way. The bitstream is demultiplexed into overhead information such as motion information, quantizer step size, macroblock type and quantized DCT coefficients. The quantized DCT coefficients are dequantized and are input to the Inverse DCT

(IDCT). The reconstructed waveform from the IDCT is added to the result of the prediction.

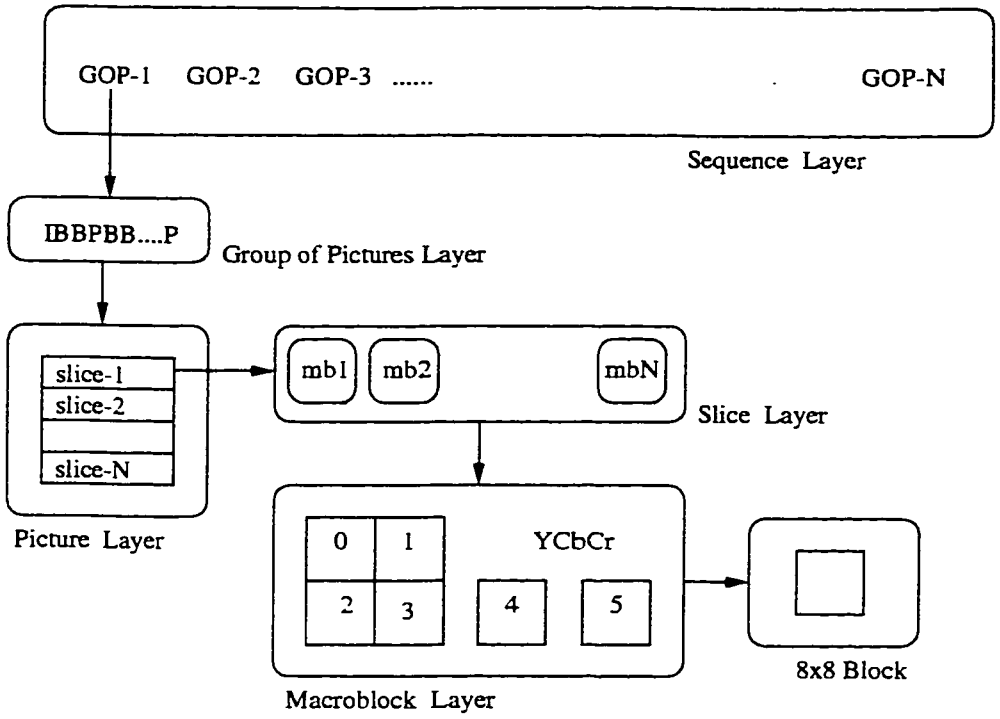


Figure 2.8: Syntax layers in MPEG2 video coding

2.1.3 Structure of the Coded Video Bitstream

The MPEG syntax for the coded video bitstream has a hierarchical representation with six layers:

1. Sequence layer
2. Group of pictures layer
3. Picture layer
4. Slice layer
5. Macroblock layer

6. Block layer

The Sequence layer is the top coding layer. It includes a sequence header and is followed by one or more groups of pictures. It ends with a sequence end code. Sequence header includes information like the vertical and horizontal picture size, the picture aspect ratio, the frame rate (pictures per second), the bit rate, and the minimum buffer size needed by the decoder. The header may also include the DCT quantization matrices for intra and non intra pictures and optional user data.

A Group Of Pictures (GOP) is a set of pictures in contiguous display order. It must contain one I frame. The header of a GOP also includes timing information and user data.

A picture in MPEG terminology is the basic unit of display and corresponds to a single frame in the sequence. The header of a picture provides a temporal reference number that can be used to define the display order of a picture. Additional header data provide information about the picture type, synchronization, and the resolution and range of the motion vectors.

Each picture is divided into slices. A slice is a horizontal strip of Macroblocks within a frame. Slices can be as big as the whole picture and as small as a single macroblock. It is the basic processing unit in MPEG2, and provides the last resynchronization layer within the bitstream. In case of data corruption, the information in the slice headers allows for a smoother recovery by the decoder. A slice header contains information for its position within a picture and a quantizer scale factor, between 1 and 31, that can be used by the decoder to dequantize the coded DCT coefficients.

A slice is divided into Macroblocks. A Macroblock is a $16 * 16$ segment in a frame, and is the basic coding unit within MPEG2. The start of a Macroblock defines its type, positional information, codes for the motion vectors and the blocks within the Macroblock that are actually coded and transmitted. The encoder may include also Macroblock stuffing. Stuffing code is a codeword, and can be placed into

	Simple	Main	SNR scalable	Spatially scalable	High
High level	no	yes	no	no	yes
High level 1440	no	yes	no	yes	yes
Main level	yes	yes	yes	no	yes
Low level	no	yes	yes	no	no

Table 2.1: Profiles and levels supported in MPEG2

the bitstream whenever the decoder detects the possibility of a buffer underflow.

The block is the smallest coding unit in the MPEG2 algorithm, and is the basic element for the DCT. It is made up of $8 * 8$ pixels and can be one of three types: luminance (Y), blue chrominance (Cb), and red chrominance (Cr).

2.1.4 MPEG2 Profiles and Levels

MPEG2 has different profiles and levels depending on the targeted application. The profiles define different subsets in the MPEG2 syntax, based on the encoding scheme, while the levels refer primarily to the resolution of the video signal produced. Initially the standard had three profiles (simple, main, and next) and four levels (High type 1, High type 2, Main, and Low). As an example, the low level refers to a standard image video format with resolution of $352 * 240$ (SIF format). The main level is targeted for CCIR-601 quality (resolution $720 * 576$), whereas the high level is intended for HDTV. More profiles were added later. The complete set of profiles now consists of the Simple, Main, SNR scalable, Spatially scalable, High, and $4 : 2 : 2$ profiles. The levels consist of the high, high-1440, main and low. The new scalable profiles can work very efficiently in a network environment, since whenever there is congestion the enhancement layer can be dropped without affecting the basic quality.

The most important profile that was first standardized is the main profile. The importance of this profile, as noted in Table 2.1, has to do with its ability to handle images from the lowest level (MPEG1 quality) to the highest, making it suitable for

Level	Max dimensions	frames/sec	Max bit-rate	Application
Low	352 * 288	30	4 mb/s	CIF, consumer tape equiv.
Main	720 * 576	30	15 Mb/s	CCIR-601, studio TV
High 1440	1440 * 1152	30	63 Mb/s	4 * 601, consumer HDTV
High	1920 * 1152	30	84 Mb/s	production SMPTE 240M std

Table 2.2: MPEG2 Main Profile

HDTV application. Typical bit-rates and targeted applications for the main profile are shown in Table 2.2.

2.2 MPEG2 System Layer

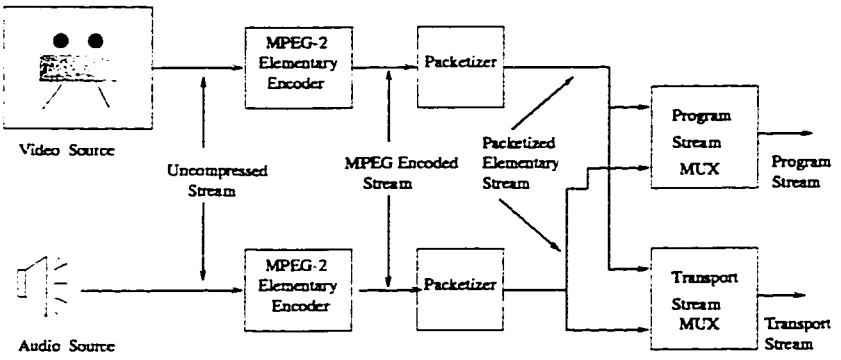


Figure 2.9: MPEG2 Systems Layer

The MPEG standard defines a way of multiplexing more than one stream (video or audio) in order to produce a program. A program consists of one or more Elementary Streams. Two data stream syntax are defined by the MPEG2 system standard [13].

Program Stream: Each Program Stream consists of one program.

Transport Stream: It combines one or more programs into a single stream. The program stream is intended for the storage and retrieval of program material from digital storage media like Digital Versatile Disk (DVD). The Transport Stream is intended for a transmission application using short fixed length packets for non error-free environments.

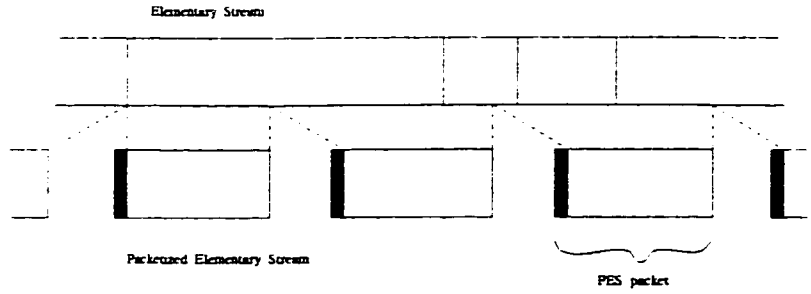


Figure 2.10: Generation of Packetized Elementary Stream

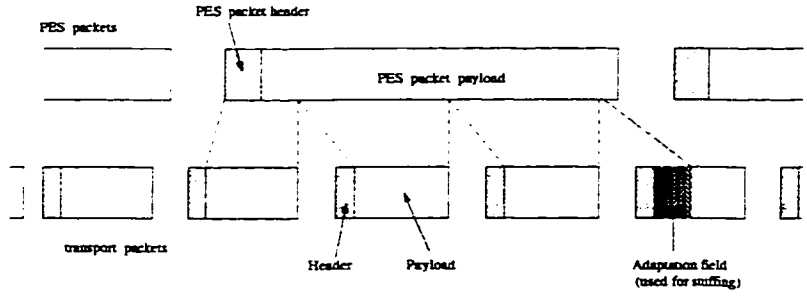


Figure 2.11: Generation of Transport Stream from PES packets

After the creation of the Elementary Stream, the next step is its packetization. The resulting stream is now called Packetized Elementary Stream (PES), and the packets are called PES packets. A simplified overview of this process is shown in Figure 2.9. A PES packet consists of a header and a payload. The payload is nothing more than data bytes taken sequentially from the elementary stream. There is no specific format for encapsulating data bytes in a PES packet.

PES packets have special identifiers to distinguish themselves from PES packets of other elementary streams. They may also carry two different timestamps that are used at the decoder end for the synchronization of related elementary streams: The Presentation Timestamp (PTS) and the Decoding Timestamp (DTS). Figure 2.10 shows the creation of PES packets.

Transport stream packets are subdivisions of PES packets with additional header information. The Transport stream consists of short fixed-length packets. Each packet has a length of 188 bytes. It comprises a 4-byte header followed by an

"adaptation field" or a payload or both. The PES packets from the various elementary streams are each divided among the payload parts of a number of transport packets. However, there is one constraint: the first byte of a PES packet must be the first byte of a transport packet payload.

Due to that constraint, a transport packet may not be completely full. The stuffing bytes needed to fill the packet are placed in the adaptation field (see Figure 2.11), except in the case of a transport packet carrying Program Specific Information (PSI) which may be placed at the end of the packet. The amount of this stuffing can be minimized by careful selection of the PES packet length. Usually, long PES packets are better in terms of bandwidth efficiency, but are more prone to synchronization problems.

Chapter 3

Concepts of Video Transmission

During transmission of compressed video over Communication networks errors may occur. As discussed in Chapter 1, because the compressed video is fragile, any error may lead to the severe degradation of the video sequence. To compensate for transmission errors, different techniques have been introduced.

The material covered here includes three parts. In Section 3.1 a technique for transmitting compressed video is presented. The performance and the complexity of the method is discussed in 3.1.1. In Section 3.2 an alternative way for recovering video data from channel errors is examined. Here the redundant information which still exists in the compressed bitstream is used to alleviate the effect of the errors. Finally in Section 3.3 the framework of a scheme which can used to compensate for transmission errors is defined.

3.1 Error Control Coding

The integrity of received data is a critical consideration in the design of digital communications and storage systems. Many applications require the absolute validity of the received message, allowing no room for errors encountered during transmission. Error Control Coding [14] provides the means to protect data from errors. For the case of video information, Standardization Organizations like Digital Audiovisual Council (DAVIC) [15] and European Telecommunication Standards Institute (ETSI) [16, 17] set the required post decoding BER at 10^{-11} for MPEG2 Quality of Service to be guaranteed. This is referred to as Quasi-Error-Free (QEF) transmission. To achieve this, Forward Error Correction needs to be added to protect the MPEG2 transport stream.

It is important to distinguish between pre decoding bit error probability and the post decoding bit error probability. The first one depends on the communication channel, while the later one is the error probability that we get after channel decoding. Since block codes (which are used) act on symbols, symbol errors will be

discussed before the bit errors. The probability of an uncorrectable error is given by P_{UE}. An uncorrectable error occurs when more than (t) received symbols are in error in a given block. An important parameter in determining P_{UE} is the channel symbol error rate P_{SE}. The channel symbol error rate is the probability that the channel will change a symbol during the transmission of the message. The expression for the probability of an uncorrectable error, assuming that the symbol errors are independent, is given by:

$$P_{UE} = 1 - \sum_{i=0}^{t} \binom{n}{i} (P_{SE})^i (1 - P_{SE})^{n-1} \tag{3.1}$$

where n is the block length, and t the number of the correctable symbol errors.

The relationship between symbol error rate (P_{SE}) and bit error rate (P_B), under the assumption of random bit errors, is given by:

$$P_{SE} = 1 - (1 - P_B)^m \tag{3.2}$$

where m is the number of bits per symbol.

Based on Equations 3.1 and 3.2 [18], and the requirement for post decoding BER of 10^{-11}, the code that will be used must ensure good performance for error environments up to 10^{-4}. To achieve the appropriate level of error protection a FEC based on RS encoding is used. In [15], and [16] DAVIC and ETSI propose the use of a shortened RS$(204, 188, t = 8)$ code. That simply means that for every 188 bytes of video data the code adds 16 bytes of redundancy data.

3.1.1 Reed-Solomon Coding

Figure 3.1 gives the general diagram of the RS based scheme. Following the energy dispersal randomization process, systematic RS encoding is performed for each randomized MPEG2 transport packet (see Figure 3.2). This process adds 16 parity

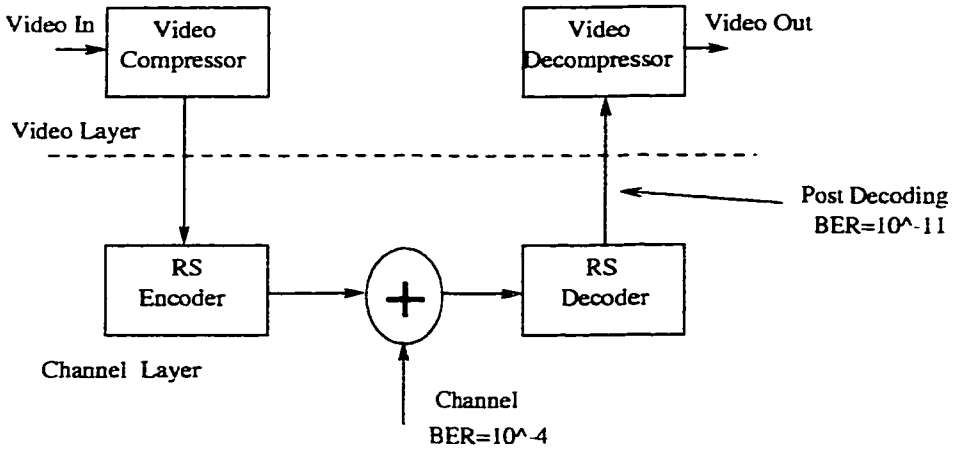


Figure 3.1: Protecting MPEG2 using a RS FEC code

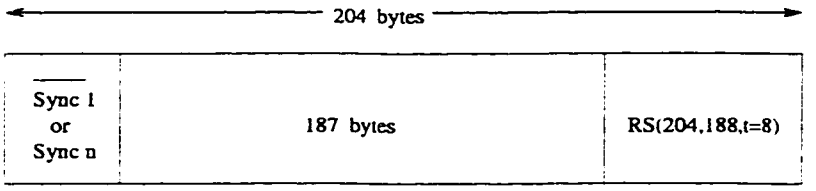


Figure 3.2: Reed-Solomon RS(204,188,t=8) error protected packet

bytes to the MPEG2 transport packet to give a codeword $(204, 188)$. The RS code has the following generator polynomials:

Code Generator Polynomial $g(x) = (x + \mu^0)(x + \mu^1)(x + \mu^2)...(x + \mu^{15})$, where $\mu = 02H$

Field Generator Polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

The shortened RS code is implemented by appending 51 bytes, all set to zero, before the information bytes at the input of an $(255, 239)$ encoder. After the coding procedure these bytes are discarded.

Performance of the Scheme

DAVIC in [15] gives the following requirements:
The amount of the white noise random bit error rate received by the user device

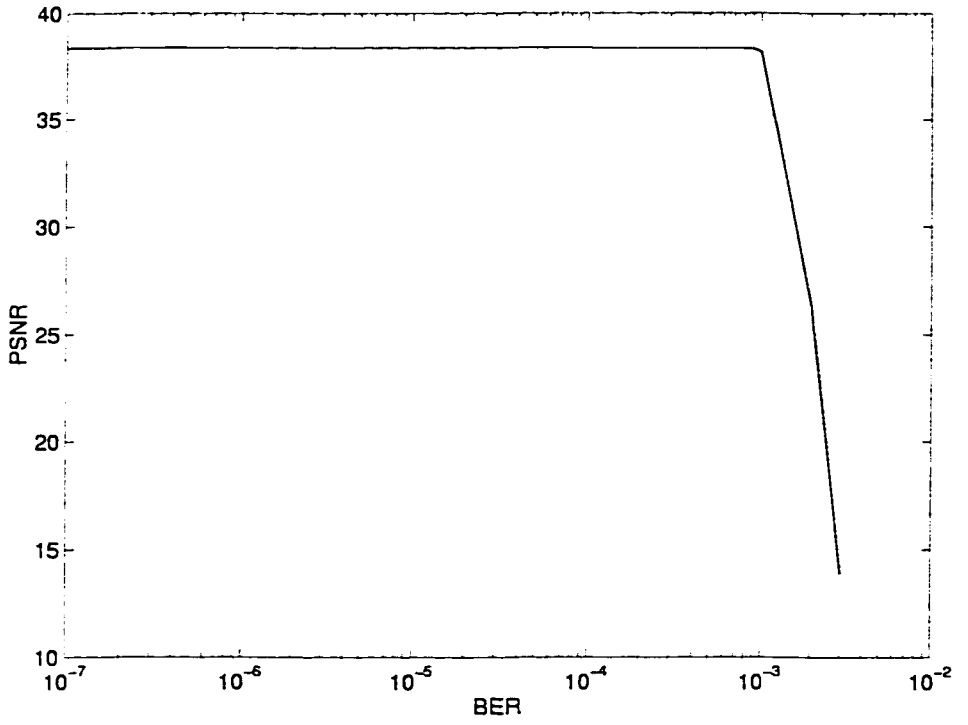


Figure 3.3: Performance of RS(204,188,t=8) code

after error correction shall be lower than 10^{-12}. With this assumption a 10 Mbits/s downstream stream flow will be corrupted less than once a day, which is considered to be acceptable for interactive multimedia services.

The network device received white noise random bit error rate after error correction shall be less than 10^{-10}. With this assumption, a 100 Kbits/s upstream data stream will be corrupted less than once a day, which is considered to be acceptable for interactive multimedia services.

Figure 3.3 shows the performance of the $RS(204, 188, t = 8)$ code. The sequence Table-Tennis, consisting of 180 frames and compressed at 8 Mbits/s, was tested in a wide range of BER. For BER up to 10^{-3} the code performs well. For higher BER however, it degrades fast.

Complexity

The computational complexity for syndrome based algorithms for decoding RS codes, with error correcting capability (t) and code length (n), is given in 3.3 and 3.4 [19]

$$\text{No. of Additions} = n(3t-1) + 4t^2 + 1 \tag{3.3}$$

$$\text{No. of Multiplications} = n(4t-1) + 5t^2 + 3t - 1 \tag{3.4}$$

For the case of the $(204, 188, t = 8)$ RS code equations 3.3 and 3.4 gives: number of additions 4949, number of multiplications 6667 per RS block. A 1 second sequence encoded at 8 Mbits/s is packed in 5578 Error Correction blocks. Channel decoding of such a bitstream requires 27605522 additions and 37188526 multiplications. The required computations introduce a decoding delay, which is aproximately 3 RS blocks. In a typical sequence like the one used during these simulations the size of the compressed frames are as follows: I-frame 600000 bits, P-frame 250000 bits, B-frame 120000 bits. In order to correct a single RS block the required delay is equivalent with 0.7% of an I-frame, or 2% of a P-frame, or 4% of a B-frame.

3.2 Error Concealment

An alternative way to alleviate channel errors from a compressed video bitstream is by using Error Concealment (EC). Error Concealment techniques [20] generally make use of the redundancies still remaining in the frequency, temporal and spatial domain after the compression. They can be divided into four categories. Simple Concealment, Frequency Concealment, Spatial Concealment, and Temporal Concealment.

3.2.1 Simple Concealment

At low bit error rate, some simple algorithms can conceal the error up to a certain extent. These include simple replacement from the previous spatially or temporally adjacent blocks or simple replacement from the previous picture. These techniques work under certain circumstances when the bit error rate is not high, and there is not much scene change and motion from frame to frame. For medium or high bit error rate, or in sequences with high motion, more sophisticated algorithms should be used.

3.2.2 Frequency Concealment

DCT transforms a block of data in the spatial domain into a block of the same size in the frequency domain. One of the most important features of the DCT, that frequency concealment uses, is the fact that the background of adjacent blocks is generally correlated. The low frequency DCT coefficients contain most of the background information. The basic frequency concealment algorithm [20] does linear or polynomial interpolation of adjacent DCT coefficients. The quality of the reconstructed picture is determined by the complexity and the accuracy of the algorithm. The more coefficients are interpolated, the more details we can have for the reconstructed block. However, a certain degree of accuracy is required since the error in

the DCT coefficients will be magnified in the spatial domain because they contain compressed information.

3.2.3 Spatial Concealment

Another technique is to interpolate directly into the spatial domain [21] using neighboring blocks within the same picture. The algorithms utilizes the spatial correlation more directly and thoroughly by performing interpolation in a large local neighborhood surrounding the block. Because interpolation is implemented in the spatial domain, finer details of the image can be reconstructed and consistency of the edges can be maintained to a certain degree. Unlike the concealment in the frequency domain, where the correlation between lower DCT coefficients mainly exists in horizontal vertical and diagonal directions, the correlation in the spatial domain can be of any angle depending on the content of the picture. Each pixel inside the block should be estimated instead of a few low frequency coefficients in the frequency domain.

3.2.4 Temporal Concealment

Temporal concealment [22] is possible for coding algorithms where the video sequence is further compressed in the temporal domain. For example we can reconstruct a lost macroblocks motion vectors and then put the reference block into the same position. The block can be further optimized by smoothing its edges. The crucial point of temporal concealment is that the estimation of motion vectors should be very accurate, otherwise the block obtained via a vector displaced too much from the original one will cause large subjective degradation.

3.3 Combined Error Detection and Concealment

Transmission of video over communication networks is prone to errors. The effect of those errors in a compressed video bitstream is annoying for the end user. To compensate for transmission errors, video data are channel coded (prior to transmission) using a strong FEC code. This code performs well, but does not take into consideration useful information that the compressed video data carry.

The main idea which drives this research effort can be summarized as follows: Why not try to relax the need for channel coding of video data (in other words use less strong code), and instead use a simpler error detecting code which will be able to locate errors in the compressed video domain. The correcting capability of the FEC code that we will give up will be replaced by an EC scheme, which uses the redundant information which still exists in the video data. Here the concealment will be seen as a correcting process, although it does not really correct errors, but tries to alleviate them. Concealment can be performed either on the compressed domain (prior to video decompression), or at the uncompressed domain (after video decompression).

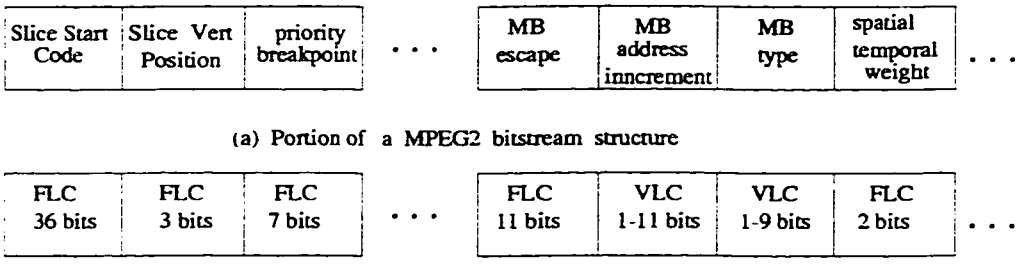
In the next chapters, two schemes for video transmission, based on the above idea, will be presented in detail. The first one is called Combined Error Detection and Error Concealment (CEDEC), and the second one Syntax Based Error Concealment (SBEC).

The objectives are:

1. Ensure quality video for the end user.
2. Reduce the required number of bits needed for transmission.
3. The scheme must have low complexity.

3.3.1 Redundancies in the MPEG2 Bitstream

In a MPEG2 compressed video bitstream, two types of redundancies can be found: Spatial-Temporal and Syntax Based.



(a) Portion of a MPEG2 bitstream structure

(b) Portion of a MPEG2 bitstream seen as a concatenation of VLC and FLC

Figure 3.4: Structure of MPEG2 bitstream

In Chapter 2 we saw how the MPEG2 compression algorithm reduces spatial and temporal redundancies using the DCT transform and motion compensative prediction. But the algorithm still does not remove all of the redundant information which exists inside a video sequence. The amount which still resides inside the compressed bitstream is used from spatial and temporal Error Concealment schemes in order to alleviate errors. Figure 3.5 demonstrates such an example. This frame was taken from the sequence "Susie" encoded at 8 Mb/s, after errors were inserted. Here there is a black stripe due to a bit error. Looking more closely (see Figure 3.6), it is obvious that the surrounding neighborhood contains similar information. Figures 3.7 and 3.8, show the same frame after Error Concealment has taken place. Here the fact that the area surrounding the damaged area had similar content helped the concealment process.

In 2.1.3 the structure of the MPEG2 bitstream was presented. Careful inspection of this bitstream reveals a concatenation (see Figure 3.4) of Variable Length codes (VLC) and Fixed Length codes (FLC). One point of interest is that if the codeword corresponding to a certain field is replaced with another codeword (of the same field), the result will be most of the time a valid MPEG2 stream. This is

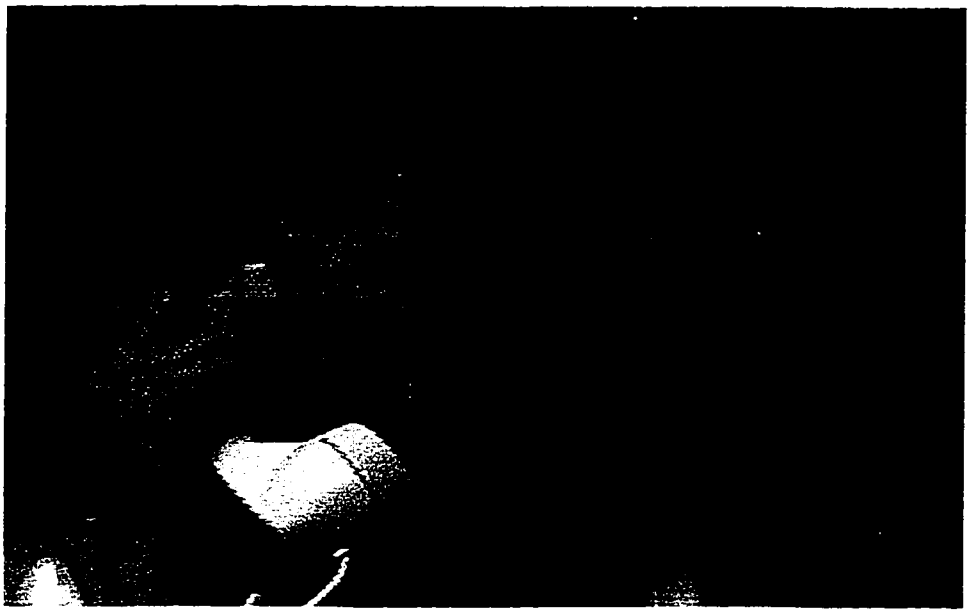


Figure 3.5: Susie Frame #5 with error

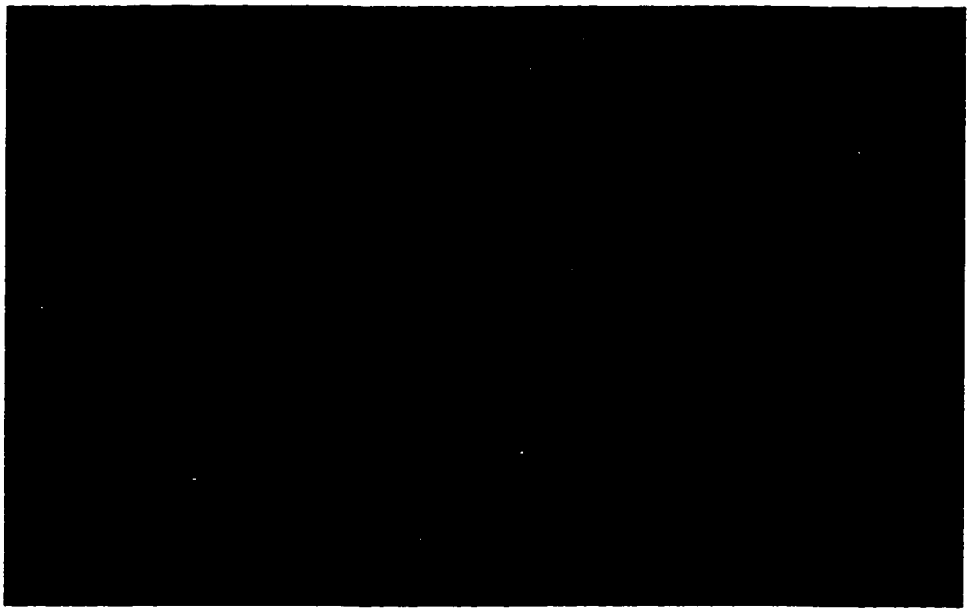


Figure 3.6: Zoomed Susie Frame

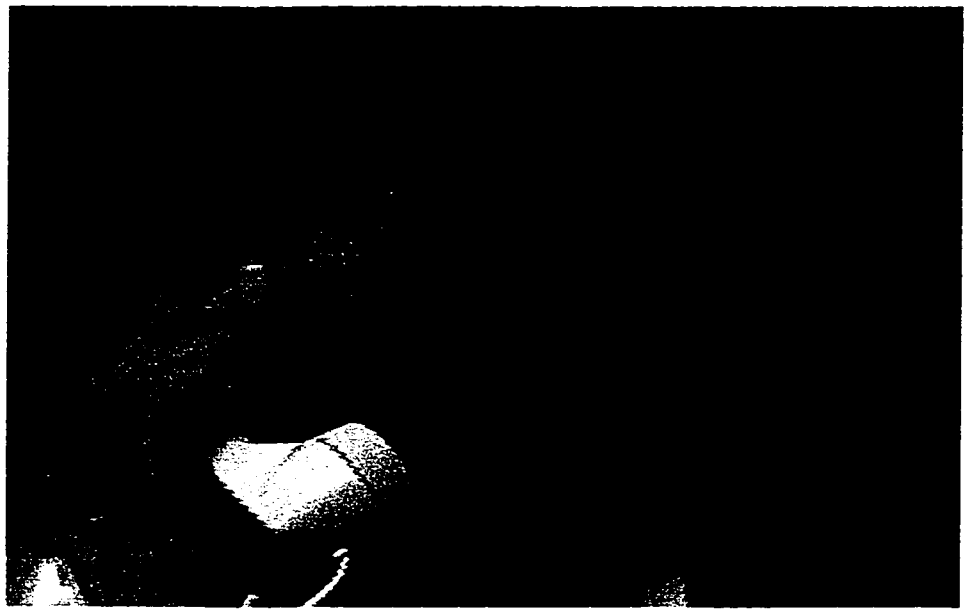


Figure 3.7: Susie Frame #5 after Error Concealment

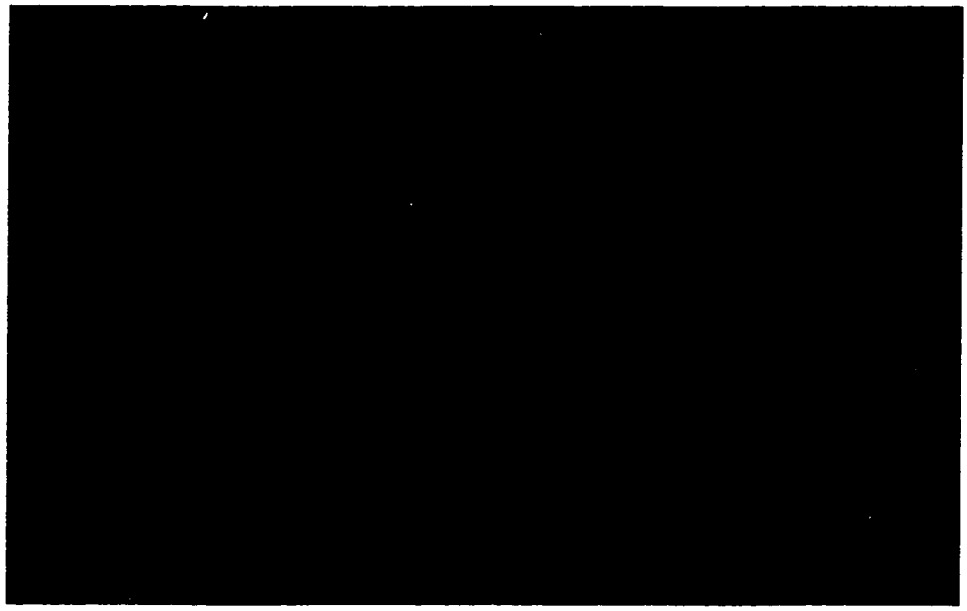


Figure 3.8: Zoomed Concealed Susie Frame

referred to as syntax based redundancy.

VLC	motion code
00011	−3
0011	−2
011	−1
1	0
010	1
0010	2
00010	3

Table 3.1: Variable length codes for motion code

This type of redundant information can be used to conceal bit error occurance in the bitstream. Suppose that such an error had occurred in the motion code field. This particular field is used to derive the motion information and is coded using VLC. Table 3.1 presents a range of valid values that this field can take. If a scheme can test a number of bits in a specified size block (which contains that field) until the MPEG2 decompressor does not find a syntax violation, the result will be a concealed sequence. Figures 3.9 and 3.10 demonstrate such an example. Figure 3.9 shows a frame from the sequence "Flower-Garden" with bit errors. Figure 3.10 shows the same frame after a concealment process, which exploits the redundancy in the syntax of the bitstream.

3.3.2 CEDEC

For CEDEC a simple error detection code is used, together with classical error concealment techniques which perform on the uncompressed domain. The scheme uses the redundancy that still remains in the video data after compression in order to conceal errors. The locations of the bit errors are provided by the error detection code.

Figure 3.9: Flower-Garden Frame #69 with Bit Errors

Figure 3.10: Frame after Syntax Based Concealment

3.3.3 SBEC

In SBEC, again a simple error detection code is used, together with a new error concealment scheme which performs on the compressed domain. The scheme uses the redundancy on the syntax of the MPEG2 video bitstream in order to conceal errors. The locations of the bit errors again are provided by the error detection code.

Chapter 4

Combined Error Detection and Concealment Technique

In Chapter 3 a scheme for protecting MPEG2 compressed video against transmission errors, based on the shortened RS$(204, 188, t = 8)$ code, was presented. In this chapter a scheme [23] for recovering transmission errors in a MPEG2 compressed bitstream, based on what was discussed in Section 3.3, is examined. The method uses ED to locate the presence of errors in the compressed bitstream. This information is then translated into location of errors in the uncompressed domain. Finally an EC method tries to hide those errors.

In Section 4.1 the different modules the scheme consists of are presented. The type of ED code which is used, and there may be more than one reason for the selection of that particular ED code, is presented in 4.1.2. In 4.1.3 the spatial translation from the compressed to the uncompressed domain is discussed. Finally 4.1.4 examines the EC algorithm which is used to alleviate the errors. Experimental results are given in Section 4.2, while a short discussion of the performance of the scheme is given in Section 4.4.

4.1 Combined Error Detection and Concealment

The scheme which will be presented here is named Combined Error Detection and Error Concealment (CEDEC). ED provides the location of the errors in the compressed bitstream. EC can be considered as an error correcting process. The difference is that it does not correct the errors, but rather alleviates them.

When a code with minimum distance d_{min} (the minimum number of of bits that two codewords differ by) is used for ED, it detects all the error patterns with $d_{min} - 1$, or fewer errors. When the same code is used for error correction, it can correct all the error patterns with $\lfloor d_{min} - 1 \rfloor / 2$ errors [14]. Thus, for the same code, more error detection can be done than correction. By combining ED with EC, the full error detecting capability of a code is used, while good error correcting capability is maintained, using error concealment.

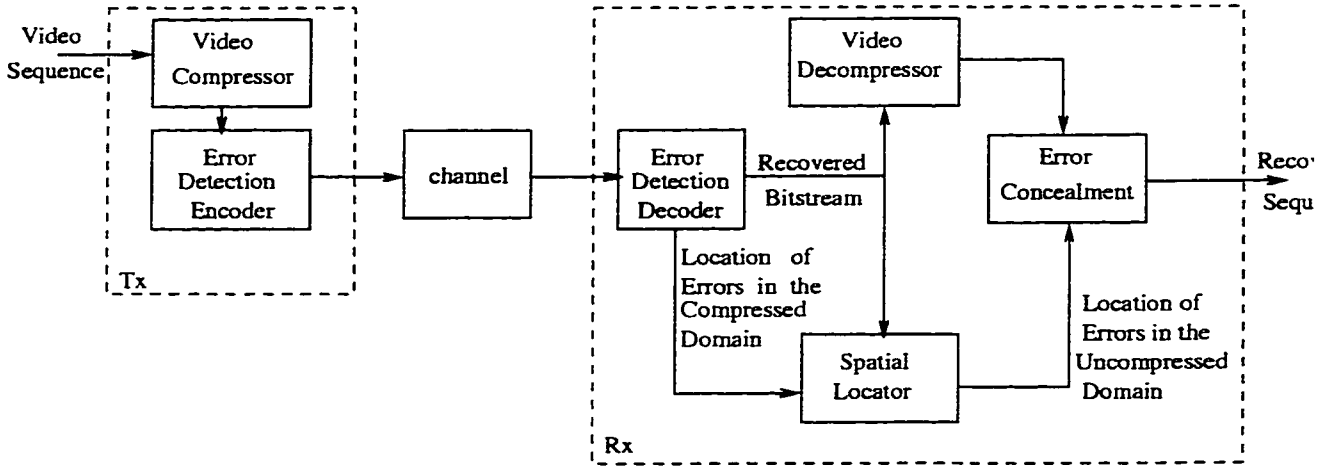


Figure 4.1: The CEDEC technique

CEDEC is shown in figure 4.1. After compression the bitstream is encoded for error detection, and then transmitted through a channel. In the receiver end, data pass through the ED decoder and are checked for errors. The bitstream, along with the error locations in the compressed domain, pass to the spatial locator. There a translation of the location of the errors, from the compressed domain in the uncompressed video, takes place. For example if we know that an error has occurred in the 1200th block, after the spatial locator we have information that looks similar to the following: ED block 1200 corresponds to picture 5, slice 10, macroblock 420. This type of information passes to the EC, together with the decompressed sequence. EC provides the restored video sequence.

CEDEC does not introduce a new Error Concealment algorithm, but uses existing methods. In the literature a lot of Concealment schemes [20, 21, 24] have been proposed, and actually the one that was used here is very simple. Also CEDEC does not bring anything new to the topic of FEC. The coupling of Error Detection and Error Concealment is the main contribution of CEDEC. While some schemes [21, 25] refer to a transport mechanism which provides error location information to the EC, in others [26] the error location is known apriori. In CEDEC, on the other hand, the location problem (able to locate the position of an error in the uncompressed

domain) is addressed. Also, here EC is used in order to provide error correction capabilities to the ED code.

4.1.1 Channel

This module simulates the channel traffic. In Section 1.1 the different types of transmission errors were presented. Here the decision was to study the case of bit error occurance.

The module takes as an input the video bitstream, and adds random white noise. The amount of added noise is controlled from the selected BER case. For the simulations, noise is added only to a portion of the bitstream. Errors are permitted only in DCT coefficients of B-frames. The reasons for that decision can be summarized as follows:

1. Errors at B-frames will have only spatial effect. So having an error at frame 4 will not cause errors at other frames of the sequence.

2. Ability to locate the position at the MB layer because important components of the bitstream will be error free.

The rest of the bitstream is considered vital for the decompressing process, and assumed well protected. At the output of the channel comes the corrupted bitstream.

4.1.2 Error Detection

After compression the bitstream is encoded for error detection. For each data block of n bits, x parity bits are added forming an ED block. At the decoder the received codeword, possibly containing transmission errors, is checked to determine whether or not the parity relationship is satisfied.

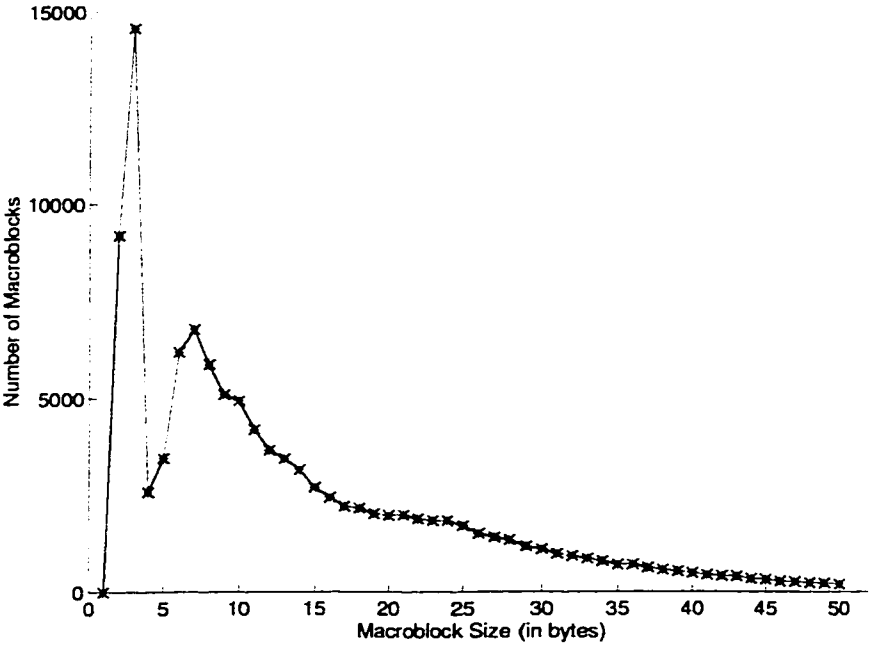


Figure 4.2: Analysis of the Macroblock Size

To design the ED code, the size of the bitstream data block that will be encoded must be specified. Since in CEDEC the requirement is to spatially locate errors down to macroblock resolution, the desired ED block size should be close to the smallest compressed macroblock size. Two Sequences, "Susie" and "Table-Tennis", encoded at 8 Mb/s (compression ratio 15 : 1), each consisting of 90 frames (3 sec.) were used to get an idea about the size of the compressed macroblocks. In Figure 4.2 a histogram of the compressed macroblock sizes is given (Mean value:14.46, Standard deviation:12.41). The analysis indicates that although there are a number of macroblocks with size less than two bytes, the peak lies between two and three bytes.

Based upon this preliminary analysis, a code with data block size equal to 16 bits (2 bytes) was chosen. By doing so, the chance of a particular ED block spanning between two macroblocks should be rare. In that way the mapping from the compressed to the uncompressed domain will perform down to the macroblock resolution. For each data block of 16 bits, one parity bit is added. In that way any

odd number of errors will be detected.

4.1.3 Spatial Locator

As mentioned earlier, the Spatial Locator is used to translate the position of a bit in the compressed bitstream into a spatial location in the uncompressed video sequence. The idea behind this is that a one-to-one correspondence exists between locations in the compressed bitstream and spatial locations in the uncompressed video. The MPEG2 [1] compressed bitstream consists of several layers (see Section 2.1.3). High level layers are associated with larger portions of the uncompressed video than lower level layers. If the error occurs in a high level layer, then it may affect a larger spatial location than if it occurred in a lower level layer. For example, an error at the Picture header may cause the whole frame to be discarded by the decompressor. While an error at the block layer may cause the loss of a single block.

To get the spatial location of the errors the bitstream must be partially decompressed. Start codes are extracted from the bitstream in order to find the slice where the erroneous ED block belongs. When that is achieved, the y-axis location of the error (inside a frame which is a two dimensional array) is known. To get the x-axis location of the error, decompression must continue until the address increment VLC of the macroblock is extracted. Note here that in order to get the x-axis location, we assume that in that slice there are no previous errors.

4.1.4 Error Concealment

Once the location of the errors in the uncompressed video are known, a standard concealment technique may be used. Here two different techniques were checked. First, a simple method based on the Boundary Matching Algorithm (BMA) [26]. As Figure 4.5 shows, the corrupted macroblock is concealed with information from the surrounding neighborhood. This method has two weak points which are the

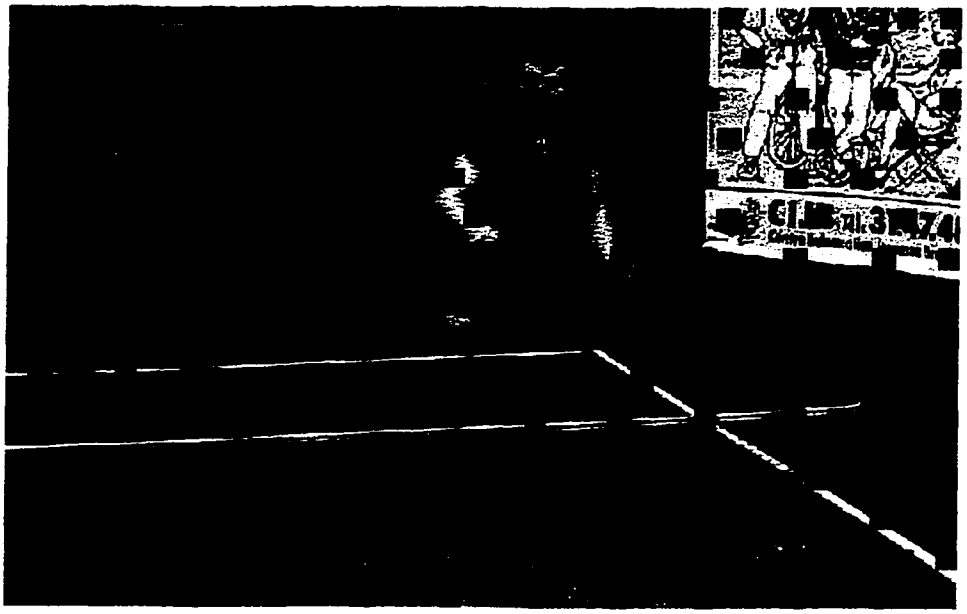


Figure 4.3: Corrupted Frame

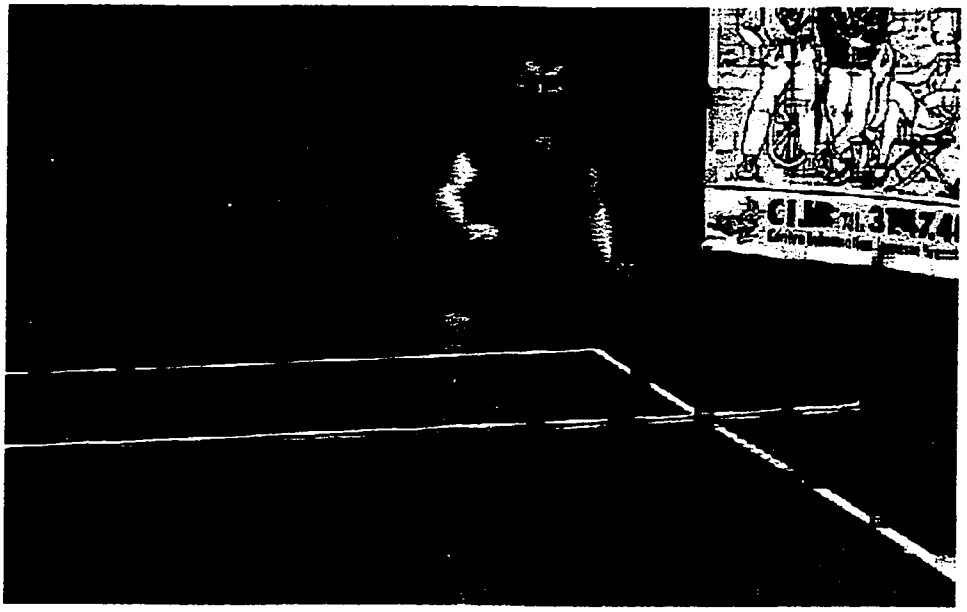


Figure 4.4: Concealed Frame using the BMA method

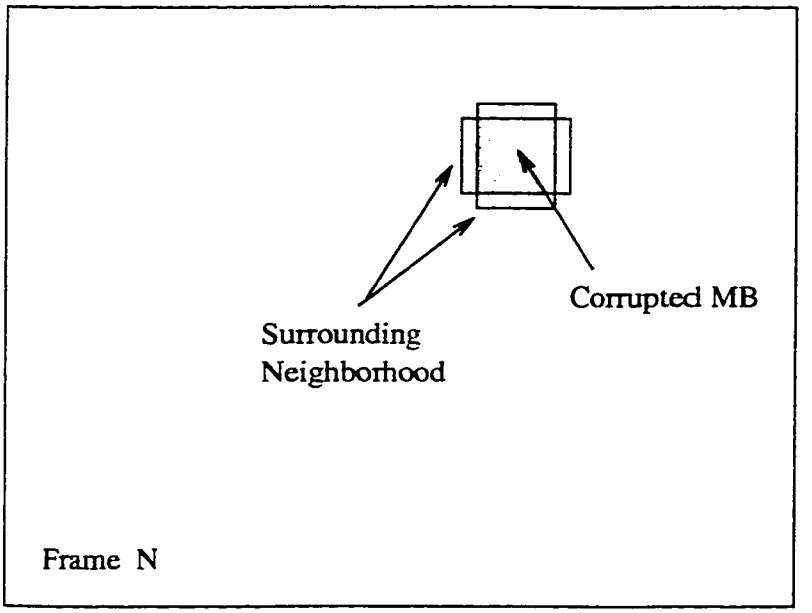


Figure 4.5: EC using BMA method

following:

1. In practice, adjacent macroblocks may be corrupted.

2. This method preserves only horizontal and vertical edges due to the bilinear interpolation method that it uses.

Figure 4.4 shows an example of the performance of the BMA method. In cases were there are horizontal and vertical edges in the missing MB the method performs well. The problem is in places were the missing MB includes edges in other directions (for example at the edge of the table).

The second method is a simple replacement method. As Figure 4.6 shows, the corrupted macroblock is concealed with information from the same location of the previous frame. This method gave better results than the first one and was used as the EC module of CEDEC. This method has also a weak point. It works only with sequences with no, or little motion (scene changes). The reason for that lies in the basic idea of this method. Here a MB located at the x,y location of frame N, which is corrupted is replaced by the x,y MB of frame $N-1$. When there is motion

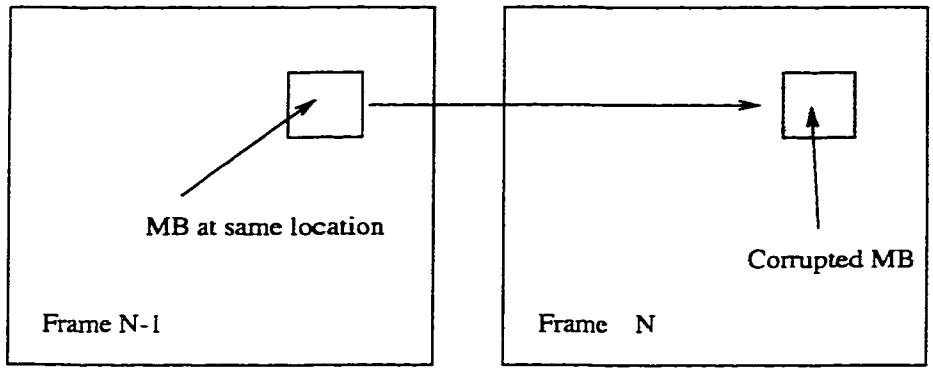


Figure 4.6: EC using simple replacement method

between frames N and $N-1$ the correct MB maybe the one located at $x+i$,$y+j$ of frame $N-1$.

4.2 Simulation Results

The performance of SBEC was tested against a general error correcting scheme which uses a Reed Solomon (RS) $(204, 188, t = 8)$ code. The sequence "Susie" encoded at 8 Mb/s (compression ratio 15:1) was used. To evaluate the scheme, a series of different experiments was conducted.

BER	RS	CEDEC
$1 * 10^{-7}$	44.64	44.64
$1 * 10^{-6}$	44.64	44.37
$1 * 10^{-5}$	44.64	43.37
$1 * 10^{-4}$	44.64	38.57
$1 * 10^{-3}$	44.51	33.04
$2 * 10^{-3}$	27.65	30.38

Table 4.1: PSNR values for Susie, compression ratio 15:1

Table 4.1 shows the average PSNR ($PSNR = 10 \log \frac{255^2}{MSE}$) over the 90-frame sequence. Figures 4.7 and 4.8 show the corrupted and concealed version of frame 16 from the sequence. As we can see here, the scheme performed well and the errors were concealed successfully. Figures 4.9 and 4.10 show the corrupted and concealed

Figure 4.7: Corrupted Frame #16 from Susie

Figure 4.8: Concealed Frame

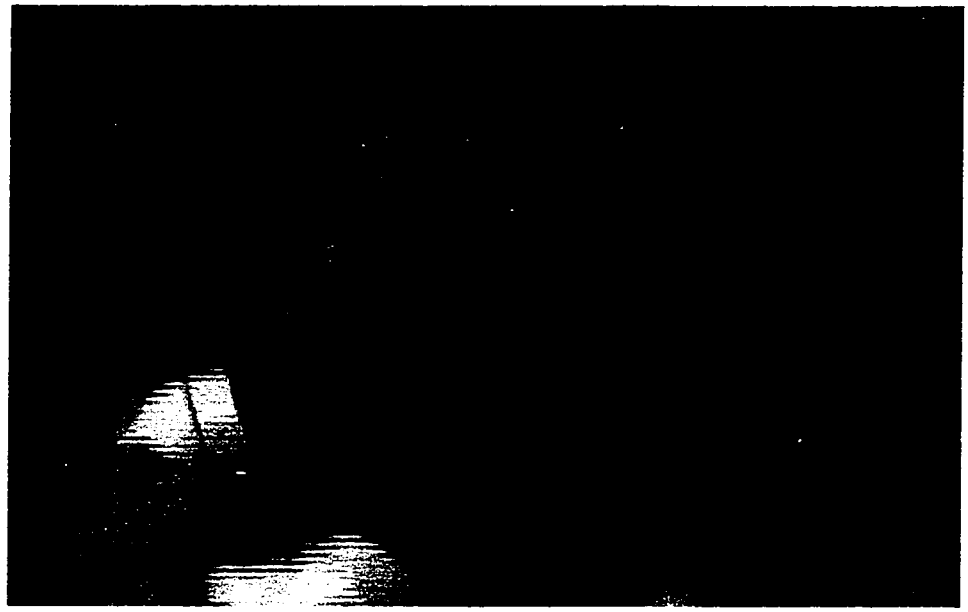


Figure 4.9: Corrupted Frame #31 from Susie

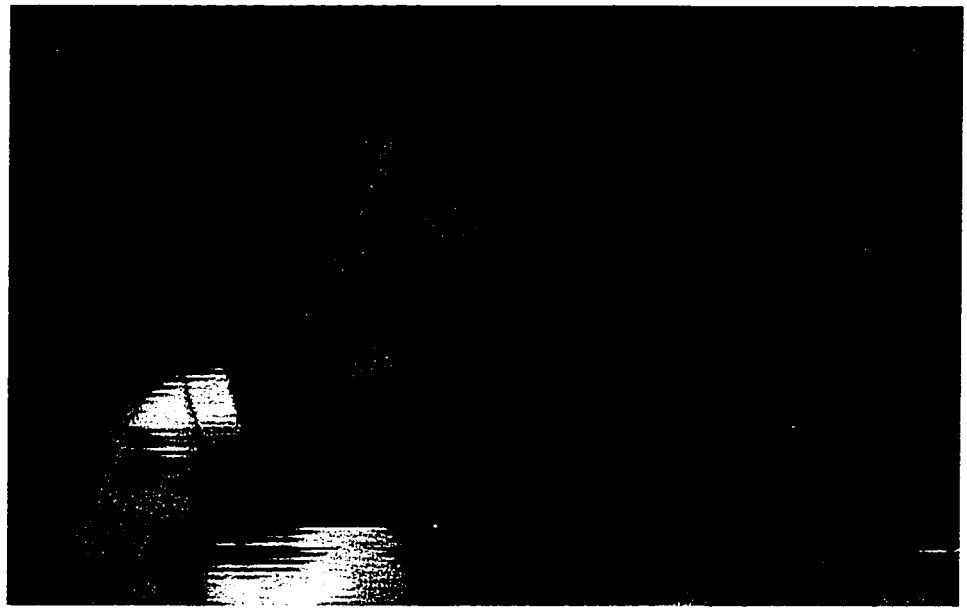


Figure 4.10: Concealed Frame

version of frame 31 from the same sequence. In this case the concealment was not successful. The main reason for that was the fact that between the corrupted frame and the previous frame used by the concealment there is a scene change.

4.3 Complexity

In Section 3.1.1 the complexity of a scheme using the shortened RS(204,188,$t = 8$) code, was presented.

Three factors contribute to the complexity of CEDEC:

- The complexity of the ED decoder
- The complexity of the Spatial locator
- The complexity of the Error Concealment

The complexity of the Spatial locator is simple. The whole process consists of extracting bits from the compressed bitstream, while tracking their location in the uncompressed video until the location of an error is reached. As the BER increases, however, this process becomes slower due to the increased number of errors present in the bitstream. The Error Concealment process also has simple complexity. What it does is to fill up a number of rectangles (of size $16 * 16$ pixels) with information preserved in a frame memory. Because it is difficult to predict the impact of an error in the video sequence, concealment performs from the reported error location, up to the end of the slice. Thus, for example if the error is located at the 20th MB of a slice (assuming frame size $704x480$ pixels, which means each slice contains 44 MBs), concealment must replace the values for 6144 pixels. Again, as the BER increases this process slows down due to the increased number of errors.

4.4 Discussion

From the simulation results it is clear both subjectively and objectively that the RS scheme performs better than CEDEC. CEDEC, however, as discussed in 4.3 seems to have lower complexity.

One thing that came out during simulations is that the EC does not perform well. This does not refer only to the concealment used here (which is a simple one), but is a general remark concerning EC. As shown in Chapter 2, there are two major types of EC: Spatial EC and Temporal EC. Spatial EC uses information from the surrounding area to hide the errors. The weak point here is what happens when the surrounding area is damaged too. Temporal EC uses information from the previous frames, obtained using motion information. For this type of EC the weak points are what happens when the motion information is damaged or non existent (I frames), or when the anchor frame is damaged too.

Another weak point of EC is the fact that the effect of errors in the bitstream cannot be predicted. The error may not cause visual degradation (see Figure 4.11), may degrade just a single block (see Figure 4.12), or may corrupt a whole slice (see Figure 4.13). To address this problem EC must perform from the point that there is report of error until the end of the slice. This process (see Figures 4.14, 4.15 and 4.16) sometimes degrades the quality of the video.

In CEDEC the error location problem is addressed. Error Concealment schemes which try to solve practical errors, in general, refer to a transport mechanism, which provides the error location [21]. In CEDEC the Spatial Locator is used, as mentioned earlier, to map error locations in the compressed domain to error locations in the uncompressed domain.

Here we choose to add errors only in DCT coefficients of B-frames. There were two reasons for this. The first reason was to avoid error propagation and so be able to deal with errors only at the spot that they occurred. The second reason was to be able to use a simple error concealment scheme in order to correct the errors. The

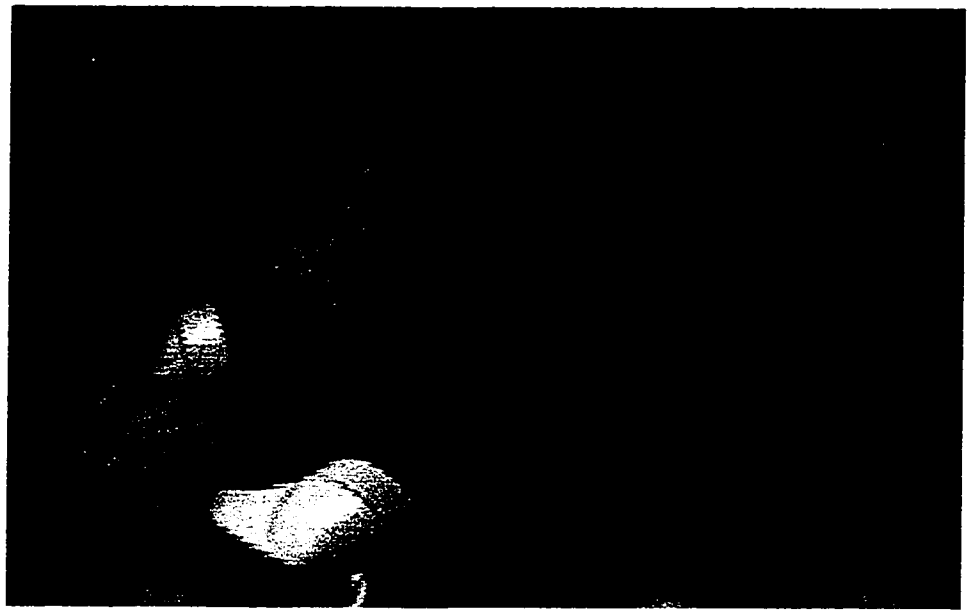


Figure 4.11: Error with no Visual Effect

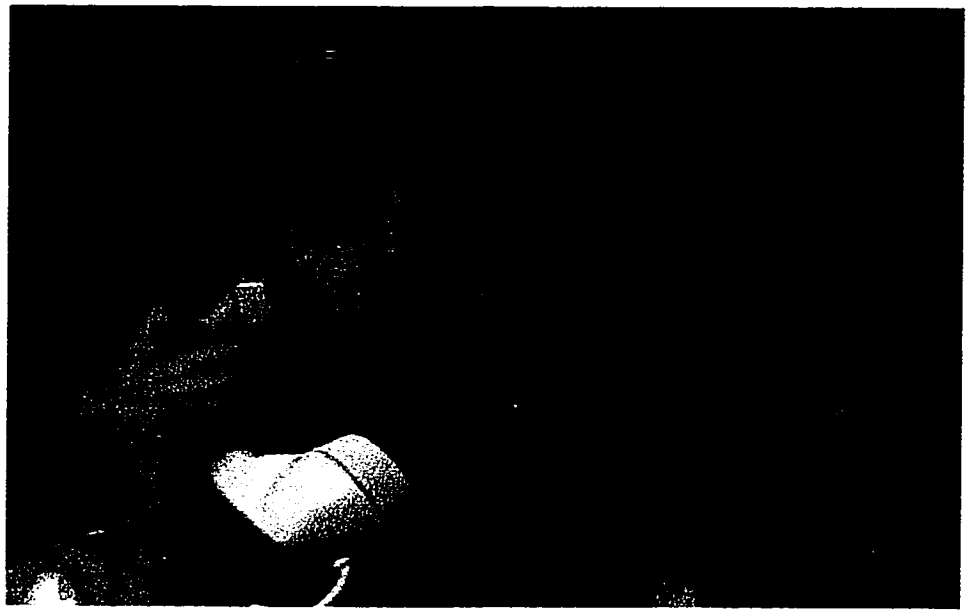


Figure 4.12: Error which degrades a Simple Block

Figure 4.13: Error which corrupts a portion of a Slice

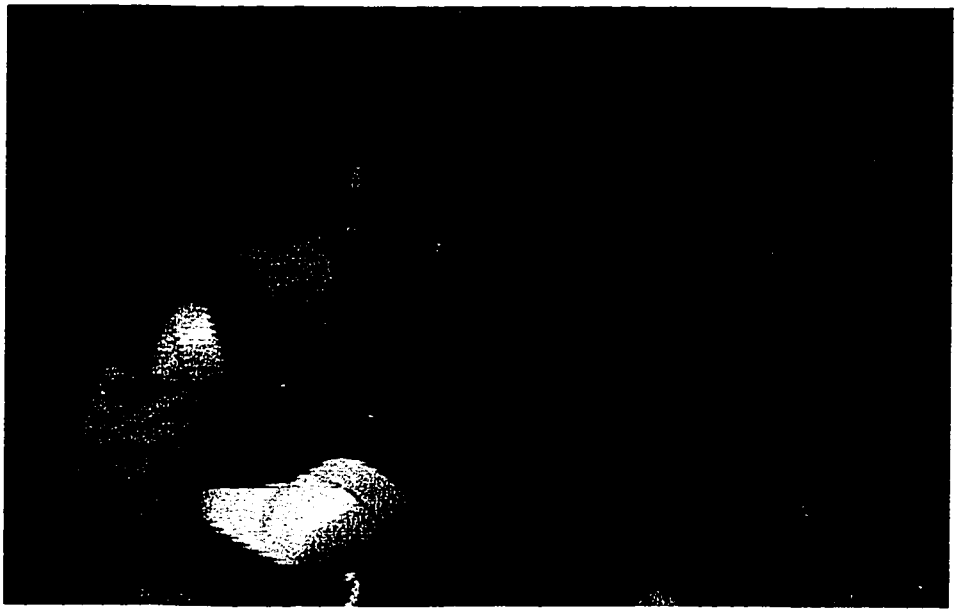


Figure 4.14: Concealed version of Figure 4.11

Figure 4.15: Concealed version of Figure 4.12

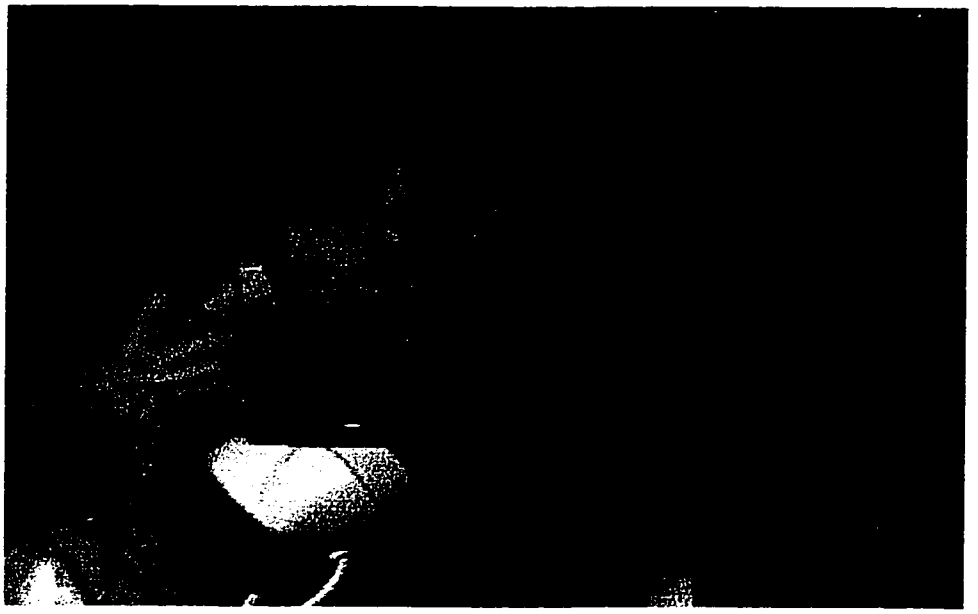


Figure 4.16: Concealed version of Figure 4.13

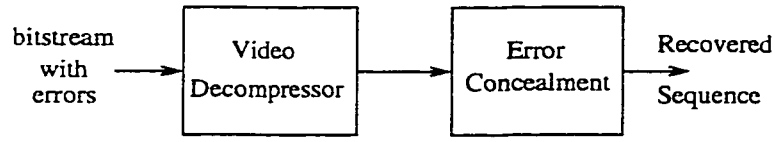
next step, after evaluating the performance of CEDEC for this simple case, was to permit errors in all the bitstream components inside the slice layer.

As shown in Table 4.2, the slice layer has a large number of components. For each entry of the Table, a study is required in order to evaluate the effect of an error at that location. For the DCT coefficients that study led to three type of errors:

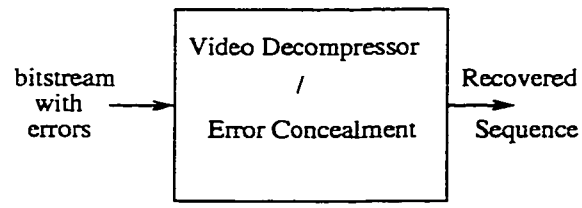
1. Errors that the video decompressor does not detect.

2. Errors that eventually, but not immediately, cause the decompressor to enter into an illegal condition.

3. Errors that immediately cause the decompressor to enter into an illegal condition.



a) Error Concealment Performed after video decompression



b) Error Concealment performed during video decompression

Figure 4.17: Performance of EC after and during the video decompression

To study errors in other components more efficient error concealment schemes must be used. Also, in order to permit errors in I and P frames, the concealment must be done during the decompression (see Figure 4.17) to reduce error propagation. That is because, for example, if an error exists in a P frame, because the P frame will be decompressed before the B frames (in a PBBP or a IBBP order), that

Name	Size	Occurance	Frequency
slice start code	36 bits	slice header	Every slice
slice vertical position extension	3 bits	slice header	If vertical size > 2800
priority breakpoint	7 bits	slice header	If scalable mode = data partitioning
quantizer scale code	5 bits	slice header	Every slice
intra slice flag	1 bit	slice header	if bit value = 1
intra slice	1 bit	slice header	if intra slice flag
reserved bits	7 bits	slice header	Every slice
extra bit slice	1 bit	slice header	Every slice
extra information slice	8 bits	slice header	If previous bit = 1
macroblock escape	11 bits	Start of MB	If more than 33 skipped MB's
macroblock address increment	1 – 11 bits	Start of MB	Every MB
macroblock type	1 – 9 bits	Start of MB	Every MB
spatial Temporal weight code	2 bits	Start of MB	For Spatial Scalability
frame motion type	2 bit	Start of MB	for P&B frames, frame DCT coded
field motion type	2 bits	Start of MB	for P&B frames, field DCT coded
dct type	1 bit	Start of MB	Every MB
quantizer scale code	5 bits	Start of MB	if MB type = quant
motion vertical field select	1 bit	Start of MB	depends on prediction type
motion code	1 – 11 bits	Start of MB	depends on prediction type
motion residual	1 – 8 bits	Start of MB	depends on prediction type
dmvector	1 – 2 bits	Start of MB	if dmv = 1
motion code	1 – 11 bits	Start of MB	depends on prediction type
motion residual	1 – 8 bits	Start of MB	depends on prediction type
dmvector	1 – 2 bits	Start of MB	if dmv = 1
marker bit	1 bits	Start of MB	for intra MB with concealment vectors
coded block pattern 420	3 – 9 bits	Start of MB	if MB pattern = 1
coded block pattern 1	2 bits	Start of MB	for 4 : 2 : 2 case
coded block pattern 2	6 bits	Start of MB	for 4 : 4 : 4 case
dct dc size luminance	2 – 9 bits	MB data	if MB intra
dct dc differential	1 – 11 bits	MB data	if previous value $\neq 0$
dct dc size chrominance	2 – 10 bits	MB data	if MB intra
dct dc differential	1 – 11 bits	MB data	if previous value $\neq 0$
dct coefficients	$2bits - \#bytes$	MB data	Every Block

Table 4.2: Bitstream Components (Slice layer)

error will not be passed to the following B frames. Some of the bitstream components, however, must be error free, in order for CEDEC to be able to work. That is because we need to have the x and y axis location of the error. The y-axis location is provided by the slice header. The x-axis location is provided by the macroblock address increment code. However it is vital for the good performance of the scheme that this code is well protected.

The study of CEDEC was not completed, in the sense that not all of the bitstream components were studied, more efficient error concealment was not introduced, and different sequences were not used to evaluate it. One reason for that was the fact that the required scheme (both error protection and error concealment) would be complicated because not all the bitstream components required the same treatment. But the main reason was the idea of exploiting the inherent redundancy of the bitstream syntax. That led to the study of a new scheme called Syntax Based Error Concealment, the detailed presentation of which comes in the next chapter.

Chapter 5

Syntax Based Error Concealment Technique

In Chapter 4 a method for error control in a compressed video sequence was examined. It relies on redundant information which still remains in the video after compression in order to alleviate transmission errors. In Chapter 3 another method was presented which adds (prior to transmission) redundant data into the compressed bitstream.

In this chapter a new method [27] for recovering transmission errors in a MPEG2 compressed bitstream is presented. The method relies on the redundancy which the syntax of the compressed bitstream provides.

In Section 5.1 the different modules of SBEC are presented. Design decisions concerning the ED code which is used are given in 5.1.1. Different choices which can be followed are examined and discussed. In 5.1.3 the idea of altering different bits in an ED block marked as in error, and checking if that violates the syntax rules, is presented. Experimental results are given in Section 5.2. In Section 5.3 the complexity of SBEC is discussed, while a short discussion of the performance of the scheme is given in Section 5.4.

5.1 Syntax Based Error Concealment

Here a new error resilience scheme is introduced, which is called Syntax Based Error Concealment (SBEC). The basic idea is to use the syntax structure of the MPEG2 compressed bitstream to remove the most serious effects of the errors from the video sequence.

The scheme is shown in Figure 5.1. After compression the bitstream is encoded for error detection, and then transmitted through a channel. At the receiver end data pass through the ED decoder and are checked for errors. The bitstream, along with the detected erroneous ED blocks, passes to the Bit Toggler. There, for each detected error, several decompressions of the slice which has the error may take place. The first decompression that will not cause a syntax violation will be accepted as

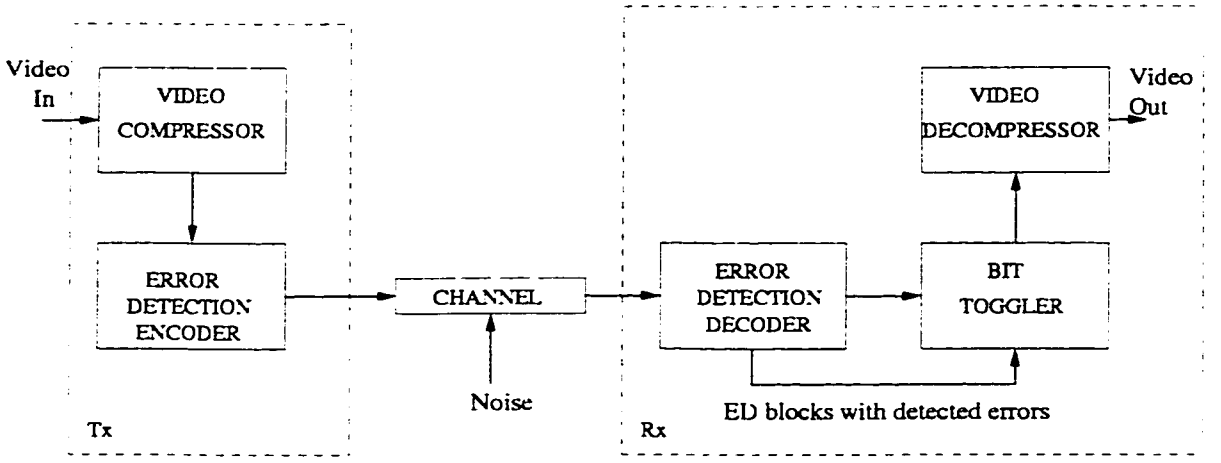


Figure 5.1: The SBEC Technique

the valid decompressed video.

5.1.1 Transmitter Part (Tx)

Video Compressor

This module is used to compress the video sequence. Because we concentrate our work to study ways for robust MPEG2 [1] video transportation and recovery, this module must be a MPEG2 compliant compressor. Currently the MPEG2 compressor V1.2 is used.

Error Detection Encoder

After compression the bitstream is encoded for error detection. Here a single parity-check code is used. For every block of k information bits a check bit is appended. The check bit is chosen to satisfy the overall parity rule for the codeword, which can be either odd or even. With such a code, any odd number of errors will be detected.

To design the ED code, the size of the bitstream data block k, that will be encoded, had to be specified. For comparison it is desirable that the Bandwidth Expansion (BE), defined as $BE = \frac{codewordsize}{datablocksize}$, that the scheme adds to the compressed

bitstream be comparable to that of a Reed Solomon based scheme. The data block size was chosen to be 12 bits. For each data block of 12 bits, one parity bit is added. With data block chosen to 12 bits, the BE is 1.083, which is slightly less than the BE the RS$(204, 188, t = 8)$ scheme adds which is 1.085.

Different sizes of data block could also be used. Changing the size of the block, however, will affect the BE that is added to the bitstream.

Data Block	BE
12	1.083
13	1.076
14	1.071

Table 5.1: Bandwidth Expansion for Different Block Size

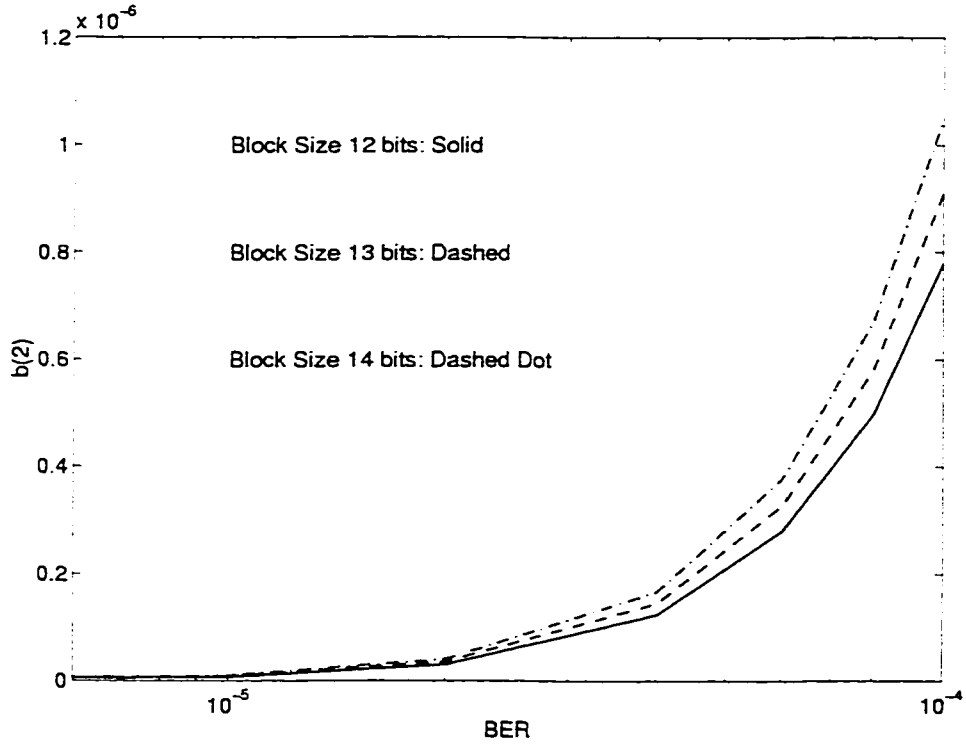


Figure 5.2: Probability of two errors per ED block, for different ED block sizes

Table 5.1 shows that by choosing bigger data block size the added BE is reduced. A bigger data block, on the other hand, means more undetected errors.

That is because for a block of n bits, the probability of x bits in error is given by:

$$b(x; n, p) = \begin{pmatrix} n \\ x \end{pmatrix} p^x q^{n-x}$$

where p is the probability of a bit error, and

$$q = 1 - p$$

Figure 5.2 gives the probabilities of two bits in errors for three different data block sizes, and for a range of bit error rates (BER). So by choosing bigger data blocks the probability of undetected errors (blocks with 2 errors) will be increased. That will degrade the performance of the scheme. Note also that choosing different data blocks will affect the complexity of SBEC (see Section 5.3).

5.1.2 Channel

This module simulates the channel traffic. As discussed in Section 1.1 there are two different types of transmission errors. Here, due to the nature of the concealment which is used, the decision was to study the case of bit error occurance in the bitstream. The Channel module takes as an input the video bitstream, and adds random white noise. The amount of added noise is controlled from the selected BER case. For the simulations that were conducted, noise is added only to a portion of the bitstream. Errors are permitted only inside slices. The rest of the bitstream - consisting of picture, GOP and sequence header information - is considered vital for the decompressing process, and assumed to be well protected.

5.1.3 Receiver Part (Rx)

Error Detection Decoder

At the decoder the received codeword, possibly containing transmission errors, is checked to determine whether or not the parity relationship is satisfied. If the

parity relationship is not satisfied, the ED block will be marked with error. This information will then pass to the Bit Toggler.

Video Decompressor

The decompressor module must have the two following characteristics:

- Ability to recognize important violations of the syntax.
- Ability to perform in the presence of errors.

Currently the MPEG2 decompressor V1.2*a* is used.

Bit Toggler

For each block error reported by ED, the slice which contains it will be decompressed up to 13 times (Block size $12 + 1$ bits). In each decompression, a different bit from the block will be altered (toggled), starting with the parity bit. Note that changing the parity bit does not alter the bitstream. The first case that will result in a decompression without violation of the bitstream syntax will be accepted as the correct bitstream.

By doing that, errors are not always removed from the bitstream. In fact there will be cases where more errors are introduced. For example, if the ninth bit is the true error, but toggling the fourth bit results in a decompression without syntax violations, then this decompression will be accepted as correct. Simulation results indicate that the fact of introducing more errors does not degrade significantly (see Table 5.2) the performance of SBEC.

Complication occurs as the BER increases. Higher BER means more errors per slice. That has a dual impact upon SBEC. More errors means increasing probability for two errors per ED block, in other words increasing probability for undetected by the ED code errors. Also, more slice errors means slower performance of the scheme. For a single slice error, we have to decompress the particular slice up to 13 times. A

BER	Total errors	Errors introduced by SBEC	SBEC PSNR	No Process PSNR
10^{-7}	6	1	38.74	37.94
10^{-6}	51	8	37.45	32.63
10^{-5}	495	85	31.86	21.80

Table 5.2: Errors introduced by SBEC and their impact in the Video Quality

second error in the same slice simply means that we may have to decompress that slice up to $13^2 = 169$ times.

5.2 Simulation Results

The performance of SBEC was tested against a general error correcting scheme which uses a Reed Solomon (RS)$(204, 188, t = 8)$ code, and a simple case where no effort is made to correct existing errors (NP). Two sequences were used during the simulations, "Flower-Garden" and "Table-Tennis", each one consisting of 180 frames. Because the frame rate which is used is 30 frames/sec, that means each sequence lasts 6sec.

To evaluate SBEC, a series of different experiments were conducted. Table 5.3 gives a summary of the contacted experiments. The purpose of the different experiments is to get an idea about how the scheme corresponds for different Sequences and in different bit rates.

Changing the bit rate means controlling the amount of data present in the bitstream. The larger that amount becomes (higher bit rates), the larger the amount of bit error occurances in the bitstream. Decreasing it (lower bit rates) reduces the number of bit error occurances.

The purpose of checking different Sequence contents is to examine whether the performance of the method depends on the presence of scene changes and motion. Classical EC, based on redundancies still present in the bitstream, usually degrades when the Sequence includes a lot of scene changes and high motion. It is interesting

to see how SBEC, which uses the syntax structure to perform EC, responds in such cases.

Experiment	Sequence Used	Bit Rate
$1st$	"Flower-Garden"	8Mb/s
$2nd$	"Table-Tennis"	8Mb/s
$3rd$	"Flower-Garden"	12Mb/s
$4th$	"Flower-Garden"	4Mb/s

Table 5.3: Summary of Simulations

5.2.1 First Experiment

For the first experiment 180 frames (6*sec*) from the Sequence "Flower-Garden", encoded at 8 Mbits/s, were used. The compression ratio is equivalent to 15 : 1.

BER	RS	BT	NP
	Y Cb Cr	Y Cb Cr	Y Cb Cr
$1 * 10^{-7}$	35.17 38.25 40.37	35.17 38.25 40.37	35.08 38.25 40.36
$2 * 10^{-7}$	35.17 38.25 40.37	35.17 38.25 40.37	34.64 38.16 40.32
$4 * 10^{-7}$	35.17 38.25 40.37	35.16 38.25 40.37	33.73 37.99 40.08
$6 * 10^{-7}$	35.17 38.25 40.37	35.12 38.24 40.37	33.20 37.40 39.93
$8 * 10^{-7}$	35.17 38.25 40.37	35.01 38.16 40.33	32.54 37.10 39.87
$1 * 10^{-6}$	35.17 38.25 40.37	34.93 38.15 40.29	32.19 36.40 39.27
$2 * 10^{-6}$	35.17 38.25 40.37	34.37 37.97 40.22	30.57 36.12 39.11
$4 * 10^{-6}$	35.17 38.25 40.37	33.93 37.63 40.02	28.06 34.47 37.41
$6 * 10^{-6}$	35.17 38.25 40.37	32.78 37.20 39.77	26.09 33.86 37.06
$8 * 10^{-6}$	35.17 38.25 40.37	32.40 37.11 39.44	25.93 33.73 34.89
$1 * 10^{-5}$	35.17 38.25 40.37	31.14 36.77 38.50	23.78 31.40 34.64
$2 * 10^{-5}$	35.17 38.25 40.37	28.88 34.17 37.53	21.14 28.17 31.98
$4 * 10^{-5}$	35.17 38.25 40.37	25.99 33.36 36.80	18.46 26.62 31.12
$6 * 10^{-5}$	35.17 38.25 40.37	24.92 32.09 36.04	16.67 24.06 27.51
$8 * 10^{-5}$	35.17 38.25 40.37	23.41 31.54 34.51	15.55 23.00 26.60
$1 * 10^{-4}$	35.17 38.25 40.37	20.94 29.42 32.83	13.82 20.22 22.15
$8 * 10^{-4}$	35.13 38.24 40.36	10.00 13.25 11.26	06.23 09.48 08.81
$9 * 10^{-4}$	34.39 37.86 40.11	09.72 12.74 09.94	05.95 09.17 08.49
$1 * 10^{-3}$	34.44 37.37 39.21	09.34 11.98 09.58	05.57 08.61 08.13
$2 * 10^{-3}$	18.61 26.32 39.21	08.47 10.76 08.56	04.70 07.59 07.14
$3 * 10^{-3}$	10.00 14.47 14.09	08.28 10.14 08.32	04.51 07.27 06.94

Table 5.4: PSNR values for "Flower-Garden", compression ratio 15 : 1

Figure 5.3 gives the average PSNR for the luminance (Y), blue chrominance (Cb), and red chrominance (Cr) parts of the frames. For BER between 10^{-7} and $2 * 10^{-6}$ the performance of SBEC was less than one db (1*db*) difference from the performance of the RS scheme. For BER between $4 * 10^{-6}$ and 10^{-5} the performance of SBEC was above the 30 db limit. Figure 5.4 shows the Original frames 38, 110, 4. Figure 5.5 shows the same frames after errors were inserted. Table 5.5 summarizes

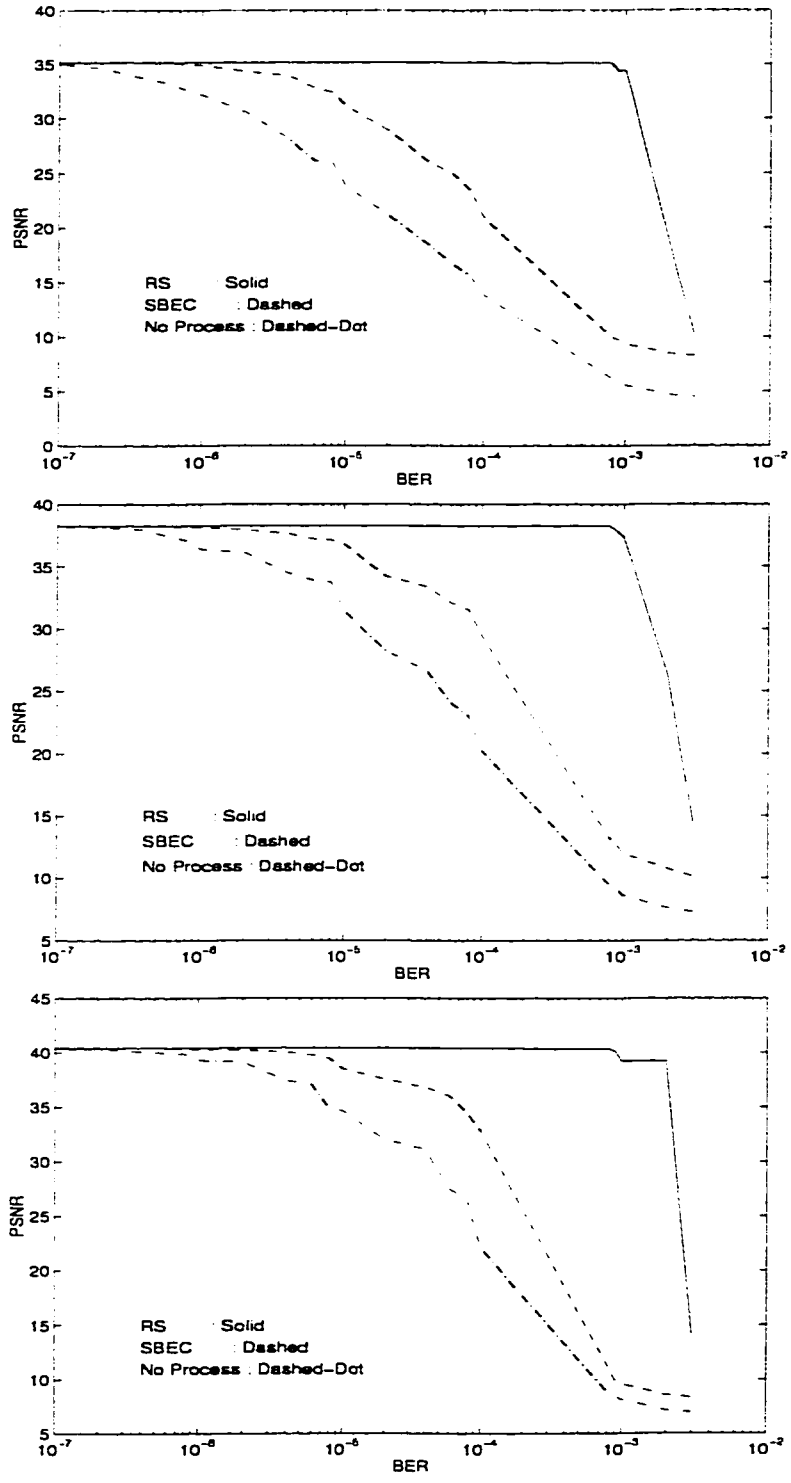


Figure 5.3: PSNR: Luminance, Blue Chrominance, and Red Chrominance for "Flower-Garden" 8 Mb/s

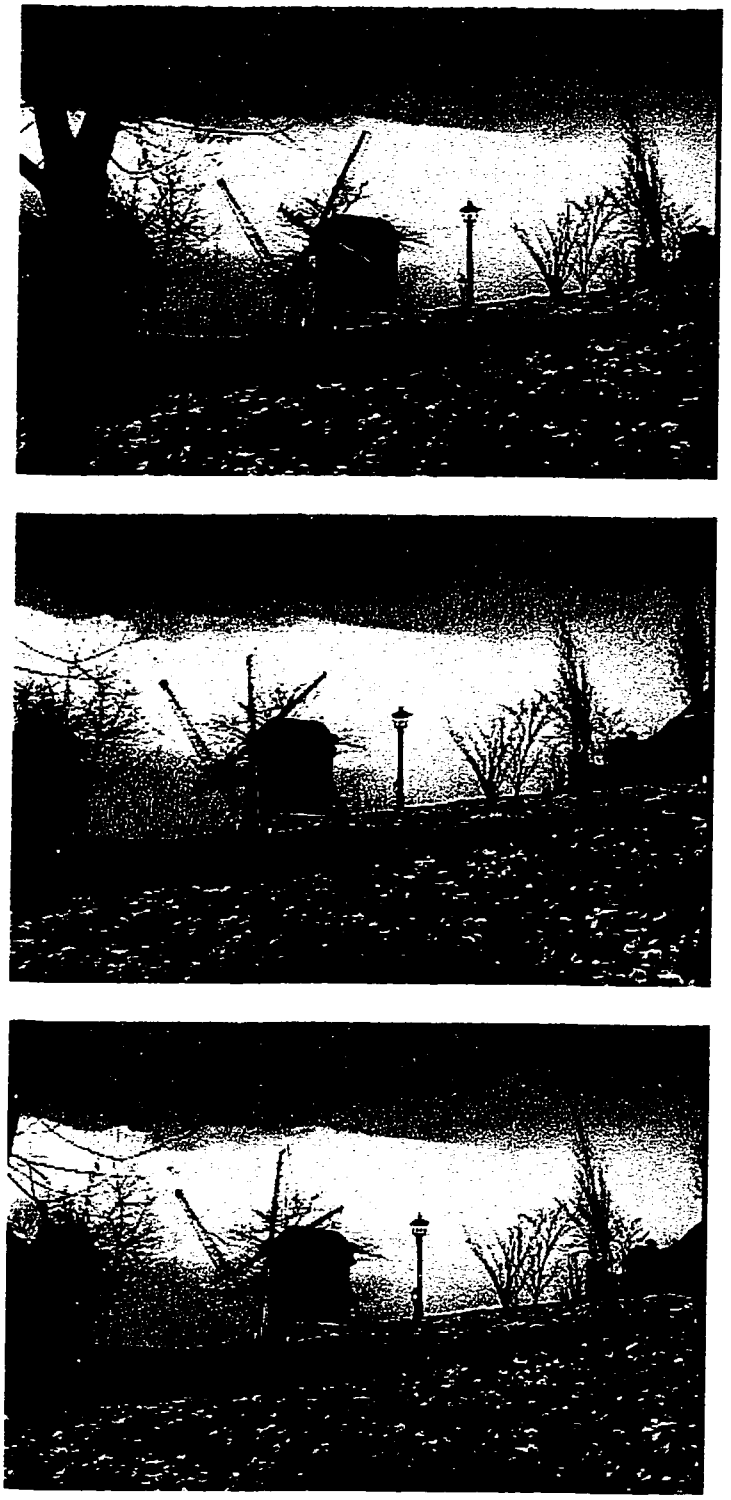


Figure 5.4: Original Frames 38, 110, 4 from "Flower-Garden", compression ratio 15 : 1

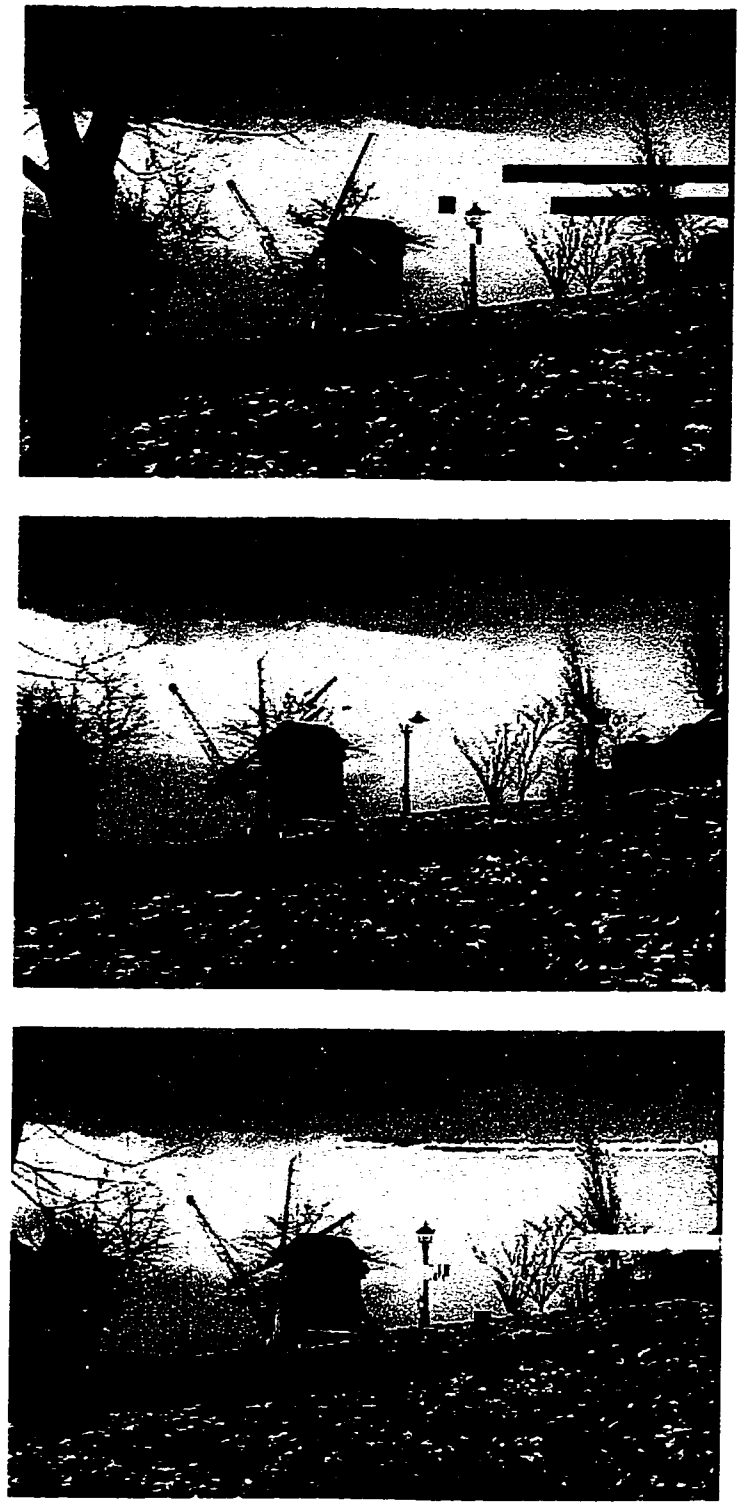


Figure 5.5: Corrupted Frames 38, 110, 4 from "Flower-Garden"

Figure 5.6: Frames 38, 110, 4 from "Flower-Garden" after Bit Toggling

the 3 examples.

BER	Frame number
$6 * 10^{-6}$	38
$1 * 10^{-5}$	110
$2 * 10^{-5}$	4

Table 5.5: Channel BER for frames $38, 110, 4$

In frame 38 the errors were serious enough to cause the discarding of a portion of the slice where they occurred. Also there is a shift to the right located at the center of the frame. In frame 110 there is an error located at the upper right of the frame. Also there are two shifts to the right. One is located at the top of the trees at the right edge of the frame, while the second is located at the center of the frame. In frame 4 there are two white stripes and also a shift to the right located at the center of the frame.

Figure 5.6 shows frames $38, 110, 4$ after the sequence was processed by SBEC. In frame 38 only a small portion of the black stripes remained, caused by undetected error which occurred in a previous anchor frame. Note also that the shift to the right has been corrected. In frame 110 the error at the upper right remained while the two shifts to the right were fixed. In frame 4 the two white stripes were not removed, but the process was able to remove the shift to the right.

5.2.2 Second Experiment

For this experiment, again, 180 frames (6*sec*) from a Sequence encoded at 8 Mbits/s, compression ratio equivalent again to 15 : 1, were used. What changed here is the Sequence itself. Instead of "Flower-Garden", the Sequence "Table-Tennis" was used.

BER	RS	BT	NP
	Y Cb Cr	Y Cb Cr	Y Cb Cr
$1 * 10^{-7}$	38.37 46.83 46.68	38.37 46.83 46.68	38.33 46.82 46.61
$2 * 10^{-7}$	38.37 46.83 46.68	38.35 46.83 46.68	38.26 46.73 46.55
$4 * 10^{-7}$	38.37 46.83 46.68	38.33 46.83 46.68	37.92 46.46 46.26
$6 * 10^{-7}$	38.37 46.83 46.68	38.32 46.83 46.68	37.80 46.37 46.25
$8 * 10^{-7}$	38.37 46.83 46.68	38.13 46.81 46.68	37.72 46.25 45.99
$1 * 10^{-6}$	38.37 46.83 46.68	38.12 46.72 46.65	36.67 44.71 44.03
$2 * 10^{-6}$	38.37 46.83 46.68	37.64 46.71 46.18	35.51 44.21 44.02
$4 * 10^{-6}$	38.37 46.83 46.68	36.15 46.41 46.09	34.53 44.11 43.91
$6 * 10^{-6}$	38.37 46.83 46.68	35.96 45.18 44.10	33.70 42.76 41.00
$8 * 10^{-6}$	38.37 46.83 46.68	34.67 44.91 44.00	32.62 42.70 40.04
$1 * 10^{-5}$	38.37 46.83 46.68	34.62 44.41 43.76	31.71 41.07 39.55
$2 * 10^{-5}$	38.37 46.83 46.68	33.77 44.25 42.85	29.85 39.74 37.55
$4 * 10^{-5}$	38.37 46.83 46.68	28.59 38.70 38.11	24.32 32.75 31.08
$6 * 10^{-5}$	38.37 46.83 46.68	26.89 38.26 36.88	21.90 30.33 28.50
$8 * 10^{-5}$	38.37 46.83 46.68	25.43 35.59 34.30	20.42 27.84 26.24
$1 * 10^{-4}$	38.37 46.83 46.68	23.95 34.99 33.53	19.83 26.34 24.57
$8 * 10^{-4}$	38.35 46.83 46.68	11.74 16.63 15.75	08.75 11.04 09.80
$9 * 10^{-4}$	38.33 46.82 46.66	11.32 15.04 14.40	08.36 10.43 09.21
$1 * 10^{-3}$	38.18 46.78 46.63	10.54 13.64 13.64	07.92 09.84 08.76
$2 * 10^{-3}$	26.41 36.93 32.58	10.01 12.37 11.37	06.87 08.59 07.27
$3 * 10^{-3}$	13.90 16.92 15.21	09.93 10.98 10.48	06.47 08.05 06.80

Table 5.6: PSNR values for "Table-Tennis", compression ratio 15 : 1

Figure 5.7 gives the average PSNR for the luminance (Y), blue chrominance (Cb), and red chrominance (Cr) parts of the frames. For BER between 10^{-7} and $2 * 10^{-6}$ the performance of SBEC was less than one db (1*db*) difference from the

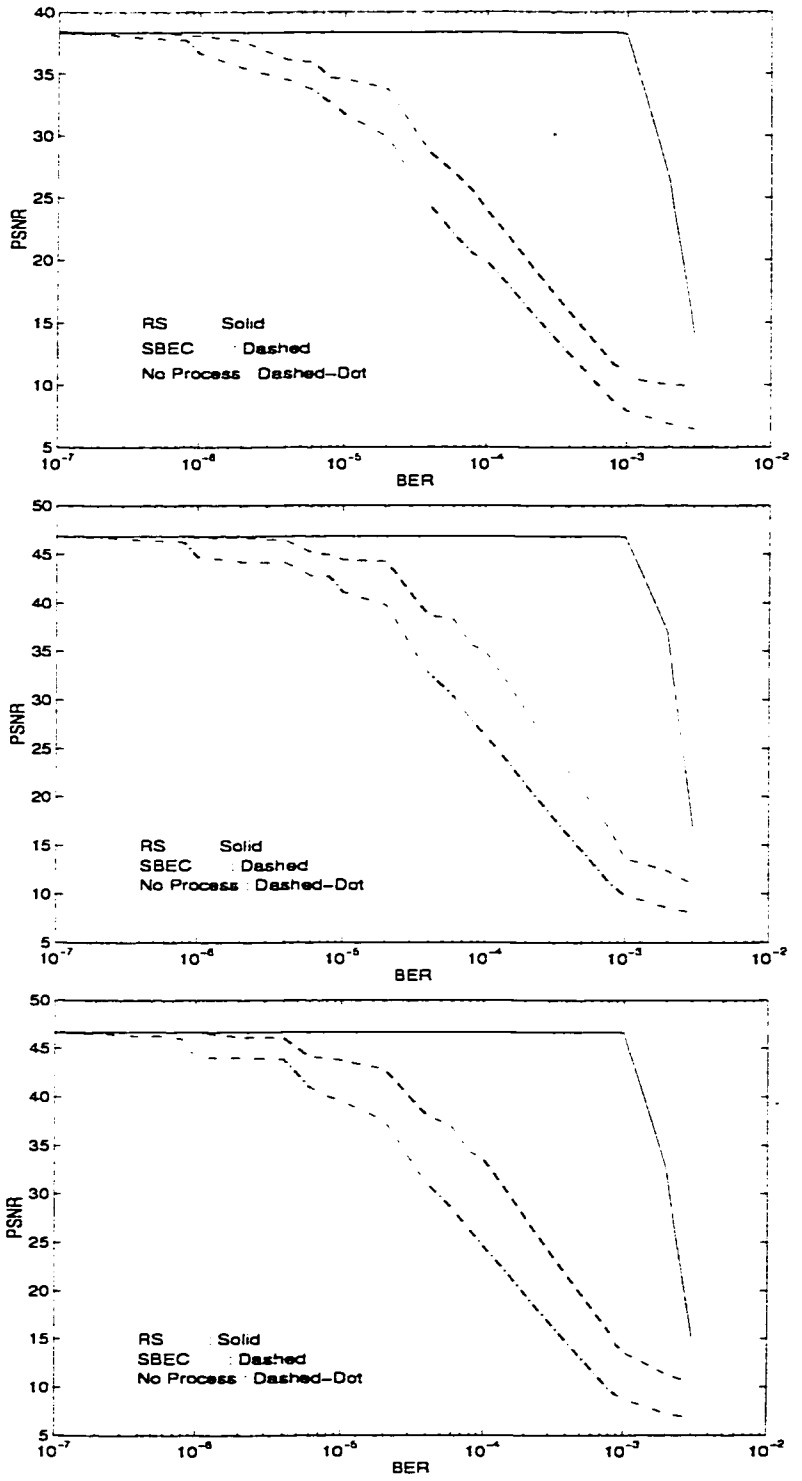


Figure 5.7: PSNR: Luminance, Blue Chrominance, and Red Chrominance for "Table-Tennis" 8 Mb/s

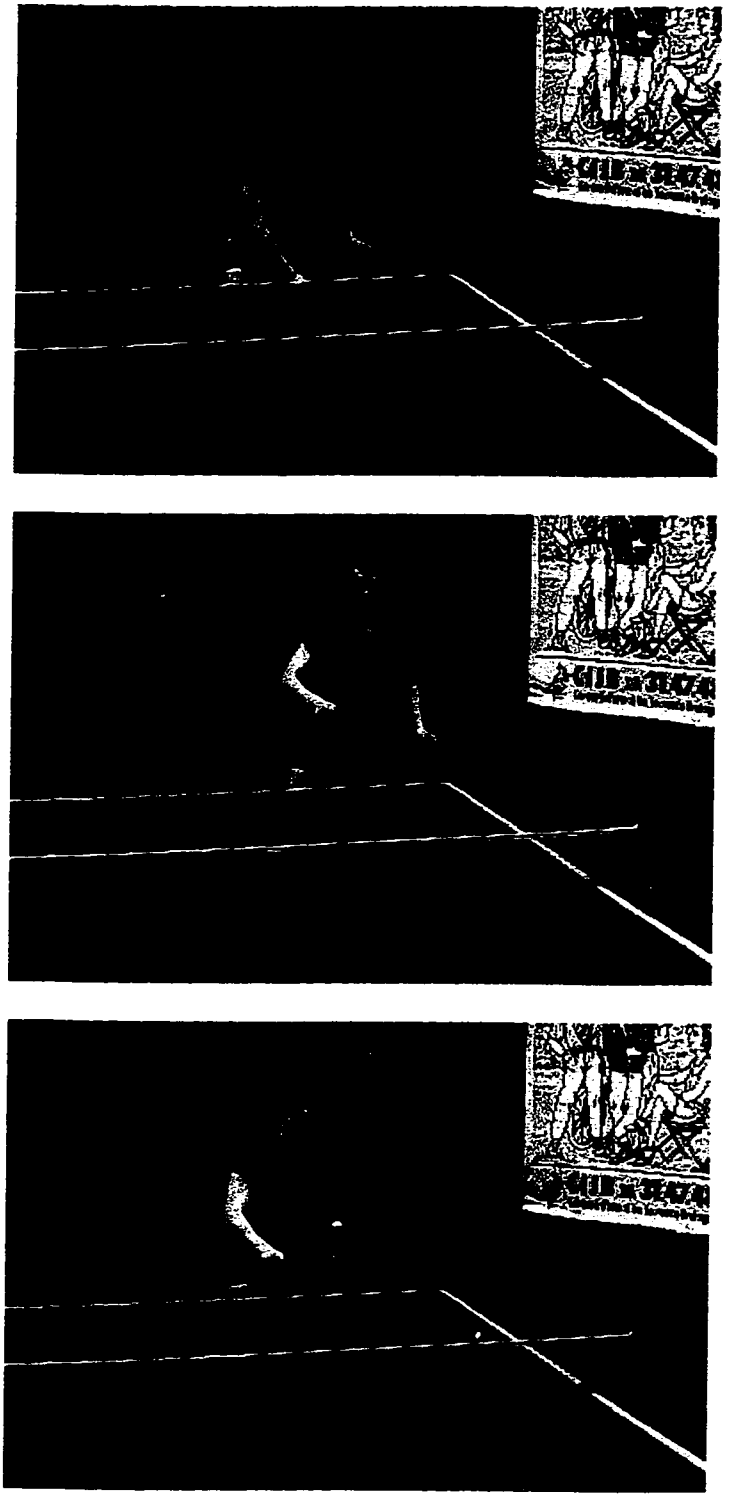


Figure 5.8: Original Frames 27, 6, 88 from "Table-Tennis", compression ratio 15 : 1

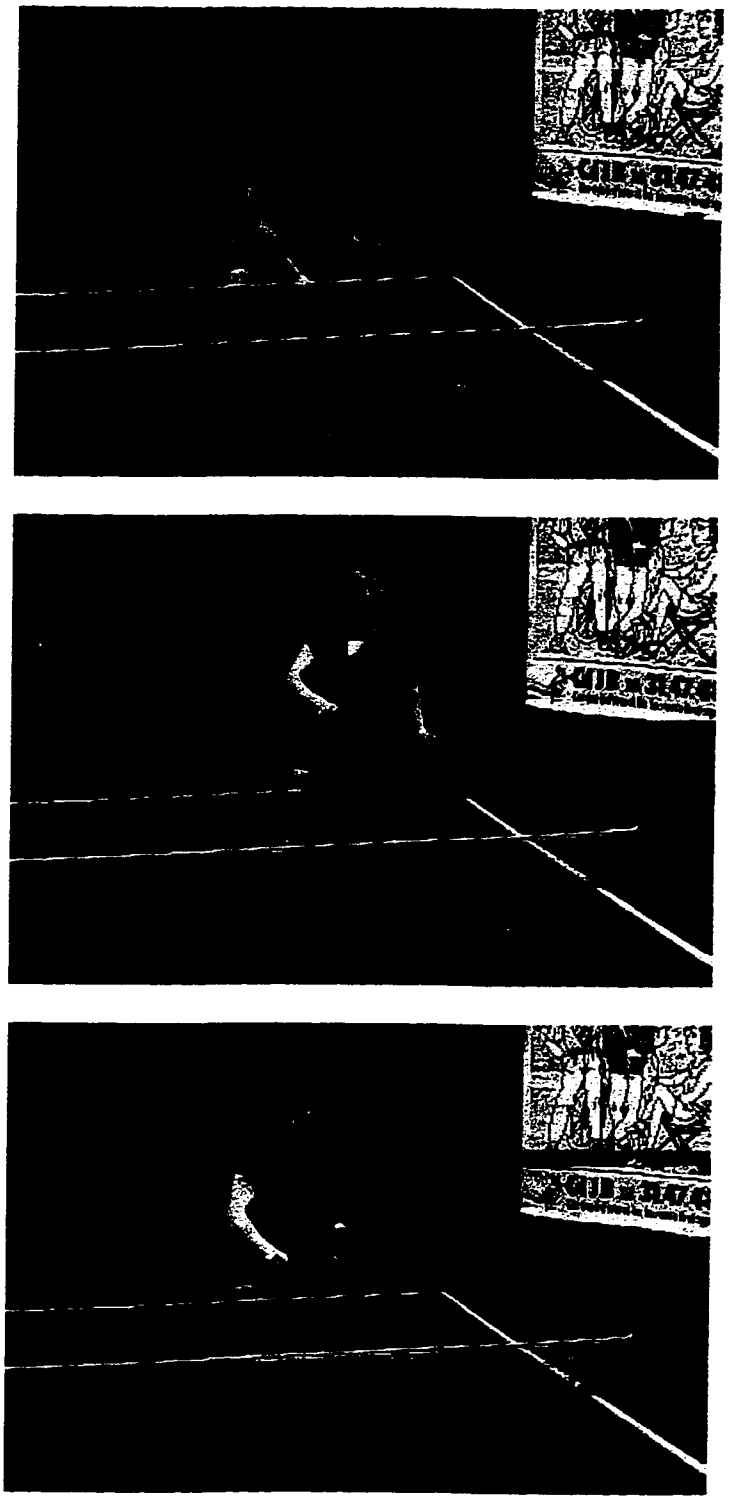


Figure 5.9: Corrupted Frames 27, 6, 88 from "Table-Tennis"

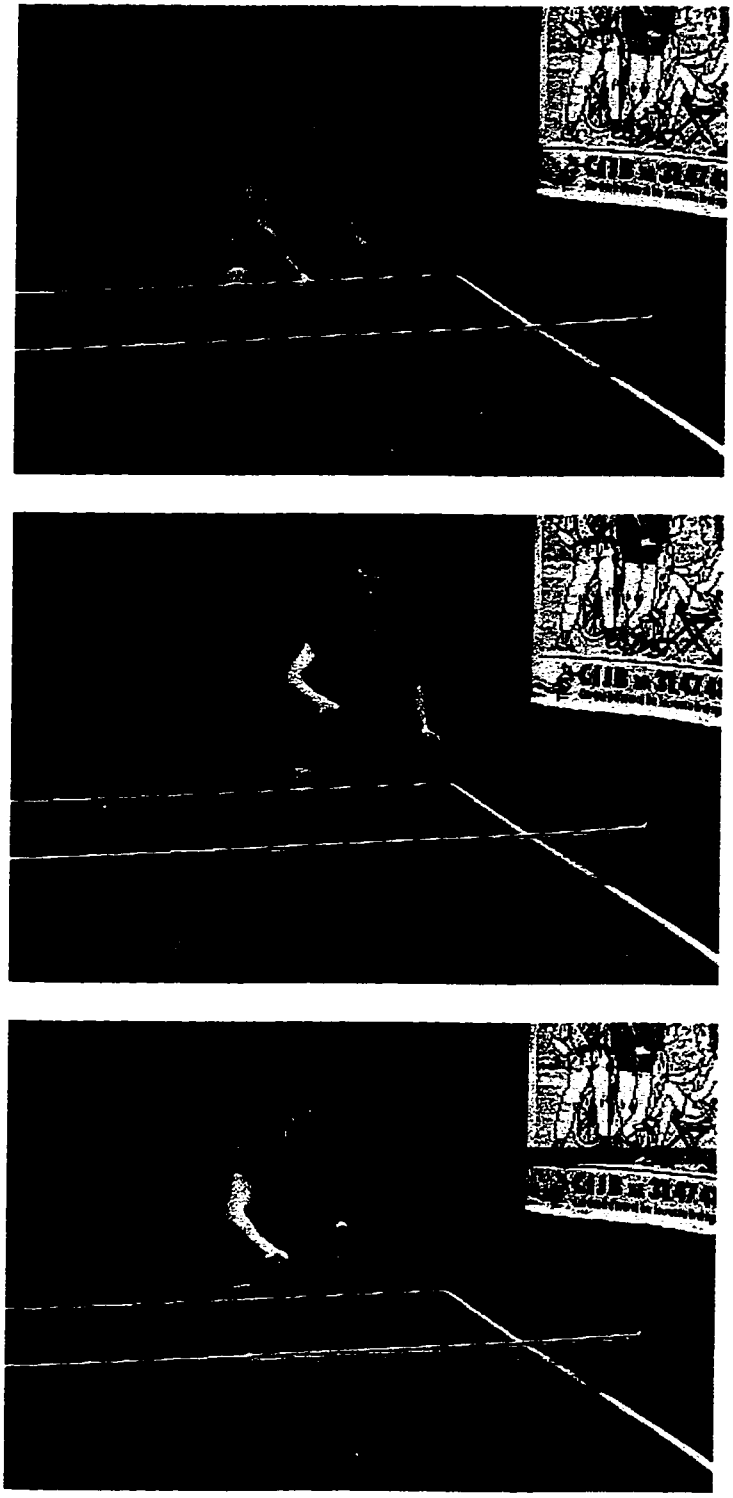


Figure 5.10: Frames 27, 6, 88 from "Table-Tennis" after Bit Toggling

performance of the RS scheme. For BER between $4 * 10^{-6}$ and $2 * 10^{-5}$ the performance of SBEC was above the 30 db limit. Figure 5.8 shows the Original frames $27, 6, 88$. Figure 5.9 shows the same frames after errors were inserted. Table 5.7 summarizes the 3 examples.

BER	Frame number
$4 * 10^{-6}$	27
$6 * 10^{-6}$	6
$8 * 10^{-6}$	88

Table 5.7: Channel BER for frames $27, 6, 88$

In frame 27 the errors caused two shifts to the right located at the center of the frame, and the degradation of a MB located at the bottom right edge of the poster. In frame 6 there are three black stripes. Also there are six MB with visual errors, together with two shifts to the right located at the upper body of the player. In frame 88 there are two stripes with visual degradations caused by errors in chrominance blocks.

Figure 5.10 shows frames $27, 6, 88$ after the sequence was processed by SBEC. In frame 27 the shifts to the right were successfully removed. Also the error in the MB was fixed. In frame 6 the black stripes were removed. Also the corrupted MBs and the shifts to the right were fixed. In frame 88 SBEC was not able to remove the two stripes possibly because the errors that caused them did not lead to syntax violations.

5.2.3 Third Experiment

Here, again, 180 frames (6*sec*) from the Sequence "Flower-Garden" were used, this time encoded at 12 Mbits/s. For this case the compression ratio achieved is equivalent to 10 : 1.

BER	RS	BT	NP
	Y Cb Cr	Y Cb Cr	Y Cb Cr
$1 * 10^{-7}$	38.74 42.86 41.18	38.74 42.86 41.18	37.94 42.74 40.93
$2 * 10^{-7}$	38.74 42.86 41.18	38.73 42.85 41.18	37.77 42.70 40.77
$4 * 10^{-7}$	38.74 42.86 41.18	38.62 42.84 41.16	36.38 42.21 40.56
$6 * 10^{-7}$	38.74 42.86 41.18	38.56 42.84 41.16	36.03 42.13 39.79
$8 * 10^{-7}$	38.74 42.86 41.18	38.39 42.76 41.05	35.81 40.77 39.13
$1 * 10^{-6}$	38.74 42.86 41.18	37.45 42.72 41.03	32.63 40.20 38.43
$2 * 10^{-6}$	38.74 42.86 41.18	37.23 42.71 40.97	30.71 39.89 37.90
$4 * 10^{-6}$	38.74 42.86 41.18	34.10 41.37 39.46	25.88 37.31 34.07
$6 * 10^{-6}$	38.74 42.86 41.18	33.27 41.34 38.99	24.53 36.43 32.99
$8 * 10^{-6}$	38.74 42.86 41.18	32.90 41.27 38.56	23.60 33.22 31.54
$1 * 10^{-5}$	38.74 42.86 41.18	31.86 41.07 37.72	21.80 32.88 28.96
$2 * 10^{-5}$	38.74 42.86 41.18	28.71 39.38 36.45	19.80 31.59 27.50
$4 * 10^{-5}$	38.74 42.86 41.18	24.12 37.71 33.35	16.12 27.47 23.95
$6 * 10^{-5}$	38.74 42.86 41.18	22.43 34.08 29.14	15.09 24.94 21.57
$8 * 10^{-5}$	38.74 42.86 41.18	19.67 31.89 28.58	13.92 22.04 20.34
$1 * 10^{-4}$	38.74 42.86 41.18	18.00 29.28 26.64	12.23 18.35 17.74
$6 * 10^{-4}$	38.55 42.84 41.15	06.00 08.31 09.16	05.83 08.19 08.89
$8 * 10^{-4}$	37.84 42.66 41.05	05.40 07.82 08.49	05.24 07.70 08.24
$1 * 10^{-3}$	37.71 42.61 40.89	05.13 07.56 08.24	04.99 07.45 08.00
$2 * 10^{-3}$	17.29 26.99 23.97	04.40 06.81 07.27	04.28 06.70 07.05
$3 * 10^{-3}$	08.32 11.00 11.78	04.29 06.68 07.11	04.17 06.58 06.91

Table 5.8: PSNR values for "Flower-Garden", compression ratio 10 : 1

Figure 5.11 gives the average PSNR for the luminance (Y), blue chrominance (Cb), and red chrominance (Cr) parts of the frames. For BER between 10^{-7} and $8 * 10^{-7}$ the performance of SBEC was less than one db (1*db*) difference from the performance of the RS scheme. For BER between $1 * 10^{-6}$ and $1 * 10^{-5}$ the performance of SBEC was above the 30 db limit. Figure 5.12 shows the Original frames

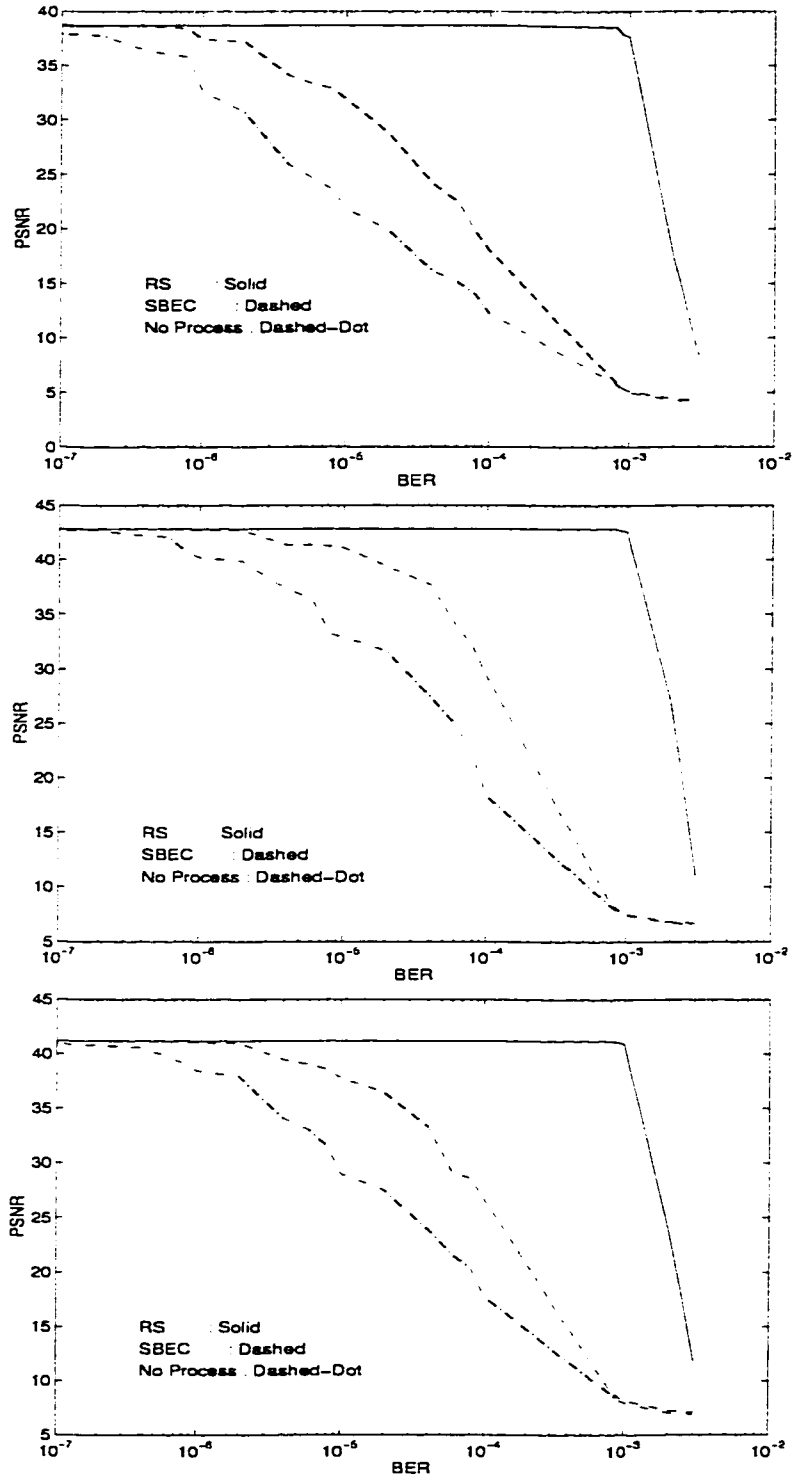


Figure 5.11: PSNR: Luminance, Blue Chrominance, and Red Chrominance for "Flower-Garden" 12 Mb/s

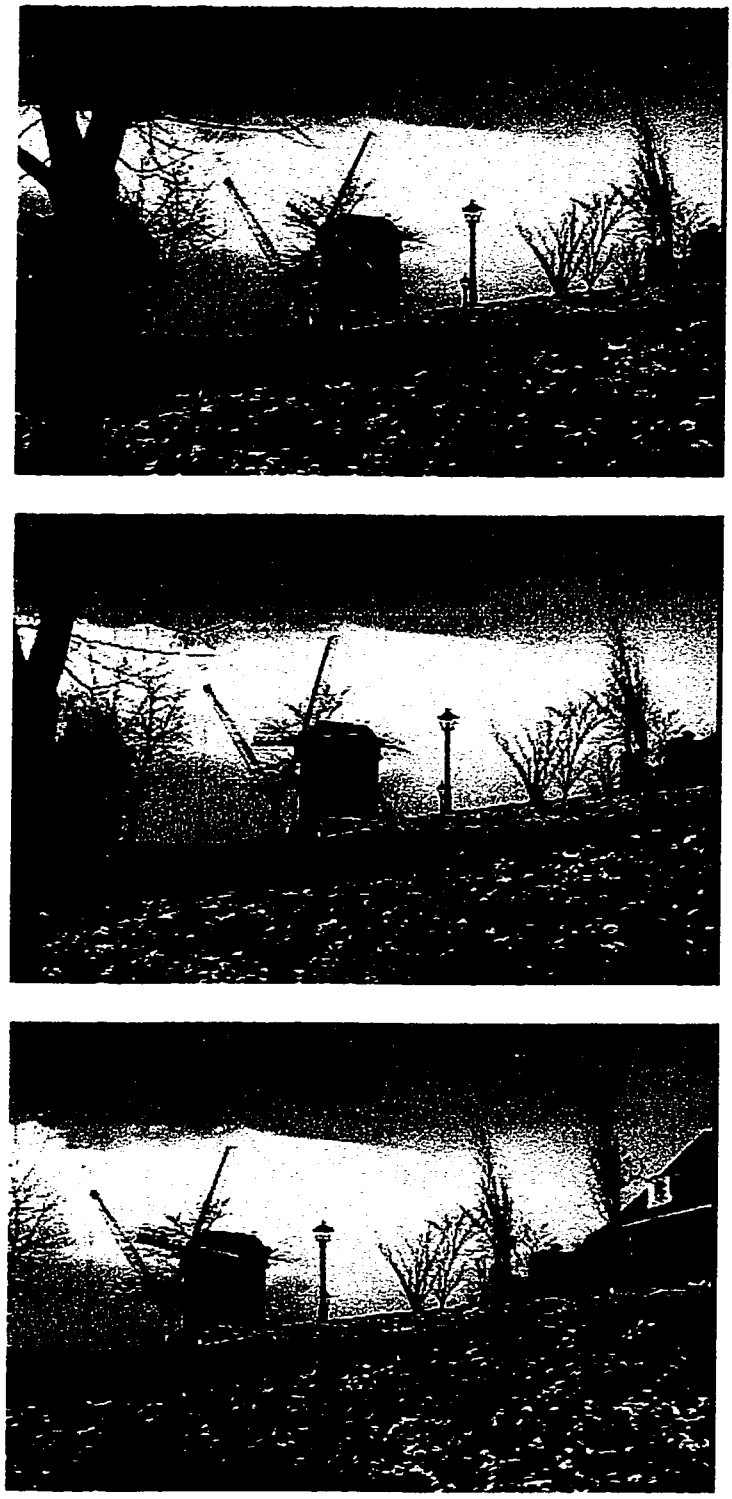


Figure 5.12: Original Frames $3, 24, 111$ from "Flower-Garden", compression ratio $10:1$

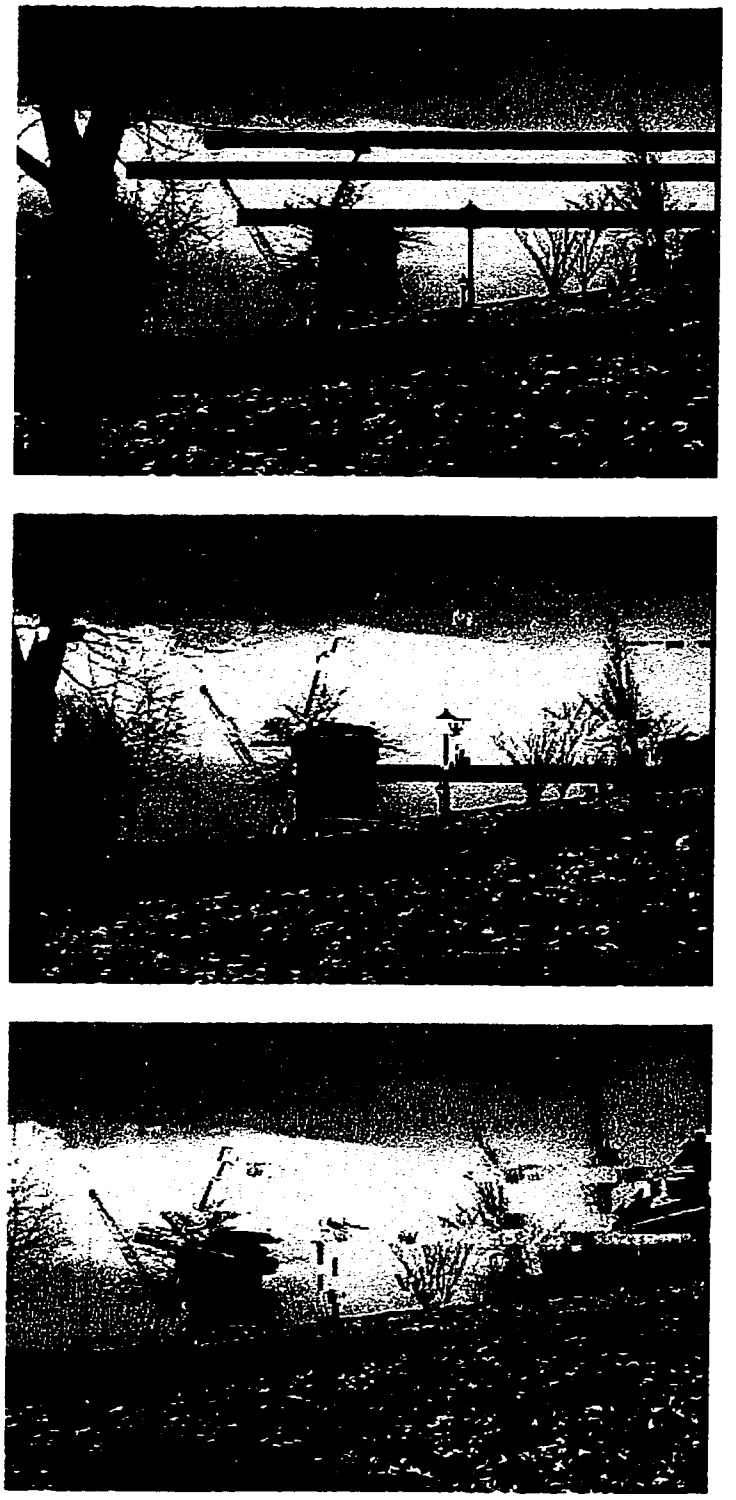


Figure 5.13: Corrupted Frames 3, 24, 111 from "Flower-Garden"

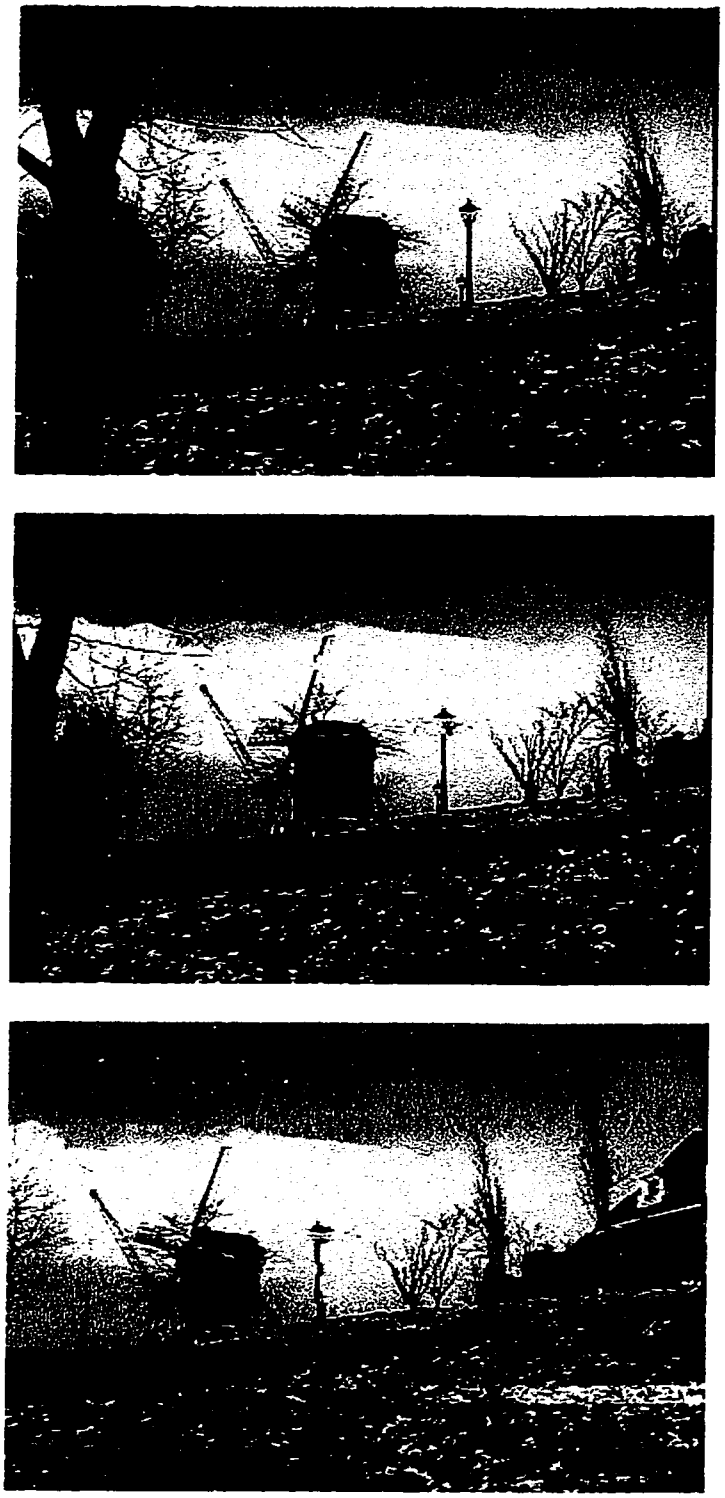


Figure 5.14: Frames 3, 24, 111 from "Flower-Garden" after Bit Toggling

3, 24, 111. Figure 5.13 shows the same frames after errors were inserted. Table 5.9 summarizes the 3 examples.

BER	Frame number
$1 * 10^{-5}$	3
$2 * 10^{-5}$	24
$6 * 10^{-5}$	111

Table 5.9: Channel BER for frames 3, 24, 111

In frame 3 there are six black stripes. In frame 24 there are two black stripes, a number of corrupted MBs and four shifts to the right. One is located at the tree (upper left edge of the frame), while the rest of them are located at the area surrounding the wind mill. In frame 111 there are several shifts to the right located at the center of the frame.

Figure 5.14 shows frames 3, 24, 111 after the sequence was processed by SBEC. In frame 3 the black stripes were successfully removed. In frame 24 the black stripes were removed and the corrupted MBs were fixed. As for the shifts to the right although there were not completely removed, they do not cause serious annoying effects to the viewer. In frame 111 also the shifts to the right were not completely removed, but the quality of the frame was improved.

5.2.4 Fourth Experiment

In the last experiment the same number of frames from the Sequence "Flower-Garden", encoded at 4 Mbits/s, were used. The equivalent compression ratio is 30 : 1.

BER	RS	BT	NP
	Y Cb Cr	Y Cb Cr	Y Cb Cr
$1 * 10^{-7}$	30.11 33.82 36.68	30.11 33.82 36.68	30.10 33.82 36.68
$2 * 10^{-7}$	30.11 33.82 36.68	30.11 33.82 36.68	30.06 33.82 36.68
$4 * 10^{-7}$	30.11 33.82 36.68	30.10 33.82 36.68	29.96 33.81 36.67
$6 * 10^{-7}$	30.11 33.82 36.68	30.07 33.82 36.68	29.84 33.73 36.60
$8 * 10^{-7}$	30.11 33.82 36.68	29.99 33.82 36.67	29.79 33.68 36.58
$1 * 10^{-6}$	30.11 33.82 36.68	29.97 33.80 36.67	29.28 33.53 36.52
$2 * 10^{-6}$	30.11 33.82 36.68	29.84 33.73 36.62	28.82 33.23 36.42
$4 * 10^{-6}$	30.11 33.82 36.68	29.76 33.66 36.60	27.80 32.86 35.77
$6 * 10^{-6}$	30.11 33.82 36.68	28.91 33.64 36.50	27.00 32.61 35.69
$8 * 10^{-6}$	30.11 33.82 36.68	28.76 33.63 36.43	25.93 31.57 34.76
$1 * 10^{-5}$	30.11 33.82 36.68	28.91 33.48 36.28	25.32 30.96 33.46
$2 * 10^{-5}$	30.11 33.82 36.68	26.04 32.16 36.05	22.44 29.49 33.45
$4 * 10^{-5}$	30.11 33.82 36.68	24.79 31.88 34.38	20.18 27.83 31.35
$6 * 10^{-5}$	30.11 33.82 36.68	23.38 29.94 33.32	19.20 26.41 30.07
$8 * 10^{-5}$	30.11 33.82 36.68	21.62 29.31 33.31	18.49 25.97 29.41
$1 * 10^{-4}$	30.11 33.82 36.68	21.45 29.27 32.37	17.08 24.37 27.85
$8 * 10^{-4}$	30.11 33.82 36.68	10.19 14.57 13.32	09.24 13.40 12.54
$9 * 10^{-4}$	29.90 33.70 36.64	08.99 13.02 12.09	08.06 11.87 11.31
$1 * 10^{-3}$	29.48 33.45 36.57	08.69 12.06 11.53	07.74 10.89 10.77
$2 * 10^{-3}$	20.43 27.18 30.97	06.76 10.12 09.11	05.83 08.98 08.36
$3 * 10^{-3}$	13.34 18.94 20.62	05.92 08.94 08.32	04.97 07.78 07.55

Table 5.10: PSNR values for "Flower-Garden", compression ratio 30 : 1

Figure 5.15 gives the average PSNR for the luminance (Y), blue chrominance (Cb), and red chrominance (Cr) parts of the frames. For BER between 10^{-7} and $4 * 10^{-6}$ the performance of SBEC was less than one db ($1db$) difference from the performance of the RS scheme. For BER between $6 * 10^{-6}$ and $1 * 10^{-5}$ the performance of SBEC although below the 30 db limit (due to the low bit rate), is very

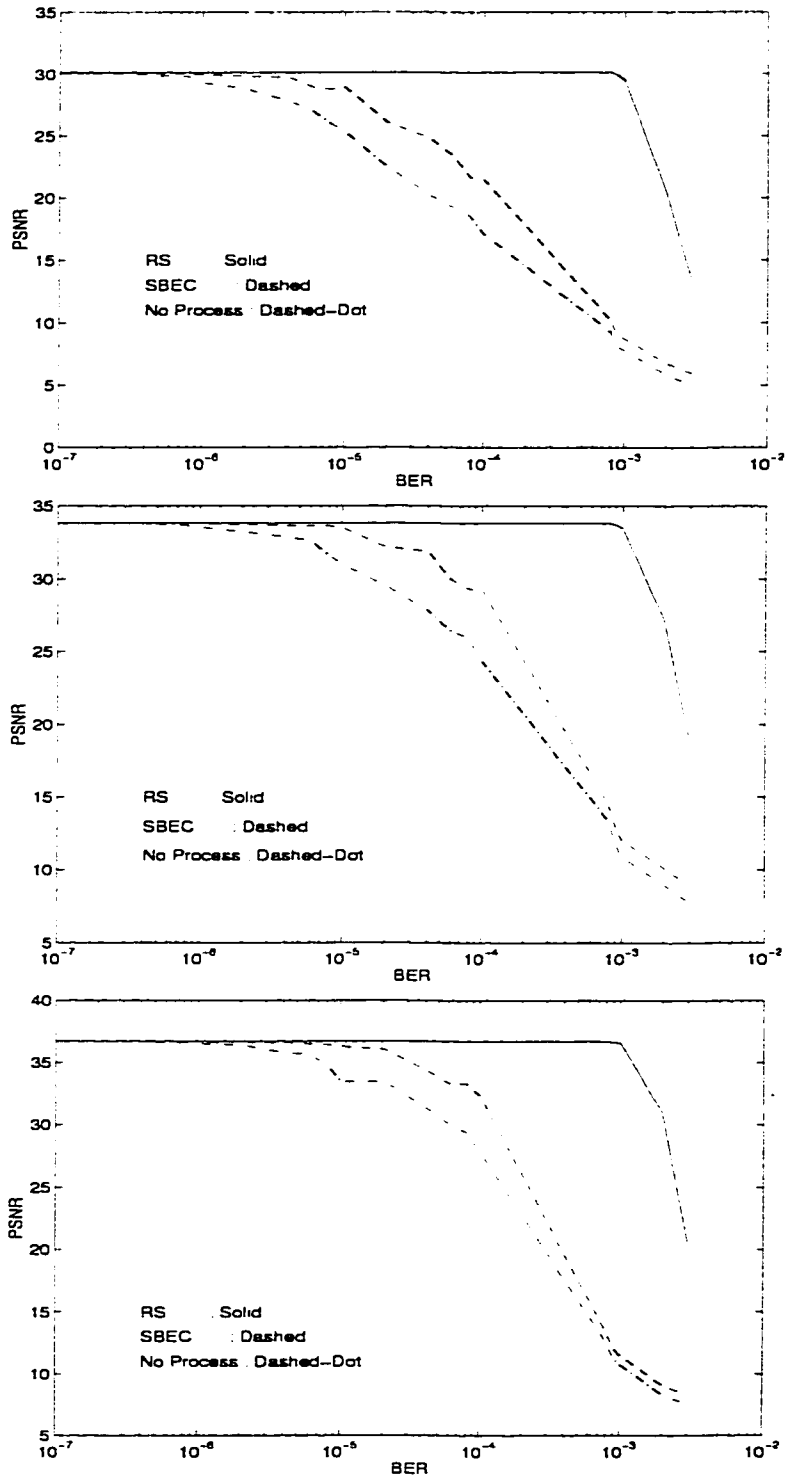


Figure 5.15: PSNR: Luminance, Blue Chrominance, and Red Chrominance for "Flower-Garden" 4 Mb/s

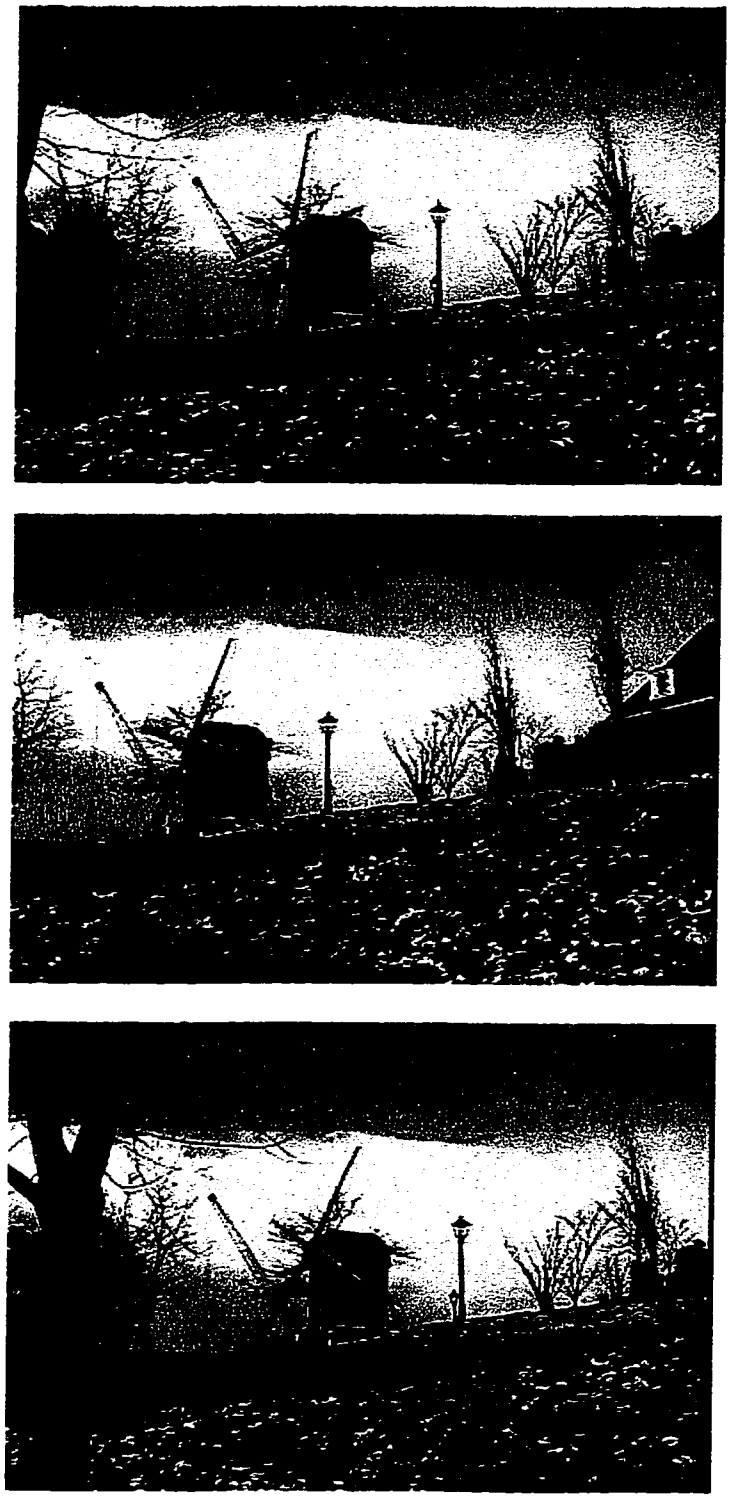


Figure 5.16: Original Frames 38, 110, 4 from "Flower-Garden", compression ratio 30 : 1

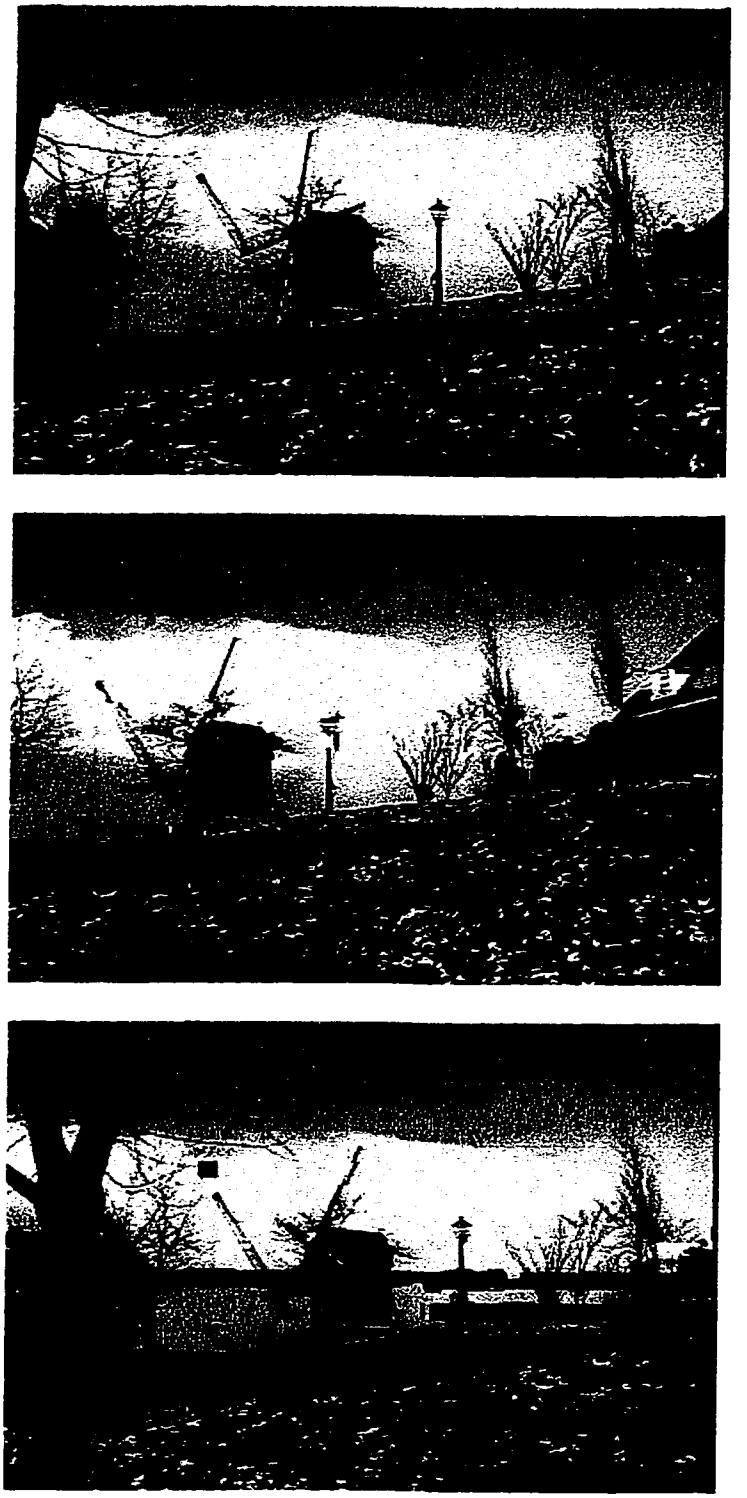


Figure 5.17: Corrupted Frames 38, 110, 4 from "Flower-Garden"

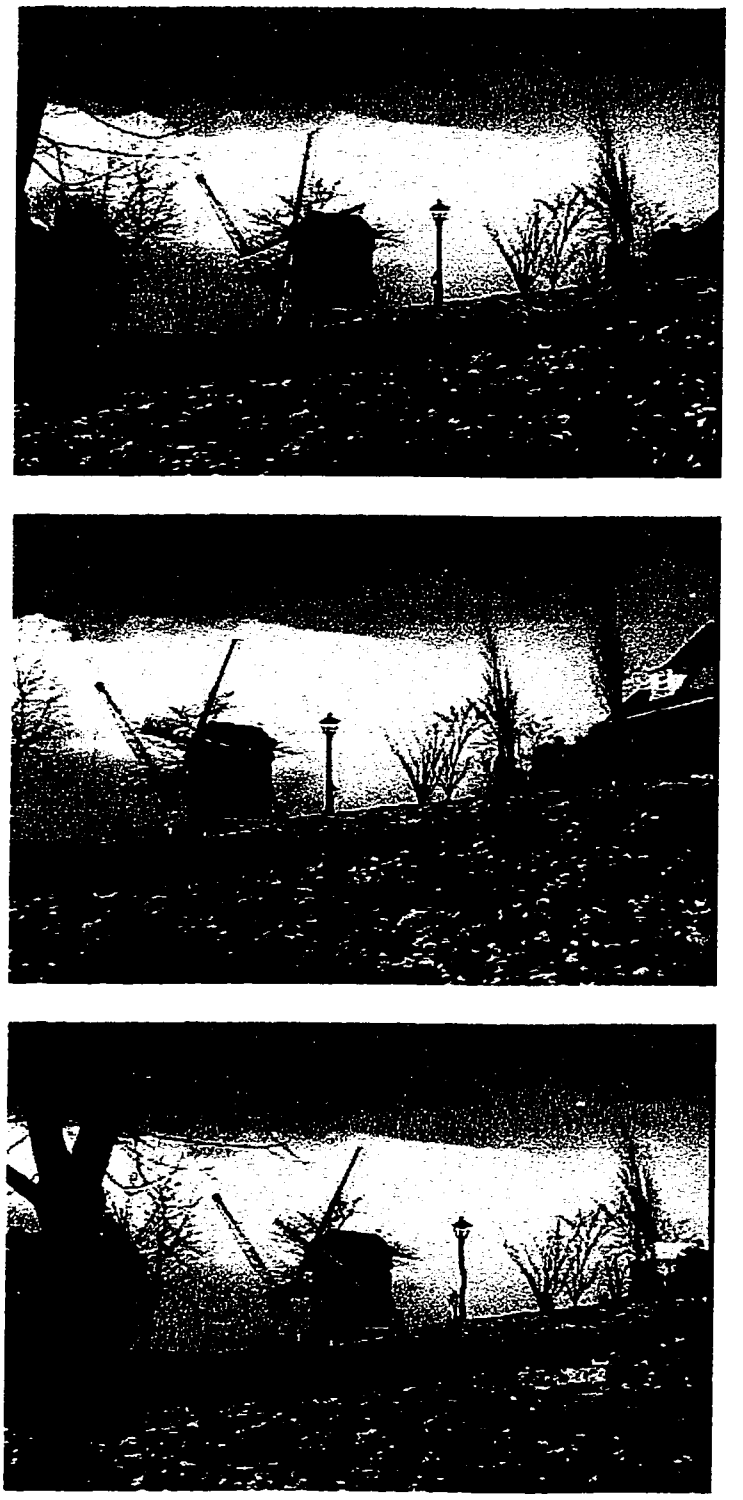


Figure 5.18: Frames 38, 110, 4 from "Flower-Garden" after Bit Toggling

close to the performance of the RS scheme. Figure 5.16 shows the Original frames 38, 110, 4. Figure 5.17 shows the same frames after errors were inserted. Table 5.11 summarizes the 3 examples.

BER	Frame number
$2 * 10^{-6}$	38
$2 * 10^{-5}$	110
$8 * 10^{-5}$	4

Table 5.11: Channel BER for frames 38, 110, 4

In frame 38 there is a shift to the right located at the upper left of the frame. In frame 110 there are two shifts to the right located at the area surrounding the wind mill. In frame 4 there are four black stripes and a number of corrupted MBs.

Figure 5.18 shows frames 38, 110, 4 after the sequence was processed by SBEC. In frame 38 the shift to the right was succesfully removed. In frame 110 the two shifts to the right were removed. In frame 4 most of the black stripes and of the corrupted MBs were fixed.

5.3 Complexity

In Section 3.1.1 the complexity of a scheme using the shortened RS(204,188,$t = 8$) code was presented. For SBEC complexity depends on two factors:

- The complexity of the ED decoder.
- The complexity of the Bit Toggler

The ED coder here is a simple parity check, which can be implemented with one table look-up per ED block.

The complexity of the Bit Toggler is harder to quantify. It can be analyzed into three sub-factors:

- The number of errors per slice.
- The number of undetected errors.
- The bytes the decompressor has to extract from the bitstream until it detects an error (during an unsuccessful decompression of a slice).

The first sub-factor is proportional to the BER. Low BER usually results in only one error in any particular corrupted slice. As the BER increases, the probability of a second (or more) error per corrupted slice increases. The second sub-factor also depends on the BER. Two errors in one ED block will not only result in an undetected error, but may result in a large increase in complexity if there are other errors in the same slice. Consider that the bit toggler will keep on toggling until it finds a bitstream that violates its syntax, or it runs out of bits to toggle. An undetected error may result in a syntax error that bit toggling elsewhere in the slice will not remove. Several errors in the same slice may result in many cases to check. Table 5.12 shows the number of errors per corrupted slice, and the number of undetected errors for a range of BER. These data were captured during the first experiment.

BER	Max Number of Errors (per corrupted slice)	Undetected Errors
$1 * 10^{-7}$	1	None
$2 * 10^{-7}$	1	None
$4 * 10^{-7}$	1	None
$6 * 10^{-7}$	1	None
$8 * 10^{-7}$	1	None
$1 * 10^{-6}$	1	None
$2 * 10^{-6}$	2	None
$4 * 10^{-6}$	2	None
$6 * 10^{-6}$	2	None
$8 * 10^{-6}$	2	None
$1 * 10^{-5}$	2	None
$2 * 10^{-5}$	3	None
$4 * 10^{-5}$	4	None
$6 * 10^{-5}$	6	2
$8 * 10^{-5}$	7	4
$1 * 10^{-4}$	8	6
$2 * 10^{-4}$	12	36
$4 * 10^{-4}$	21	116
$6 * 10^{-4}$	28	218
$8 * 10^{-4}$	35	394

Table 5.12: Number of errors per corrupted slice and number of undetected errors for a range of BER

We attempt to estimate the complexity of the third sub-factor by examining how many bits must be extracted from the bitstream during the multiple decompressions. The complexity is taken to be proportional to the number of bits extracted, since operations like DCT and motion compensation need not be done in the simple syntax checking done here. This number depends on the type of the error in the bitstream. Errors in important parts of the bitstream are caught earlier than those in not so important parts. Figure 5.19 shows a plot of that distance from data gathered during the simulations. Although there are cases which took significantly longer time to be detected, the majority of them were solved within a few hundred bits distance.

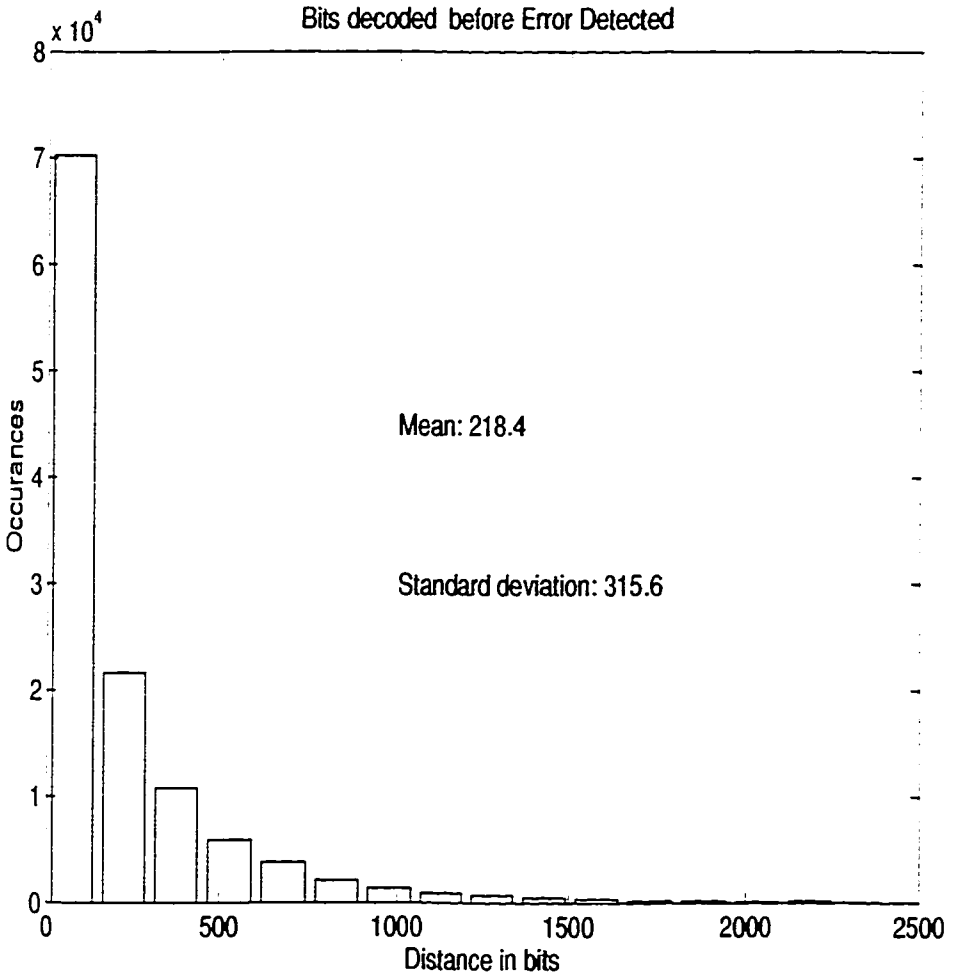


Figure 5.19: Analysis of the Distance covered by the decompressor until it detects an error

If SBEC was to run for an infinite time, then all recoverable errors would be resolved no matter how long it would take. What is interesting to see is the probability for SBEC to resolve a case, given a limited number of operations ($Operation = n * bit$) where n can be any integer. Clearly this is the worst case scenario, because usually the decompressor extracts bits in blocks of variable size. To put a bound on the number of operations in our scheme, we assume that for each bit extracted SBEC uses 10 operations ($n = 10$). We then give SBEC a fraction of the operations that a classical RS decoder would use, and calculate the probability of event U: that SBEC does not finish within the given time window. The probability is given in Figure 5.20. These estimates indicate that SBEC has lower complexity than the RS scheme for lower BER.

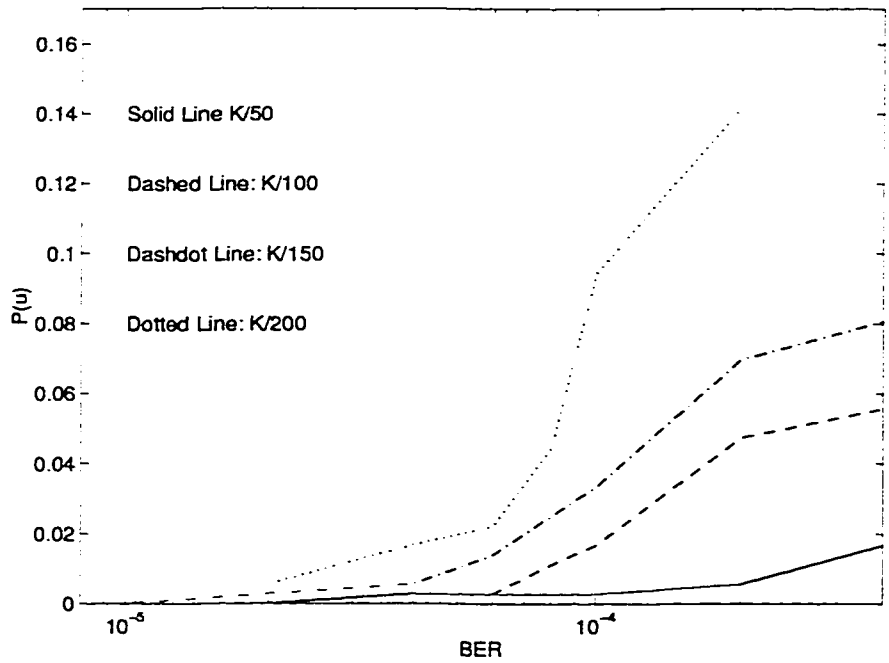


Figure 5.20: Probability of not finishing within a given number of operations

5.4 Discussion

Figures on pages 69, 75, 81 and 87 give the plot of PSNR values for the luminance (corresponds to the Y blocks) and the two chrominance (corresponds to the Cb and the Cr blocks) type frames for the four experiments. These figures indicate that the performance of the scheme does not depend on the content of the Sequence, or the compression ratio. One thing that needs to be mentioned is that Sequences compressed at high bit rates (higher bit rate means lower compression ratio) seem to degrade faster at higher BER than those compressed at low bit rates (lower bit rate means higher compression ratio).

Bit Chosen	Number of Times	Percentage
Parity Bit	29,903	62.8%
Bit 1	3,862	8.1%
Bit 2	2,687	5.6%
Bit 3	2,187	4.6%
Bit 4	1,756	3.7%
Bit 5	1,476	3.1%
Bit 6	1,232	2.6%
Bit 7	1,073	2.3%
Bit 8	884	1.9%
Bit 9	777	1.6%
Bit 10	703	1.5%
Bit 11	607	1.3%
Bit 12	471	0.9%

Table 5.13: Bit Selection Percentage

Figures on pages (70, 71, 72),(76, 77, 78),(82, 83, 84),(88, 89, and 90) show frames from the two Sequences which were used for the simulations. The first page always shows original frames, the second page shows the same frames after channel degradation, and the last page shows the same frames after the Bit Toggling process. In the majority of the cases examined the most egregious errors (black stripes, shifting) were removed from the Sequences. There were cases, however, where the scheme did not manage to remove the errors. This was because either the error did

not lead to syntax violation or because the scheme decided to toggle a wrong bit (see Table 5.13), which did not cause syntax violation. One other point to notice is that although sometimes the scheme does introduces new errors, which happens when toggling different bits other than the actual error, this situation does not create serious visual errors.

From the simulation results it is clear that the RS scheme performs better than SBEC for a wider range of BER. However, the performance of SBEC can be rated as follows: for BER from 10^{-7} to 10^{-6} errors were not noticeable, for BER from 10^{-6} to $6*10^{-6}$ errors were noticeable but not very annoying, for BER from $6*10^{-6}$ to 10^{-5} errors were annoying, while for BER from 10^{-5} to 10^{-3} errors were very annoying. In other words for a BER range from 10^{-7} to 10^{-5} the performance of SBEC, both subjectively and objectively, is good. Also, as discussed in Section 5.3, results indicate that SBEC has simpler complexity than the RS (204,188,t = 8) based scheme.

Chapter 6

Conclusions

6.1 Discussion

During transmission of compressed video over communication networks errors may occur. As discussed in Chapter 1, because the compressed video is fragile, any error may lead to the severe degradation of the video sequence.

One method to compensate for transmission errors is to apply a FEC code to the data prior to transmission. Standardization Organizations like DAVIC [15] and ETSI [16] proposed the use of a shortened RS$(204, 188, t = 8)$ code. As shown in 3.1.1 the code ensures good performance for error environments up to 10^{-3}. The cost associated with such strong code is its complexity, the BE and the delay that is imposed on the receiver.

An alternative way for transmission of compressed MPEG2 video over communication networks was examined in this thesis. To compensate for the possible transmission errors, a strategy of combining Error Detection (ED) together with Error Concealment (EC) is applied. Based on that idea, two schemes, namely CEDEC and SBEC, were introduced and evaluated against a scheme which uses the RS$(204, 188, t = 8)$ code. Based on the simulations results, we may conclude the following for the performance of both CEDEC and SBEC.

It is clear from the presented results that the RS scheme performs better than both CEDEC and SBEC. For a BER range between 10^{-7} and $2 * 10^{-5}$ the performance of SBEC is comparable. CEDEC on the other hand, although it seems to have the lowest complexity of the three schemes, has a moderate performance. As discussed in Section 5.3 SBEC seems to have lower complexity than the RS scheme. Another advantage of SBEC is that it does not impose any delay on the receiver.

6.1.1 Contributions

In Chapter 4 a method to handle transmission errors was presented. The method combines Error Detection with Error Concealment performed on the uncompressed

video domain. The main idea here is the translation of the location of the errors from the compressed to the uncompressed domain. That is accomplished by the Spatial Locator module. Error Detection encoding provides information for bit error occurance in the bitstream. That information passes then to the Spatial Locator, which translates it to a location in the uncompressed video. This is done by partially decompressing the sequence, and keeping track of the locations in both the compressed and the uncompressed domain. Depending on where in the compressed bitstream the error occurred, the spatial location can be as big as a whole frame or as small as a simple block.

In 3.3.1 the idea of Syntax Based redundancies was introduced. In Chapter 5 that type of redundant information was used as the basis of an Error Concealment process performed on the compressed domain. Again, here an Error Detection encoding process provides information for ED blocks with errors. For each such block, the Bit Toggling module will decompress the slice which carries that block a number of times. That number is upper bounded by the size of the ED block. For each decompression the method concentrates whether the syntax of the compressed bitstream is violated. The very first decompression which does not cause a syntax violation is accepted as the corrected video.

6.2 Future Work

Future work related to this thesis includes two parts. First, for CEDEC, a more efficient Error Concealment technique, which will make it possible to address the problem of cell losses during transmission, is necessary. Second, in the case of SBEC, a feed back loop from the uncompressed domain is necessary. The purpose of this loop is to carry extra information for the bit toggling process.

6.2.1 Error Concealment

The two methods which were introduced in Chapters 4 and 5 were addressing the problem of bit error occurrence during transmission. But as was discussed in Section 1.1 cell loss is another type of error that can occur during transmission in packet switched networks.

For that type of error Error Concealment methods seem to be more appropriate than the SBEC method. For example, suppose that a cell of n bytes was lost. For an Error Concealment scheme, the task would be to fill up some space. For a method similar to the one introduced in Chapter 5, there should be a testing of a number of different patterns for the lost cell, and acceptance of the one that does not violate the syntax of the compressed bitstream. For a worst case scenario there must be 2^n tests. Considering that the size of each cell for ATM networks is equivalent to 53 bytes, it is obvious that such a task is forbidden.

The Error Concealment must be performed during the decompression process. By doing so error propagation will be reduced because errors in anchor frames will be resolved before the decompression of the predicted frames. Also, motion information will be available, which will improve the performance of the Concealment process.

6.2.2 Bit Toggling

In Chapter 5, a new method for compensating transmission errors based on the syntax redundancy of the compressed bitstream was presented. The performance of the method as shown in illustrative examples is promising. The only problem, as stated in 5.1.3, is the fact that there are errors that, although they do not violate the syntax, cause degradation of the video sequence. This category includes the errors for which the SBEC scheme does not take any action. The same result may have errors for which the bit toggler decides to alter a wrong bit.

To handle such cases information from the uncompressed video domain is

needed. For each slice with error the neighboring slices must be decompressed and stored. Each time the bit toggler will come up with an accepted slice control will pass to a checking process. There the content of the erroneous slice will be compared against the content of the stored neighboring slices. Then a decision will be made whether the defected slice can be accepted or not. Depending on the decision the slice will be accepted as the corrected video or will undergo more bit toggling processes.

One other field that may need further research is the concept of using Reversible VLCs. At the draft of MPEG4, among other data recovery tools they also refer to the Reversible Variable Length Codes (RVLC). These codewords are designed such that they can be read both in the forward as well as the reverse direction. In the context of SBEC, the use of RVLC, instead of VLC, will help the toggling process even though it will reduce the coding efficiency. That is because RVLC will reduce the candidate number of the accepted toggled bits.

Bibliography

[1] S. M. E. Group, *Generic Coding of Moving Pictures and Associated Audio Information: Video.* ISO-IEC/JTC1/SC29/WG11 N0702, May 1994.

[2] T.Sikora, "MPEG Digital Video Coding Standards," *IEEE Signal Processing Magazine*, pp. 82–100, September 1997.

[3] F.Jeng and S.Lee, "Concealment of Bit Error and Cell Loss in Inter Frame Video Transmission," in *IEEE International Conference on Communications*, pp. 496–500, 1991.

[4] A.K.Jain, *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[5] A.N.Netravali and B.G.Haskell, *Digital Pictures Representation Compression and Standards.* Plenum Press, 1995.

[6] A.B.Watson, *Digital Images and Human Vision.* MIT Press, 1993.

[7] M. H. Page, "http://www.cselt.it/mpeg."

[8] D. Gall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, vol. 34, no. 4, pp. 47–58, April 1991.

[9] R.Schafer and T.Sikora, "Digital Video Coding Standards and their Role in Video Communications," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 907–924, June 1995.

[10] F.Dufaux and F.Moscheni, "Motion Estimation Techniques for Digital TV:A Review and a new Contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, June 1995.

[11] H.G.Musmann, P.Pirdch, and H.J.Grallert, "Advances in Picture Coding," *Proc. IEEE*, vol. PROC-73, pp. 523–548, April 1985.

[12] M.I.Sezen and R.L.Lagendijk, *Motion Analysis and Image Sequence Processing.* Boston MA, Klawer, 1993.

[13] S. M. E. Group, *Generic Coding of Moving Pictures and Associated Audio Information: System.* ISO-IEC/JTC1/SC29/WG11 N0702, May 1994.

[14] S. Lin and J. C. Jr., *Error Control Coding.* Prentice Hall, 1983.

[15] D. A. Council, *DAVIC 1.3 Specification Part 8: Lower Layer Protocols and Physical Interfaces*, 1997. Revision 6.32.2.

[16] E. JTC, *Digital Broadcasting Systems for Television, Sound and Data Services: Framing Structure, Channel Coding and Modulation for Cable Systems.* ETSI, December 1994.

[17] E. JTC, *Digital Broadcasting Systems for Television, Sound and Data Services: Framing Structure, Channel Coding and Modulation for 11/12 GHz Satellite Services.* ETSI, December 1994.

[18] A.Michelson and A.Levesque, *Error Control Techniques for Digital Communication.* John Wiley & Sons, 1985.

[19] R.S.Brunton, "A Comparative Analysis of Reed Solomon Decoding Techniques," Master's thesis, University of Waterloo, Canada, 1981.

[20] W.Luo and M.E.Zarki, "Analysis of Error Concealment Schemes for MPEG2 Video Transmission over ATM based Networks," *SPIE Video Coding and Image Processing*, vol. 2501, pp. 1358–1368, 1995.

[21] W.Kwok and H.Sun, "Multi Directional Interpolation for Spatial Error Concealment," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 455–460, August 1993.

[22] W.-M. Lam and A. R. Reibman, "Recovery of Lost or Erroneously Received Motion Vectors," in *ICASSP-93*, vol. V, pp. 417–420, 1993.

[23] V.Papadakis, T.Le-Ngoc, and W.E.Lynch, "Combined Error Detection and Error Concealment for MPEG2 Over ATM," in *CCECE-97*, vol. I, pp. 137–140, May 1997.

[24] W.Kwok and H.Sun, "Concealment of Damaged Block Transform Coded Images Using Projections Onto Convex Sets," *IEEE Transactions on Image Processing*, vol. 4, no. 4, pp. 470–477, April 1995.

[25] S.Aign and K.Fazel, "Temporal and Spatial Error Concealment Techniques for Hierarchical MPEG2 Video Coder," in *Proceedings of IEEE International Conference on Communications*, vol. III, pp. 1778–1783, June 1995. Seattle.

[26] P.Salama, N. Shroff, E.Coyle, and E.Delp, "Error Concealment Techniques for Encoded Video Streams," in *Proceedings of IEEE International Conference of Image Processing*, September 1995.

[27] V.Papadakis, W.E.Lynch, and T.Le-Ngoc, "Syntax Based Error Concealment," To be published in ISCAS98.

Appendix A

Software Modules Block Diagrams

In Chapters 4 and 5, two techniques for transmitting MPEG2 Video over Communication Networks were presented. Figure A.2 shows both CEDEC and SBEC schemes. The Transmitter Part (Tx) in both schemes includes the Video and the ED encoder modules. The Receiver Part (Rx) includes the Video and ED decoder modules, while CEDEC also has the MPEG2 Mapper and the EC modules and SBEC has the Bit Toggler module. Note that in both schemes the ED encoder, the Channel and the ED decoder modules were implemented in one Software packet (Traffic Simulator). Next a brief manual of the following Software components will be presented:

1. Traffic Simulator
2. MPEG2 Mapper
3. Error Concealment
4. Bit Toggler

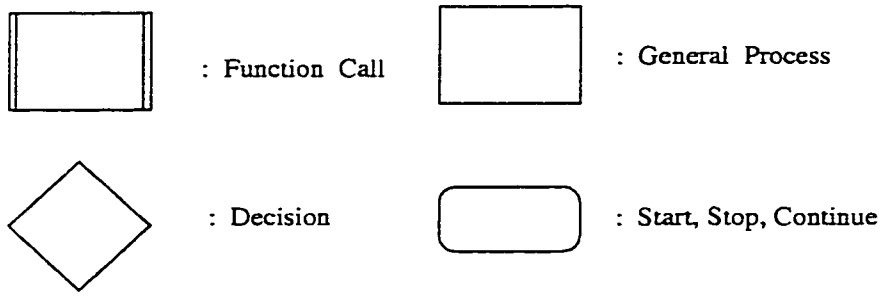
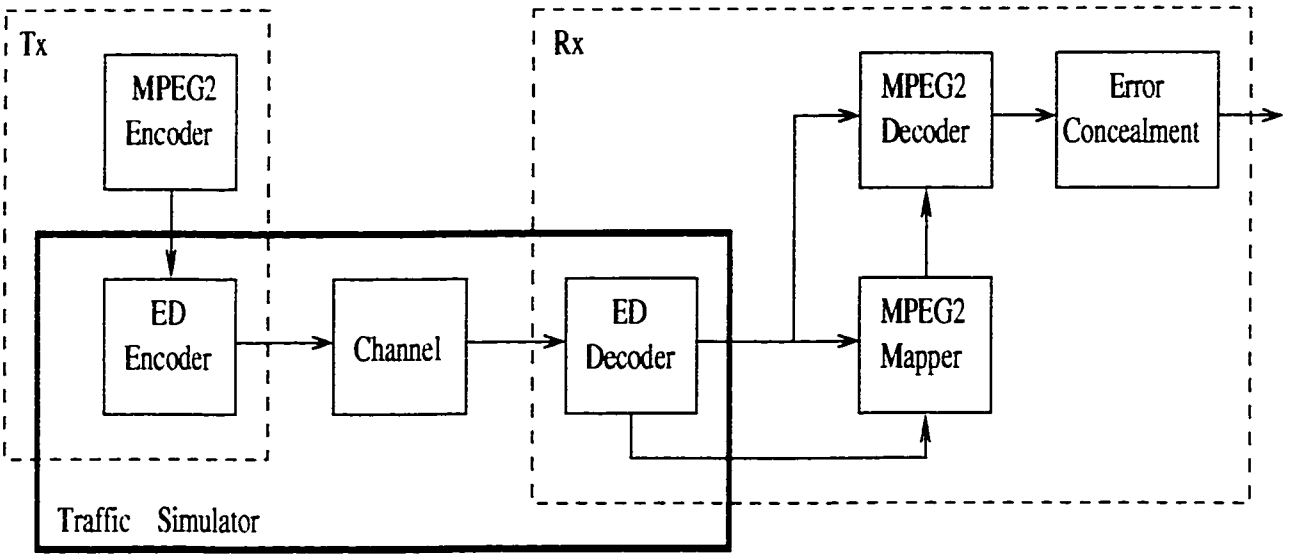
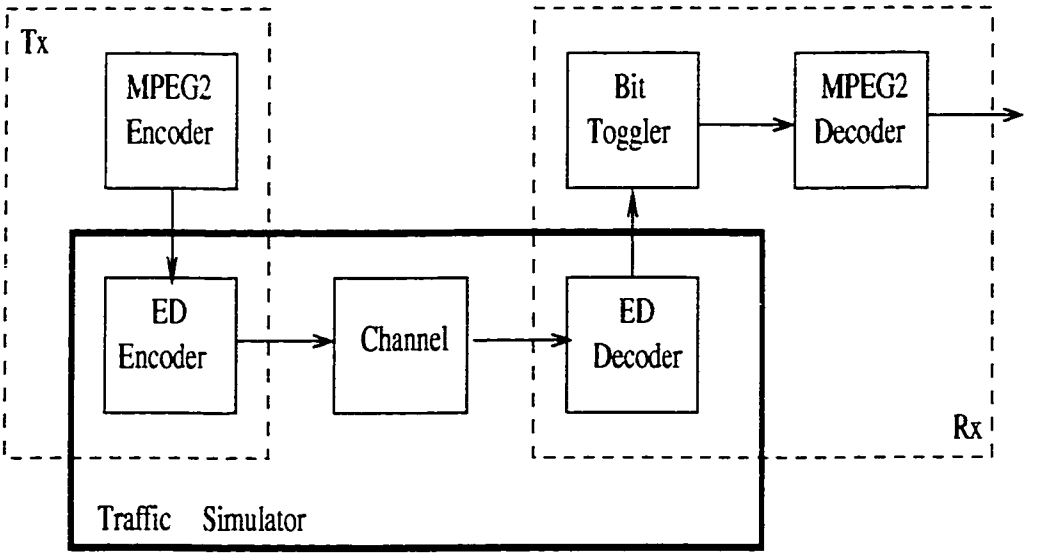


Figure A.1: Logic Diagram Components

Figure A.1 presents the notation which will be used to present the Logic Diagrams of the Software components.



a) Block Diagram of the CEDEC Technique



b) Block Diagram of the SBEC Technique

Figure A.2: Block Diagram of CEDEC and SBEC Techniques

A.1 Traffic Simulator

Usage: Traffic -b InputBitstream -c OutBitstream -e Detection OutFile -f Statistics Outfile -i BER case -p Factor

InputBitstream: The MPEG2 bitstream at the input of Traffic Simulator.

OutBitstream: The MPEG2 bitstream after the addition of noise.

Detection OutFile: File containing the ED blocks in error (To be used by MPEG2 Mapper or by Bit Toggler).

Statistics Outfile: File containing information regarding the bits in error.

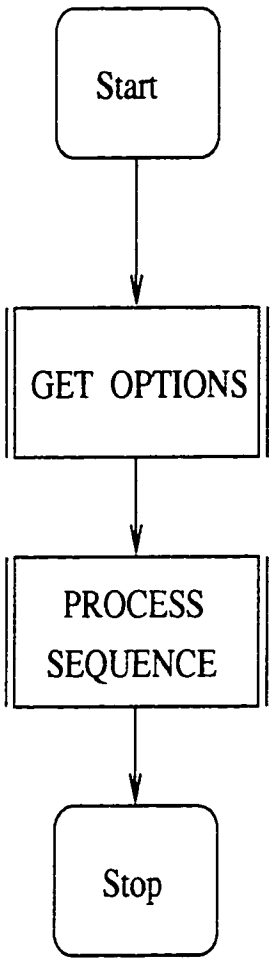
BER case: Selects a desire case within the limits $10^{-1}...10^{-12}$.

Factor: Selects a desire case within the limits 1...9

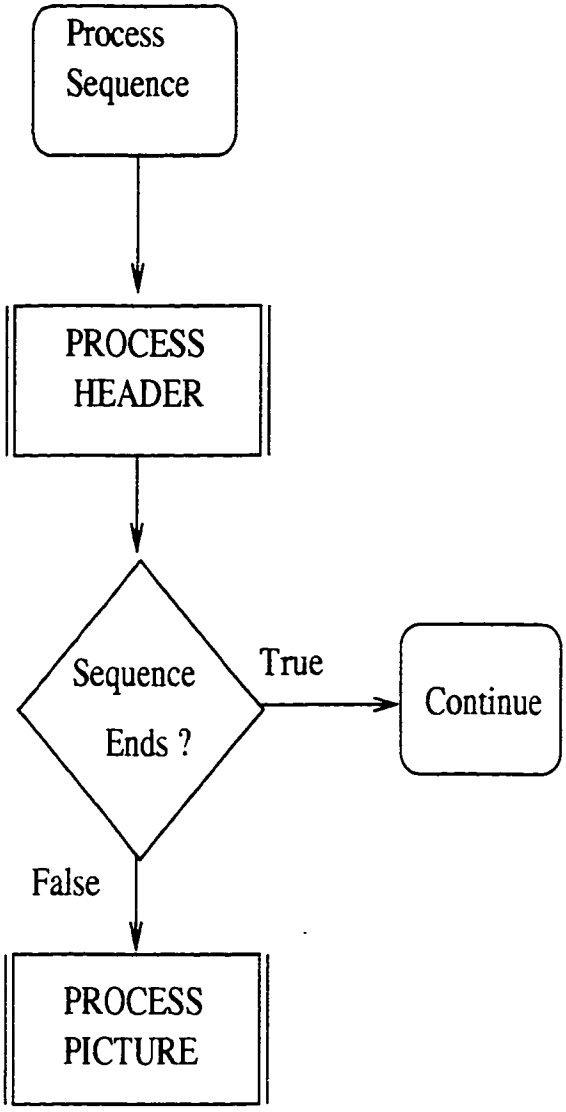
Example: Traffic -b tennis.m2v -c tennis_corrupt.m2v -e detect.out -f stat.out -i6 -p5

Traffic Simulator is based on a MPEG2 video decoder. Because the purpose is to add noise to a bitstream, basic functions of the decoder like IDCT, Inverse Quantization, Frame Storage are removed. Each bit extracted from the InputBitstream goes either to the function putbits() or to the function puterror(). The decision depends on the location of the bit inside the Bitstream. For CEDEC DCT coefficient of B frames, while for SBEC every bit inside the slice layer will go into the puterror() function. In all other cases the bit goes into the putbits() function. Both puterror() and putbits() functions will finally store the bit to a buffer. The size of the buffer depends on the size of the data block (16 bits for CEDEC 12 bits for SBEC).

When the buffer is full its contents goes to the OutBitstream. Depending on the number of bit errors (counter total_errors) in the present buffer, a number (counter blocks_counter) is written into the Detection OutFile. That number corresponds to the number of the ED blocks needed to carry the bits already extracted from the InputBitstream. After each write operation all the counters and flags are appropriately set. Figures A.3, A.4, A.5 and A.6 give the Logic Diagram of the Traffic Simulator.

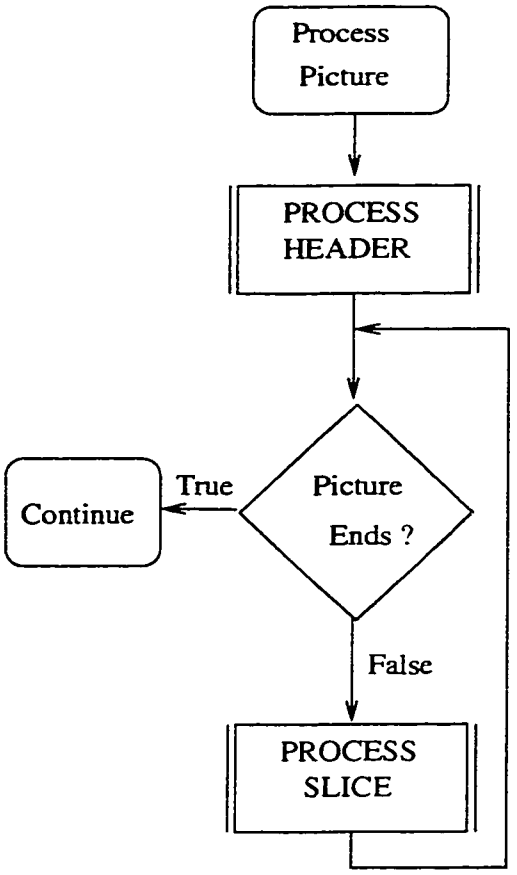


a) Traffic Simulator : Main Process

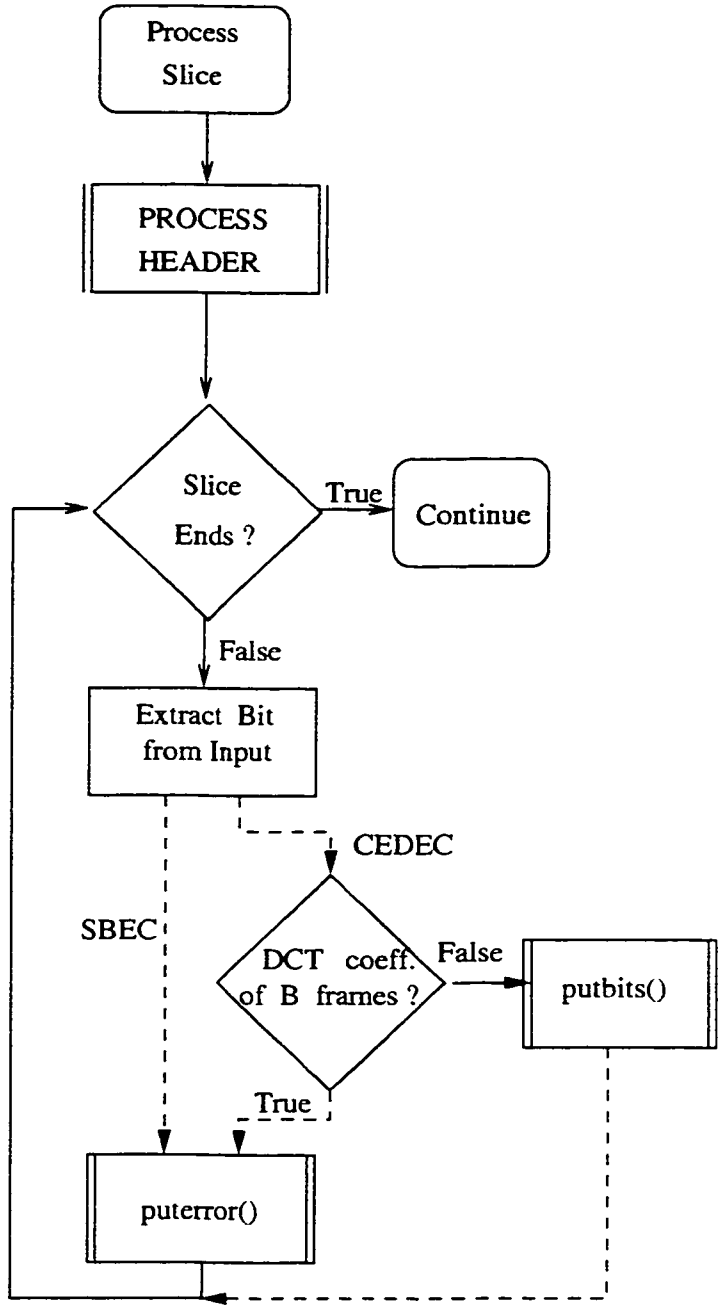


b) Traffic Simulator : Sequence Layer

Figure A.3: Traffic Simulator

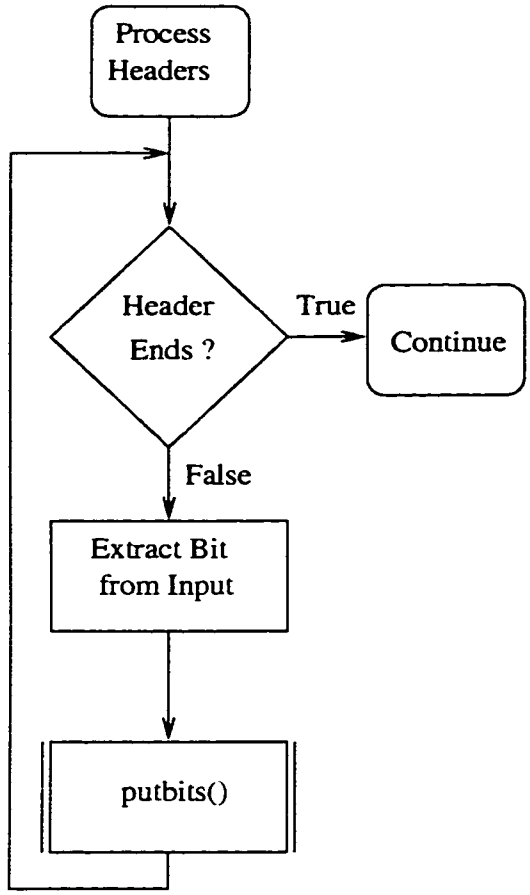


c) Traffic Simulator : Picture Layer

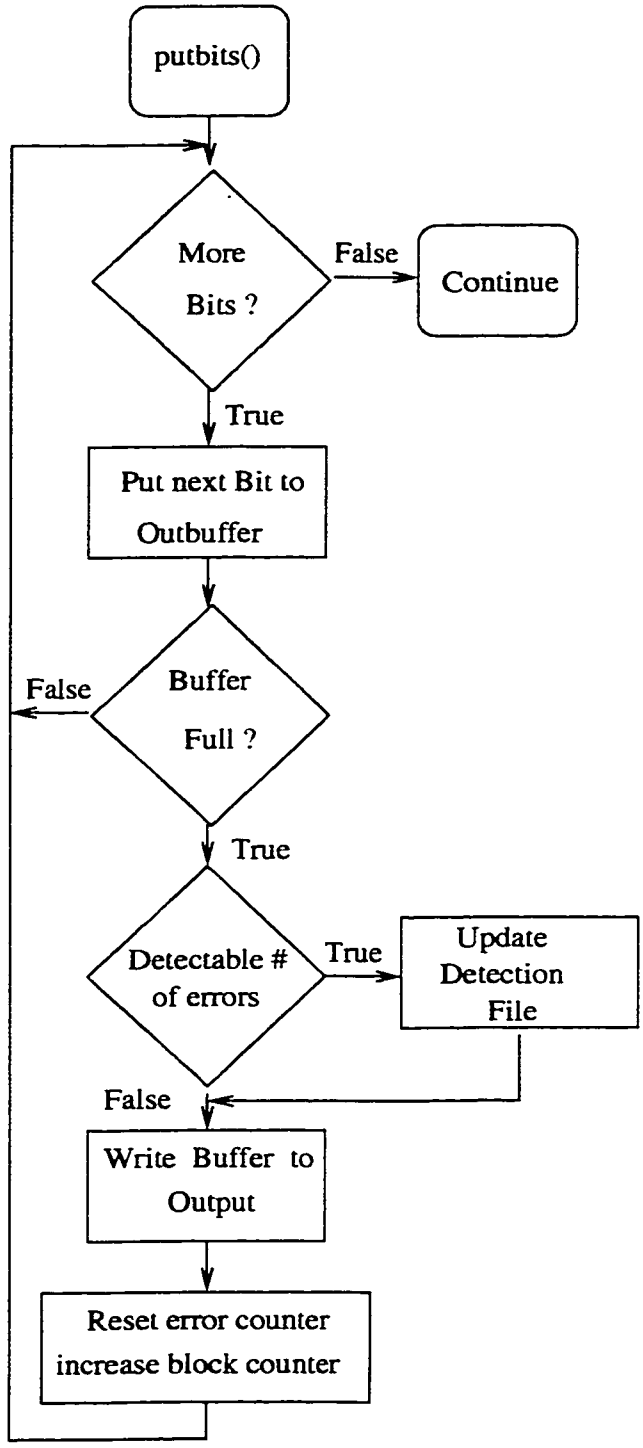


d) Traffic Simulator : Slice Layer

Figure A.4: Traffic Simulator

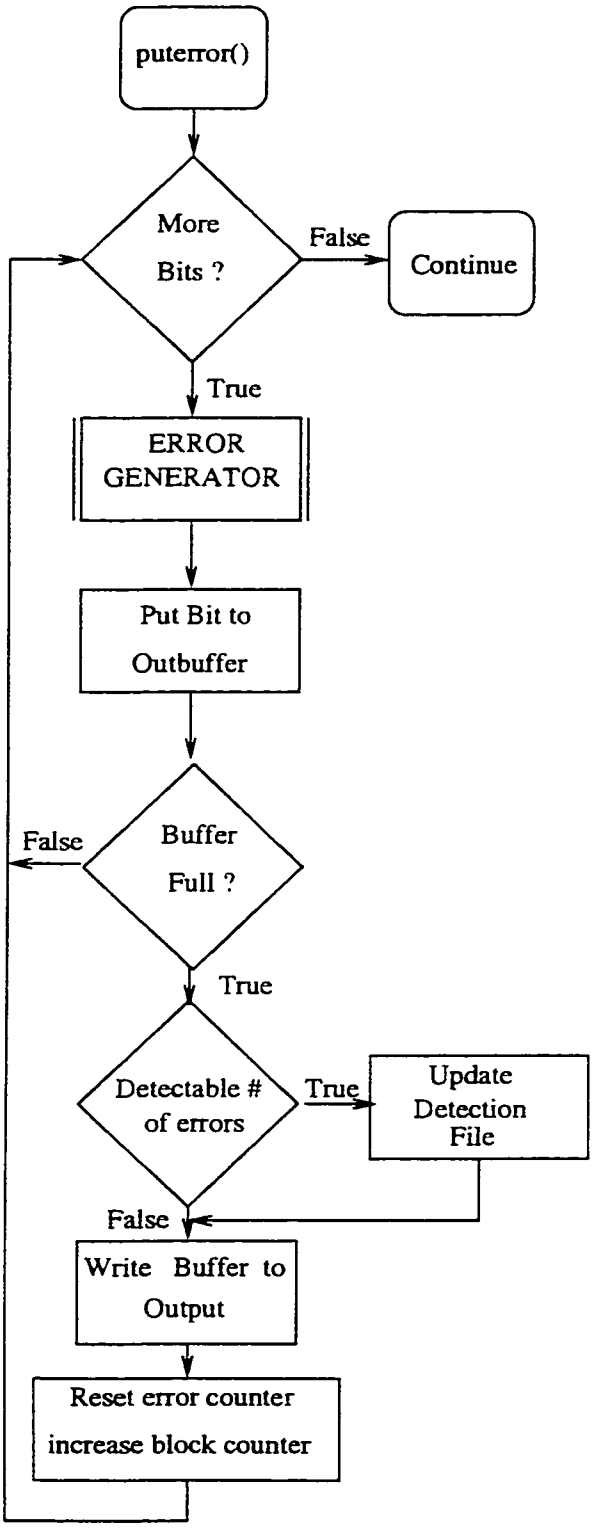


e) Traffic Simulator : Process of Header Info

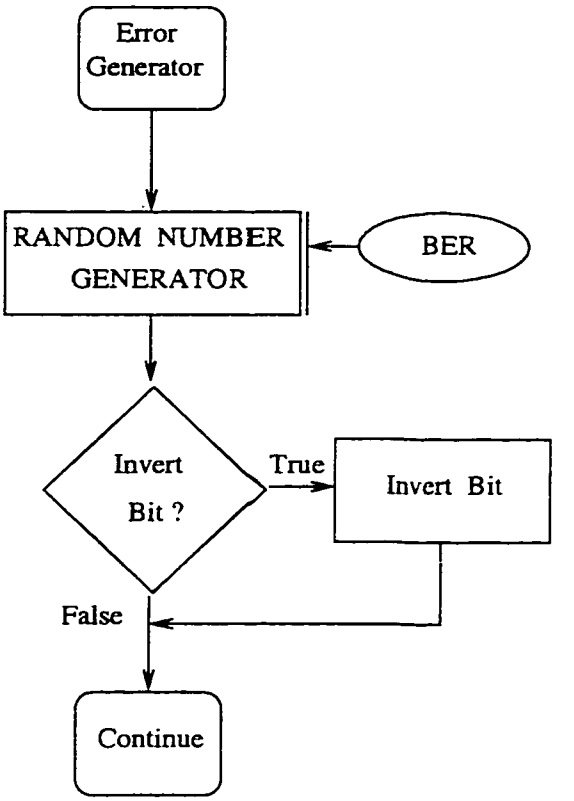


f) Traffic Simulator : putbits()

Figure A.5: Traffic Simulator



g) Traffic Simulator : puterror()



h) Traffic Simulator : Error Generator

Figure A.6: Traffic Simulator

A.2 MPEG2 Mapper

Usage: mpeg2map -b InputBitstream -c I/P distance -f Detection InputFile -g CheckByte -o OutFile

InputBitstream: The MPEG2 bitstream at the input of Traffic Simulator.

I/P distance: The distance between successive I and P frames.

Detection InputFile: File containing the ED blocks in error.

CheckByte: Used in cases where the location of a single byte is needed.

Outfile: File to be used by Error Concealment. Entries should have the following format: Picture x, Slice y MB z.

Example: mpeg2map -b tennis.m2v -c3 -f detect.in -o conceal.out

MPEG2 Mapper is based on a MPEG2 video decoder. Because the purpose is to map a location in the compressed bitstream to a location to the uncompressed video, basic functions of the decoder like IDCT, Inverse Quantization, Frame Storage are removed. For each bit extracted from the InputBitstream we check if that bit corresponds to the beginning of an ED block reported in error (condition strm_bit == check_byte * 8). When that condition becomes True, the current location in the uncompressed domain defined by the variables *current_picture*, *current_slice*, and *current_MB*, are recorded in the Outfile. In the case where only one location needed to be checked we may input the information using the online argument CheckByte. Figures A.7, A.8, and A.9 give the Logic Diagram of the MPEG2 Mapper.

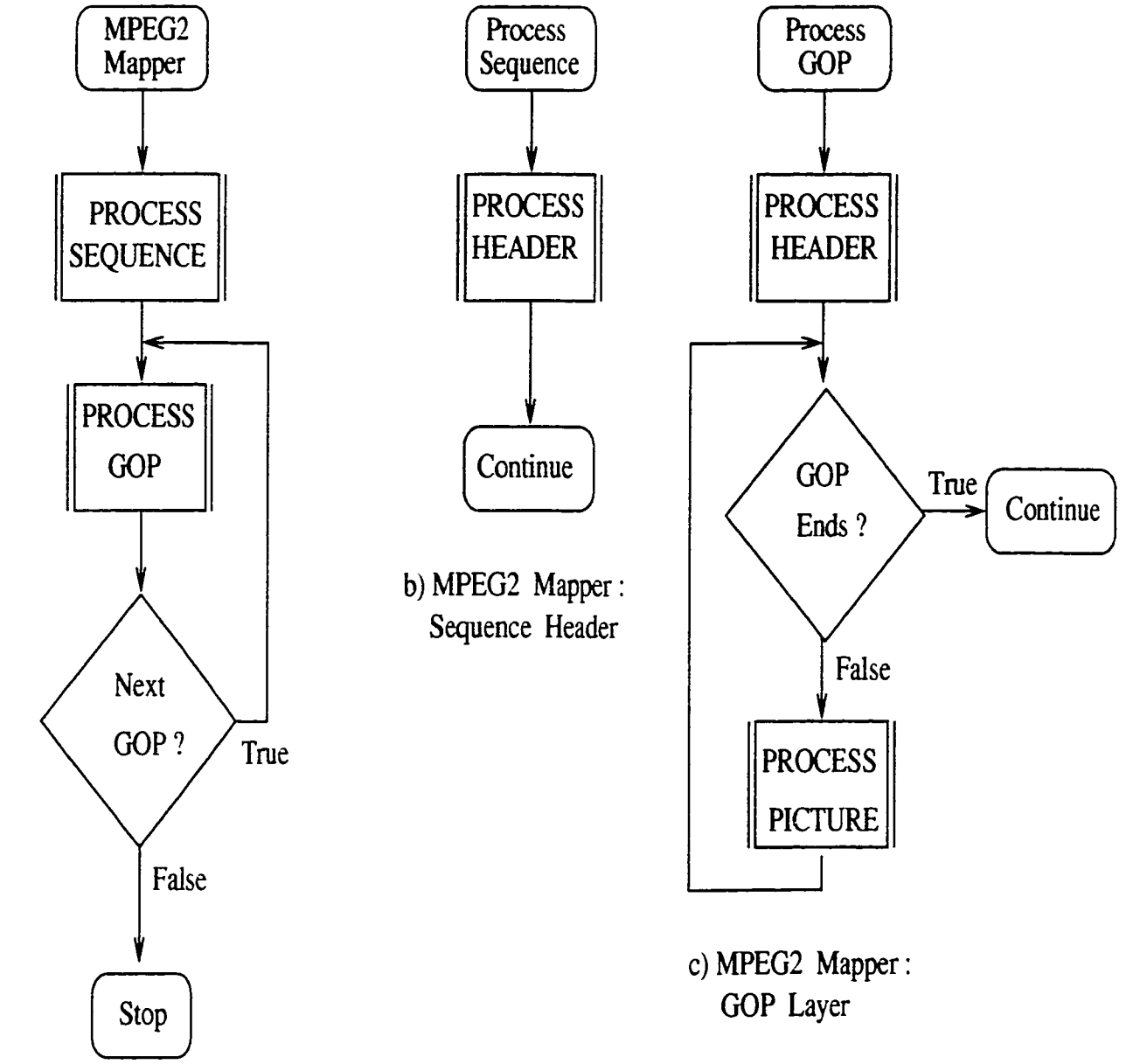
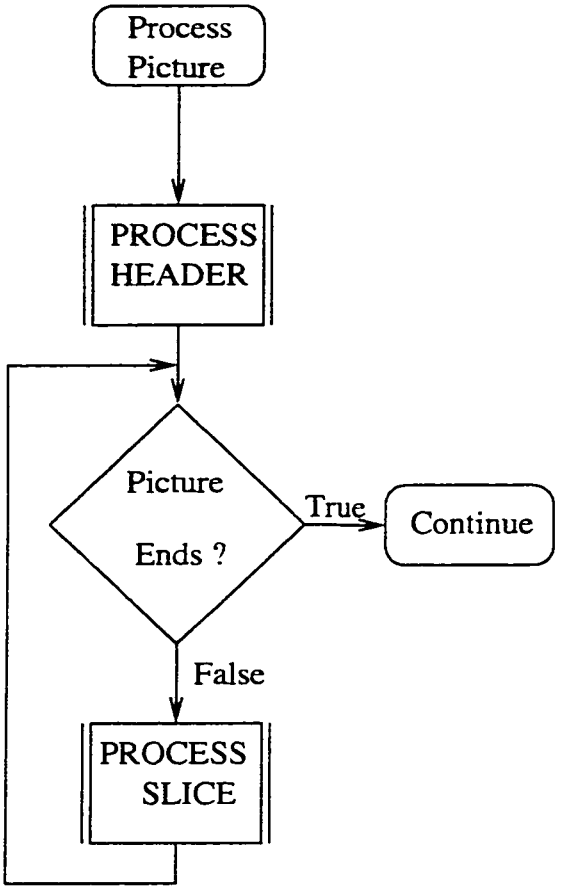
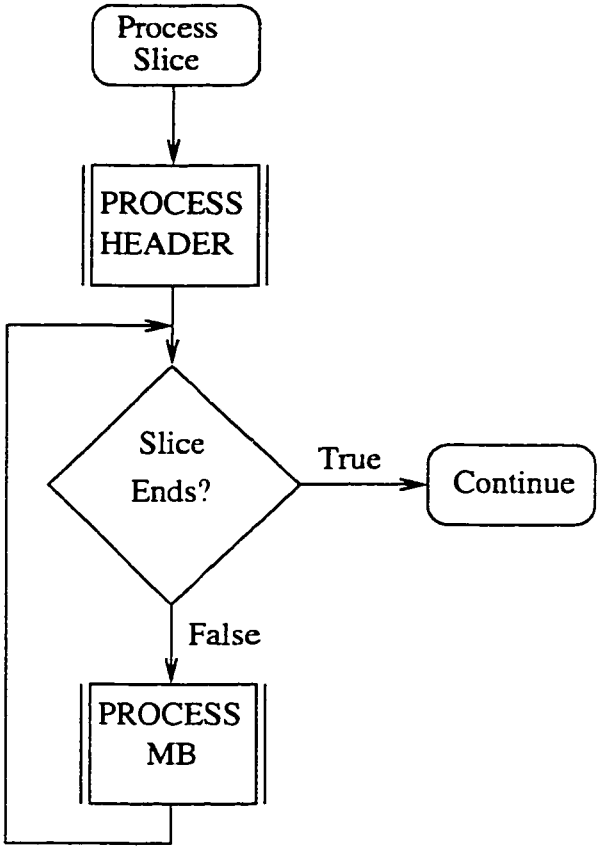


Figure A.7: MPEG2 Mapper

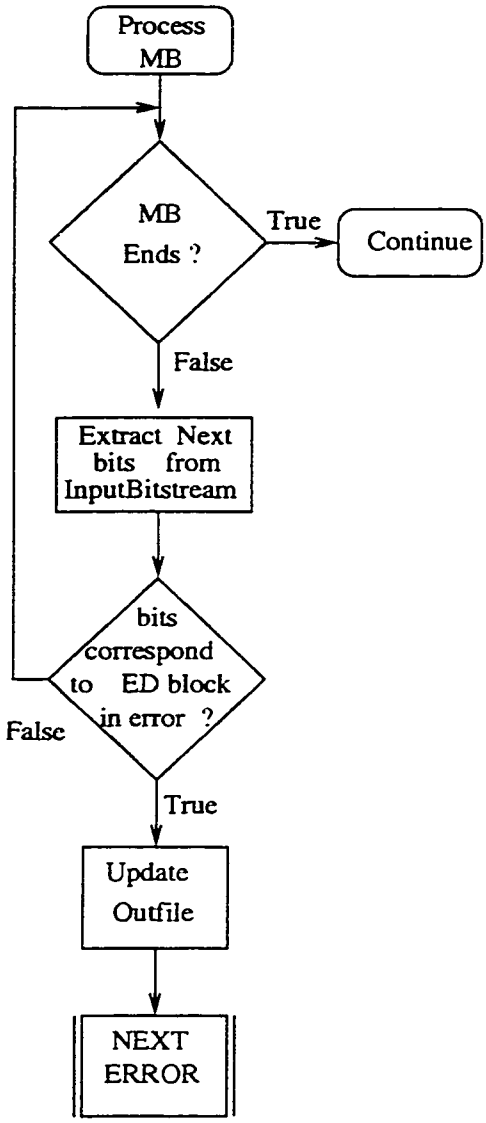


d) MPEG2 Mapper :
Picture Layer

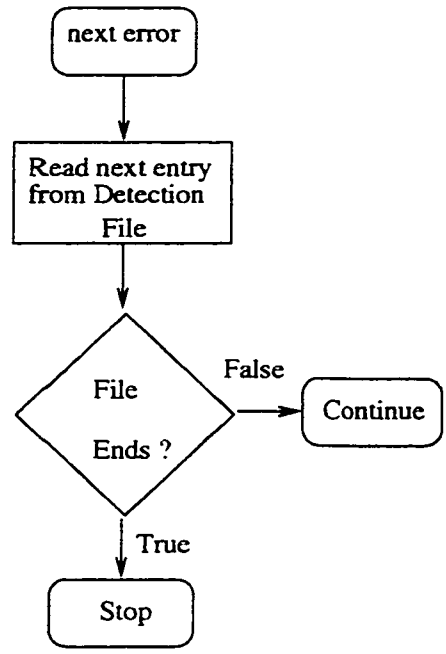


e) MPEG2 Mapper :
Slice Layer

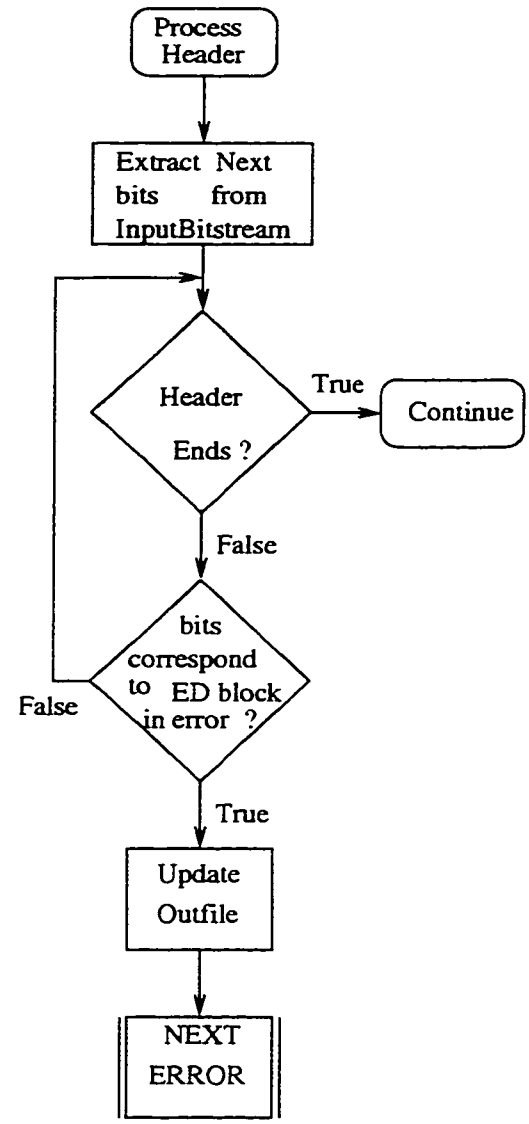
Figure A.8: MPEG2 Mapper



f) MPEG2 Mapper : MB Layer



g) MPEG2 Mapper : next error



h) MPEG2 Mapper : Header Layer

Figure A.9: MPEG2 Mapper

A.3 Error Concealment

Usage: conceal Parameter file Error Location file base_name

Parameter File: File containing information regarding the number of GOPs that the sequence has, the number of the first frame, the number of frames in a GOP, the difference between I and P frames, the horizontal and vertical size of each frame, and the chroma format.

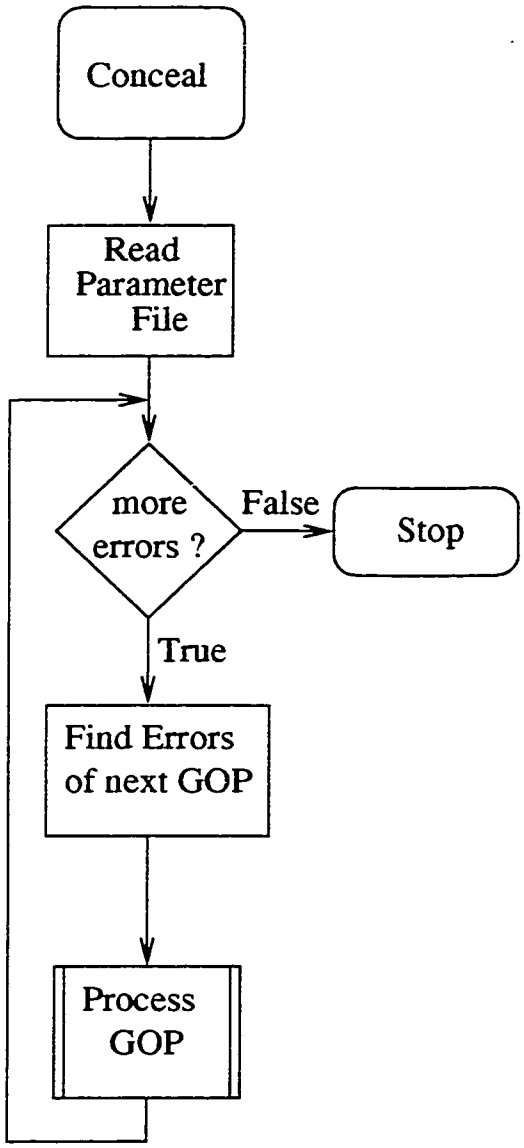
Error Location File: The Output from the MPEG2 Mapper

base_name The base name of the frames

Example: conceal conc.par conceal.in susie

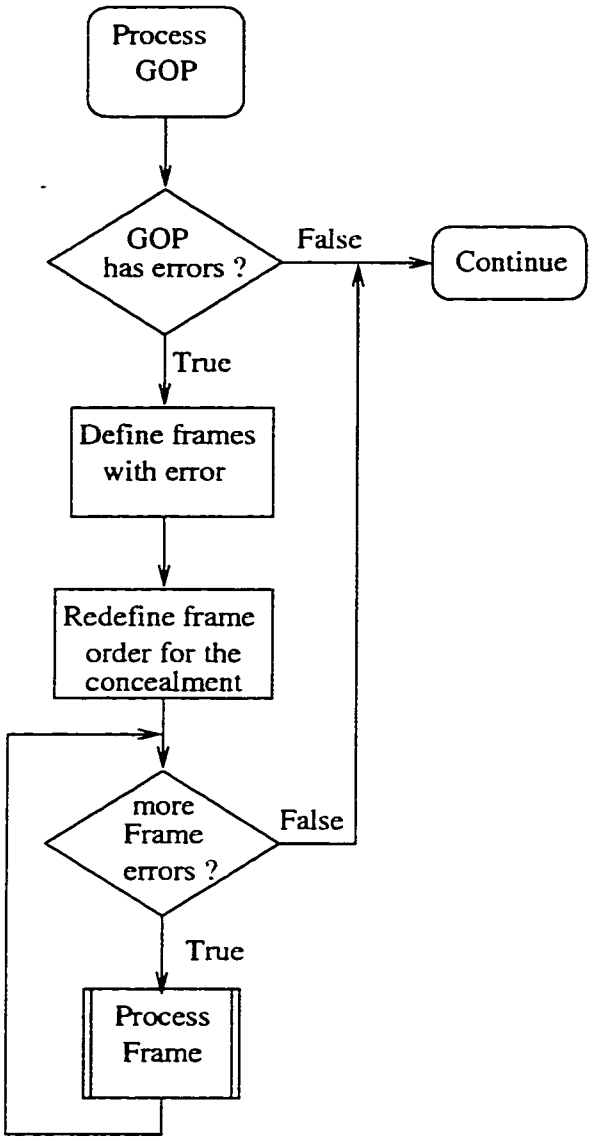
Concealment starts with the processing of the Parameter file. After the parameters of the sequence are defined, table error_table which contains the frames with error, is defined. Then control will pass to the GOP module which will process all the GOPs of the sequence. There gop_error_table is defined which contains the frames of the next GOP, reported with errors. If there are no frames in error the next GOP will be skipped.

For each frame of the next GOP, reported with errors, table frame_error_table is formed. This table contains the location of errors in the next frame to be processed. One thing that needs to be mentioned is that because concealment is performed in the uncompressed domain, error propagation will be present. That means that errors in I or P frames may pass into the subsequent P or B frames. To deal with that problem the following strategy was followed. First the frames in a GOP are reordered. By doing that, anchor frames will be processed before the predicted frames. Also if an error occurred in a anchor frame (I or P frame), it is assumed that it will be present (at the same location) for the rest of the GOP. Then control will pass to the frame module. There for each entry of the frame_error_table a number of MBs from the previous frame will be copied in the current frame. Figures A.10, and A.11 give the Logic Diagram of the Error Concealment.

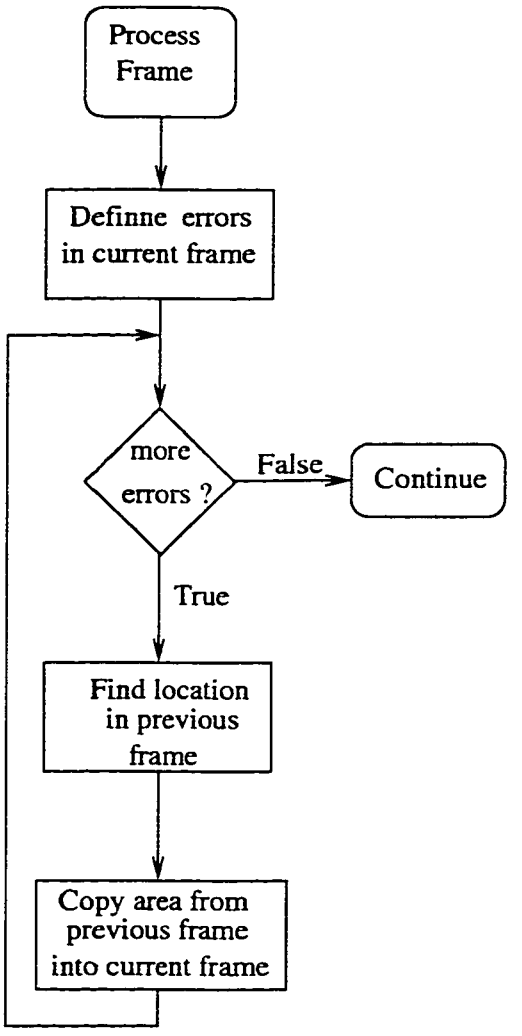


a) Conceal: Main Process

Figure A.10: Error Concealment



b) Error Concealment : Process GOP



c) Error Concealment : Process frame

Figure A.11: Error Concealment

A.4 Bit Toggler

Usage: bitoggle -b InputBitstream -d OutBitstream -e Detection InputFile -g Statistics Outfile -g Disable warnings to stderr

InputBitstream: The MPEG2 bitstream at the input of the Bitoggler.

OutBitstream: The MPEG2 bitstream after the bitoggling process.

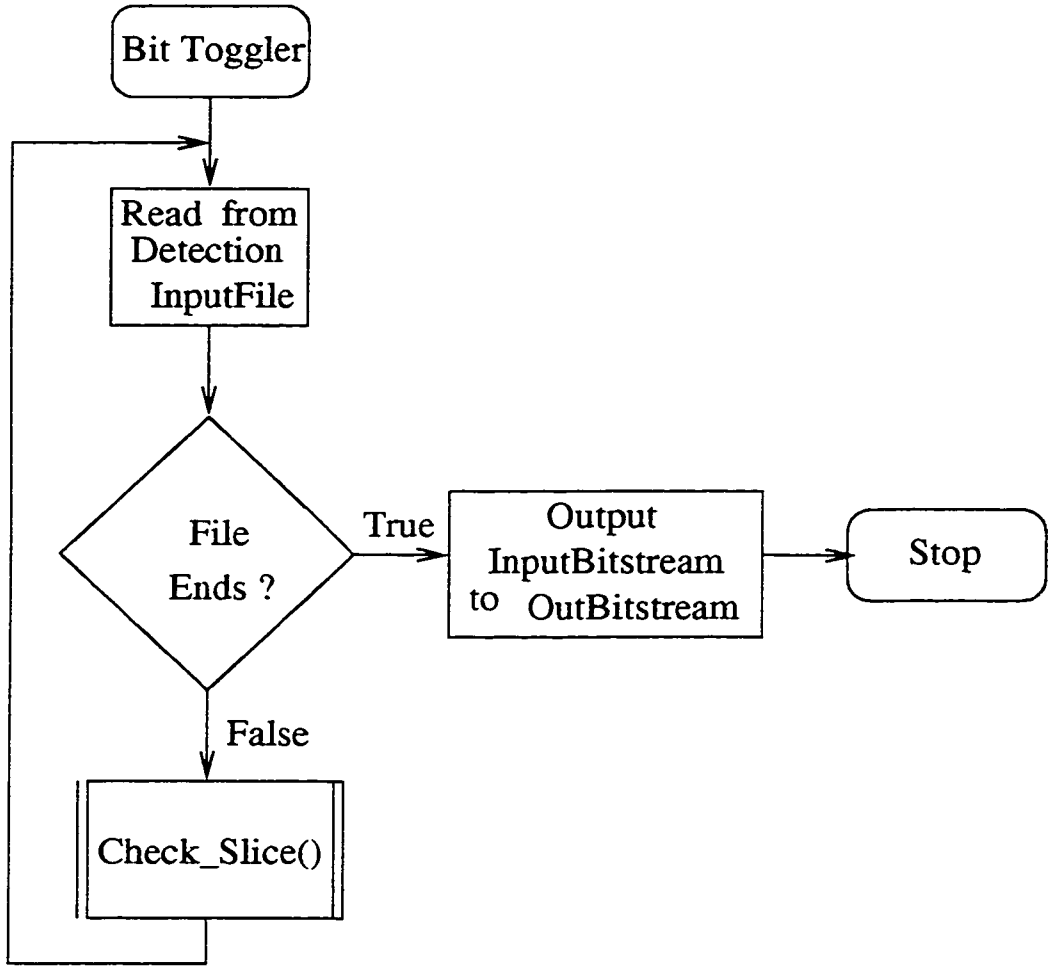
Detection InputFile: File containing information, regarding blocks in error.

Statistics Outfile: Statistics File.

Bitoggler is again a program based on the MPEG2 video decoder. The main process here is done by the Check_Slice(). For every slice (as far as there are more errors reported), this function will decompress the slice twice. The first time the size of the slice is determined by checking the distance in bits between the two nearest headers. Then the location of the next reported error is checked if it belongs in the current slice.

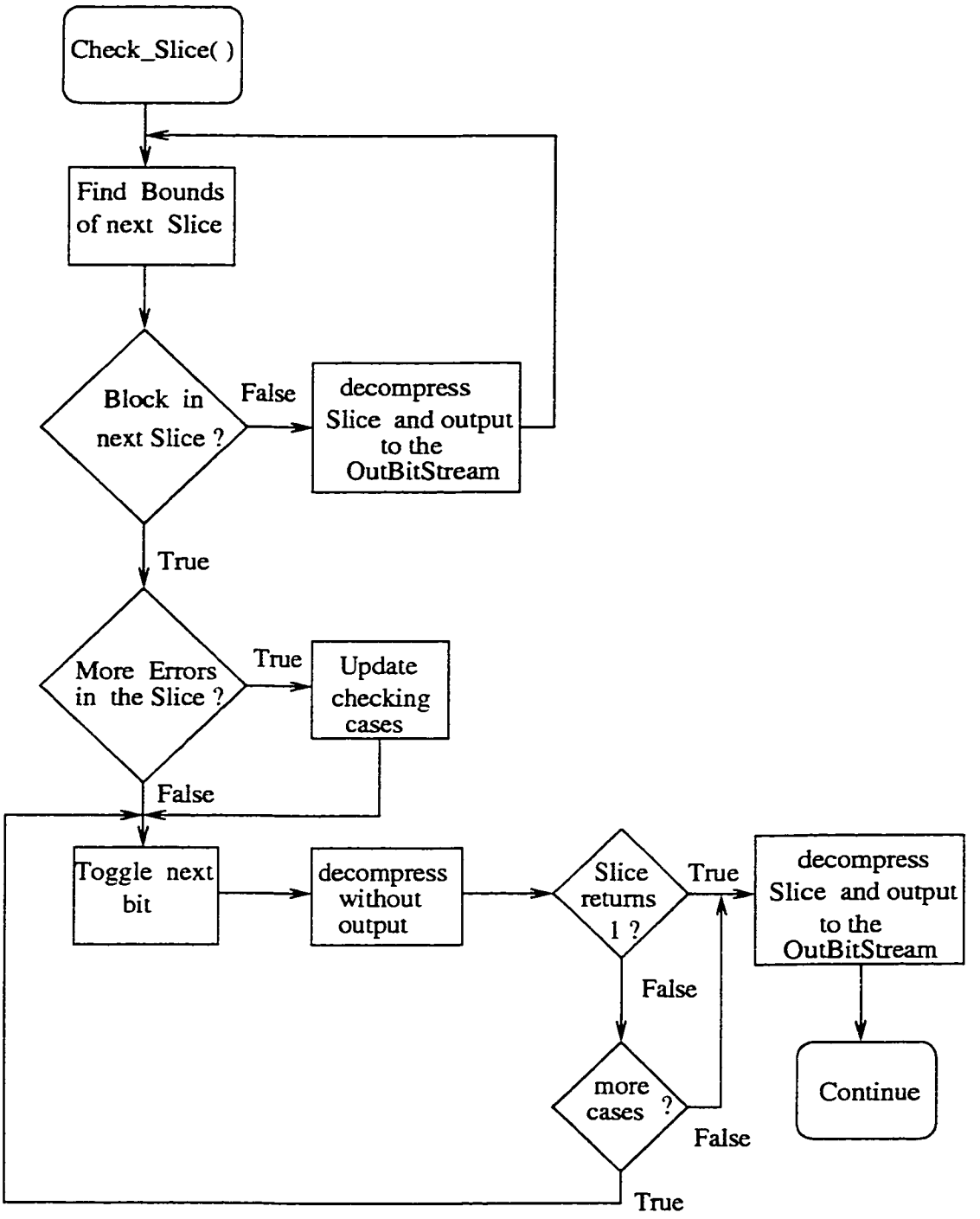
If it does first we check if subsequent errors belong in the same slice. Then decompression of the slice starts (control pass to the function slice() which performs the decompression) and the output is disabled. When control returns back to the Check_Slice the return value is checked. If it is 1 no syntax errors were found and slice is ready for output. If the return value is 0 syntax problems were found. This process continues until the return value becomes 1 or we run out of bits to toggle (13 bits for a single slice error).

The process of altering bits is performed by the function Show_Bits(). This function is modified and runs in two modes. In mode 0 is used to show the next N bits from the buffer. In mode 1 is also is used to show the next N bits from the buffer, but also to check if the condition (bits_extracted == ED_block_number * 12) is satisfied by the next bits extracted from the bitstream. If it is satisfied, that means that the following bits belong to the ED block in error and depending on the case the appropriate bit is toggled. Figures A.12, and A.13 give the Logic Diagram of the Bit Toggler.



a) Bit Toggler : Main Process

Figure A.12: Bit Toggler



b) Bit Toggler : Check_Slice( )

Figure A.13: Bit Toggler