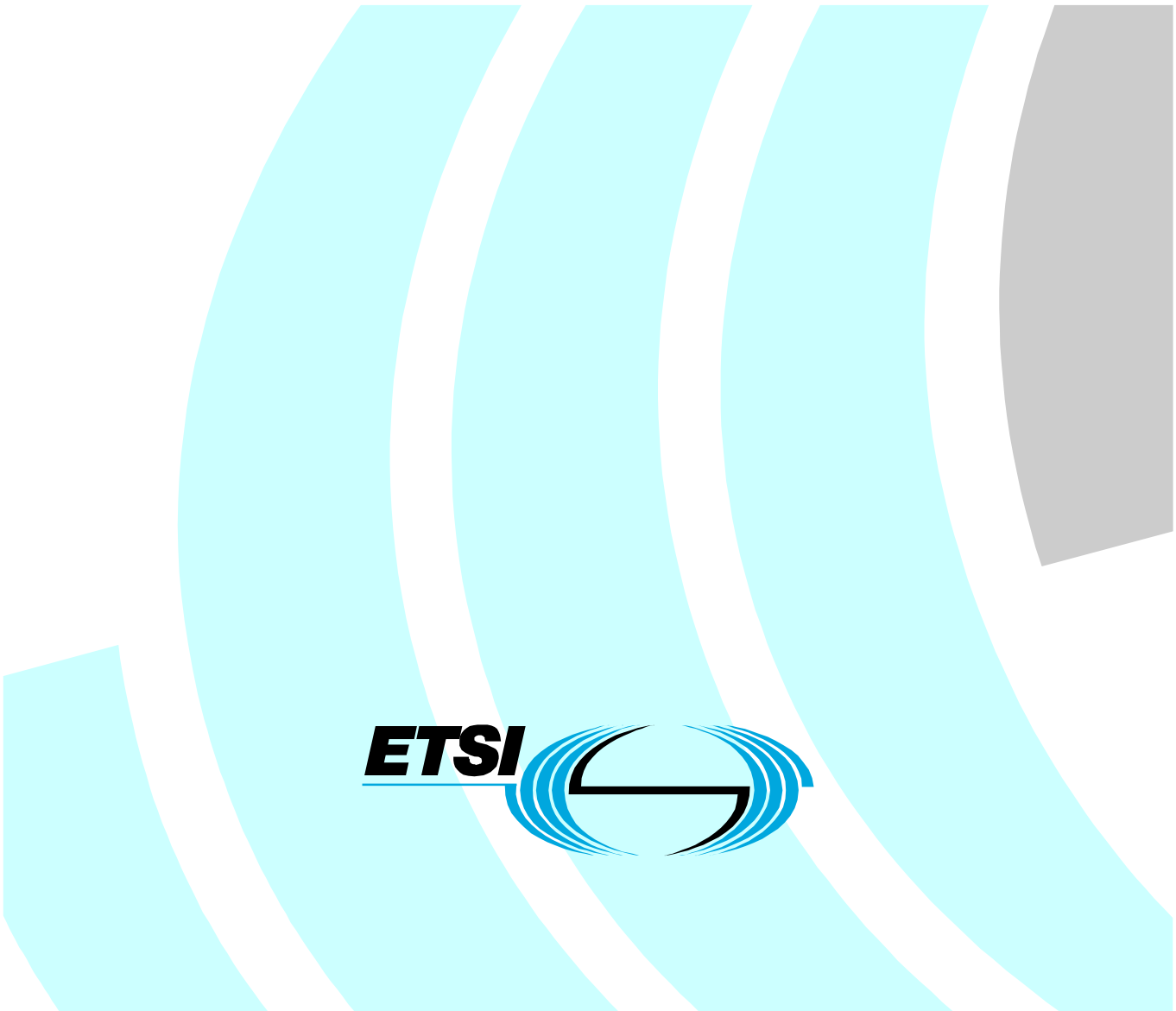
ETSI ES 202 050 V1.1.2 (2003-10)

ETSI Standard

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



Reference
RES/STQ-00041

Keywords
Algorithm, speech

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - 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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, send your comment to:
editor@etsi.org

Copyright Notification

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 ETSI 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://webapp.etsi.org/IPR/home.asp).

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 ETSI 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 degradation in performance due to transcoding on the voice channel is removed and channel invariability is achieved.

The present document presents a standard for a front-end to ensure compatibility between the terminal and the remote recognizer. The first ETSI standard DSR front-end ES 201 108 [1] was published in February 2000 and is based on the Mel-Cepstrum representation that has been used extensively in speech recognition systems. This second standard is for an Advanced DSR front-end that provides substantially improved recognition performance in background noise. Evaluation of the performance during the selection of this standard showed an average of 53 % reduction in speech recognition error rates in noise compared to ES 201 108 [1].

1 Scope

The present document specifies algorithms for advanced 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 advanced 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 advanced 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 advanced front-end features.

The algorithms are defined in a mathematical form or as flow diagrams. Software implementing these algorithms written in the 'C' programming language is contained in the ZIP file es_202050v010102p0.zip which accompanies the present document. 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. In particular, for packet data transmission, it is anticipated that the IETF AVT RTP DSR payload definition (see bibliography) will be used to transport DSR features using the frame pair format described in clause 7.

The Advanced DSR standard is designed for use with discontinuous transmission and to support the transmission of Voice Activity information. Annex A describes a VAD algorithm that is recommended for use in conjunction with the Advanced DSR standard, however it is not part of the present document and manufacturers may choose to use an alternative VAD algorithm.

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 and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

[1] ETSI ES 201 108: "Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms".

[2] ETSI EN 300 903: "Digital cellular telecommunications system (Phase 2+) (GSM); 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

blind equalization: process of compensating the filtering effect that occurs in signal recording

NOTE: In the present document blind equalization is performed in the cepstral domain.

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

discrete cosine transform: process of transforming the log filter-bank 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

NOTE: 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 ES 202 050: 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 frequency 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 predefined frequency

NOTE: 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

power spectral density: squared magnitude spectrum of the signal

pre-emphasis: filtering process in which the frequency response of the filter has emphasis at a given frequency range

NOTE: 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

SNR-dependent Waveform Processing (SWP): processing of signal waveform with objective to emphasize high-SNR waveform portions and de-emphasize low-SNR waveform portions

voice activity detection: process of detecting voice activity in the signal

NOTE: In the present document one voice activity detector is used for noise estimation and a second one is used for non-speech frame dropping.

wiener filtering: filtering of signal by using Wiener filter (filter designed by using Wiener theory).

NOTE: In this work, objective of Wiener filtering is to de-noise signal.

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:

bin	FFT frequency index
$c(i)$	cepstral coefficients; used with appropriate subscript
$E(k)$	filter-bank energy; used with appropriate subscript
$H(bin)$ or $H(k)$	Wiener filter frequency characteristic; used with appropriate subscript
$h(n)$	Wiener filter impulse response; used with appropriate subscript
k	filter-bank band index
K_{FB}	number of bands in filter-bank
lnE	log-compressed energy feature appended to cepstral coefficients
n	waveform signal time index
N	length, (e.g. frame length, FFT length, ...); used with appropriate subscript
$P(bin)$	power spectrum; used with appropriate subscript
$S(k)$	log filter-bank energy; used with appropriate subscript
$s(n)$	waveform signal; used with appropriate subscript
t	frame time index
T_{PSD}	number of frames used in the PSD Mean technique
$w(n)$	windowing function in time domain; used with appropriate subscript
$W(bin)$	frequency window
$X(bin)$	FFT complex output

For compression:

$Idx^{i, i+1}(t)$	codebook index
$N^{i, i+1}$	size of the codebook (compression)
$Q^{i, i+1}$	compression codebook
$q_j^{i, i+1}$	jth codevector in the codebook $Q^{i, i+1}$
$y(t)$	feature vector with 14 components

3.3 Abbreviations

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

ADC	Analog-to-Digital Conversion
AVT	Audio/Video Transport
CC	Cepstral Computation
CRC	Cyclic Redundancy Code
DCT	Discrete Cosine Transform
DSR	Distributed Speech Recognition

FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FVS	Feature Vector Selection
HFB	High Frequency Band
IDCT	Inverse Discrete Cosine Transform
IETF	Internet Engineering Task Force
LFB	Low Frequency Band
LSB	Least Significant Bit
MEL-FB	MEL Filter Bank
MSB	Most Significant Bit
NR	Noise Reduction
PSD	Power Spectral Density
QMF	Quadrature-Mirror Filters
RTP	Real Time Protocol
SNR	Signal to Noise Ratio
SWP	SNR-dependent Waveform Processing
VAD	Voice Activity Detection (used for non-speech frame dropping)
VADNest	Voice Activity Detection (used for Noise estimation)
VQ	Vector Quantizer

4 System overview

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 kHz, 11 kHz 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 characteristics as specified in EN 300 903 [2]. DSR terminal developers should be aware that reduced recognition performance may be obtained if they operate outside the recommended tolerances.

Figure 4.1 shows the block scheme of the proposed front-end and its implementation in both the terminal and server sides. In the terminal part, which is shown in figure 4.1(a), speech features are computed from the input signal in the Feature Extraction part. Then, features are compressed and further processed for channel transmission.

In the Feature Extraction part, noise reduction is performed first. Then, waveform processing is applied to the de-noised signal and cepstral features are calculated. At the end, blind equalization is applied to the cepstral features. The Feature Extraction part also contains an 11 and 16 kHz extension block for handling these two sampling frequencies. Voice activity detection (VAD) for the non-speech frame dropping is also implemented in Feature Extraction.

At the server side (see figure 4.1(b)), bit-stream decoding, error mitigation and decompression are applied. Before entering the back-end, an additional server feature processing is performed. All blocks of the proposed front-end are described in detail in the following clauses.

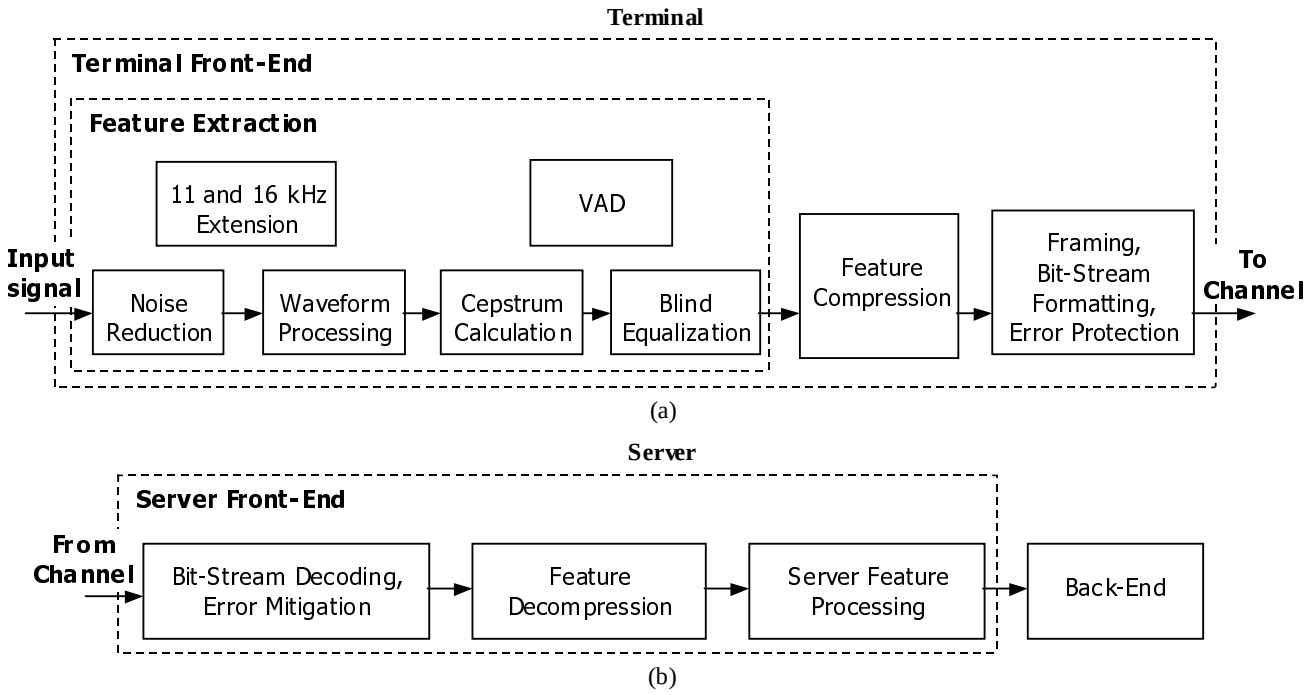


Figure 4.1: Block scheme of the proposed front-end. Figure (a) shows blocks implemented at the terminal side and (b) shows blocks implemented at the server side

5 Feature Extraction Description

5.1 Noise Reduction

5.1.1 Two stage mel-warped Wiener filter approach

Noise reduction is based on Wiener filter theory and it is performed in two stages. Figure 5.1 shows the main components of the Noise Reduction block of the proposed front-end. The input signal is first de-noised in the first stage and the output of the first stage then enters the second stage. In the second stage, an additional, dynamic noise reduction is performed, which is dependent on the signal-to-noise ratio (SNR) of the processed signal.

Noise reduction is performed on a frame-by-frame basis. After framing the input signal, the linear spectrum of each frame is estimated in the Spectrum Estimation block. In PSD Mean block (Power Spectral Density), the signal spectrum is smoothed along the time (frame) index. Then, in the WF Design block, frequency domain Wiener filter coefficients are calculated by using both the current frame spectrum estimation and the noise spectrum estimation. The noise spectrum is estimated from noise frames, which are detected by a voice activity detector (VADNest). Linear Wiener filter coefficients are further smoothed along the frequency axis by using a Mel Filter-Bank, resulting in a Mel-warped frequency domain Wiener filter. The impulse response of this Mel-warped Wiener filter is obtained by applying a Mel IDCT (Mel-warped Inverse Discrete Cosine Transform). Finally, the input signal of each stage is filtered in the Apply Filter block. Notice from figure 5.1 that the input signal to the second stage is the output signal from the first stage. At the end of Noise Reduction, the DC offset of the noise-reduced signal is removed in the OFF block.

Additionally, in the second stage, the aggression of noise reduction is controlled by Gain Factorization block.

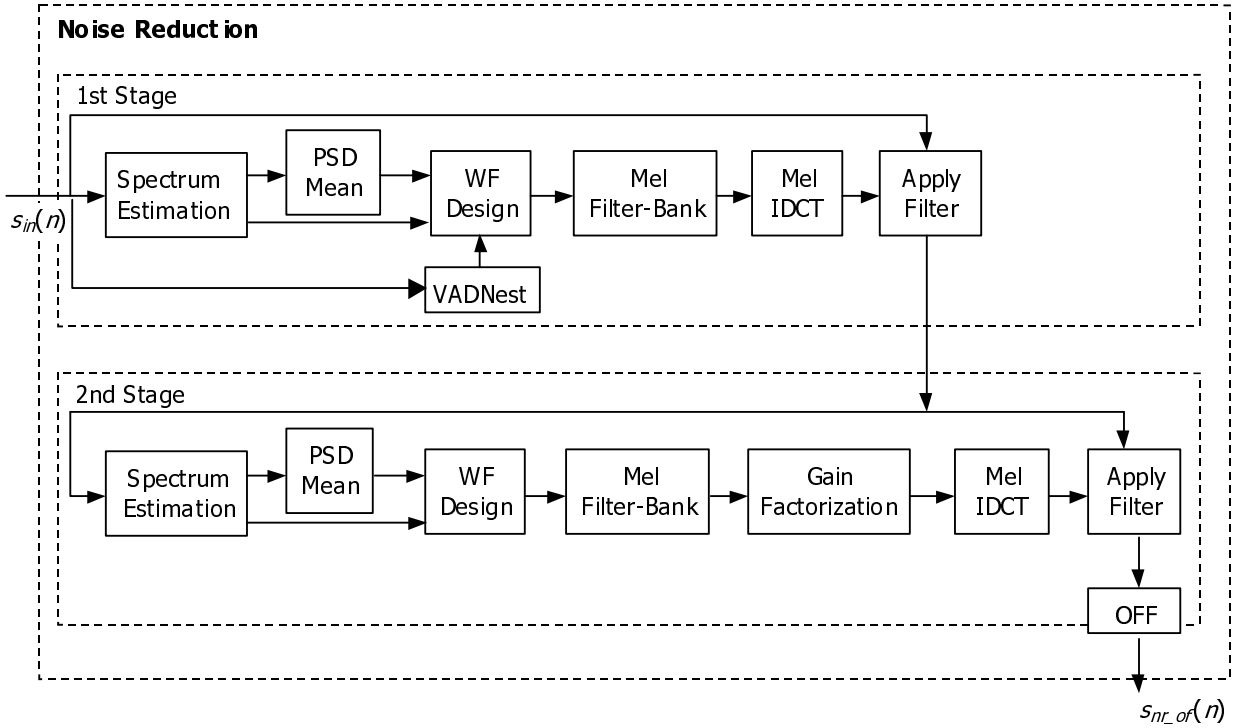


Figure 5.1: Block scheme of noise reduction

5.1.2 Buffering

The input of the noise reduction block is a 80-sample frame. A 4-frame (frame 0 to frame 3) buffer is used for each stage of the noise reduction. At each new input frame, the 2 buffers are shifted by one frame. The new input frame becomes frame 3 of the first buffer. Then the frame 1 (from position 80 to position 159 in the buffer) of the first buffer is denoised and this denoised frame becomes frame 3 of the second buffer. The frame 1 of the second buffer is denoised and this denoised frame is the output of the noise reduction block. Hence at each stage of the noise reduction block, there is a latency of 2 frames (20 ms). For each stage of the noise reduction block, the spectrum estimation is performed on the window which starts at position 60 and ends at position 259.

5.1.3 Spectrum estimation

Input signal is divided into overlapping frames of N_{in} samples. 25ms (N_{in}=200) frame length and 10ms (80 samples) frame shift are used. Each frame $s_{in}(n)$ is windowed by a Hanning window of length N_{in}, $w_{Hann}(n)$, like

$$s_w(n) = s_{in}(n) \times w_{Hann}(n), \quad 0 \le n \le N_{in} - 1 \tag{5.1}$$

where

$$w_{Hann}(n) = 0{,}5 - 0 \times 5 \times \cos\left(\frac{2 \times \pi \times (n + 0{,}5)}{N_{in}}\right) \tag{5.2}$$

Then, zeros are padded from the sample N_{in} up to the sample N_{FFT}-1, where N_{FFT}=256 is the fast Fourier transform (FFT) length:

$$s_{FFT}(n) = \begin{cases} s_w(n), & 0 \le n \le N_{in} - 1 \\ 0, & N_{in} \le n \le N_{FFT} - 1 \end{cases} \tag{5.3}$$

To get the frequency representation of each frame, the FFT is applied to $s_{FFT}(n)$ like:

$$X(bin) = FFT\{s_{FFT}(n)\} \tag{5.4}$$

where *bin* denotes the FFT frequency index.

The power spectrum of each frame, $P(bin)$ $0 \le bin \le N_{FFT}/2$ is computed by applying the power of 2 function to the FFT bins:

$$P(bin) = |X(bin)|^2, \quad 0 \le bin \le N_{FFT}/2 \tag{5.5}$$

The power spectrum $P(bin)$ is smoothed like

$$P_{in}(bin) = \frac{P(2 \times bin) + P(2 \times bin + 1)}{2}, \quad 0 \le bin < N_{FFT}/4 \tag{5.6}$$

$$P_{in}(N_{FFT}/4) = P(N_{FFT}/2)$$

By this smoothing operation, the length of the power spectrum is reduced to $N_{SPEC} = N_{FFT}/4 + 1$

5.1.4 Power spectral density mean

This module computes for each power spectrum bin $P_{in}(bin)$ the mean over the last T_{PSD} frames.

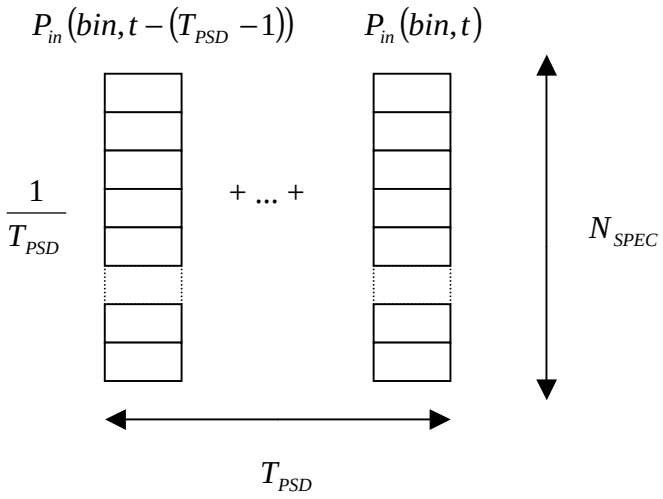


Figure 5.2: Mean computation over the last T_{PSD} frames as performed in PSD mean

Power spectral density mean (PSD mean) is calculated as

$$P_{in\_PSD}(bin, t) = \frac{1}{T_{PSD}} \sum_{i=0}^{T_{PSD}-1} P_{in}(bin, t-i), \quad \text{for} \quad 0 \le bin \le N_{SPEC} - 1 \tag{5.7}$$

where the chosen value for T_{PSD} is 2 and *t* is frame (time) index. Note that throughout this document, we use frame index *t* only if it is necessary for explanation. If the frame index is dropped, current frame is referred.

5.1.5 Wiener filter design

A forgetting factor $lambdaNSE$ (used in the update of the noise spectrum estimate in first stage of noise reduction) is computed for each frame depending on the frame time index t:

$$\begin{aligned} &if\ (t < NB\_FRAME\_THRESHOLD\_NSE) \\ &then \\ &\quad lambdaNSE = 1 - 1/t \\ &else \\ &\quad lambdaNSE = LAMBDA\_NSE \end{aligned} \tag{5.8}$$

where $NB\_FRAME\_THRESHOLD\_NSE$ equals 100 and $LAMBDA\_NSE$ equals 0,99.

In first stage the noise spectrum estimate is updated according to the following equation, dependent on the $flagVAD_{Nest}$ from VADNest:

$$\begin{cases} P_{noise}^{1/2}(bin,t_n) = max\left(lambdaNSE \times P_{noise}^{1/2}(bin,t_n - 1) + (1 - lambdaNSE) \times P_{in\_PSD}^{1/2}(bin,t_n), EPS\right) \\ P_{noise}^{1/2}(bin,t) = P_{noise}^{1/2}(bin,t_n) \end{cases} \tag{5.9}$$

where EPS equals exp(–10,0), t represents the current frame index, t_n represents the index of the last non-speech frame and $P_{in\_PSD}(bin,t)$ is the output of the PSD Mean module. $P_{noise}^{1/2}(bin,-1)$ is initialized to EPS.

In the second stage the noise spectrum estimate is updated permanently according to the following equation:

$$\begin{aligned} &if\ (t < 11) \\ &then \\ &\quad lambdaNSE = 1 - 1/t \\ &\quad P_{noise}(bin,t) = lambdaNSE \times P_{noise}(bin,t-1) + (1 - lambdaNSE) \times P_{in\_PSD}(bin,t) \\ &else \\ &\quad upDate = 0,9 + 0,1 \times P_{in\_PSD}(bin,t) \Big/ \left(P_{in\_PSD}(bin,t) + P_{noise}(bin,t-1)\right) \\ &\qquad \times \left(1 + 1 \Big/ \left(1 + 0,1 \times P_{in\_PSD}(bin,t) / P_{noise}(bin,t-1)\right)\right) \\ &\quad P_{noise}(bin,t) = P_{noise}(bin,t-1) \times upDate \\ &if\ \left(P_{noise}^{1/2}(bin,t) < EPS\right) \\ &then \\ &\quad P_{noise}^{1/2}(bin,t) = EPS \end{aligned} \tag{5.10}$$

Then the noiseless signal spectrum is estimated using a "decision-directed" approach:

$$P_{den}^{1/2}(bin,t) = BETA \times P_{den3}^{1/2}(bin,t-1) + (1 - BETA) \times T\left[P_{in\_PSD}^{1/2}(bin,t) - P_{noise}^{1/2}(bin,t)\right] \tag{5.11}$$

$P_{den}^{1/2}(bin,-1)$ is initialized to 0, $BETA$ equals 0,98 and the threshold function T is given by:

$$T\left[z(bin,t)\right] = \begin{cases} z(bin,t) & \text{if } z(bin,t) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

Then the a priori SNR $\eta(bin,t)$ is computed as:

$$\eta(bin,t) = \frac{P_{den}(bin,t)}{P_{noise}(bin,t)} \tag{5.13}$$

The filter transfer function $H(bin,t)$ is obtained according to the following equation:

$$H(bin,t) = \frac{\sqrt{\eta(bin,t)}}{1+\sqrt{\eta(bin,t)}} \tag{5.14}$$

The filter transfer function $H(bin,t)$ is used to improve the estimation of the noiseless signal spectrum:

$$P_{den2}^{1/2}(bin,t) = H(bin,t)\,P_{in\_PSD}^{1/2}(bin,t) \tag{5.15}$$

Then an improved a priori SNR $\eta_2(bin,t)$ is obtained:

$$\eta_2(bin,t) = \max\left(\frac{P_{den2}(bin,t)}{P_{noise}(bin,t)}, {\eta_{TH}}^2\right) \tag{5.16}$$

where η_{TH} equals 0,079 432 823 (value corresponding to a SNR of -22 dB).

The improved transfer function $H_2(bin,t)$ is then obtained according to the following equation:

$$H_2(bin,t) = \frac{\sqrt{\eta_2(bin,t)}}{1+\sqrt{\eta_2(bin,t)}},\ 0 \le bin \le N_{SPEC} - 1 \tag{5.17}$$

The improved transfer function $H_2(bin,t)$ is then used to calculate the noiseless signal spectrum $P_{den3}^{1/2}(bin,t)$ that will be used for the next frame in Equation (5.11):

$$P_{den3}^{1/2}(bin,t) = H_2(bin,t)\,P_{in}^{1/2}(bin,t) \tag{5.18}$$

5.1.6 VAD for noise estimation (VADNest)

A forgetting factor $lambdaLTE$ (used in the update of the long-term energy) is computed for each frame using the frame time index t:

$$\begin{aligned} &if\ (t < NB\_FRAME\_THRESHOLD\_LTE)\\ &then\\ &\quad lambdaLTE = 1 - 1/t\\ &else\\ &\quad lambdaLTE = LAMBDA\_LTE \end{aligned} \tag{5.19}$$

where $NB\_FRAME\_THRESHOLD\_LTE$ equals 10 and $LAMBDA\_LTE$ equals 0,97.

Then the logarithmic energy $frameEn$ of the M (M = 80) last samples of the input signal $s_{in}(n)$ is computed:

$$frameEn = 0,5 + \frac{16}{\ln 2} \times \ln\left(\frac{\left(64 + \sum_{i=0}^{M-1} s_{in}(n)^2\right)}{64}\right) \tag{5.20}$$

Then $frameEn$ is used in the update of $meanEn$:

$$
\begin{aligned}
&if \left(\begin{array}{l} \left(\left(frameEn - meanEn \right) < SNR\_THRESHOLD\_UPD\_LTE \right) \\ OR \\ \left(t < MIN\_FRAME \right) \end{array} \right) \\
&then \\
&\quad if \left(\left(frameEn < meanEN \right) OR \left(t < MIN\_FRAME \right) \right) \\
&\quad then \\
&\qquad meanEn = meanEn + \left(1 - lambdaLTE \right) \times \left(frameEn - meanEn \right) \\
&\quad else \\
&\qquad meanEn = meanEn + \left(1 - lambdaLTEhigherE \right) \times \left(frameEn - meanEn \right) \\
&\quad if \left(meanEn < ENERGY\_FLOOR \right) then\ meanEn = ENERGY\_FLOOR
\end{aligned}
\qquad (5.21)
$$

where $SNR\_THRESHOLD\_UPD\_LTE$ equals 20, $ENERGY\_FLOOR$ equals 80, MIN_FRAME equals 10 and $lambdaLTEhigherE$ equals 0,99.

Then $frameEn$ and $meanEn$ are used to decide if the current frame is speech ($flagVAD_{Nest} = 1$) or not ($flagVAD_{Nest} = 0$):

$$
\begin{aligned}
&if \left(t > 4 \right) \\
&then \\
&\quad if \left(\left(frameEn - meanEn \right) > SNR\_THRESHOLD\_VAD \right) \\
&\quad then \\
&\qquad flagVAD_{Nest} = 1 \\
&\qquad nbSpeechFrame = nbSpeechFrame + 1 \\
&\quad else \\
&\qquad if \left(nbSpeechFrame > MIN\_SPEECH\_FRAME\_HANGOVER \right) \\
&\qquad then \\
&\qquad\qquad hangOver = HANGOVER \\
&\qquad nbSpeechFrame = 0 \\
&\qquad if \left(hangOver != 0 \right) \\
&\qquad then \\
&\qquad\qquad hangOver = hangOver - 1 \\
&\qquad\qquad flagVAD_{Nest} = 1 \\
&\qquad else \\
&\qquad\qquad flagVAD_{Nest} = 0
\end{aligned}
\qquad (5.22)
$$

where $SNR\_THRESHOLD\_VAD$ equals 15, $MIN\_SPEECH\_FRAME\_HANGOVER$ equals 4 and $HANGOVER$ equals 15.

$nbSpeechFrame$, $meanEn$, $flagVAD_{Nest}$ and $hangOver$ are initialized to 0. The frame time index t is initialised to 0 and is incremented each frame by 1 so that it equals 1 for the first frame processed.

5.1.7 Mel filter-bank

The linear-frequency Wiener filter coefficients $H_2(bin),\ 0 \le bin \le N_{SPEC} - 1,$ (computed by formula (5.17)) are smoothed and transformed to the Mel-frequency scale. Mel-warped Wiener filter coefficients $H_{2\_mel}(k)$ are estimated by using triangular-shaped, half-overlapped frequency windows applied on $H_2(bin)$. To obtain the central frequencies of FB bands in terms of FFT bin indices, $bin_{centr}(k)$, the linear frequency scale f_{lin} was transformed to mel scale by using the following formula:

$$MEL\{f_{lin}\} = 2\,595 \times \log_{10}(1 + f_{lin}/700) \tag{5.23}$$

Then, the central frequency of the *k*-th band, $f_{centr}(k)$, is calculated as

$$f_{centr}(k) = 700 \times \left(10^{\frac{f_{mel}(k)}{2595}} - 1\right), \quad \text{for} \, 1 \le k \le K_{FB} \tag{5.24}$$

with K_{FB} = 23 and

$$f_{mel}(k) = k \times \frac{MEL\{f_{lin\_samp}/2\}}{K_{FB} + 1} \tag{5.25}$$

where $f_{lin\_samp} = 8\,000$ is the sampling frequency. Additionally, two marginal FB bands with central frequencies $f_{centr}(0) = 0$ and $f_{centr}(K_{FB} + 1) = f_{lin\_samp}/2$ are added to the K_{FB} = 23 Mel FB bands for purposes of following DCT transformation to the time domain; thus, in total we calculate K_{FB} + 2 = 25 Mel-warped Wiener filter coefficients. The FFT bin index corresponding to central frequencies is obtained as

$$bin_{centr}(k) = round\left(\frac{f_{centr}(k)}{f_{lin\_samp}} \times 2 \times (N_{SPEC} - 1)\right) \tag{5.26}$$

Frequency windows $W(k,i)$ for $1 \le k \le K_{FB}$ are calculated as

$$W(k,i) = \frac{i - bin_{centr}(k-1)}{bin_{centr}(k) - bin_{centr}(k-1)}, \quad \text{for} \quad bin_{centr}(k-1) + 1 \le i \le bin_{centr}(k) \tag{5.27a}$$

$$W(k,i) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k)}, \quad \text{for} \quad bin_{centr}(k) + 1 \le i \le bin_{centr}(k+1) \tag{5.27b}$$

and $W(k,i) = 0$ for other *i*. For $k = 0$

$$W(0,i) = 1 - \frac{i}{bin_{centr}(1) - bin_{centr}(0)}, \quad \text{for} \ \ 0 \le i \le bin_{centr}(1) - bin_{centr}(0) - 1 \tag{5.27c}$$

and $W(0,i) = 0$ for other *i*. For $k = K_{FB} + 1$

$$W(K_{FB}+1,i) = \frac{i - bin_{centr}(K_{FB})}{bin_{centr}(K_{FB}+1) - bin_{centr}(K_{FB})}, \ \text{for} \ bin_{centr}(K_{FB}) + 1 \le i \le bin_{centr}(K_{FB}+1) \tag{5.27d}$$

and $W(K_{FB}+1,i)=0$ for other *i*. Mel-warped Wiener filter coefficients $H_{2\_mel}(k)$ for $0 \le k \le K_{FB}+1$ are computed as:

$$H_{2\_mel}(k) = \frac{1}{\sum_{i=0}^{N_{SPEC}-1} W(k,i)} \sum_{i=0}^{N_{SPEC}-1} W(k,i) \times H_2(i) \tag{5.28}$$

5.1.8 Gain factorization

In this block, factorization of the Wiener filter Mel-warped coefficients (or gains), $H_{2\_mel}(k)$, is performed to control the aggression of noise reduction in the second stage.

In the first stage, de-noised frame signal energy $E_{den}(t)$, where *t* is frame index starting with 1, is calculated by using the de-noised power spectrum $P_{den3}(bin,t)$ computed by (5.18) as

$nbSpeechFrame$, $meanEn$, $flagVAD_{Nest}$ and $hangOver$ are initialized to 0.

$$E_{den}(t) = \sum_{bin=0}^{N_{SPEC}-1} P_{den3}^{1/2}(bin,t) \tag{5.29}$$

In the second stage, the noise energy at the current frame index *t* is estimated by using the noise power spectrum $P_{noise}(bin,t)$ as

$$E_{noise}(t) = \sum_{bin=0}^{N_{SPEC}-1} P_{noise}^{1/2}(bin,t) \tag{5.30}$$

Then, smoothed SNR is evaluated by using three de-noised frame energies (notice there is two frames delay between the first and the second stage) and noise energy like

$$\begin{aligned} &Ratio = \frac{E_{den}(t-2) \times E_{den}(t-1) \times E_{den}(t)}{E_{noise}(t) \times E_{noise}(t) \times E_{noise}(t)} \\ &if\ (Ratio > 0{,}0001) \\ &then \\ &\qquad SNR_{aver}(t) = 20/3 \times \log_{10}(Ratio) \\ &else \\ &\qquad SNR_{aver}(t) = -100/3 \end{aligned} \tag{5.31}$$

To decide the degree of aggression of the second stage Wiener filter for each frame, the low SNR level is tracked by using the following logic:

$$\begin{aligned} &if\ \{(SNR_{aver}(t) - SNR_{low\_track}(t-1)) < 10 \quad \text{or} \quad t < 10\} \\ &\qquad \text{calculate}\ \ \lambda_{SNR}(t) \\ &\qquad SNR_{low\_track}(t) = \lambda_{SNR}(t) \times SNR_{low\_track}(t-1) + (1-\lambda_{SNR}(t)) \times SNR_{aver}(t) \\ &else \\ &\qquad SNR_{low\_track}(t) = SNR_{low\_track}(t-1) \end{aligned} \tag{5.32}$$

with SNR_{low_track} initialized to zero. The forgetting factor $\lambda_{SNR}(t)$ is calculated by the following logic:

$$
\begin{aligned}
&if \quad \{t < 10\} \\
&\qquad \lambda_{SNR}(t) = 1 - 1/t \\
&else \\
&\qquad if \quad \{SNR_{aver}(t) < SNR_{low\_track}(t)\} \\
&\qquad\qquad \lambda_{SNR}(t) = 0{,}95 \\
&\qquad else \\
&\qquad\qquad \lambda_{SNR}(t) = 0{,}99
\end{aligned}
\tag{5.33}
$$

The intention of gain factorization is to apply more aggressive noise reduction to purely noisy frames and less aggressive noise reduction to frames also containing speech. At this point, the current SNR estimation, $SNR_{aver}(t)$ is compared to the low SNR tracked value, $SNR_{low\_track}(t)$, and the Wiener filter gain factorization coefficient $\alpha_{GF}(t)$ is updated. This is done by the following logic:

$$
\begin{aligned}
&if\,(E_{den}(t) > 100) \\
&then \\
&\qquad if \quad \{SNR_{aver}(t) < (SNR_{low\_track}(t) + 3{,}5)\} \\
&\qquad\qquad \alpha_{GF}(t) = \alpha_{GF}(t-1) + 0{,}15 \\
&\qquad\qquad if \quad \{\alpha_{GF}(t) > 0{,}8\} \\
&\qquad\qquad\qquad \alpha_{GF}(t) = 0{,}8 \\
&\qquad else \\
&\qquad\qquad \alpha_{GF}(t) = \alpha_{GF}(t-1) - 0{,}3 \\
&\qquad\qquad if \quad \{\alpha_{GF}(t) < 0{,}1\} \\
&\qquad\qquad\qquad \alpha_{GF}(t) = 0{,}1
\end{aligned}
\tag{5.34}
$$

with $\alpha_{GF}(0) = 0{,}8$.

The second stage Wiener filter gains are multiplied by $\alpha_{GF}(t)$ like

$$H_{2\_mel\_GF}(k,t) = (1 - \alpha_{GF}(t)) + \alpha_{GF}(t) \times H_{2\_mel}(k,t), \quad 0 \le k \le K_{FB} + 1 \tag{5.35}$$

The coefficient $\alpha_{GF}(t)$ takes values from 0,1 to 0,8, which means that the aggression of the second stage Wiener filter is reduced to 10 % for speech + noise frames and to 80 % for noise frames.

5.1.9 Mel IDCT

The time-domain impulse response of Wiener filter $h_{WF}(n)$ is computed from the Mel Wiener filter coefficients $H_{2\_mel}(k)$ from clause 5.1.6 (in the second stage, $H_{2\_mel\_GF}(k)$ from equation (5.35)) by using Mel-warped inverse DCT:

$$h_{WF}(n) = \sum_{k=0}^{K_{FB}+1} H_{2\_mel}(k) \times IDCT_{mel}(k,n), \qquad 0 \le n \le K_{FB} + 1 \tag{5.36}$$

where $IDCT_{mel}(k,n)$ are Mel-warped inverse DCT basis computed as follows.

First, central frequencies of each band are computed for $1 \le k \le K_{FB}$ like:

$$f_{centr}(k) = \frac{1}{\sum_{i=0}^{N_{SPEC}-1} W(k,i)} \sum_{i=0}^{N_{SPEC}-1} W(k,i) \times i \times \frac{f_{samp}}{2 \times (N_{SPEC} - 1)} \quad (5.37)$$

where f_{samp} = 8 000 is sampling frequency. $f_{centr}(0) = 0$ and $f_{centr}(K_{FB} + 1) = f_{samp} / 2$. Then, Mel-warped inverse DCT basis are obtained as

$$IDCT_{mel}(k,n) = \cos\left(\frac{2 \times \pi \times n \times f_{centr}(k)}{f_{samp}}\right) \times df(k), \quad 0 \le k \le K_{FB} + 1, \ 0 \le n \le K_{FB} + 1 \quad (5.38)$$

where $f_{centr}(k)$ is central frequency corresponding to the Mel FB index k and $df(k)$ is computed like

$$df(k) = \frac{f_{centr}(k+1) - f_{centr}(k-1)}{f_{samp}}, \quad 1 \le k \le K_{FB} \quad (5.39)$$

$$df(0) = \frac{f_{centr}(1) - f_{centr}(0)}{f_{samp}} \quad \text{and} \quad df(K_{FB}+1) = \frac{f_{centr}(K_{FB}+1) - f_{centr}(K_{FB})}{f_{samp}}$$

The impulse response of Wiener filter is mirrored as

$$h_{WF\_mirr}(n) = \begin{cases} h_{WF}(n), & 0 \le n \le K_{FB} + 1 \\ h_{WF}(2 \times (K_{FB} + 1) + 1 - n), & K_{FB} + 2 \le n \le 2 \times (K_{FB} + 1) \end{cases} \quad (5.40)$$

5.1.10 Apply filter

The causal impulse response $h_{WF\_caus}(n,t)$ is obtained from $h_{WF\_mirr}(n,t)$ according to the following relations:

$$\begin{cases} h_{WF\_caus}(n,t) = h_{WF\_mirr}(n + K_{FB} + 1, t), & n = 0, \cdots, K_{FB} \\ h_{WF\_caus}(n,t) = h_{WF\_mirr}(n - K_{FB} - 1, t), & n = K_{FB} + 1, \cdots, 2 \times (K_{FB} + 1) \end{cases} \quad (5.41)$$

The causal impulse response $h_{WF\_caus}(n,t)$ is then truncated giving $h_{WF\_trunc}(n,t)$:

$$h_{WF\_trunc}(n,t) = h_{WF\_caus}\left(n + K_{FB} + 1 - (FL-1)/2, t\right), \quad n = 0, \cdots, FL - 1 \quad (5.42)$$

where the filter length FL equals 17.

The truncated impulse response is weighted by a Hanning window:

$$h_{WF\_w}(n,t) = \left\{0{,}5 - 0{,}5 \times \cos\left(\frac{2 \times \pi \times (n + 0{,}5)}{FL}\right)\right\} \times h_{WF\_trunc}(n,t), \ 0 \le n \le FL - 1 \quad (5.43)$$

Then the input signal s_{in} is filtered with the filter impulse response $h_{WF\_w}(n,t)$ to produce the noise-reduced signal s_{nr}:

$$s_{nr}(n) = \sum_{i=-(FL-1)/2}^{(FL-1)/2} h_{WF\_w}\left(i+(FL-1)/2\right) \times s_{in}(n-i),\ 0 \le n \le M-1 \tag{5.44}$$

where the filter length FL equals 17 and the frame shift interval M equals 80.

5.1.11 Offset compensation

To remove the DC offset, a notch filtering operation is applied to the noise-reduced signal like:

$$s_{nr\_of}(n) = s_{nr}(n) - s_{nr}(n-1) + (1-1/1024) \times s_{nr\_of}(n-1),\ \ 0 \le n \le M-1 \tag{5.45}$$

where $s_{nr}(-1)$ and $s_{nr\_of}(-1)$ correspond to the last samples of the previous frame and equal 0 for the first frame, and $M = 80$ is the frame shift interval.

5.2 Waveform Processing

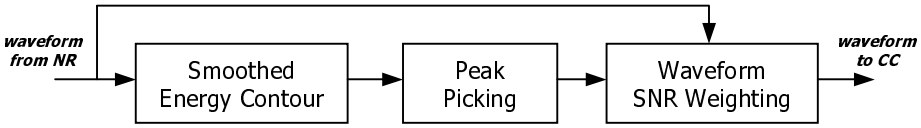


Figure 5.3: Main components of SNR-dependent waveform processing

SNR-dependent Waveform Processing (SWP) is applied to the noise reduced waveform that comes out from the Noise Reduction (NR) block. The noise reduction block outputs 80-sample frames that are stored in a 240-sample buffer (from sample 0 to sample 239). The waveform processing block is applied on the window that starts at sample 1 and ends at sample 200. Figure 5.3 describes the basic components of SWP. In the Smoothed Energy Contour block, the instant energy contour is computed for each input frame by using the Teager operator like

$$E_{Teag}(n) = \left| s_{nr\_of}^2(n) - s_{nr\_of}(n-1) \times s_{nr\_of}(n+1) \right|,\ \ 1 \le n < N_{in}-1 \tag{5.46a}$$

$$E_{Teag}(0) = \left| s_{nr\_of}^2(0) - s_{nr\_of}(0) \times s_{nr\_of}(1) \right| \tag{5.46b}$$

and

$$E_{Teag}(N_{in}-1) = \left| s_{nr\_of}^2(N_{in}-1) - s_{nr\_of}(N_{in}-2) \times s_{nr\_of}(N_{in}-1) \right| \tag{5.46c}$$

The energy contour is smoothed by using a simple FIR filter of length 9 like

$$E_{Teag\_Smooth}(n) = \frac{1}{9} \sum_{i=-4}^{4} E_{Teag}(n+i) \tag{5.47}$$

At the beginning or ending edge of $E_{Teag}(n)$, the $E_{Teag}(0)$ or $E_{Teag}(N_{in}\text{-}1)$ value is repeated, respectively.

In the Peak Picking block, maxima in the smoothed energy contour related to the fundamental frequency are found. First, the global maximum over the entire energy contour $E_{Teag\_Smooth}(n),\ 0 \le n \le N_{in}-1$, is found. Then, maxima on both left and right sides of the global maximum are identified. Each maximum is expected to be between 25 and 80 samples away from its neighbour.

In the block Waveform SNR Weighting, a weighting function is applied to the input frame. Having the number of maxima N_{MAX} of the smoothed energy contour $E_{Taeg\_Smooth}(n)$ and their positions $pos_{MAX}(n_{MAX}),\ 0 \le n_{MAX} < N_{MAX}$, a weighting function $w_{swp}(n)$ of length N_{in} is constructed, which equals 1,0 for *n* from intervals

$$\langle [pos_{MAX}(n_{MAX}) - 4],\ [pos_{MAX}(n_{MAX}) - 4] + 0{,}8 \times [pos_{MAX}(n_{MAX}+1) - pos_{MAX}(n_{MAX})] \rangle,\ 0 \le n_{MAX} < N_{MAX}$$

and equals 0 otherwise. At the transitions (from 0,0 to 1,0 or from 1,0 to 0,0), the $w_{swp}(n)$ function has value 0,5.

Finally, the following weighting is applied to the input noise-reduced frame

$$s_{swp}(n) = 1{,}2 \times w_{swp}(n) \times s_{nr\_of}(n) + 0{,}8 \times (1 - w_{swp}(n)) \times s_{nr\_of}(n),\quad 0 \le n \le N_{in} - 1 \tag{5.48}$$

5.3 Cepstrum Calculation

This block performs cepstrum calculation. Cepstrum calculation is applied on the signal that comes out from the waveform processing block. The following figure shows main components of the Cepstrum Calculation block.

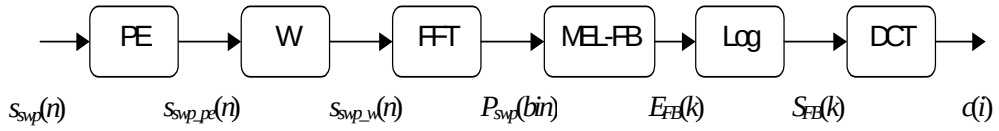


Figure 5.4: Main components of the cepstrum calculation block

5.3.1 Log energy calculation

For each frame, a log energy parameter is calculated from the de-noised signal as

$$lnE = \begin{cases} \ln(E_{swp}) & \text{if } E_{swp} \ge E_{THRESH} \\ \ln(E_{THRESH}) & \text{otherwise} \end{cases} \tag{5.49a}$$

where E_{THRESH} = exp(-50) and E_{swp} is computed as

$$E_{swp} = \sum_{n=0}^{N_{in}-1} s_{swp}(n) \times s_{swp}(n) \tag{5.49b}$$

5.3.2 Pre-emphasis (PE)

A pre-emphasis filter is applied to the output of the waveform processing block $s_{swp}(n)$ like

$$s_{swp\_pe}(n) = s_{swp}(n) - 0{,}9 \times s_{swp}(n-1) \tag{5.50}$$

where $s_{swp\_of}(-1)$ is the last sample from the previous frame and equals 0 for the first frame.

5.3.3 Windowing (W)

A Hamming window of length N_{in} =200 is applied to the output of the pre-emphasis block:

$$s_{swp\_w}(n) = \left[0{,}54 - 0{,}46 \times \cos\left(\frac{2\pi \times (n+0{,}5)}{N_{in}}\right)\right] \times s_{swp\_pe}(n), \quad 0 \le n \le N_{in} - 1 \tag{5.51}$$

5.3.4 Fourier transform (FFT) and power spectrum estimation

Each frame of N_{in} samples is zero padded to form an extended frame of 256 samples. An FFT of length N_{FFT} = 256 is applied to compute the complex spectrum $X_{swp}(bin)$ of the de-noised signal:

$$X_{swp}(bin) = FFT\{s_{swp\_w}(n)\} \tag{5.52}$$

Corresponding power spectrum $P_{swp}(bin)$ is calculated as

$$P_{swp}(bin) = \left|X_{swp}(bin)\right|^2, \quad 0 \le bin \le N_{FFT}/2 \tag{5.53}$$

5.3.5 Mel filtering (MEL-FB)

Purpose

The leading idea of the *MEL-FB* module is to recombine the information contained in the frequency-dependent representation (*FFT*) by regrouping it in a Mel-band representation.

The FFT-bins are linearly recombined for each Mel-band. The useful frequency band lies between f_{start} and f_{samp} / 2. This band is divided into K_{FB} channels equidistant in the Mel frequency domain. Each channel has a triangular-shaped frequency window. Consecutive channels are half-overlapping.

Frequencies and index

In the FFT calculation, index value $bin = N_{FFT}$ corresponds to the frequency f_{samp}. The formula that accounts for the index calculation of frequencies is then:

$$index\{f\} = round\left\{\frac{f}{f_{samp}} \times N_{FFT}\right\} \tag{5.54}$$

where $round\{\cdot\}$ stands for rounding towards the nearest integer.

Mel-function

The Mel-function is the operator which rescales the frequency domain.

$$Mel\{x\} = \Lambda \times \log_{10}\left(1 + \frac{x}{\mu}\right) = \lambda \times \ln\left(1 + \frac{x}{\mu}\right), \quad \text{with} \quad \lambda = \frac{\Lambda}{\ln(10)} \tag{5.55a}$$

The inverse Mel-function is:

$$Mel^{-1}\{y\} = \mu \times \left(\exp\left(\frac{y}{\lambda}\right) - 1\right) \tag{5.55b}$$

Central frequencies of the filters

The central frequencies of the filters are calculated from the Mel-function, in order to have an equidistant distribution of the bands in the Mel domain.

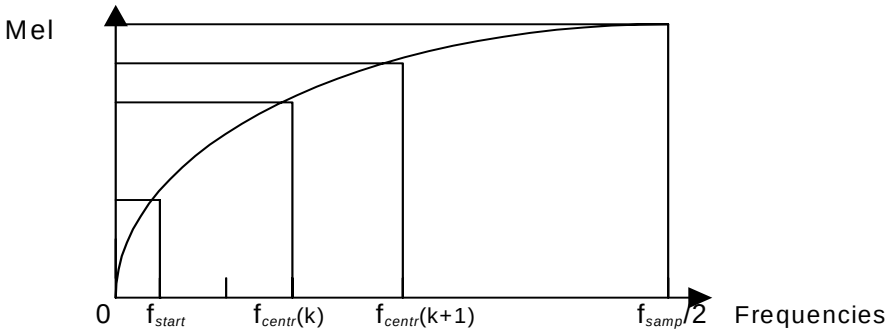


Figure 5.5: Linear to Mel frequency mapping

$$f_{centr}(k) = Mel^{-1}\left\{ Mel\{f_{start}\} + k \times \frac{Mel\{f_{samp}/2\} - Mel\{f_{start}\}}{K_{FB}+1} \right\}, \quad 1 \le k \le K_{FB} \qquad (5.56)$$

In our proposal, parameters are chosen as follows:

$f_{start} = 64\,Hz$, $f_{samp} = 8\,kHz$
$\mu = 700$, $\Lambda = 2\,595$, $\lambda = 1\,127$
$K_{FB} = 23$

In terms of FFT index, the central frequencies of the filters correspond to:

$$bin_{centr}(k) = index\{f_{centr}(k)\} = round\left\{ \frac{f_{centr}(k)}{f_{samp}} \times N_{FFT} \right\}, \quad 1 \le k \le K_{FB} \qquad (5.57)$$

For the *k*-th Mel-band, the frequency window is divided into two parts. The former part (i.e, frequencies $f_{centr}(k-1) < f < f_{centr}(k)$) accounts for increasing weights, whereas the latter part (i.e, frequencies $f_{centr}(k) < f < f_{centr}(k+1)$) accounts for decreasing weights. Each frequency window is applied to the de-noised power spectrum $P_{swp}(bin)$ computed by (5.53). Frequency window weights for each band are calculated depending on the position of each frequency bin with respect to the corresponding band central frequency like:

if the bin *i* is from $bin_{centr}(k-1) \le i \le bin_{centr}(k)$, then

$$W_{left}(i,k) = \frac{i - bin_{centr}(k-1) + 1}{bin_{centr}(k) - bin_{centr}(k-1) + 1}, \qquad \text{for } 1 \le k \le K_{FB} \qquad (5.58)$$

if the bin *i* is from $bin_{centr}(k) < i \le bin_{centr}(k+1)$, then

$$W_{right}(i,k) = 1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k) + 1}, \qquad \text{for } 1 \le k \le K_{FB} \qquad (5.59)$$

For other situations, weights equal zero.

Output of *MEL-FB*

The output of each Mel filter is the weighted sum of the de-noised power spectrum values $P_{swp}(bin)$ from equation (5.53) in each band. Triangular, half-overlapped windowing is used as follows:

$$E_{FB}(k) = \sum_{i=bin_{centr}(k-1)}^{bin_{centr}(k)} W_{left}(i,k) \times P_{swp}(i) + \sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)} W_{right}(i,k) \times P_{swp}(i), \quad \text{for } 1 \le k \le K_{FB} \qquad (5.60)$$

5.3.6 Non-linear transformation (Log)

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

$$S_{FB}(k) = \ln(E_{FB}(k)), \quad \text{for } 1 \le k \le K_{FB} \qquad (5.61)$$

A flooring is applied in such a way that the log filter bank outputs cannot be smaller than -10.

5.3.7 Cepstral coefficients (DCT)

13 cepstral coefficients are calculated from the output of the Non-linear transformation block by applying a DCT.

$$c(i) = \sum_{k=1}^{K_{FB}} S_{FB}(k) \times \cos\left(\frac{i \times \pi}{K_{FB}} \times (k - 0{,}5)\right), \quad 0 \le i \le 12 \qquad (5.62)$$

Notice that in the case of 16 kHz input signal, number of FB bands K_{FB} is increased by 3 (see clause 5.5 for more details).

5.3.8 Cepstrum calculation output

The final feature vector consists in 14 coefficients: the log-energy coefficient *lnE* and the 13 cepstral coefficients *c*(0) to *c*(12).

The $c(0)$ coefficient is often redundant when the log-energy coefficient is used. However, the feature extraction algorithm is defined here for both energy and $c(0)$. Depending on the application, either the coefficient $c(0)$, or the log-energy coefficient, or a combination of $c(0)$ and lnE may be used.

5.4 Blind Equalization

12 cepstral coefficients ($c(1),\ldots,c(12)$) are equalized according to the following LMS algorithm:

$$weightingPar = Min(1, Max(0, \ln E - 211/64)), \qquad (5.63)$$

$$stepSize = 0{,}008\,789\,062\,5 \times weightingPar, \qquad (5.64)$$

$$c_{eq}(i) = c(i) - bias(i), \ 1 \le i \le 12 \qquad (5.65)$$

$$bias(i) += stepSize \times (c_{eq}(i) - RefCep(i)), \ 1 \le i \le 12 \qquad (5.66)$$

where *lnE* is the log energy of the current frame as computed by (5.49a) and the values of $bias(i)$ and $RefCep(i)$ at the initialization stage are the following:

$$bias(i) = 0,0, \quad 1 \le i \le 12,$$

$$\begin{aligned} &RefCep(1) = -6,618\,909, \; RefCep(2) = \; 0,198\,269, \; RefCep(3) = -0,740\,308 \\ &RefCep(4) = \; 0,055\,132, \; RefCep(5) = -0,227\,086, \; RefCep(6) = \; 0,144\,280, \\ &RefCep(7) = -0,112\,451, \; RefCep(8) = -0,146\,940, \; RefCep(9) = -0,327\,466, \\ &RefCep(10) = \; 0,134\,571, \; RefCep(11) = \; 0,027\,884, \; RefCep(12) = -0,114\,905, \end{aligned} \tag{5.67}$$

The reference cepstrum corresponds to the cepstrum of a flat spectrum.

5.5 Extension to 11 kHz and 16 kHz sampling frequencies

For the 11 kHz sampling frequency, we perform downsampling from 11 kHz to 8 kHz and all front-end processing is the same as in the case of the 8 kHz sampling frequency.

For the 16 kHz sampling frequency, we extended the 8 kHz front-end as shown on figure 5.6. In this approach, the 8 kHz feature extraction part processes the signal from the low-frequency band (LFB, 0 kHz to 4 kHz) and it is re-used without significant changes. The signal from the high frequency band (HFB, 4 kHz to 8 kHz) is processed separately and the high-frequency information is added to the low-frequency information just before transforming the log FB energies to cepstral coefficients. Additionally, the whole-band log energy parameter *lnE* is also computed by using both the low-frequency and high-frequency information.

5.5.1 FFT-based spectrum estimation

As it can be observed from figure 5.6, the input signal, $s_{in\_16}(n)$, is first filtered by a couple of quadrature-mirror filters (QMF), $h_{LFB\_QMF}(n)$ and $h_{HFB\_QMF}(n)$, to get both the LFB and HFB signal portions:

$$s_{LFB}(n) = s_{in\_16}(n) \times h_{LFB\_QMF}(n), \; s_{HFB}(n) = s_{in\_16}(n) \times h_{HFB\_QMF}(n) \tag{5.68}$$

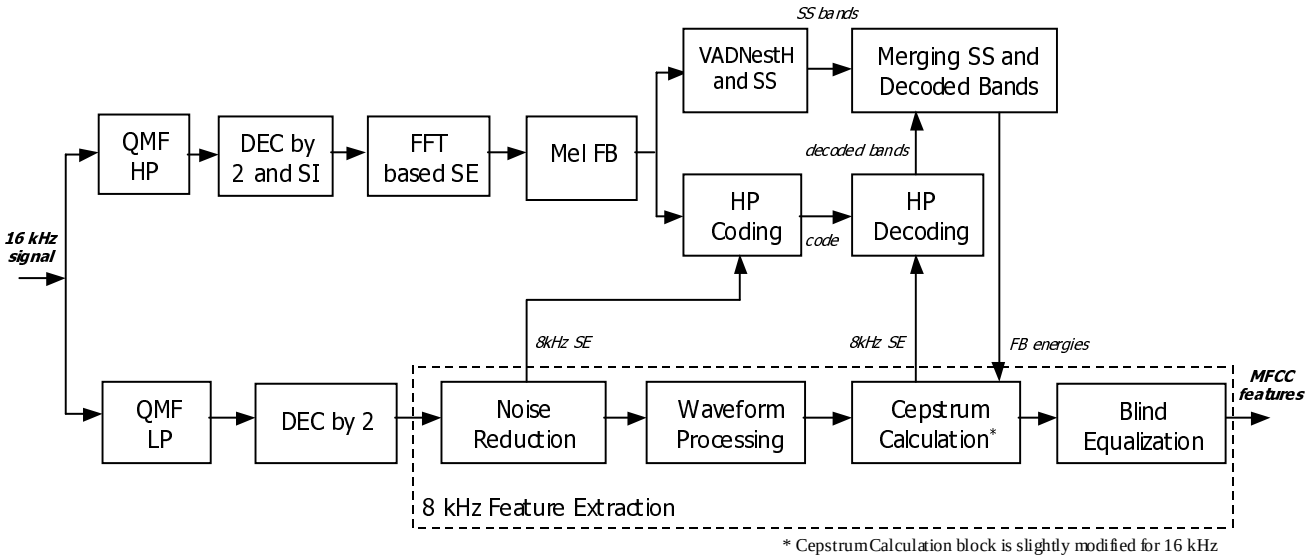


Figure 5.6: Extension of 8 kHz front-end for 16 kHz sampling frequency

The LFB QMF is a finite impulse response (FIR) filter of length 118 from the ITU-T standard software tools library for downsampling. The HFB QMF is an FIR filter obtained from the LFB QMF by multiplying each sample of its impulse response by $(-1)^n$, where n is sample index. Both LFB and HFB signals are decimated by factor 2 by choosing only every second sample from the corresponding filtered signal. Additionally, the HFB signal is frequency-inverted (spectrum inversion, SI on figure 5.6) by multiplying the HFB signal sequence by the sequence $(-1)^n$, where n is the sample index. The LFB signal enters the Noise Reduction part of Feature Extraction and it is processed up to the cepstral coefficient computation in the same way as in the case of 8 kHz sampling frequency.

By downsampling and spectral-inversion, the HFB signal is shifted to the frequency range 0 kHz to 4 kHz. This shifted HFB signal $s_{SI\_HFB}(n)$ is further processed on frame-by-frame basis, where the frame length and frame shift are synchronized with the LFB processing and are the same as in the case of 8 kHz input signal (i.e., 25ms/10ms). Each frame of length N_{in} = 200 is windowed by a Hamming window:

$$s_{W\_HFB}(n) = s_{SI\_HFB}(n) \times w_{Hamm}(n), \quad 0 \le n \le N_{in} - 1 \tag{5.69}$$

and zeros are padded from the sample N_{in} up to the sample N_{FFT}-1, where N_{FFT} = 256 is the FFT length:

$$s_{W\_HFB\_FFT}(n) = \begin{cases} s_{W\_HFB}(n), & 0 \le n \le N_{in} - 1 \\ 0, & N_{in} \le n \le N_{FFT} - 1 \end{cases} \tag{5.70}$$

A smoothed HFB power spectrum, $P_{Smooth\_HFB}(bin)$, is estimated by using an FFT followed by power of 2 like:

$$X_{HFB}(bin) = FFT\{s_{W\_HFB\_FFT}(n)\} \tag{5.71}$$

$$P_{HFB}(bin) = |X_{HFB}(bin)|^2, \quad 0 \le bin \le N_{FFT}/2 \tag{5.72}$$

$$P_{Smooth\_HFB}(bin) = \frac{P_{HFB}(2 \times bin) + P_{HFB}(2 \times bin + 1)}{2}, \quad 0 \le bin < N_{FFT}/4 \tag{5.73}$$

$$P_{Smooth\_HFB}(N_{FFT}/4) = P_{HFB}(N_{FFT}/2)$$

By the smoothing operation, the length of the power spectrum is reduced to $N_{SPEC} = N_{FFT}/4 + 1$

5.5.2 Mel filter-bank

The entire high-frequency band is divided into K_{HFB} = 3 filter-bank (FB) bands, which are equidistantly distributed in the Mel-frequency domain. Energies within the FB bands, $E_{HFB}(k)$, are estimated by using triangular-shaped, half-overlapped frequency windows applied on the HFB power spectrum. To obtain the central frequencies of FB bands in terms of FFT bin indices, $bin_{centr}(k)$, we used the following relationship between the linear and mel frequency scales:

$$f_{mel} = MEL\{f_{lin}\} = 2\,595 \times \log_{10}(1 + f_{lin}/700) \tag{5.74}$$

Then, the central frequency of the k-th band, $f_{centr}(k)$, is calculated as:

$$f_{centr}(k) = 700 \times \left(10^{\frac{f_{mel}(k)}{2\,595}} - 1 \right), \quad 1 \le k \le K_{HFB} \tag{5.75}$$

with

$$f_{mel}(k) = MEL\{f_{lin\_start}\} + k \times \frac{MEL\{f_{lin\_samp}/2\} - MEL\{f_{lin\_start}\}}{K_{HFB} + 1} \tag{5.76}$$

where $f_{lin\_start} = 80$ is the starting frequency and $f_{lin\_samp} = 8\,000$ is the sampling frequency. The corresponding FFT bin index is obtained as:

$$bin_{centr}(k) = round\left(\frac{f_{centr}(k)}{f_{lin\_samp}} \times 2 \times (N_{SPEC} - 1) \right) \tag{5.77}$$

Having the central frequencies, $bin_{centr}(k)$, the energy within the *k*-th FB band, $E_{HFB}(k)$, is computed as:

$$\begin{aligned} E_{HFB}(k) = & \sum_{i=bin_{centr}(k-1)+1}^{bin_{centr}(k)} \frac{i - bin_{centr}(k-1)}{bin_{centr}(k) - bin_{centr}(k-1)} \times P_{Smooth\_HFB}(i) + \\ & + \sum_{i=bin_{centr}(k)+1}^{bin_{centr}(k+1)} \left(1 - \frac{i - bin_{centr}(k)}{bin_{centr}(k+1) - bin_{centr}(k)} \right) \times P_{Smooth\_HFB}(i) \end{aligned} \tag{5.78}$$

where $1 \le k \le K_{HFB}$. $bin_{centr}(0)$ and $bin_{centr}(K_{HFB}+1)$ are the FFT indices corresponding to the starting frequency f_{lin_start}, and half of the sampling frequency $f_{lin\_samp}/2$

5.5.3 High-frequency band coding and decoding

Before coding, the natural logarithm is applied to the HFB mel FB energies $E_{HFB}(k)$ as:

$$S_{HFB}(k) = \ln(E_{HFB}(k)), \quad 1 \le k \le K_{HFB} \tag{5.79}$$

with a flooring avoiding values of $S_{HFB}(k)$ lower than -10. The HFB log FB energies, $S_{HFB}(k)$, are coded and decoded by using three auxiliary bands computed from 2 kHz to 4 kHz frequency interval of LFB power spectrum. For coding, the auxiliary bands are calculated before applying both noise reduction (NR) and waveform processing (SWP) to the LFB signal. For decoding, the auxiliary bands are calculated after applying both NR and SWP to the LFB signal. Auxiliary bands are approximately logarithmically spaced in the given frequency interval.

The three auxiliary log FB energies for coding are computed from the input signal power spectrum $P_{in}(bin)$, $0 \le bin < N_{SPEC}$, calculated in the first stage of Noise Reduction block (see equation (5.6) in clause 5.1.2) as:

$$S_{LFB\_aux}(1) = \ln\left(\sum_{bin=33}^{38} P_{in}(bin) \right), \; S_{LFB\_aux}(2) = \ln\left(\sum_{bin=39}^{48} P_{in}(bin) \right) \text{ and} \tag{5.80}$$

$$S_{LFB\_aux}(3) = \ln\left(\sum_{bin=49}^{64} P_{in}(bin) \right)$$

with flooring that avoids values of $S_{LFB\_aux}(k)$ lower than -10. Then, coding is performed as:

$$Code(k,l) = S_{LFB\_aux}(k) - S_{HFB}(l), \quad 1 \le k,l \le K_{HFB} \tag{5.81}$$

The three auxiliary bands for decoding are computed from the de-noised power spectrum $P_{swp}(bin)$, $0 \le bin \le N_{FFT}/2$, calculated in the Cepstrum Calculation block (see clause 5.3.5) as:

$$S_{swp\_LFB\_aux}(1) = \ln\left(\frac{1}{2}\sum_{bin=66}^{76} P_{swp}(bin)\right),\ S_{swp\_LFB\_aux}(2) = \ln\left(\frac{1}{2}\sum_{bin=77}^{96} P_{swp}(bin)\right), \text{ and} \quad (5.82)$$

$$S_{swp\_LFB\_aux}(3) = \ln\left(\frac{1}{2}\sum_{bin=97}^{128} P_{swp}(bin)\right)$$

with flooring that avoids values of $S_{swp\_LFB\_aux}(k)$ lower than -10. The decoded HFB bands, $S_{code\_HFB}(k)$, are obtained by using the code $Code(k,l)$ and the three de-noised auxiliary LFB log FB energies $S_{swp\_LFB\_aux}(k)$ like:

$$S_{code\_HFB}(k) = \sum_{l=1}^{K_{HFB}} w_{code}(l)\left(S_{swp\_LFB\_aux}(l) - Code(l,k)\right),\ \ 1 \le k \le K_{HFB} \quad (5.83)$$

where $w_{code}(l)$ is a frequency-dependent weighting with:

$$\sum_{l=1}^{K_{HFB}} w_{code}(l) = 1 \quad (5.84)$$

In the current implementation, frequency weights are $w_{code}(1) = 0{,}1,\ w_{code}(2) = 0{,}2,\ w_{code}(3) = 0{,}7$

5.5.4 VAD for noise estimation and spectral subtraction in high-frequency bands

A simple, energy-based voice activity detector for noise estimation (VADNestH) is designed for noise estimation in the HFB signal. A forgetting factor for a) updating the noise estimation and b) tracking the low log energy level is computed for each frame *t* according to the logic:

$$\begin{aligned} &if \quad \{t < 100\} \\ &\quad \lambda_{NSE}(t) = 1 - 1/t \\ &else \\ &\quad \lambda_{NSE}(t) = 0{,}99 \end{aligned} \quad (5.85)$$

The low log energy level is tracked by using the following logic:

$$\begin{aligned} &if \quad \{[(E_{\log}(t) - E_{\log\_low\_track}(t-1)) < 1{,}2] \quad \text{or} \quad [t < 10]\} \\ &\quad if \quad \{t < 10\} \\ &\quad\quad E_{\log\_low\_track}(t) = \lambda_{NSE}(t) \times E_{\log\_low\_track}(t-1) + (1 - \lambda_{NSE}(t)) \times E_{\log}(t) \\ &\quad else \\ &\quad\quad if \quad \{E_{\log}(t) < E_{\log\_low\_track}(t-1)\} \\ &\quad\quad\quad E_{\log\_low\_track}(t) = 0{,}98 \times E_{\log\_low\_track}(t-1) + (1 - 0{,}98) \times E_{\log}(t) \\ &\quad\quad else \\ &\quad\quad\quad E_{\log\_low\_track}(t) = 0{,}995 \times E_{\log\_low\_track}(t-1) + (1 - 0{,}995) \times E_{\log}(t) \end{aligned} \quad (5.86)$$

where $E_{\log\_low\_track}$ is initialized to 0 and the log energy $E_{\log}(t)$ is computed like:

$$E(t) = \sum_{k=1}^{K_{HFB}} E_{HFB}(t,k) \tag{5.87a}$$

$$E_{\log}(t) = \begin{cases} \ln(E(t)) & \text{for } E(t) > 0{,}001 \\ \ln(0{,}001) & \text{for } E(t) \le 0{,}001 \end{cases} \tag{5.87b}$$

VADNestH flag $flagVAD_{NestH}(t)$ is updated by using the current frame log energy $E_{\log}(t)$ and the low log energy level $E_{\log\_low\_track}(t)$ as follows:

$$\begin{aligned} &if \quad \{(E_{\log}(t) - E_{\log\_low\_track}(t)) > 2{,}2\} \\ &\quad flagVAD_{NestH}(t) = 1 \\ &\quad nbSpeechFrame(t) = nbSpeechFrame(t-1) + 1 \\ &else \\ &\quad if \quad \{nbSpeechFrame(t-1) > 4\} \\ &\quad\quad hangOver(t) = 5 \\ &\quad nbSpeechFrame(t) = 0 \\ &\quad if \quad \{hangOver(t) \; != 0\} \\ &\quad\quad hangOver(t+1) = hangOver(t) - 1 \\ &\quad\quad flagVAD_{NestH}(t) = 1 \\ &\quad else \\ &\quad\quad flagVAD_{NestH}(t) = 0 \end{aligned} \tag{5.88}$$

VADNestH flag is used for estimating the HFB noise spectrum in terms of FB energies like:

$$\begin{aligned} &if \quad \{flagVAD_{NestH}(t) = 0\} \\ &\quad \hat{N}_{HFB}(k,t) = \lambda_{NSE}(t) \times E_{HFB}(k,t) + (1 - \lambda_{NSE}(t)) \times \hat{N}_{HFB}(k,t-1), \quad 1 \le k \le K_{HFB}, \end{aligned} \tag{5.89}$$

where *t* is the frame index and the noise FB energy vector is initialized to a zero vector.

Spectral subtraction is performed like:

$$E_{SS\_HFB}(k) = \max\{E_{HFB}(k) - \alpha \times \hat{N}_{HFB}(k),\ \beta \times E_{HFB}(k)\}, \quad 1 \le k \le K_{HFB} \tag{5.90}$$

where α = 1,5 and β = 0,1 were set empirically.

5.5.5 Merging spectral subtraction bands with decoded bands

In the Cepstrum Calculation block, log FB energies from both LFB and HFB are joined and cepstral coefficients representing the entire frequency band are calculated. It is obvious that the noise reduction performed on the LFB signal is more complex than the spectral subtraction (SS) algorithm applied on HFB FB bands, and thus FB energies resulting from these two processes are not entirely compatible. To reduce the differences between the FB energies from the HFB and LFB, the SS HFB log FB energies are used in combination with the HFB log FB energies resulting from the coding scheme described in clause 5.5.3.

First, rough pre-emphasis correction and log non-linearity are applied on HFB energies resulting from spectral subtraction like:

$$S_{SS\_HFB}(k) = \ln\left(\left(1 + a_{pre}\right) \times E_{SS\_HFB}(k)\right) \quad 1 \le k \le K_{FB} \tag{5.91}$$

where $a_{pre} = 0{,}9$ is pre-emphasis constant. The HFB log FB energies $S_{HFB}(k)$ are then obtained by combining both $S_{SS\_HFB}(k)$ and $S_{code\_HFB}(k)$, like:

$$S_{HFB}(k) = \lambda_{merge} \times S_{code\_HFB}(k) + \left(1 - \lambda_{merge}\right) \times S_{SS\_HFB}(k), \quad 1 \le k \le K_{HFB} \tag{5.92}$$

where $\lambda_{merge} = 0{,}7$ is an empirically set constant.

For each frame, a cepstrum is calculated from a vector of log FB energies that is formed by appending the three HFB log FB energies to the LFB log FB energies. Before joining the LFB and HFB log FB energies, the transition between the last LFB band $S_{FB}(K_{FB})$ (computed as in clause 5.3.7) and the first HFB $S_{HFB}(1)$ is smoothed by modifying the two transition log energies like:

$$S'_{FB}(K_{FB}) = 0{,}6 \times S_{FB}(K_{FB}) + 0{,}4 \times S_{aver} \tag{5.93a}$$

and

$$S'_{HFB}(1) = 0{,}6 \times S_{HFB}(1) + 0{,}4 \times S_{aver} \tag{5.93b}$$

where

$$S_{aver} = \frac{S_{FB}(K_{FB}) + S_{HFB}(1)}{2} \tag{5.93c}$$

Finally, the log FB energy vector for cepstrum calculation $S_{cep}(k),\ 1 \le k \le K_{FB} + K_{HFB}$, is formed like:

$$S_{cep}(k) = \left\{S_{FB}(1), S_{FB}(2), \ldots, S_{FB}(K_{FB} - 1), S'_{FB}(K_{FB}), S'_{HFB}(1), S_{HFB}(2), S_{HFB}(3)\right\} \tag{5.94}$$

5.5.6 Log energy calculation for 16 kHz

Log energy parameter is computed by using information from both the LFB and HFB. We used the HFB log FB energies, $S_{HFB}(k)$, to modify the log energy parameter. First, we computed the HFB energy E_{HFB} by using pre-emphasis corrected, de-noised HFB log FB energies like:

$$E_{HFB} = \sum_{k=1}^{K_{HFB}} \exp\left(S_{HFB}(k) - preem\_corr\right) \tag{5.95}$$

where

$$preem\_corr = \ln\left(1 + a_{pre}\right) \tag{5.96}$$

and a_{pre} = 0,9 is the pre-emphasis constant. Then, the energy parameter is computed as the natural logarithm of the sum of the de-noised LFB energy E_{swp} and the de-noised HFB energy E_{HFB}

$$lnE = \ln\left(E_{swp} + E_{HFB}\right) \tag{5.97}$$

6 Feature Compression

6.1 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 5. Its purpose is to reduce the number of bits needed to represent each front-end feature vector.

6.2 Compression algorithm description

6.2.1 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 5.4.

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

$$\mathbf{c_{eq}}(t) = \left[c_{eq}(1,t), c_{eq}(2,t), ..., c_{eq}(12,t)\right]^T \tag{6.1}$$

where *t* denotes the frame index, plus the zeroth cepstral coefficient *c*(0) and a log energy term *lnE*(*t*) as defined in clause 5.3.2. The final input to the compression algorithm is the VAD flag. These parameters are formatted as:

$$y(t) = \begin{bmatrix} \mathbf{c_{eq}}(t) \\ VAD(t) \\ c(0,t) \\ lnE(t) \end{bmatrix} \tag{6.2}$$

6.2.2 Vector quantization

The feature vector ***y***(*t*) is directly quantized with a split vector quantizer. The 14 coefficients (c(1) to c(12), c(0) & *lnE*) 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 6.1, along with the codebook size used for each pair. The VAD flag is transmitted as a single bit. *c*(1) to *c*(10) are quantized with 6 bits per pair, while *c*(11) and *c*(12) are quantized with 5 bits. The closest VQ centroid is found using a weighted Euclidean distance to determine the index:

$$d_j^{i,i+1} = \begin{bmatrix} y_i(t) \\ y_{i+1}(t) \end{bmatrix} - q_j^{i,i+1} \tag{6.3}$$

$$idx^{i,i+1}(t) = \underset{0 \le j \le \left(N^{i,i+1}-1\right)}{\operatorname{argmin}} \left\{\left(d_j^{i,i+1}\right) W^{i,i+1} \left(d_j^{i,i+1}\right)\right\}, \quad i = \{0, 2, 4...12\} \tag{6.4}$$

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 $idx^{i,i+1}(t)$ denotes the codebook index chosen to represent the vector $\left[y_i(t), y_{i+1}(t)\right]^T$. The indices are then retained for transmission to the back-end.

Table 6.1: Split Vector Quantization Feature Pairings

Codebook	Size ($N^{l,l+1}$)	Weight Matrix ($W^{l,l+1}$)	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)
$Q^{10,11}$	32	I	c(11)	c(12)
$Q^{12,13}$	256	Non-identity	c(0)	lnE

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 $$W^{12,13} = \begin{bmatrix} 10\,498{,}7 & 0 \\ 0 & 15{,}1 \end{bmatrix}$$

16 kHz sampling rate $$W^{12,13} = \begin{bmatrix} 10\,617{,}2 & 0 \\ 0 & 21{,}8 \end{bmatrix}$$

7 Framing, Bit-Stream Formatting and Error Protection

7.1 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. The basic unit for transmission consists of a pair of speech frames and associated error protection bits with the format defined in clause 7.2.4. This frame pair unit can be used either for circuit data systems or packet data systems such as the IETF real-time protocols (RTP). For circuit data transmission a multiframe format is defined consisting of 12 frame pairs in each multiframe and is described in clauses 7.2.1 to 7.2.3. The formats for DSR transmission using RTP are defined in the IETF Audio Video Transport, Internet-Draft (see bibliography) where the number of frame pairs sent per payload is flexible and can be designed for a particular application.

7.2 Algorithm description

7.2.1 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 7.1, consists of a synchronization sequence, a header field, and a stream of frame packets.

Table 7.1: 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 7.1) fields are transmitted left to right.

7.2.2 Synchronization sequence

Each multiframe begins with the 16-bit synchronization sequence 0 x 87B2 (sent LSB first, as shown in table 7.2).

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 synchronization is not required if a multiframe is immediately followed by the synchronization sequence for the next multiframe.

Table 7.2: 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

7.2.3 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 7.3, and definition of the fields appears in table 7.4. 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 clause 7.2.4).

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 7.3: 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 7.4: 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:

$$g_1(X) = 1 + X^8 + X^{12} + X^{14} + X^{15} \tag{7.1}$$

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:

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ P_{15} \\ P_{16} \end{bmatrix} = \begin{bmatrix} 1&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1 \\ 1&1&0&0&0&0&0&0&1&1&0&0&1&1&1&0 \\ 1&1&1&0&0&0&0&0&1&1&1&0&1&1&0&1 \\ 0&1&1&1&0&0&0&0&0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&0&0&1&0&1&1&0&0&0&0 \\ 0&1&0&1&1&1&0&0&0&1&0&1&1&0&0&0 \\ 0&0&1&0&1&1&1&0&0&0&1&0&1&1&0&0 \\ 0&0&0&1&0&1&1&1&0&0&0&1&0&1&1&0 \\ 1&0&0&0&1&0&1&1&0&0&0&0&0&0&0&1 \\ 0&1&0&0&0&1&0&1&1&0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&1&0&1&1&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1&0&1&1&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0&1&0&1&1&0&0&0&1 \\ 0&0&0&0&0&1&0&0&0&1&0&1&1&0&0&1 \\ 0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1 \\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1 \end{bmatrix}^T \begin{bmatrix} SampRate1 \\ SampRate2 \\ feType \\ MFrameCnt1 \\ MFrameCnt2 \\ MFrameCnt3 \\ MFrameCnt4 \\ EXP1 \\ EXP2 \\ EXP3 \\ EXP4 \\ EXP5 \\ EXP6 \\ EXP7 \\ EXP8 \\ EXP9 \end{bmatrix} \tag{7.2}$$

where *T* denotes the matrix transpose.

7.2.4 Frame packet stream

Each 10 ms frame from the front-end is represented by the codebook indices specified in clause 7.2.2. The indices and the VAD flag for a single frame are formatted for a frame according to table 7.5.

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

Table 7.5: Frame information for *t*-th frame

Bit 8	7	6	5	4	3	2	1	Octet
$idx^{2,3}(t)$		$Idx^{0,1}(t)$						1
$idx^{4,5}(t)$				$idx^{2,3}(t)$ (cont)				2
$idx^{6,7}(t)$						$idx^{4,5}(t)$ (cont)		3
$idx^{10,11}(t)$	VAD (t)	$idx^{8,9}(t)$						4
$idx^{12,13}(t)$				$idx^{10,11}(t)$ (cont)				5
				$idx^{12,13}(t)$ (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 7.6 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 7.6: 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.

8 Bit-Stream Decoding and Error Mitigation

8.1 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.

8.2 Algorithm description

8.2.1 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.

8.2.2 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.

8.2.3 Feature decompression

The indices and the VAD flag 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 clause 8.2.4.) Using the indices received, estimates of the front-end features are extracted with a VQ codebook lookup:

$$\begin{bmatrix} \hat{y}_i(t) \\ \hat{y}_{i+1}(t) \end{bmatrix} = q^{i,i+1}_{idx^{i,i+1}(m)} \quad i = \{0, 2, 4...12\} \tag{8.1}$$

8.2.4 Error mitigation

8.2.4.1 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:

$$badindexflag_i = \begin{cases} 1 & \text{if } \left(y_i(t+1) - y_i(t) > T_i\right) \; OR \; \left(y_{i+1}(t+1) - y_{i+1}(t) > T_{i+1}\right) \\ 0 & otherwise \end{cases} \quad i = \{0,2....12\} \qquad (8.2)$$

The thresholds T_i have been determined based on measurements of error free 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:

$$\sum_{i=0,2,...12} badindexflag_i \geq 2 \qquad (8.3)$$

The data consistency check for erroneous 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 8.1 and 8.2.

8.2.4.2 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. It should be noted that the speech vector includes the 12 static cepstral coefficients, the zeroth cepstral coefficient, the log energy term and the VAD flag, and all are therefore replaced together. 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.

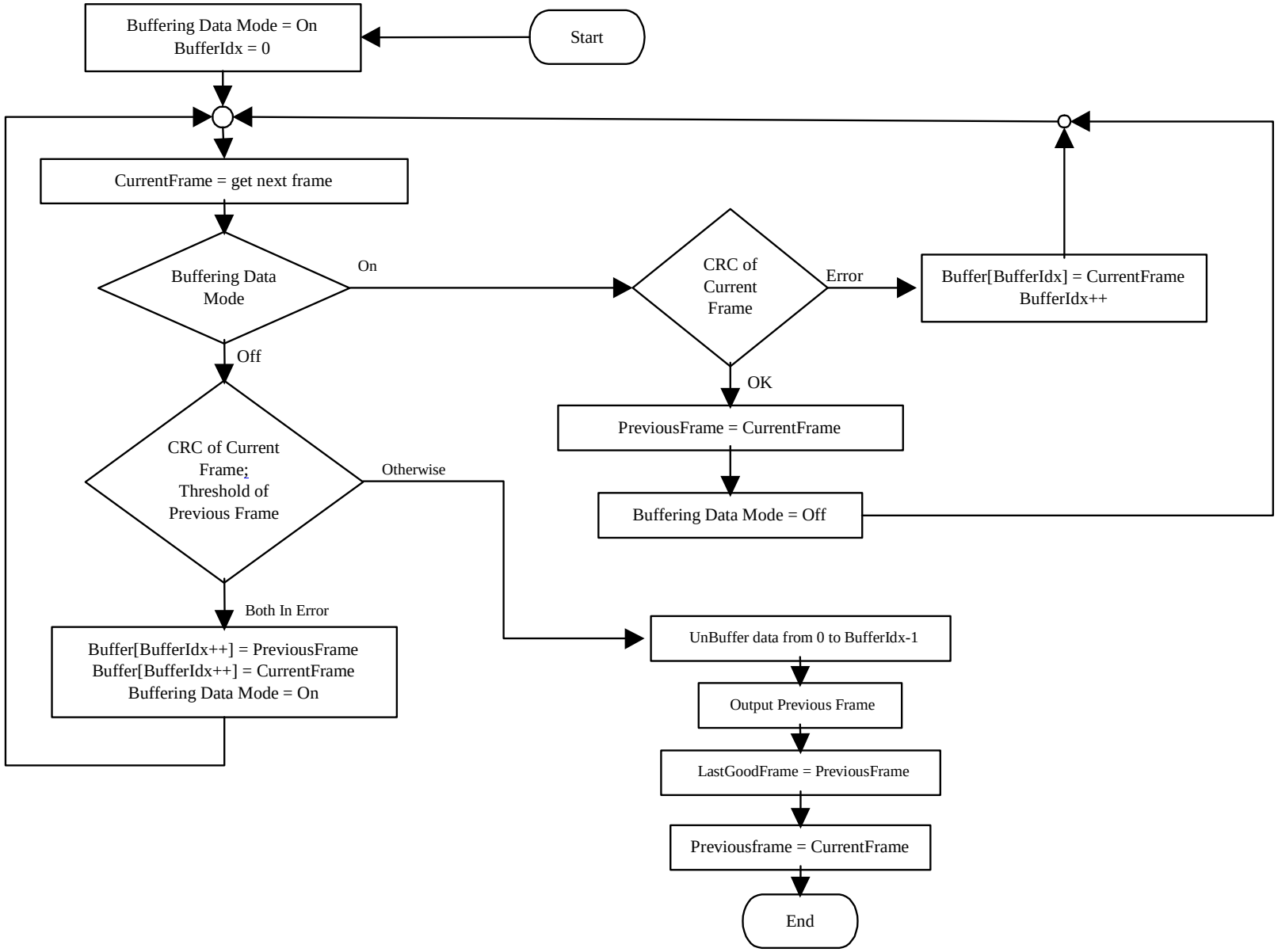


Figure 8.1: Error mitigation initialization flow chart

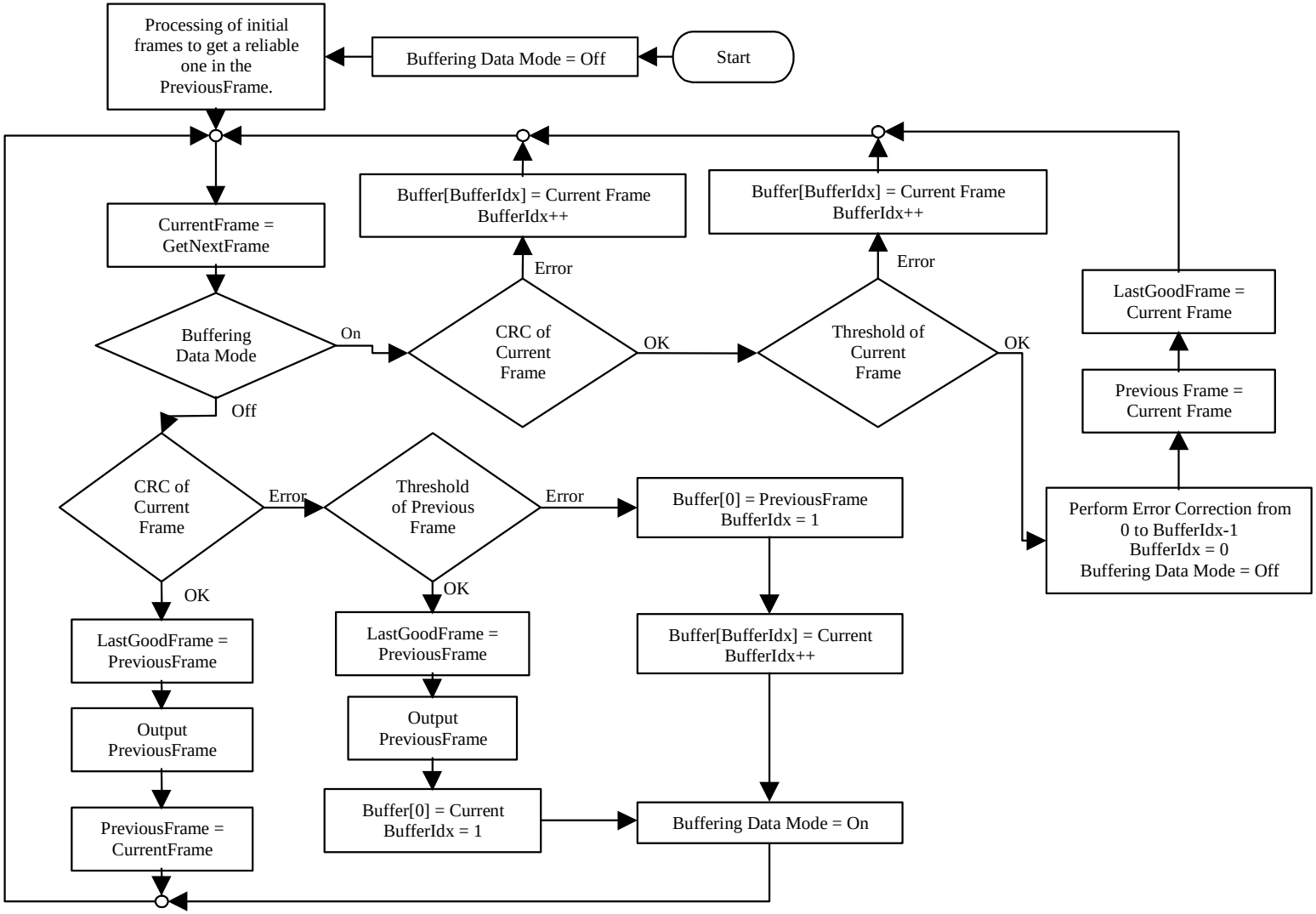


Figure 8.2: Main error mitigation flow chart

9 Server Feature Processing

lnE and $c(0)$ combination, derivatives calculation and feature vector selection (FVS) processing are performed at the server side. $c(0)$, $c(1)$, …, $c(12)$, lnE are received in the back-end. $c(0)$ is combined with lnE then the first and second order derivatives of $c(1)$, …, $c(12)$, $lnE \,\&\, c(0)$ are calculated resulting in a 39 dimensional feature vector. A feature vector selection procedure is then performed according to the VAD information transmitted.

9.1 lnE and c(0) combination

$c(0)$ and lnE are combined in the following way:

$$lnE \,\&\, c(0) = 0,6 \times c(0)/23 + 0,4 \times lnE \tag{9.1}$$

9.2 Derivatives calculation

First and second derivatives are computed on a 9-frame window. Velocity and acceleration components are computed according the following formulas:

$$\begin{aligned} vel(i,t) = {} & -1,0 \times c(i,t-4) - 0,75 \times c(i,t-3) - 0,50 \times c(i,t-2) - 0,25 \times c(i,t-1) \\ & + 0,25 \times c(i,t+1) + 0,50 \times c(i,t+2) + 0,75 \times c(i,t+3) + 1,0 \times c(i,t+4), \\ & 1 \le i \le 12 \end{aligned} \tag{9.2}$$

$$\begin{aligned} acc(i,t) = {} & 1,0 \times c(i,t-4) + 0,25 \times c(i,t-3) - 0,285\,714 \times c(i,t-2) \\ & - 0,607\,143 \times c(i,t-1) - 0,714\,286 \times c(i,t) - 0,607\,143 \times c(i,t+1) \\ & - 0,285\,714 \times c(i,t+2) + 0,25 \times c(i,t+3) + 1,0 \times c(i,t+4), \\ & 1 \le i \le 12 \end{aligned} \tag{9.3}$$

where t is the frame time index.

The same formulae are applied to obtain $lnE \,\&\, c(0)$ velocity and acceleration components.

9.3 Feature vector selection

A FVS algorithm is used to select the feature vectors that are sent to the recognizer. All the feature vectors are computed and the feature vectors that are sent to the back-end recognizer are those corresponding to speech frames, as detected by a VAD module (described in annex A).

Annex A (informative): Voice Activity Detection

A.1 Introduction

The voice activity detector has two stages - a frame-by-frame detection stage consisting of three measurements, and a decision stage in which the pattern of measurements, stored in a circular buffer, is analysed to indicate speech likelihood. The final decision from this second stage is applied retrospectively to the earliest frame in the buffer, so providing a short look-ahead facility. A hangover facility is also provided, with hangover duration related to speech likelihood.

A.2 Stage 1 - Detection

In non-stationary noise, long-term (stationary) energy thresholds based, for example, on initial noise estimates are not a reliable indicator of speech. In addition, in high noise conditions the structure of the speech (e.g. harmonics) cannot be wholly relied upon as an indicator of speech as they may be corrupted by noise, or structured noises may confuse a detector based on this method.

The voice activity detector presented here uses a comparatively noise-robust characteristic of the speech, namely the energy acceleration associated with voice onset. This acceleration is measured in three ways:

i. from energy values across the whole spectrum of each frame,
ii. from energy values over a sub-region of the spectrum of each frame considered likely to contain the fundamental pitch, and
iii. from the "acceleration" of the variance of energy values within the lower half of the spectrum of each frame.

Due to the presence of the fundamental pitch, the sub-region (characterised typically as the second third and fourth Mel-spectum bands as defined within the body of this document) generally experiences higher signal to noise ratios than the full spectum. Consequently the sub-region measurement is potentially more noise robust than the measurement based on the full spectrum.

However, the sub-region measurement is vulnerable to the effects of high-pass microphones, significant speaker variability and band-limited noise within the sub-region. Consequently it cannot be relied upon in all circumstances and is treated here instead as an augmentation of the whole spectrum measure rather than as a substitute for it.

The variance measure detects structure within the lower half of the spectrum as harmonic peaks and troughs provide a greater variance than most noise, making it particularly sensitive to voiced speech. This complements the whole spectrum measure, which is better able to detect unvoiced and plosive speech.

Measurement 1 - Whole spectrum

The whole-spectrum measurement uses the Mel-warped Wiener filter coefficients generated by the first stage of the double Wiener filter (see clause 5.1.7). A single input value is obtained by squaring the sum of the Mel filter banks.

The voice activity detector applies each of the following steps to the input from each frame, as described below:

1. `If Frame<15 AND Acceleration<2,5, Tracker=MAX(Tracker,Input)`
   Step one initialises the noise level estimate *Tracker*. The acceleration measure prevents *Tracker* being updated if speech commences within the lead-in of *Frame* < 15 frames.

2. `If Input<TrackerxUpperBound and Input>TrackerxLowerBound, Tracker=`*a*`xTracker+(1-`*a*`)xInput`
   Step two updates *Tracker* if the current input is similar to the noise estimate.

3. If Input<TrackerxFloor, Tracker=*b*xTracker+(1-*b*)xInput
   Step three is a failsafe for those instances where there is speech or an uncharacteristically large noise within the first few frames, allowing the resulting erroneously high noise estimate in *Tracker* to decay. Note there is no update of *Tracker* if the value is greater than *UpperBound*, or between *LowerBound* and *Floor*.

4. If Input>TrackerxThreshold, output TRUE else output FALSE.
   Step four returns true if the current input is more than 165 % the size of the value in *Tracker*.

Where a = 0,8 and b = 0,97, *UpperBound* is 150 % and *LowerBound* 75 %. *Floor* is 50 % and *Threshold* is 165 %. *Input* is obtained by squaring the sum of the Mel filter banks as described above.

While *Acceleration* could be calculated using the double-differentiation of successive inputs, it is estimated here by tracking the ratio of two rolling averages of successive inputs. The ratio of fast and slow-adapting rolling averages reflects the acceleration of successive inputs. The contribution rates for the averages used here were `(0 x mean + 1 x input)` i.e. instantaneous, and `((Frame - 1) x mean + 1 x input) / Frame` for fast and slow rolling averages respectively, making the acceleration measure increasingly sensitive over the first 15 frames of step one.

As noted above, the ratio of the instantaneous input value to the short-term mean value *Tracker* in step 4 is a function of the acceleration of successive inputs.

Measurement 2 - Spectral sub-region

The sub-region measurement uses as its input the average of the second, third and fourth Mel-warped Wiener filter coefficients generated by the first stage of the double Wiener filter (see clause 5.1.7). The detector then applies each of the following steps to the input from each frame, as described below:

1. Input=*p*xCurrentInput+(1-*p*)xPreviousInput
   Step one uses a rolling average to generate a smoothed version of the current input, where p = 0,75.

2. If Frame<15, Tracker=MAX(Tracker,Input)
   Step two initialises the noise estimate as the maximum of the smoothed input over the first 15 frames. Note that the variables such as 'Input' and 'Tracker' are distinct for each measurement.

Steps 3 to 5 are functionally the same as steps 2 to 4 of measurement 1, with the exception that *Threshold* now equals 3,25.

3. If Input<TrackerxUpperBound and Input>TrackerxLowerBound, Tracker=*a*xTracker+(1-*a*)xInput

4. If Input<TrackerxFloor, Tracker=*b*xTracker+(1-*b*)xInput

5. If Input>TrackerxThreshold, output TRUE else output FALSE.

Measurement 3 - Spectral Variance

The spectral variance measurement uses as its input the variance of the values comprising the whole frequency range of the linear-frequency Wiener filter coefficients for each frame. This variance is calculated as:

$$\frac{1}{N_{SPEC}} \sum_{i=0}^{N_{SPEC}-1} \left(H_2(bin)\right)^2 - \left(\sum_{i=0}^{N_{SPEC}-1} H_2(bin) \right)^2 / N_{SPEC}^2 \qquad \text{(A.1)}$$

where $N_{SPEC} = N_{FFT}/4$, and $H_2(bin)$ are the values of the linear-frequency Wiener filter coefficients as calculated by equation (5.17) in clause 5.1.5.

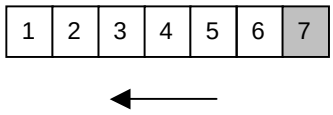
In step 1, the detector takes the maximum input value of the first 15 frames as in step 2 of Measurement 2.

Steps 2 to 4 are then the same as steps 2-4 of Measurement 1, to give a true/false output.

A.3 Stage 2 - VAD Logic

The three measurements discussed above are input to a VAD decision algorithm. The algorithm generates a single 1/0 (true/false or T/F) result based on these measurements, and stores it in a buffer. Successive results populate the buffer, so providing for contextual analysis of the buffer pattern. The VAD decision algorithm does not begin output until the buffer is fully populated with valid results. This process introduces a frame delay equal to the length of the buffer minus one.

For an *N = 7 frame* buffer, the most recent result is stored at position *N* as illustrated below. For subsequent results, the buffer contents shift left.



The VAD decision algorithm applies each of the following steps:

Step 1: V_N = Measurement 1 OR Measurement 2 OR Measurement 3

Thus result V_N is true if any of the three measurements returns true. Result V_N is then stored at position *N* on the buffer.

Step 2: $$M = MAX\left\{ \begin{array}{ll} C++, & V_i = TRUE \\ C = 0, & V_i = FALSE \end{array} , 1 < i < N \right\} \Big|_c$$

The decision algorithm then analyses the resulting buffer pattern. It searches for the longest single sequence of 'true' values in the buffer; moving along the buffer, a counter *C* is incremented if the next buffer value is true, and is reset to 0 if it is false. The maximum value *C* attains over the whole buffer is then taken as the value of *M*. For example, the sequence T T F T T T F would generate a value of 3 for *M*.

Step 3: If $M >= S_P$ AND $T < L_S$, $T = L_S$

Where S_P is a 'speech possible' threshold, corresponding to a sequence of 3 or more 'true' values found in the buffer at step 2. A short hangover timer *T* of L_S = 5 frames is activated if no hangover is already present.

Step 4: If $M >= S_L$ AND $F > F_S$, $T = L_M$ else if $M >= S_L$, $T = L_L$

Where S_L is a 'speech likely' threshold, corresponding to a sequence of 4 or more 'true' values found in the buffer at step 2. A medium hangover timer *T* of L_M = 23 frames is activated if the current frame number *F* is outside an initial lead-in safety period of F_S frames. Otherwise, a failsafe long hangover timer *T* of L_L = 40 frames is used in case the early presence of speech in the utterance has caused the initial noise estimates of the detectors to be too high.

Step 5: If $M < S_P$ AND $T > 0$, T--

If the lesser of the speech likelihood thresholds is not reached, reduce any current hangover time by 1. Thus the hangover timer T only decrements in the likely absence of speech.

Step 6: If T>0 output TRUE else output FALSE

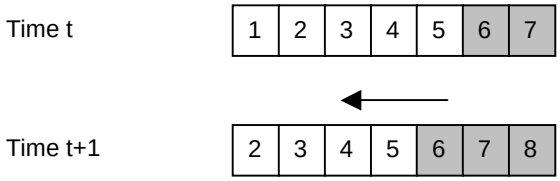
Unless hangover timer T has reached a value of zero, the algorithm outputs a positive speech decision. Because T is given a value immediately upon speech detection and only decrements in the absence of speech, step 6 provides a 'true' output both during speech and for the duration of any hangover. Because the output is applied to the frame about to leave the buffer, it also provides the look-ahead facility.

Step 7: Frame++, Shift buffer left and return to step 1

In preparation for the next frame, left-shift the buffer to accommodate the next input.

As noted above, the output speech decision is applied to the frame being ejected from the buffer. The look-ahead effect this provides is detailed below.

The figure below illustrates the buffer, labelled with the frame number of the result V_N found at that position:



Thus at time t, seven frames have populated the buffer, and the result V_N for frames 6 and 7 was True. Applying the algorithm above, a negative speech decision is applied to frame 1.

At time t+1, left-shifting of the buffer has ejected frame 1, and the result V_N from new frame 8 is True. Applying the algorithm above, a positive speech decision is applied to frame 2. This will also be the case for frames 3, 4 and 5 as subsequent new frames arrive, so forming a 4-frame look-ahead preceding the possible speech in frames 6, 7 and 8.

Assuming only these three inputs are 'True', the full speech decision sequence will be:

Frame No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
V_N result	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Timer value	0	5	5	5	5	5	4	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Speech decision	F	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F

Where frames 2-5 form a look-ahead in anticipation of further incoming speech, whilst frames 9 and 10 provide only a short hangover as this short isolated sequence may not actually be speech. Empirically the value of short hangover duration L_S is a compromise between minimising unwanted noise and providing a couple of frames to bridge speech that is broken up by noise or classification error.

To illustrate this, consider a possible alternative subsequent V_N sequence, for which the full speech decision sequence will be:

Frame No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
V_N result	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
Timer value	0	5	5	5	5	5	4	3	2	1	5	23	23	23	23	22	21	20	19	18	17	16	15	14
Speech decision	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T

The buffer length and hangover timers can be adjusted to suit needs, although the buffer should always be greater than or equal to the SL threshold value. Once results from all frames in the utterance have been added, the buffer shifts until empty whilst still applying the algorithm.

Annex B (informative): Bibliography

IETF Audio Video Transport, Internet-Draft: "RTP Payload Format for ETSI ES 201 108 Distributed Speech Recognition Encoding". http://www.ietf.org/internet-drafts/draft-ietf-avt-dsr-05.txt

History

Document history		
V1.1.1	October 2002	Publication
V1.1.2	October 2003	Publication