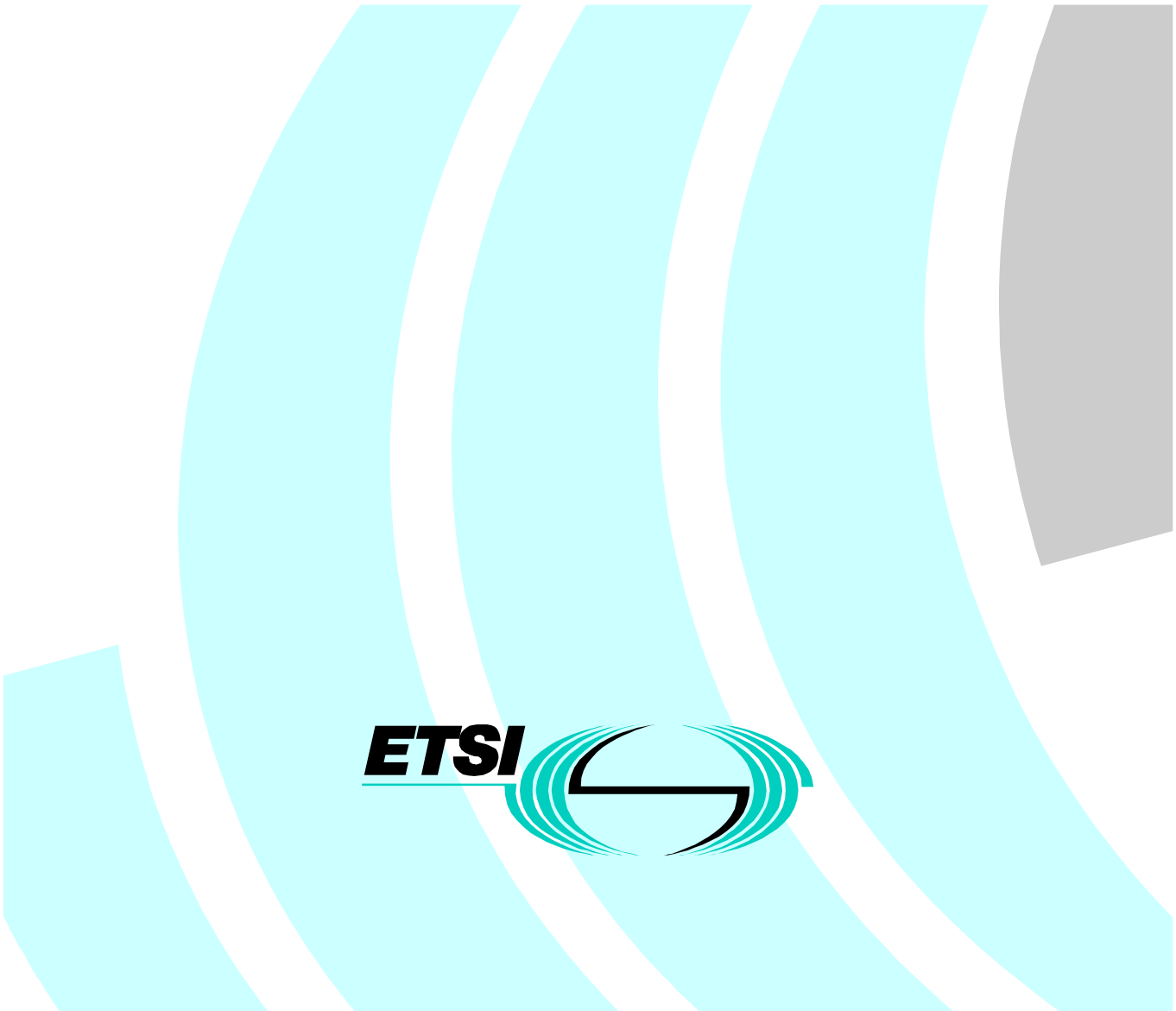
ETSI ES 201 108 V1.1.1 (2000-02)

ETSI Standard

Speech Processing, Transmission and Quality aspects (STQ);
Distributed speech recognition;
Front-end feature extraction algorithm;
Compression algorithms



Reference
DES/STQ-00007

Keywords
speech, performance, transmission

ETSI

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Internet
secretariat@etsi.fr
Individual copies of this ETSI deliverable
can be downloaded from
http://www.etsi.org
If you find errors in the present document, send your
comment to: editor@etsi.fr

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Speech processing, Transmission and Quality aspects (STQ).

Introduction

The performance of speech recognition systems receiving speech that has been transmitted over mobile channels can be significantly degraded when compared to using an unmodified signal. The degradations are as a result of both the low bit rate speech coding and channel transmission errors. A Distributed Speech Recognition (DSR) system overcomes these problems by eliminating the speech channel and instead using an error protected data channel to send a parameterized representation of the speech, which is suitable for recognition. The processing is distributed between the terminal and the network. The terminal performs the feature parameter extraction, or the front-end of the speech recognition system. These features are transmitted over a data channel to a remote "back-end" recognizer. The end result is that the transmission channel does not affect the recognition system performance and channel invariability is achieved.

The present document presents the standard for a front-end to ensure compatibility between the terminal and the remote recognizer. The particular front-end used is called the Mel-Cepstrum which has been used extensively in speech recognition systems.

1 Scope

The present document specifies algorithms for front-end feature extraction and their transmission which form part of a system for distributed speech recognition. The specification covers the following components:

- the algorithm for front-end feature extraction to create Mel-Cepstrum parameters;
- the algorithm to compress these features to provide a lower data transmission rate;
- the formatting of these features with error protection into a bitstream for transmission;
- the decoding of the bitstream to generate the front-end features at a receiver together with the associated algorithms for channel error mitigation.

The present document does not cover the "back-end" speech recognition algorithms that make use of the received DSR front-end features.

The algorithms are defined in a mathematical form or as flow diagrams. Software implementing these algorithms is available from ETSI written in the 'C' programming language. Conformance tests are not specified as part of the standard. The recognition performance of proprietary implementations of the standard can be compared with those obtained using the reference 'C' code on appropriate speech databases.

It is anticipated that the DSR bitstream will be used as a payload in other higher level protocols when deployed in specific systems supporting DSR applications.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

[1] GTS GSM 03.50: "Digital cellular telecommunications system (Phase 2+); Transmission planning aspects of the speech service in the GSM Public Land Mobile Network (PLMN) system (GSM 03.50)".

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

analog-to-digital conversion: electronic process in which a continuously variable (analog) signal is changed, without altering its essential content, into a multi-level (digital) signal

DC offset: direct current (DC) component of the waveform signal

discrete cosine transform: process of transforming the log filterbank amplitudes into cepstral coefficients

fast Fourier transform: fast algorithm for performing the discrete Fourier transform to compute the spectrum representation of a time-domain signal

feature compression: process of reducing the amount of data to represent the speech features calculated in feature extraction

feature extraction: process of calculating a compact parametric representation of speech signal features which are relevant for speech recognition. The feature extraction process is carried out by the front-end algorithm

feature vector: set of feature parameters (coefficients) calculated by the front-end algorithm over a segment of speech waveform

framing: process of splitting the continuous stream of signal samples into segments of constant length to facilitate blockwise processing of the signal

Frame pair packet: definition is specific to the present document: The combined data from two quantized feature vectors together with 4 bits of CRC

front-end: part of a speech recognition system which performs the process of feature extraction

magnitude spectrum: absolute-valued Fourier transform representation of the input signal

multiframe: grouping of multiple frame vectors into a larger data structure

mel-frequency warping: process of non-linearly modifying the scale of the Fourier transform representation of the spectrum

mel-frequency cepstral coefficients: cepstral coefficients calculated from the mel-frequency warped Fourier transform representation of the log magnitude spectrum

notch filtering: filtering process in which the otherwise flat frequency response of the filter has a sharp notch at a pre-defined frequency. In the present document, the notch is placed at the zero frequency, to remove the DC component of the signal

offset compensation: process of removing DC offset from a signal

pre-emphasis: filtering process in which the frequency response of the filter has emphasis at a given frequency range. In the present document, the high-frequency range of the signal spectrum is pre-emphasized

sampling rate: number of samples of an analog signal that are taken per second to represent it digitally

windowing: process of multiplying a waveform signal segment by a time window of given shape, to emphasize pre-defined characteristics of the signal

zero padding: method of appending zero-valued samples to the end of a segment of speech samples for performing a FFT operation

3.2 Symbols

For the purposes of the present document, the following symbols apply:

For feature extraction (clause 4):

bin_k	absolute value of complex-valued FFT output vector *k*
C_i	*i*th cepstral coefficient
$cbin_i$	Center frequency of the *i*th Mel channel in terms of FFT bin indices
$fbank_k$	output of Mel filter for filter bank *k*
f_{c_i}	Center frequency of the *i*th Mel channel
f_i	log filter bank output for the *i*th Mel channel
f_s	input signal sampling rate
f_{s1}, f_{s2}, f_{s3}	symbols for specific input signal sampling rates (8 kHz, 11 kHz, 16 kHz)
f_{start}	starting frequency of Mel filter bank
FFTL	Length of FFT block
$\ln(\ )$	natural logarithm operation
$\log_{10}(\ )$	10-base logarithm operation
M	frame shift interval
$Mel\{\ \}$	Mel scaling operator
$Mel^{-1}\{\ \}$	inverse Mel scaling operator
N	frame length
$round\{\ \}$	operator for rounding towards nearest integer
s_{in}	input speech signal
s_{of}	offset-free input speech signal
s_{pe}	speech signal after pre-emphasis operation
s_w	windowed speech signal

For compression (clause 5):

$idx^{i,i+1}(m)$	codebook index
m	framenumber
$N^{i,i+1}$	compression: size of the codebook
$Q^{i,i+1}$	compression codebook
$q_j^{i,i+1}$	*j*th codevector in the codebook $Q^{i,i+1}$
$W^{i,i+1}$	weight matrix
y(m)	Feature vector with 14 components

For error mitigation:

$badframeindex_i$	indicator if received VQ index is suspected to be received with transmission error
T_i	threshold on cepstral coefficient

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ADC	analog-to-digital conversion
CRC	cyclic redundancy code
DSR	distributed speech recognition
logE	logarithmic frame energy
LSB	least significant bit
MSB	most significant bit
VQ	vector quantizer

4 Front-end feature extraction algorithm

4.1 Introduction

This clause describes the distributed speech recognition front-end algorithm based on mel-cepstral feature extraction technique. The specification covers the computation of feature vectors from speech waveforms sampled at different rates (8, 11, and 16 kHz).

The feature vectors consist of 13 static cepstral coefficients and a log-energy coefficient.

The feature extraction algorithm defined in this clause forms a generic part of the specification while clauses 4 to 6 define the feature compression and bit-stream formatting algorithms which may be used in specific applications.

The characteristics of the input audio parts of a DSR terminal will have an effect on the resulting recognition performance at the remote server. Developers of DSR speech recognition servers can assume that the DSR terminals will operate within the ranges of caracteristics as specified in GSM 03.50 [1]. DSR terminal developers should be aware that reduced recognition performance may be obtained if they operate outside the recommended tolerances.

4.2 Front-end algorithm description

4.2.1 Front-end block diagram

The following block diagram shows the different blocks of the front-end algorithm.

The details of the analog-to-digital conversion (ADC) are not subject to the present document, but the block has been included to account for the different sampling rates. The blocks Feature Compression and Bit Stream Formatting are covered in clauses 4 to 6 of the present document.

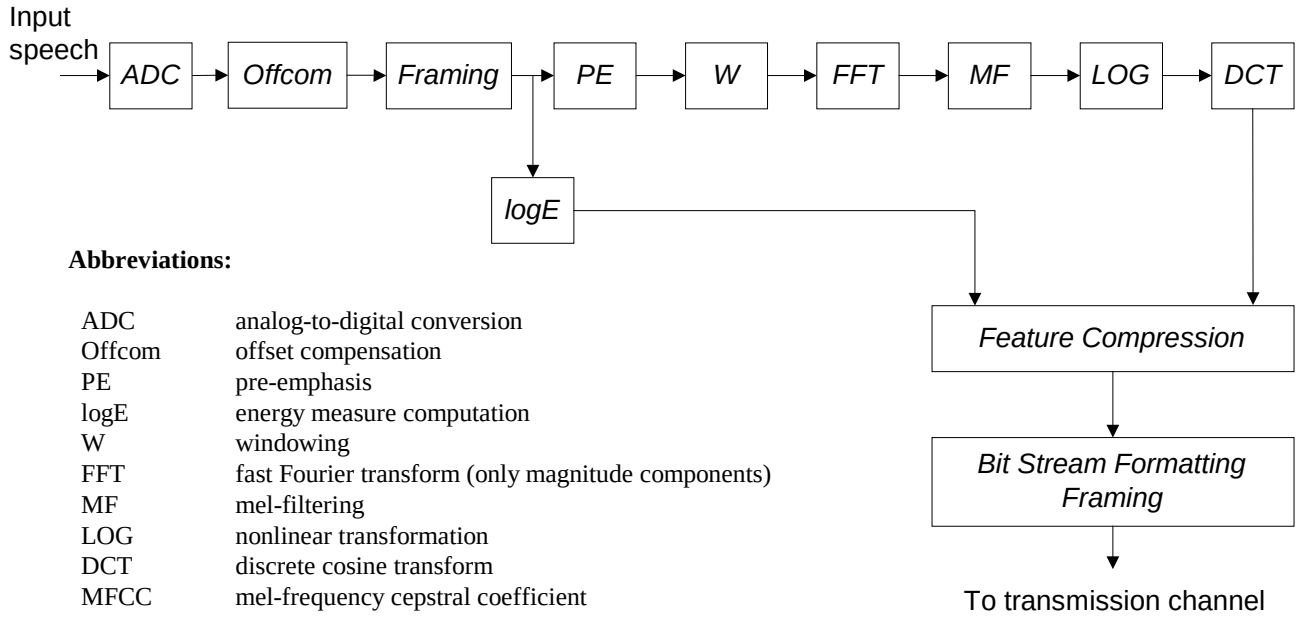


Figure 4.1: Block diagram of the front-end algorithm

4.2.2 Analog-to-digital conversion

The specifics of the analog-to-digital conversion are not part of the present document. Different word-lengths can be used depending on the application.

The output sampling rates of the ADC block are f_{s1} = 8 kHz, f_{s2} = 11 kHz, and f_{s3} = 16 kHz.

4.2.3 Offset compensation

Prior to the framing, a notch filtering operation is applied to the digital samples of the input speech signal s_{in} to remove their DC offset, producing the offset-free input signal s_{of}.

$$s_{of}(n) = s_{in}(n) - s_{in}(n-1) + 0{,}999 \cdot s_{of}(n-1) \qquad (4.1)$$

4.2.4 Framing

The offset-free input signal s_{of} is divided into overlapping frames of N samples. The frame shift interval (difference between the starting points of consecutive frames) is M samples. The parameter M defines the number of frames per unit time.

The specific values of *N* and *M* depend on the sampling rate according to table 4.1. The frame length is 25 ms for 8 and 16 kHz sampling rates, and 23,27 ms for 11 kHz.

Table 4.1: Values of Frame Length and Frame Shift Interval Depending on the Sampling Rate

Sampling rate (kHz)	**f_{s3} = 16**	**f_{s2} = 11**	**f_{s1} = 8**
Frame length *N* (samples)	**400**	**256**	**200**
Shift interval *M* (samples)	**160**	**110**	**80**

4.2.5 Energy measure

The logarithmic frame energy measure (logE) is computed after the offset compensation filtering and framing for each frame:

$$\mathrm{logE} = \ln\left(\sum_{i=1}^{N} s_{of}(i)^2\right) \tag{4.2}$$

Here N is the frame length and s_{of} is the offset-free input signal.

A floor is used in the energy calculation which makes sure that the result is not less than -50. The floor value (lower limit for the argument of ln) is approximately 2e-22.

4.2.6 Pre-emphasis

A pre-emphasis filter is applied to the framed offset-free input signal:

$$s_{pe}(n) = s_{of}(n) - 0{,}97 \cdot s_{of}(n-1) \tag{4.3}$$

Here s_{of} and s_{pe} are the input and output of the pre-emphasis block, respectively.

4.2.7 Windowing

A Hamming window of length N is applied to the output of the pre-emphasis block:

$$s_w(n) = \left\{0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi(n-1)}{N-1}\right)\right\} \cdot s_{pe}(n),\ 1 \le n \le N \tag{4.4}$$

Here N is the frame length and s_{pe} and s_w are the input and output of the windowing block, respectively.

4.2.8 FFT

Each frame of N samples is zero padded to form an extended frame of 256 samples for 8 and 11 kHz sampling rate, and 512 samples for 16 kHz. An FFT of length 256 or 512, respectively, is applied to compute the magnitude spectrum of the signal.

$$bin_k = \left|\sum_{n=0}^{FFTL-1} s_w(n) e^{-jnk\frac{2\pi}{FFTL}}\right|, \quad k = 0,\ldots,FFTL-1. \tag{4.5}$$

Here $s_w(n)$ is the input to the FFT block, $FFTL$ is the block length (256 or 512 samples), and bin_k is the absolute value of the resulting complex vector.

NOTE: Due to symmetry, only $bin_{0..FFTL/2}$ are used for further processing.

4.2.9 Mel filtering

The low-frequency components of the magnitude spectrum are ignored. The useful frequency band lies between 64 Hz and half of the actual sampling frequency. This band is divided into 23 channels equidistant in mel frequency domain. Each channel has triangular-shaped frequency window. Consecutive channels are half-overlapping.

The choice of the starting frequency of the filter bank, f_{start} = 64 Hz, roughly corresponds to the case where the full frequency band is divided into 24 channels and the first channel is discarded using any of the three possible sampling frequencies.

The centre frequencies of the channels in terms of FFT bin indices ($cbin_i$ for the ith channel) are calculated as follows:

$$Mel\{x\} = 2595 \cdot \log_{10}\left(1 + \frac{x}{700}\right), \tag{4.6a}$$

$$f_{c_i} = Mel^{-1}\left\{Mel\{f_{start}\} + \frac{Mel\{f_s/2\} - Mel\{f_{start}\}}{23+1} i\right\}, \quad i = 1,...,23, \tag{4.6b}$$

$$cbin_i = round\left\{\frac{f_{c_i}}{f_s} FFTL\right\}, \tag{4.6c}$$

where *round(.)* stands for rounding towards the nearest integer.

The output of the mel filter is the weighted sum of the FFT magnitude spectrum values (bin_i) in each band. Triangular, half-overlapped windowing is used as follows:

$$fbank_k = \sum_{i=cbin_{k-1}}^{cbin_k} \frac{i - cbin_{k-1} + 1}{cbin_k - cbin_{k-1} + 1} bin_i + \sum_{i=cbin_k+1}^{cbin_{k+1}} \left(1 - \frac{i - cbin_k}{cbin_{k+1} - cbin_k + 1}\right) bin_i, \tag{4.7}$$

where k = 1, ..., 23, $cbin_0$ and $cbin_{24}$ denote the FFT bin indices corresponding to the starting frequency and half of the sampling frequency, respectively,

$$cbin_0 = round\left\{\frac{f_{start}}{f_s} FFTL\right\}, \tag{4.8a}$$

$$cbin_{24} = round\left\{\frac{f_s/2}{f_s} FFTL\right\} = FFTL/2. \tag{4.8b}$$

4.2.10 Non-linear transformation

The output of mel filtering is subjected to a logarithm function (natural logarithm).

$$f_i = \ln(fbank_i), i = 1,\ldots,23 \tag{4.9}$$

The same flooring is applied as in the case of energy calculation, that is, the log filter bank outputs cannot be smaller than -50.

4.2.11 Cepstral coefficients

13 cepstral coefficients are calculated from the output of the Non-linear Transformation block.

$$C_i = \sum_{j=1}^{23} f_j \cdot \cos\left(\frac{\pi \cdot i}{23}(j - 0{,}5)\right), \qquad 0 \le i \le 12 \tag{4.10}$$

4.2.12 Front-end output

The final feature vector consists of 14 coefficients: the log-energy coefficient (subclause 4.2.5) and the 13 cepstral coefficients (subclause 4.2.11).

The C0 coefficient is often redundant when the log-energy coefficient is used. However, the feature extraction algorithm is defined here for both energy and C0. Depending on the application, either the C0 coefficient or the log-energy coefficient may be used.

5 Feature Compression Algorithm

5.1 Introduction

This clause describes the distributed speech recognition front-end feature vector compression algorithm. The algorithm makes use of the parameters from the front-end feature extraction algorithm of clause 4. Its purpose is to reduce the number of bits needed to represent each front-end feature vector.

5.2 Compression algorithm description

5.2.1 Input

The compression algorithm is designed to take the feature parameters for each short-time analysis frame of speech data as they are available and as specified in clause 4.

The input parameters used are the first twelve static mel cepstral coefficients:

$$\mathbf{c}(m) = [c_1(m), c_2(m) \ldots c_{12}(m)]^T \tag{5.1}$$

where *m* denotes the frame number, plus the zeroth cepstral coefficient and a log energy term as defined in subclause 4.2.12. These parameters are formatted as:

$$y(m) = \begin{bmatrix} \mathbf{c}(m) \\ c_0(m) \\ \log[E(m)] \end{bmatrix} \tag{5.2}$$

5.2.2 Vector quantization

The feature vector ***y(m)*** is directly quantized with a split vector quantizer. Coefficients are grouped into pairs, and each pair is quantized using its own VQ codebook. The resulting set of index values is then used to represent the speech frame. Coefficient pairings (by front-end parameter) are shown in table 5.1, along with the codebook size used for each pair.

The closest VQ centroid is found using a weighted Euclidean distance to determine the index:

$$d_j^{i,i+1} = \begin{bmatrix} y_i(m) \\ y_{i+1}(m) \end{bmatrix} - q_j^{i,i+1} \tag{5.3}$$

$$idx^{i,i+1}(m) = \underset{0 \le j \le (N^{i,i+1}-1)}{\operatorname{argmin}} \left\{ \left(d_j^{i,i+1} \right) W^{i,i+1} \left(d_j^{i,i+1} \right) \right\}, \quad i = 0,2,4 \ldots 12 \tag{5.4}$$

where $q_j^{i,i+1}$ denotes the *j*th codevector in the codebook $Q^{i,i+1}$, $N^{i,i+1}$ is the size of the codebook, $W^{i,i+1}$ is the (possibly identity) weight matrix to be applied for the codebook $Q^{i,i+1}$, and $idx^{i,i+1}(m)$ denotes the codebook index chosen to represent the vector $[y_i(m), y_{i+1}(m)]^T$. The indices are then retained for transmission to the back-end.

Table 5.1: Split Vector Quantization Feature Pairings

	Size	Weight Matrix		
Codebook	**($N^{l,l+1}$)**	**($W^{l,l+1}$)**	**Element 1**	**Element 2**
$Q^{0,1}$	64	I	c_1	c_2
$Q^{2,3}$	64	I	c_3	c_4
$Q^{4,5}$	64	I	c_5	c_6
$Q^{6,7}$	64	I	c_7	c_8
$Q^{8,9}$	64	I	c_9	c_{10}
$Q^{10,11}$	64	I	c_{11}	c_{12}
$Q^{12,13}$	256	Non-identity	c_0	log[E]

Two sets of VQ codebooks are defined; one is used for speech sampled at 8 kHz or 11 kHz while the other for speech sampled at 16 kHz. The numeric values of these codebooks and weights are specified as part of the software implementing the standard. The weights used (to one decimal place of numeric accuracy) are:

8 kHz or 11 kHz sampling rate $$W^{12,13} = \begin{bmatrix} 1446,0 & 0 \\ 0 & 14,7 \end{bmatrix}$$

16 kHz sampling rate $$W^{12,13} = \begin{bmatrix} 1248,9 & 0 \\ 0 & 12,7 \end{bmatrix}$$

6 Framing, Bit-Stream Formatting, and Error Protection

6.1 Introduction

This clause describes the format of the bitstream used to transmit the compressed feature vectors. The frame structure used and the error protection that is applied to the bitstream is defined.

6.2 Algorithm description

6.2.1 Multiframe Format

In order to reduce the transmission overhead, each multiframe message packages speech features from multiple short-time analysis frames. A multiframe, as shown in table 6.1, consists of a synchronization sequence, a header field, and a stream of frame packets.

Table 6.1: Multiframe format

Sync Sequence	Header Field	Frame Packet Stream
<- 2 octets ->	<- 4 octets ->	<- 138 octets ->
		<- 144 octets ->

In order to improve the error robustness of the protocol, the multiframe has a fixed length (144 octets). A multiframe represents 240 ms of speech, resulting in a data rate of 4 800 bits/s.

In the specification that follows, octets are transmitted in ascending numerical order; inside an octet, bit 1 is the first bit to be transmitted. When a field is contained within a single octet, the lowest-numbered bit of the field represents the lowest-order value (or the least significant bit). When a field spans more than one octet, the lowest-numbered bit in the first octet represents the lowest-order value (LSB), and the highest-numbered bit in the last octet represents the highest-order value (MSB). An exception to this field mapping convention is made for the cyclic redundancy code

(CRC) fields. For these fields, the lowest numbered bit of the octet is the highest-order term of the polynomial representing the field. In simple stream formatting diagrams (e.g., table 6.1) fields are transmitted left to right.

6.2.2 Synchronization Sequence

Each multiframe begins with the 16-bit synchronization sequence 0 x 87B2 (sent LSB first, as shown in table 6.2).

The inverse synchronization sequence 0 x 784D can be used for synchronous channels requiring rate adaptation. Each multiframe may be preceded or followed by one or more inverse synchronization sequences. The inverse sync is not required if a multiframe is immediately followed by the sync sequence for the next multiframe.

Table 6.2: Multiframe Synchronization Sequence

Bit	8	7	6	5	4	3	2	1	Octet
	1	0	0	0	0	1	1	1	1
	1	0	1	1	0	0	1	0	2

6.2.3 Header Field

Following the synchronization sequence, a header field is transmitted. Due to the critical nature of the data in this field, it is represented in a (31, 16) extended systematic codeword. This code will support 16-bits of data and has an error correction capability for up to three bit errors, an error detection capability for up to seven bit errors, or a combination of both error detection and correction.

Ordering of the message data and parity bits is shown in table 6.3, and definition of the fields appears in table 6.4. The 4 bit multiframe counter gives each multiframe a modulo-16 index. The counter value for the first multiframe is "0001". The multiframe counter is incremented by one for each successive multiframe until the final multiframe. The final multiframe is indicated by zeros in the frame packet stream (see subclause 6.2.4).

NOTE: The remaining nine bits which are currently undefined are left for future expansion. A fixed length field has been chosen for the header in order to improve error robustness and mitigation capability.

Table 6.3: Header field format

Bit	8	7	6	5	4	3	2	1	Octet
	EXP1	MframeCnt				feType	SampRate		1
	EXP9	EXP8	EXP7	EXP6	EXP5	EXP4	EXP3	EXP2	2
	P8	P7	P6	P5	P4	P3	P2	P1	3
	P16	P15	P14	P13	P12	P11	P10	P9	4

Table 6.4: Header field definitions

Field	No. Bits	Meaning	Code	Indicator
SampRate	2	sampling rate	00	8 kHz
			01	11 kHz
			10	undefined
			11	16 kHz
FeType	1	Front-end specification	0	standard
			1	noise robust
MframeCnt	4	multiframe counter	xxxx	Modulo-16 number
EXP1 - EXP9	9	Expansion bits (TBD)	0	(zero pad)
P1 - P16	16	Cyclic code parity bits	(see below)	

The generator polynomial used is:

$$g_1(X) = 1 + X^8 + X^{12} + X^{14} + X^{15} \quad (6.1)$$

The proposed (31, 16) code is extended, with the addition of an (even) overall parity check bit, to 32 bits. The parity bits of the codeword are generated using the calculation:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ P_{15} \\ P_{16} \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1 \\ 1&1&0&0&0&0&0&0&1&1&0&0&1&1&1&0 \\ 1&1&1&0&0&0&0&0&1&1&1&0&1&1&0&1 \\ 0&1&1&1&0&0&0&0&0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&0&0&1&0&1&1&0&0&0&0 \\ 0&1&0&1&1&1&0&0&0&1&0&1&1&0&0&0 \\ 0&0&1&0&1&1&1&0&0&0&1&0&1&1&0&0 \\ 0&0&0&1&0&1&1&1&0&0&0&1&0&1&1&0 \\ 1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0&1&0&1&1&0&0&0&1 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1 \end{bmatrix}^T \cdot \begin{bmatrix} SampRate1 \\ SampRate2 \\ feType \\ MFrameCnt1 \\ MFrameCnt2 \\ MFrameCnt3 \\ MFrameCnt4 \\ EXP1 \\ EXP2 \\ EXP3 \\ EXP4 \\ EXP5 \\ EXP6 \\ EXP7 \\ EXP8 \\ EXP9 \end{bmatrix} \quad (6.2)$$

where T denotes the matrix transpose.

6.2.4 Frame Packet Stream

Each 10 ms frame from the front-end is represented by the codebook indices specified in subclause 5.2.2. The indices for a single frame are formatted for a frame according to table 6.5.

NOTE: The exact alignment with octet boundaries will vary from frame to frame.

Table 6.5: Frame information for *m*th frame

Bit 8	7	6	5	4	3	2	1	Octet
$idx^{2,3}(m)$		$Idx^{0,1}(m)$						1
$idx^{4,5}(m)$				$idx^{2,3}(m)$ (cont)				2
$idx^{6,7}(m)$						$idx^{4,5}(m)$ (cont)		3
$idx^{10,11}(m)$		$Idx^{8,9}(m)$						4
$idx^{12,13}(m)$				$idx^{10,11}(m)$ (cont)				5
				$idx^{12,13}(m)$ (cont)				6

Two frames worth of indices, or 88 bits, are then grouped together as a pair. A 4-bit CRC ($g(X) = 1 + X + X^4$) is calculated on the frame pair and immediately follows it, resulting in a combined frame pair packet of 11,5 octets. Twelve of these frame pair packets are combined to fill the 138 octet feature stream. Table 6.6 illustrates the format of the protected feature packet stream. When the feature stream is combined with the overhead of the synchronization sequence and the header, the resulting format requires a data rate of 4 800 bits/s.

Table 6.6: CRC protected feature packet stream

Frame #1	Frame #2	CRC #1-2	...	Frame #23	Frame #24	CRC #23-24
<- 44 bits->	<- 44 bits ->	<- 4 bits ->				
		<- 138 octets / 1 104 bits ->				

All trailing frames within a final multiframe that contain no valid speech data will be set to all zeros.

7 Bit-Stream Decoding and Error Mitigation

7.1 Introduction

This clause describes the algorithms used to decode the received bitstream to regenerate the speech feature vectors. It also covers the error mitigation algorithms that are used to minimize the consequences of transmission errors on the performance of a speech recognizer.

7.2 Algorithm description

7.2.1 Synchronization Sequence Detection

The method used to achieve synchronization is not specified in the present document. The detection of the start of a multiframe may be done by the correlation of the incoming bit stream with the synchronization flag. The output of the correlator may be compared with a correlation threshold (the value of which is not specified in this definition). Whenever the output is equal to or greater than the threshold, the receiver should decide that a flag has been detected. For increased reliability in the presence of errors the header field may also be used to assist the synchronization method.

7.2.2 Header Decoding

The decoder used for the header field is not specified in the present document. When the channel can be guaranteed to be error-free, the systematic codeword structure allows for simple extraction of the message bits from the codeword. In the presence of errors, the code may be used to provide either error correction, error detection, or a combination of both moderate error correction capability and error detection capability.

In the presence of errors, the decoding of the frame packet stream in a multiframe is not started until at least two headers have been received in agreement with each other. Multiframes are buffered for decoding until this has occurred. The header block in each received multiframe has its cyclic error correction code decoded and the "common information carrying bits" are extracted. With the header defined in the present document the "common information carrying bits" consist of SampRate, FeType, and EXP1- EXP9 (expansion bits).

NOTE: The use of EXP1 - EXP9 depends on the type of information they may carry in the future. Only those bits which do not change between each multiframe are used in the check of agreement described above).

Once the common information carrying bits have been determined then these are used for all the multiframes in a contiguous sequence of multiframes.

7.2.3 Feature Decompression

Indices are extracted from the frame packet stream, and the CRC is optionally checked. (Back-end handling of frames failing the CRC check is specified in subclause 7.2.4.) Using the indices received, estimates of the front-end features are extracted with a VQ codebook lookup:

$$\begin{bmatrix} \hat{y}_i(m) \\ \hat{y}_{i+1}(m) \end{bmatrix} = Q^{i,i+1}_{idx^{i,i+1}(m)} \qquad i = 0,2,4\ldots12 \tag{7.1}$$

7.2.4 Error Mitigation

7.2.4.1 Detection of frames received with errors

When transmitted over an error prone channel then the received bitstream may contain errors. Two methods are used to determine if a frame pair packet has been received with errors:

- CRC: The CRC recomputed from the indices of the received frame pair packet data does not match the received CRC for the frame pair.
- Data consistency: A heuristic algorithm to determine whether or not the decoded parameters for each of the two speech vectors in a frame pair packet are consistent. The details of this algorithm are described below.

The parameters corresponding to each index, $idx^{i, i+1}$, of the two frames within a frame packet pair are compared to determine if either of the indices are likely to have been received with errors:

$$badindexflag_i = \begin{cases} 1 & if\ \left(y_i(m+1) - y_i(m) \rangle T_i\right)\ OR\ \left(y_{i+1}(m+1) - y_{i+1}(m) \rangle T_{i+1}\right) \\ 0 & otherwise \end{cases} \qquad i = 0, 2, \ldots 12 \qquad (7.2)$$

The thresholds T_i have been determined based on measurements on unerrored speech. A voting algorithm is applied to determine if the whole frame pair packet is to be treated as if it had been received with transmission errors. The frame pair packet is classified as received with error if:

$$\sum_{i=0,2,\ldots 12} badindexflag_i \geq 2 \qquad (7.3)$$

The data consistency check for errored data is only applied when frame pair packets failing the CRC test are detected. It is applied to the frame pair packet received before the one failing the CRC test and successively to frames after one failing the CRC test until one is found that passes the data consistency test. The details of this algorithm are shown in the flow charts of figures 6.1 and 6.2.

7.2.4.2 Substitution of parameter values for frames received with errors

The parameters from the last speech vector received without errors before a sequence of one or more "bad" frame pair packets and those from the first good speech vector received without errors afterwards are used to determine replacement vectors to substitute for those received with errors. If there are B consecutive bad frame pairs (corresponding to 2B speech vectors) then the first B speech vectors are replaced by a copy of the last good speech vector before the error and the last B speech vectors are replaced by a copy of the first good speech vector received after the error.

In the presence of errors, the decoding of the frame packet stream in a multiframe is not started until at least two headers have been received in agreement with each other. Multiframes are buffered for decoding.

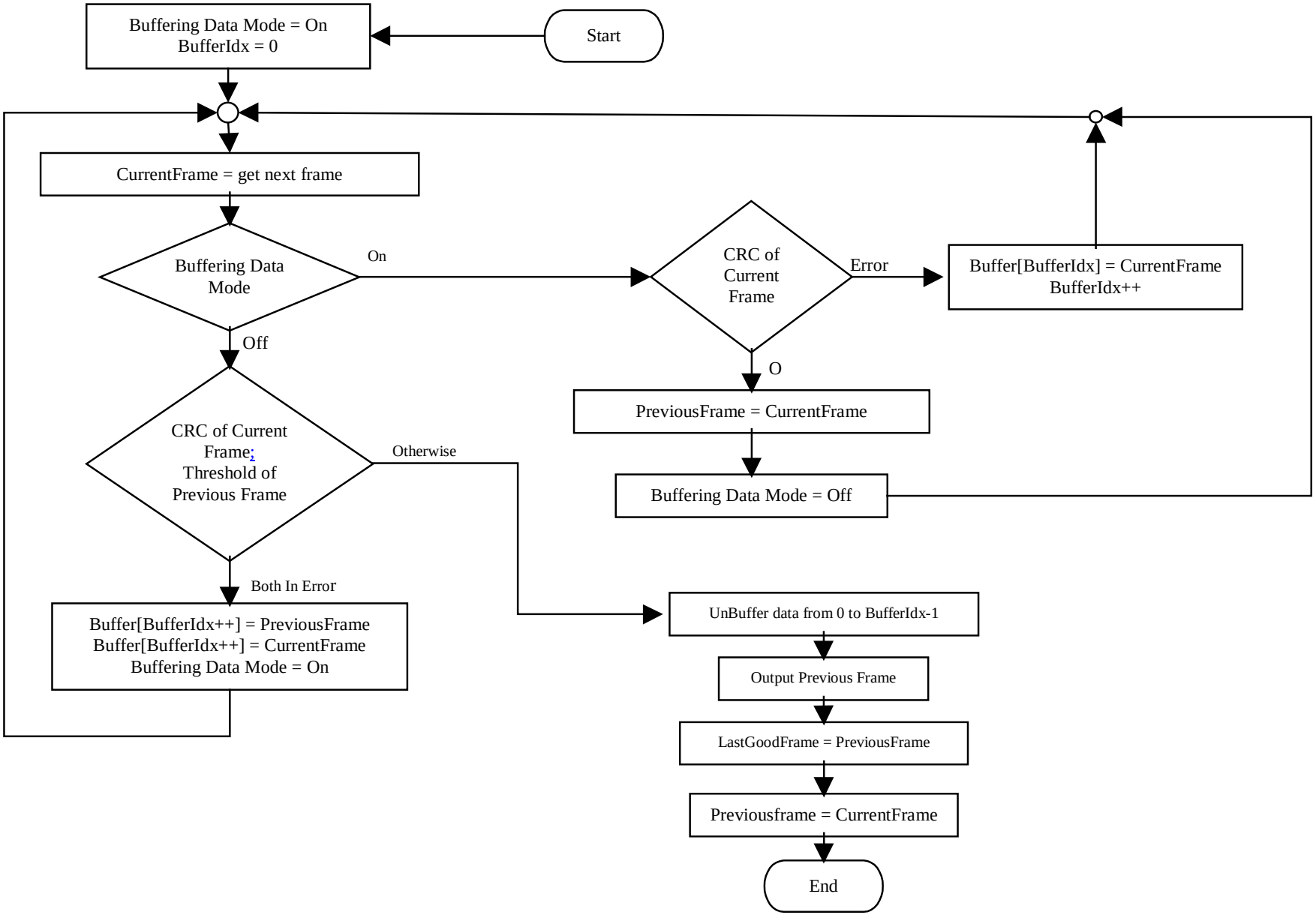


Figure 7.1: Error mitigation initialization flow chart

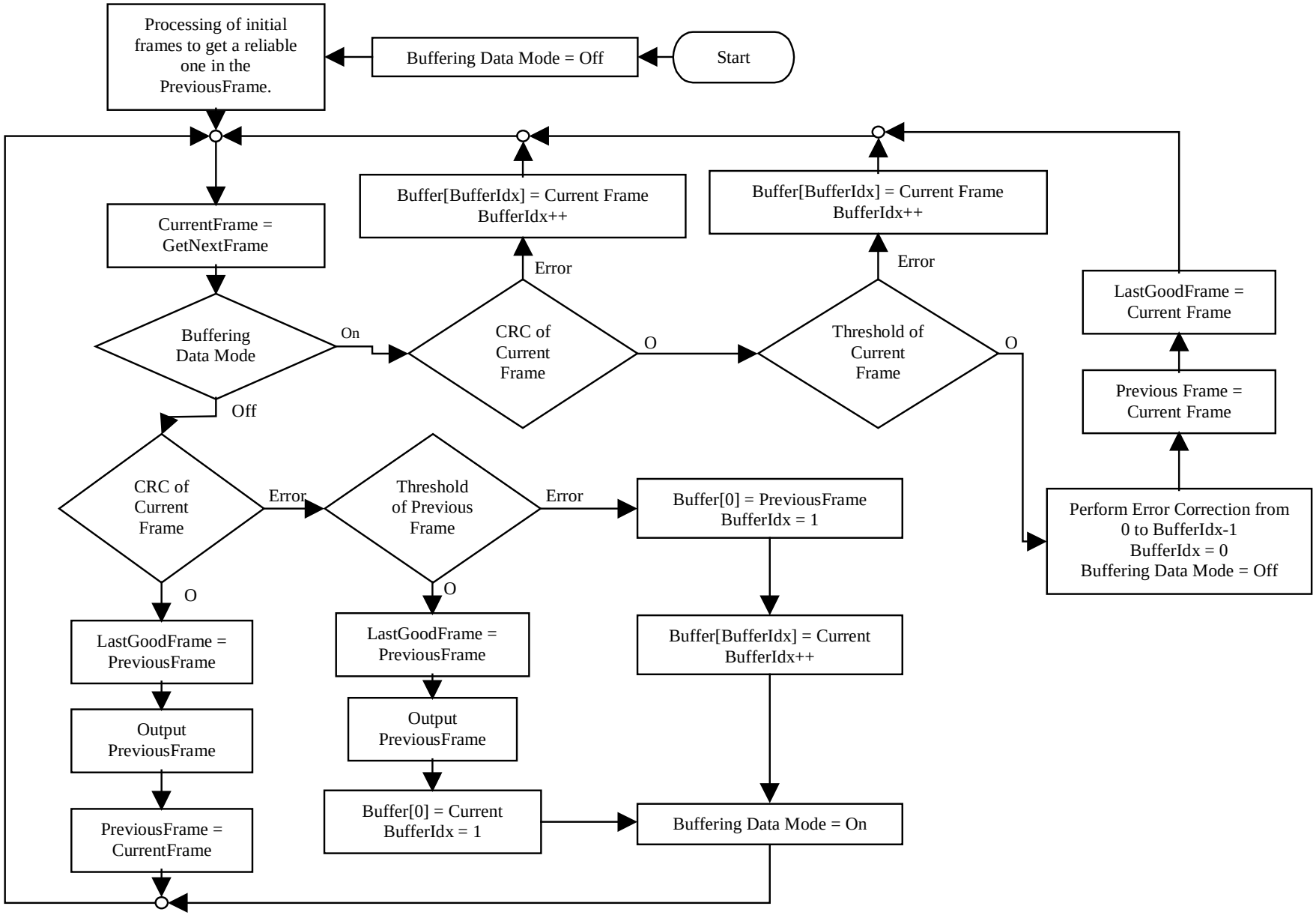


Figure 7.2: Main error mitigation flow chart

History

Document history			
V1.1.1	November 1999	Membership Approval Procedure	MV 200003: 1999-11-23 to 2000-01-21
V1.1.1	February 2000	Publication	