RL-TR-97-97
Final Technical Report
October 1997

UNIX SPEECH PROCESSING DEVELOPMENT

Atlantic Coast Technologies, Inc.

Howard Sabrin

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Rome, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-97-97 has been reviewed and is approved for publication.

APPROVED: Edward J. Cupples

EDWARD J. CUPPLES
Project Engineer

FOR THE DIRECTOR:

JOSEPH CAMERA, Technical Director
Intelligence & Reconnaissance Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL/IRAA, 32 Hangar Rd, Rome, NY 13441-4114. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Oct 97	Final Sep 94 - Aug 95

4. TITLE AND SUBTITLE

UNIX SPEECH PROCESSING DEVELOPMENT

5. FUNDING NUMBERS

C -F30602-94-C-0280
PE - 62702F
PR - 4594
TA - 15
WU- M3

6. AUTHOR(S)

Howard Sabrin

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Atlantic Coast Technologies, Inc.
624 Concerto Lane
Silver Springs, MD 20901

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Rome Laboratory/IRAA
32 Hangar Rd
Rome, NY 13441-4114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

RL-TR-97-97

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Edward J. Cupples, IRAA, 315-330-4025

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of the program was threefold: First, develop a Speech Enhancement Unit (SEU) workstation that would run ANSI Standard C versions of the SEU algorithms on a UNIX platform using a user-friendly Graphical User Interface. Second, test the algorithms and quantify their performance, finding, wherever possible, unpredicted aspects of performance. Compare the latest versions of the algorithms with earlier versions. Third, develop, in Matlab, Automatic Gain Control (AGC) algorithms to ensure dynamic range preservation and prevent clipping, at the back end to improve listening levels for the user, and as a spectral tilt compensation mechanism.

14. SUBJECT TERMS

Speech Enhancement, Interference Reduction

15. NUMBER OF PAGES

68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

List of Figures

Algorithm Descriptions

Test and Evaluation

AGC Improvement

Introduction

This document is the final report for Rome Laboratory Contract Number F30602-94-C-0280, entitled **UNIX Speech Processing Development**, and performed by Atlantic Coast Technologies, Inc. Its purpose is to summarize the results of the program and give the reader some insight into the operation of the algorithms implemented or developed under the effort.

The purpose of the program was threefold:

- Develop an SEU workstation that would run ANSI Standard C versions of the Speech Enhancement Unit algorithms on a UNIX platform using a user-friendly Graphical User Interface.

- Test the algorithms and quantify their performance, finding, wherever possible, unpredicted aspects of performance. Compare the latest versions of the algorithms with earlier versions.

- Develop, in Matlab, Automatic Gain Control (AGC) algorithms that can be used
  - at the front end of the SEU algorithms to ensure dynamic range preservation and prevent clipping,
  - at the back end to improve listening levels for the user, and
  - as a spectral tilt compensation mechanism.

This final report will describe the work performed and the accomplishments achieved in each of these categories.

The SEU Algorithms

The Speech Enhancement Unit Algorithms are computer programs that are designed to improve the quality and intelligibility of speech recorded in the harsh environment of tactical operations. The most troublesome interferences encountered during such operations include impulses (caused by electrical interferences coming from environmental static or engines), narrowband tones (caused by communication channel noise, channel drift, competing signals, or outside electrical sources), and broadband noise. Three algorithms were developed to lessen the effects of these noise sources. IMP removes impulse noise through time-domain editing, HITONES (formerly called DSS) removes tonal noises through spectral domain editing, and INTEL removes broadband noise through subtraction in the pseudo-cepstral (see description below) domain.

Development of the SEU algorithms has been a long, ongoing process involving several researchers, developers, and platforms. The purpose of this effort was to develop an ANSI C version of the algorithms that could be used on UNIX machines for both research and operational tasks. The versions were developed to use file inputs and outputs, and do not operate in true

real-time. The following paragraphs describe the workings of the algorithms, and the Graphical User Interface that calls them.

Description of the SEU Algorithms

1. Impulse Modulation Process (IMP) algorithm

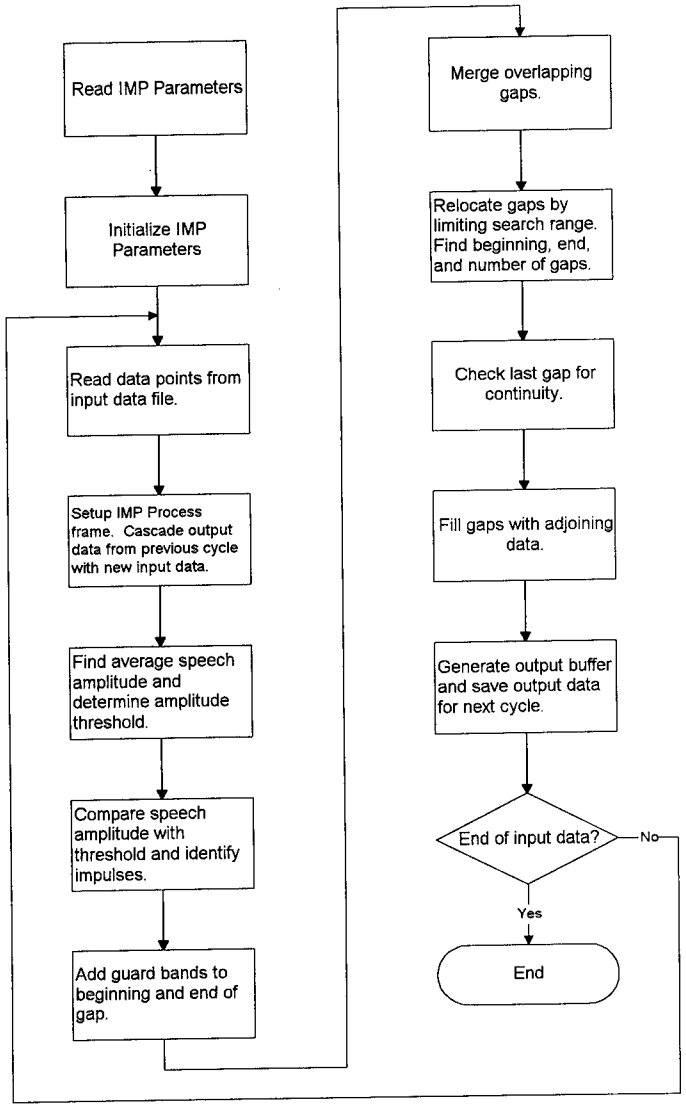
The IMP algorithm examines a noisy input speech in the time domain to determine the presence of impulses - high amplitude, short duration spikes that are typically caused by engines, electrical static, atmospheric interferences, and other sources. They are characterized by "popping sounds" that degrade speech quality and intelligibility and cause listener fatigue. IMP looks at a short strip of the input signal and uses a series of thresholds to identify an impulse. If an impulse is detected, the algorithm removes it by replacing the points in the waveform with a linear interpolation of end points.

IMP operates in the time domain on frames of 256 points (16 msec at 16 KHz sampling). The 256 points used in IMP are derived from an analysis frame of 512 points which are overlapped by 50% to obtain an operational frame rate of 16 msec. IMP operates in two steps:

- Impulse Identification
- Impulse removal and Signal Interpolation.

In the first step, a threshold is set as a user-defined constant multiple of the average absolute amplitude value. All data points above that threshold are marked and buffered by a 200 μsec onset region and a 700 μsec offset region. The marked segments are zeroed, and, in the second step the gaps are filled in with adjoining data that is triangularly windowed and folded over to cover the gap. Figure 1.1 gives a schematic diagram of the IMP algorithm.

Figure 1.1 Data Flow Diagram for IMP Algorithm



1.1 Impulse Identification

The impulse identification scheme initially sets up a 512 point processing window. The first 256 data points of the processing window comes from output data processed during the previous cycle and the second 256 data points of the window from the second half of the current input data window. During each cycle of IMP operation, gaps are filled between the data range defined from point 128 to point 384 of the processing window, thus allowing 128 points of data on either side of the frame for gap filling calculation. An average absolute amplitude value is then computed from the second 256 data points of the processing window. A decision threshold is chosen from an adjustable input threshold range of 2 - 6 times the average absolute amplitude. Regions in the input signal waveform where the absolute amplitude exceeds the decision threshold value are identified as impulses.

1.2. Impulse Removal and Signal Interpolation

Identified impulses are removed from the input signal by replacing the corresponding signal values with zeroes. The zeroed impulse region is further expanded by adding two guard bands LL and LR preceding and following the impulse region. The guard bands LL and LR are set to compensate for the impulse response characteristics of the anti-aliasing filter used in the speech enhancement unit. In the current IMP implementation, LL can vary from 200 to 600 μs and LR could be varied from 700 to 900 μseconds. Overlapping impulse regions and impulse regions too close to each other are combined. The next step involves signal interpolation in the zeroed regions to maintain signal continuity. In the signal interpolation step, signal amplitude points equal to the width of the zeroed region on both sides of the impulse are weighted linearly. The weighting function resembles a ramp, whose weighting factors are computed based on the width of the region and the distance between the signal point from the edge of the region. The weighted signal amplitudes are then folded over and summed over the impulse region. The resulting linear interpolation produces a signal to fill the impulse region. Since IMP only operates on the middle 256 points of the 512 point processing window (points 128-384), it has 128 points of data on both sides of the operational range to validate proper folding of the data for signal interpolation.

2. Tone Removal (Hitones) Algorithm

The Hitones algorithm is a frequency domain peak editor that attenuates narrowband (tonal) noise from the input waveform. Hitones relies on accurate tone detection and careful editing in the frequency domain to ensure maximum noise reduction with minimum speech distortion. The tone detection process exploits the differences between speech and tonal noise. Tones are relatively stationary in both frequency and amplitude when compared with the quasi-stationary speech. In the magnitude spectrum of tone noise, peaks result at the frequencies of tones. In contrast, speech spectrum is smooth over the entire frequency band with smooth peaks at the formant frequencies and finite non-zero bandwidths. Hitones detects tonal noises by identifying narrow

Data Flow Diagram For Hitones Algorithm (page 1)

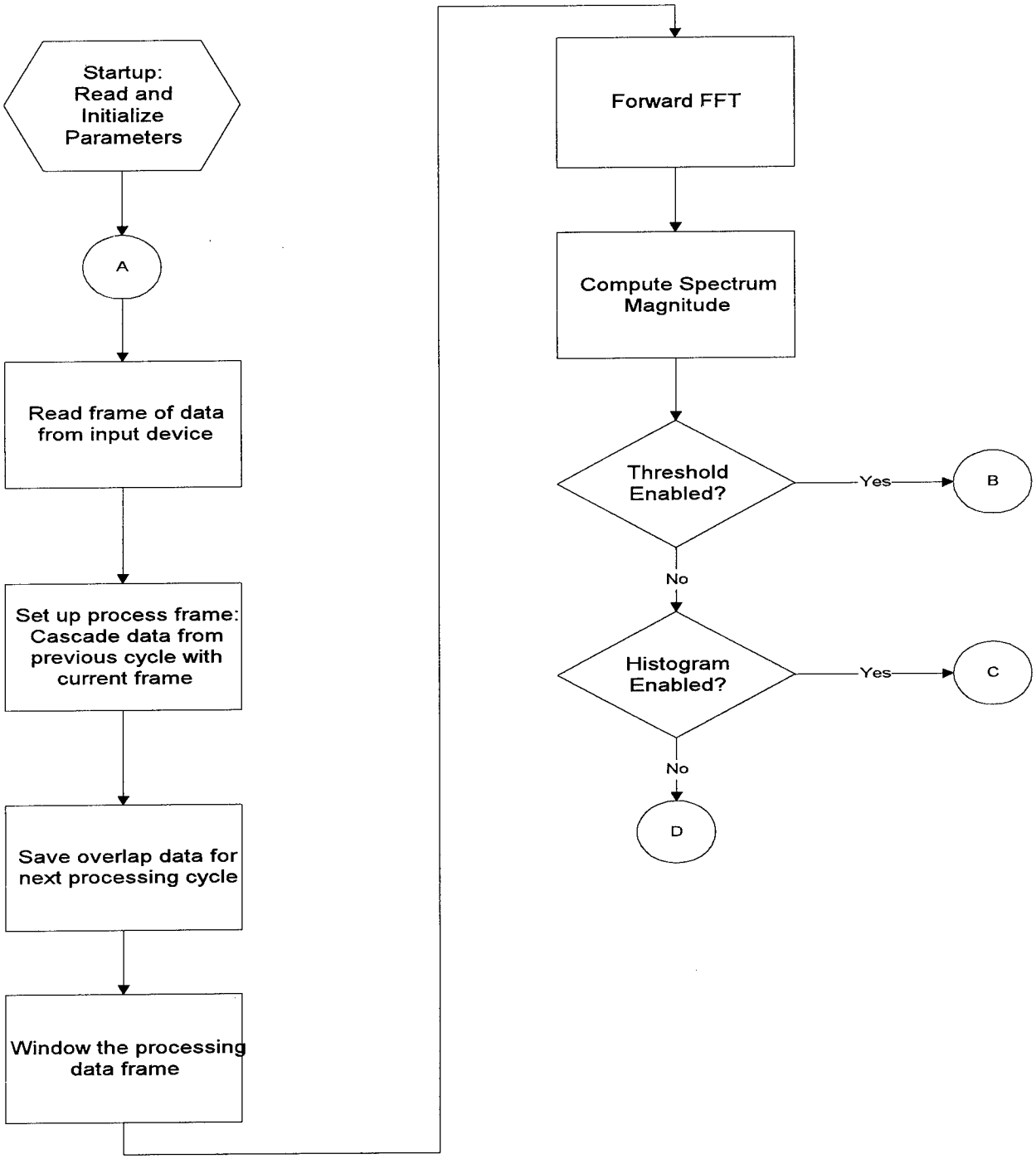


Figure 2-1 Schematic diagram of the Hitones algorithm.

Data Flow Diagram For Hitones Algorithm (page 2)

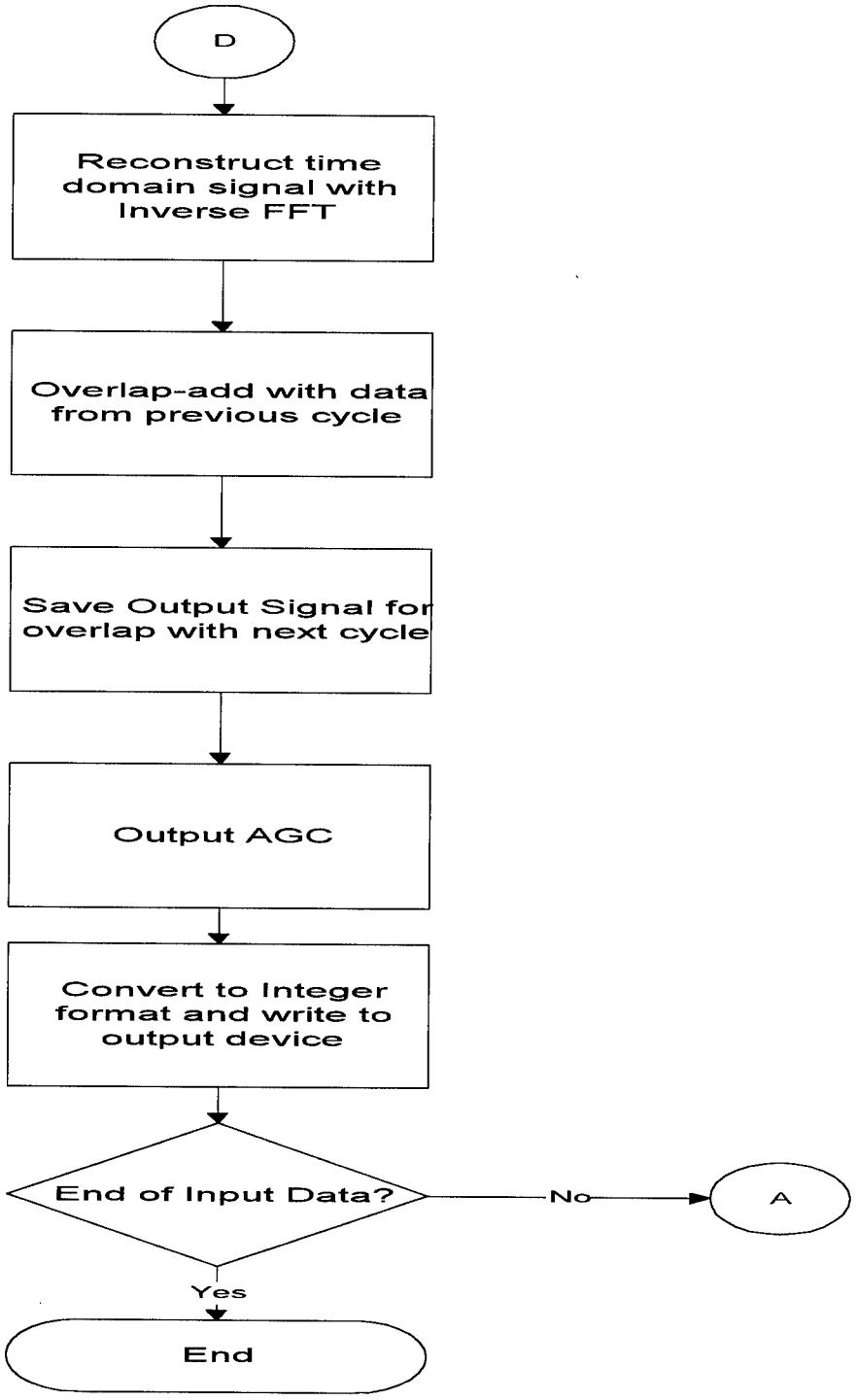


Figure 2-1 (continued)

Data Flow Diagram For Hitones Algorithm (page 3)

Histogram Method for Tone Identification and Removal

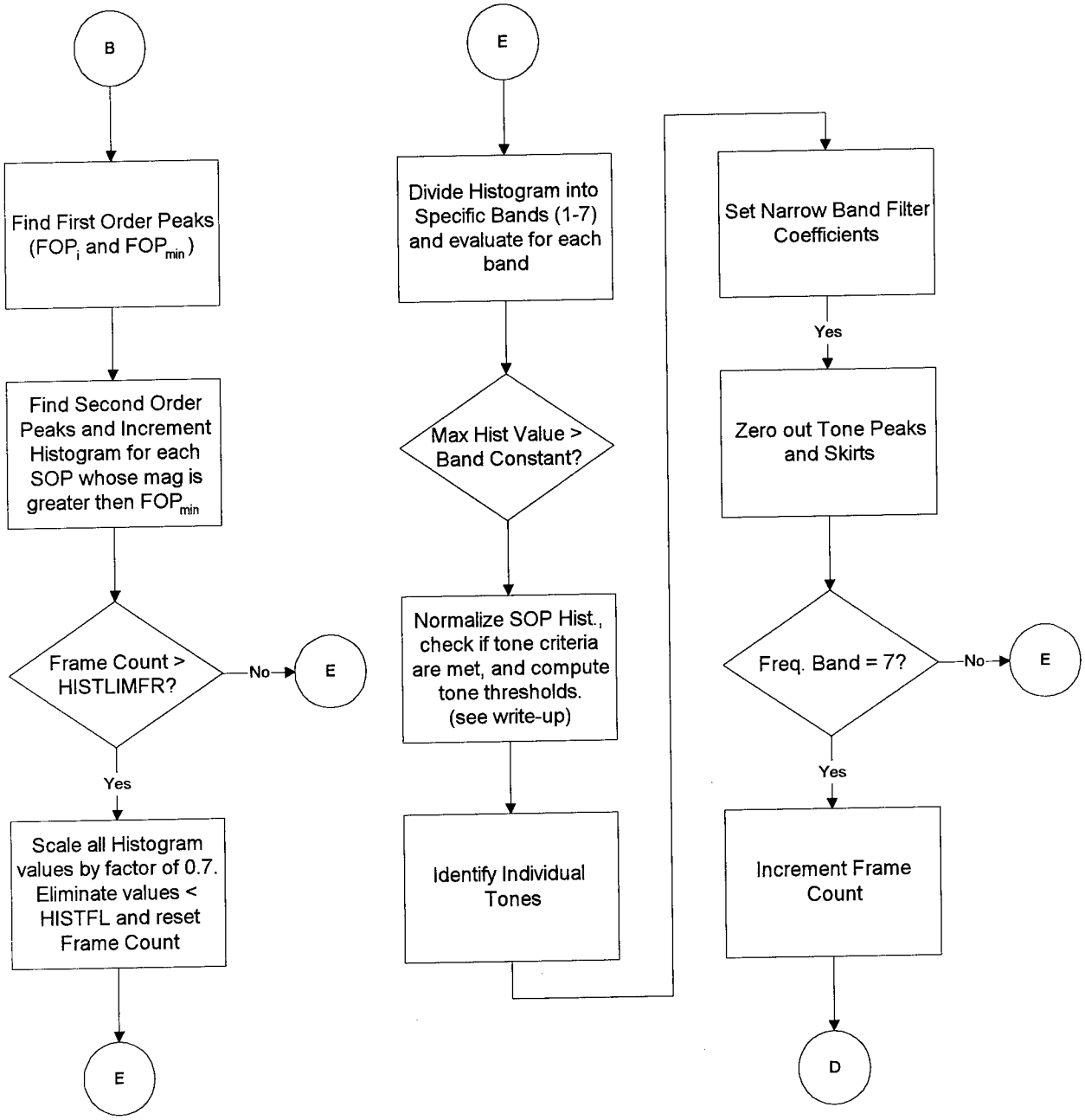


Figure 2-1 (continued)

Data Flow Diagram For Hitones Algorithm (page 4)

Threshold Method for Tone Identification and Removal

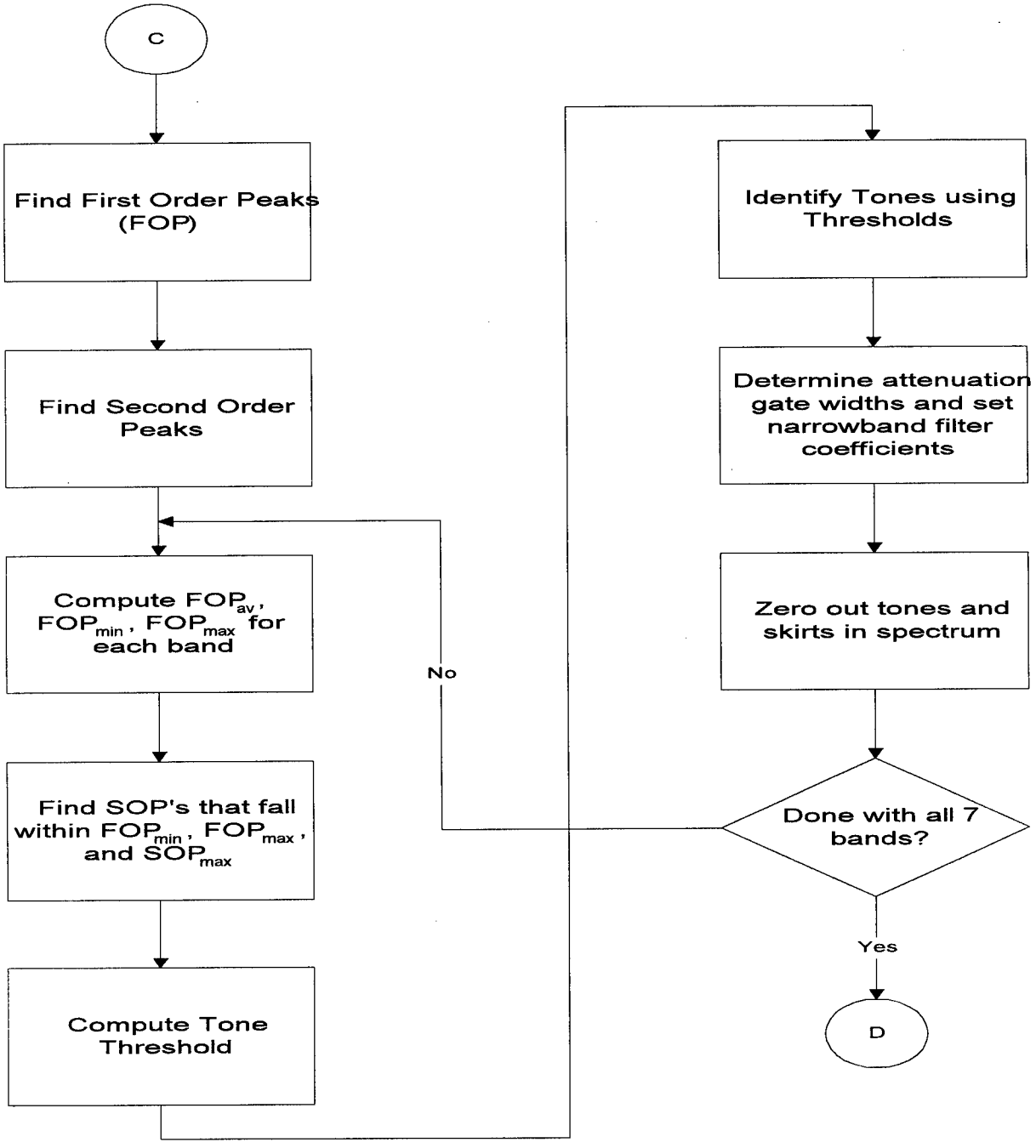


Figure 2-1 (continued)

high energy spectral peaks using two methods: (1) Thresholding, and (2) Histograms. In the threshold technique, the entire frequency band is divided into 7 frequency bands, and, for each band, a fixed threshold factor is set. First order peaks in the spectral domain are defined as local maxima in the spectrum. Second order peaks are local maxima among the first order peaks. A stable tone threshold within the band is determined using the maximum second order peak magnitude, together with the average first order spectral peak magnitude and average first order spectral peak magnitude scaled by the respective threshold factor for the band. The spectral magnitudes that exceed the stable tone threshold are identified as tones and are edited out of the spectrum by multiplying the spectral values within the peak region through multiplication by a constant that corresponds to a desired attenuation level. In the histogramming approach, a spectral peak histogram is derived and any detected tone is polled into the histogram count. The histogram is divided into seven bands and by thresholding in each band, the stable tones are identified and removed. This attenuation can be applied to multiple peaks, allowing for efficient one-pass filtering of all interfering tones that are detected. Figure 2-1 gives a schematic diagram of the Hitones algorithm.

Hitones works on windowed and overlapped data, with a switchable analysis window size of 1024 points or 2048 points which are overlapped by either 50% or 25 %. Table 2.1 shows the processing rates achieved through different combinations of frame size and overlap.

Table 2.1. Processing rate of Hitones (msec/cycle) versus Frame size and overlap.

Frame Size \ Overlap	25%	50%
1024	48	32
2048	96	64

The larger analysis window gives a higher frequency resolution (7.825 Hz/point at a 16 KHz sampling rate) at the expense of a lower time resolution while the smaller analysis window gives a lower frequency resolution (15.625 Hz/point at a 16 KHz sampling rate) in return for a higher time resolution. Overlap segment size input data from previous processing cycle is cascaded with the data read in during the current processing cycle to form the Hitones analysis frame. The analysis data is windowed using a triangular or a cosine window and transformed to the frequency domain by an forward FFT operation for tone identification and tone editing. After tone editing, an inverse FFT is performed on the spectrum to generate time domain data. Final Hitones output is derived by adding the overlap segment of the processed analysis frame with the overlap segment of analysis data processed during previous cycle. This overlap add procedure de-emphasizes the effect of the windowing process.

2.1 Tone Identification by Thresholding

In threshold detection, the entire frequency band is divided into 7 bands and for each band, a stable tone threshold is computed. Spectral magnitudes that exceed the stable tone threshold are identified as tones. The steps in the method are (1) Tone threshold computation and tone identification, (2) Tone extraction/removal.

2.1.1 Tone Threshold computation and identification of tones

First, Hitones must find the first and second order spectral peaks from the spectrum magnitude. The first order peaks (FOP) are identified as local maxima in the spectrum found by peak picking using a three point sliding window. Within the window, the spectral point whose magnitude exceeds that of the adjacent points is retained as a first order peak. The second order peaks (SOP) are identified as the local maxima of the first order spectral peak magnitudes. For each frequency band, the average spectral magnitude of the first order peaks FOP_{av} is computed excluding the maximum first order peak as in Equation 2.1.1

$$FOP_{av} = \frac{1}{N} \sum_{i=1}^{N} S_{fop}(i) - FOP_{\max} \qquad (2.1.1)$$

where $S_{fop}(k)$ is the spectral magnitude at the first order peak locations
FOP_{max} is the maximum first order spectral peak in the band
N is the number of first order peaks

Using FOP_{av}, the minimum first order peak magnitude FOP_{min} and maximum first order peak magnitude FOP_{max} are computed as in Equation 2.1.2 a-b

$$FOP_{min} = FOP_{av} * 1.001 \qquad (2.1.2\ a)$$
$$FOP_{max} = FOP_{av} * r \qquad (2.1.2\ b)$$

where r is the band threshold factor

The band threshold factors are given in table 2.1-1.

Table 2.1-1 Frequency bands and threshold factors

Bands	Frequency Range in KHz	Threshold Factors
1	0.000 to 0.793	9.1

Bands	Frequency Range in KHz	Threshold Factors
2	0.780 to 1.859	5.2
3	1.833 to 2.743	5.2
4	2.730 to 3.640	5.2
5	3.627 to 5.200	5.2
6	5.070 to 6.500	5.2
7	6.370 to 8.000	5.2

The second order peak magnitudes that fall within the range FOP_{min} and FOP_{max} are searched and the maximum SOP magnitude SOP_{max} is found. A tone identification threshold b_{st} is calculated as in Equation 2.1.3

$$b_{st} = 1.1 * SOP_{max} \quad \text{if } SOP_{max} > 0 \qquad (2.1.3\ a)$$

$$b_{st} = 3.9 * FOP_{av} \quad \text{if } SOP_{max} = 0 \qquad (2.1.3\ b)$$

The region in the band where the spectrum magnitude exceeds the tone identification threshold is classified as a tonal region and tone removal/extraction is carried out.

2.1.2 Tone Attenuation/Extraction

The tone identification threshold is used to find the tonal region in the band as follows: Find the maximum spectral magnitude in the region where the spectrum magnitude exceeds the tone threshold and define the maximum as tone peak magnitude. Find the location where the spectral magnitude exceeds the tone identification threshold (beginning location) to the location where the spectral magnitude falls 30 dB below the tonal peak magnitude (end location) around the tonal peak. This region around the tonal peak is defined as tonal region and the mid point of the tonal region is determined. Attenuation gates are set for each tone for tone removal. The width of the attenuation gate is defined as the region around the tonal peak where the spectral magnitude is 30 dB below the tonal peak magnitude. The width of the attenuation gate is determined as follows: Start with the beginning and end of tone region and search over a region MAXFILTERWIDTH (set as 7) on either side from mid point of the tonal region. The search is terminated if the spectral magnitude within the search region falls below the 30 dB tone amplitude. Thus the maximum width of the attenuation gate is limited to MAXFILTERWIDTH. Within the attenuation region, narrow band filter weights are set to attenuate/extract the tone and tonal peak and the skirts in the spectrum are zeroed out.

Once tones are identified and respective attenuation gates are set, Hitones operation allows for either tone attenuation or tone extraction. If tone attenuation is desired, the real and imaginary spectrum within the set attenuation gates are multiplied by 0.001 to accomplish the desired 10 dB attenuation and four frequency points to the left and right of the attenuation gates are

multiplied by 0.03, 0.1, 0.2 and 0.4 to smooth out the discontinuities in the spectrum around the attenuated tone. If tone extraction is desired, the real and imaginary spectrum within the attenuation gates are multiplied by 1.0 and four frequency points to the left and right of the attenuation gates are multiplied by 0.4, 0.2, 0.1 and 0.003 to effect complete tone extraction.

2.2 Tone Identification using Histograms

In this method, a frequency histogram of spectral peaks is derived to track the occurrence of tones. If a tone is stable over more than few frames, its histogram value, which is updated and incremented each frame, will indicate a stable tone that is a candidate for removal. The histogram is divided into seven bands and by thresholding in each band, the stable tones are identified. The steps in the method are (1) Derive a second order spectral peak occurrence histogram, (2) Compute stable tone threshold and identification of tones in the band, (3) Tone Attenuation/Extraction

2.2.1 Second order spectral peak occurrence histogram

First and second order spectral peaks from the spectrum magnitude in the same manner as described above for the threshold method. The first order peaks are identified as local maxima in the spectrum by peak picking using a three point sliding window. Within the window, the spectral magnitude point that exceeds the adjacent magnitude points is retained as first order peak. A first order minimum peak value FOP_{min} is determined by the Equation 2.2.1 as

$$FOP_{min} = \frac{1}{N} \sum_{k=1}^{N} S_{fop}(k) \qquad (2.2.1)$$

where N_{fop} is the number of non-zero first order peaks
$S_{fop}(k)$ is the spectral magnitude of the first order peaks

The second order peaks, SOP are identified as the local maxima of the first order spectral peak magnitudes by peak picking. Once the second order peaks are identified, a SOP occurrence histogram is updated if the second order peak magnitude $S_{sop}(k)$ exceeds the minimum first order peak magnitude value FOP_{min} in the frequency band below 3.9 KHz. For the SOP's identified in the frequency band above 3.9 KHz, the SOP occurrence histogram is updated if SOP magnitude exceeds 0.5 times FOP_{min}. Thus, the SOP histogram entries are updated as given by Equation 2.2.2.

$$\begin{aligned} S_{sop}(k) &> FOP_{min} && \text{for } k < 3.9 \text{ KHz} \\ S_{sop}(k) &> 0.5 * FOP_{min} && \text{for } k > 3.9 \text{ KHz} \end{aligned} \qquad (2.2.2)$$

Every HISTLIM (currently set to 7) frames, the SOP histogram is scaled by a factor of HISTSF (set to 0.7) and histogram values below HISTFL (set as 1.0) are eliminated. This scaling serves to allow histogram values to decay out after a tone ceases to be present.

2.2.2 Stable tone threshold computation and identification of tones in the band

The SOP histogram is divided into 7 frequency bands and each band is analyzed separately for identifying stable tones. Within each frequency band, the maximum value in the histogram H_{max} is found. This maximum value is compared with a predefined histogram band constant for the particular band. The histogram band constants for each band is given in Table 2.2-1.

Table 2.2-1 Frequency bands and histogram band constants

Bands	Frequency Range in KHz	Band constants
1	0.000 to 0.793	7.0
2	0.780 to 1.859	7.0
3	1.833 to 2.743	7.0
4	2.730 to 3.640	5.0
5	3.627 to 5.200	5.0
6	5.070 to 6.500	5.0
7	6.370 to 8.000	5.0

If the maximum value in the histogram is less than the band constant, no stable tones are found in the band and the next frequency band is searched. But if the maximum value in the histogram exceeds the histogram band constant, it is determined that stable tones are present in the frequency band and the algorithm proceeds to identify those tones. A stable tone threshold value is determined using the following steps: Normalize the SOP histogram to a value of 10. The normalized histogram entries that are above a value of 4.81 are summed and a normalized histogram average H_{nav} is computed as in Equation 2.2.1

$$H_{nav} = \frac{1}{N} \sum_{k=1}^{N} H_n(k) \quad if\ N > 3 \qquad (2.2.1a)$$

$$H_{nav} = 0 \qquad if\ N < 3 \qquad (2.2.1b)$$

where H_n is the normalized histogram values for the band
N is the number of entries that are above a value of 4.81

The average histogram values are further readjusted as follows :

$$H_{nav} = H_{nav} + 0.39 \quad \text{if } H_{nav} > 0 \qquad (2.2.2)$$

$$H_{nav} = 5.2 \quad \text{if } H_{nav} = 0 \text{ and } H_{max} > 10.4 \qquad (2.2.3)$$

$$H_{nav} = h \quad \text{if } H_{nav} = 0 \text{ and } H_{max} > 19.5 \qquad (2.2.4)$$

where h is minimum normalized histogram average constant.

The minimum normalized histogram average constant for each band is given in Table 2.2-2.

Table 2.2-2 Frequency bands and minimum normalized histogram average constants

Band	Frequency Range in KHz	Min. Average Constant
1	0.000 to 0.793	3.25
2	0.780 to 1.859	3.25
3	1.833 to 2.743	3.25
4	2.730 to 3.640	3.25
5	3.627 to 5.200	2.60
6	5.070 to 6.500	2.60
7	6.370 to 8.000	2.60

If Equations 2.2.3 and 2.2.4 are not satisfied, H_{nav} is set as

$$H_{nav} = 6.5 \qquad (2.2.5)$$

Using the readjusted normalized average histogram value H_{nav} , the stable tone threshold b_{st} is determined and is given in Equation 2.2.6 as

$$b_{st} = \text{MIN} (H_{nav} , 9.88) \qquad (2.2.6)$$

The stable tone threshold determines if the candidate peaks have sufficient energy to be called a tone. Once the stable tone threshold is determined, the normalized histogram is searched and values that exceed the stable tone threshold for the band are identified as tones. The spectral magnitude corresponding to identified tone in normalized histogram give the tone amplitude to be attenuated.

2.2.3 Tone Extraction/Removal

Narrow band filter coefficients are set up at identified tone locations and locations to the left and right of the tone. The width of the attenuation gate is set as shown in Table 2.2-3.

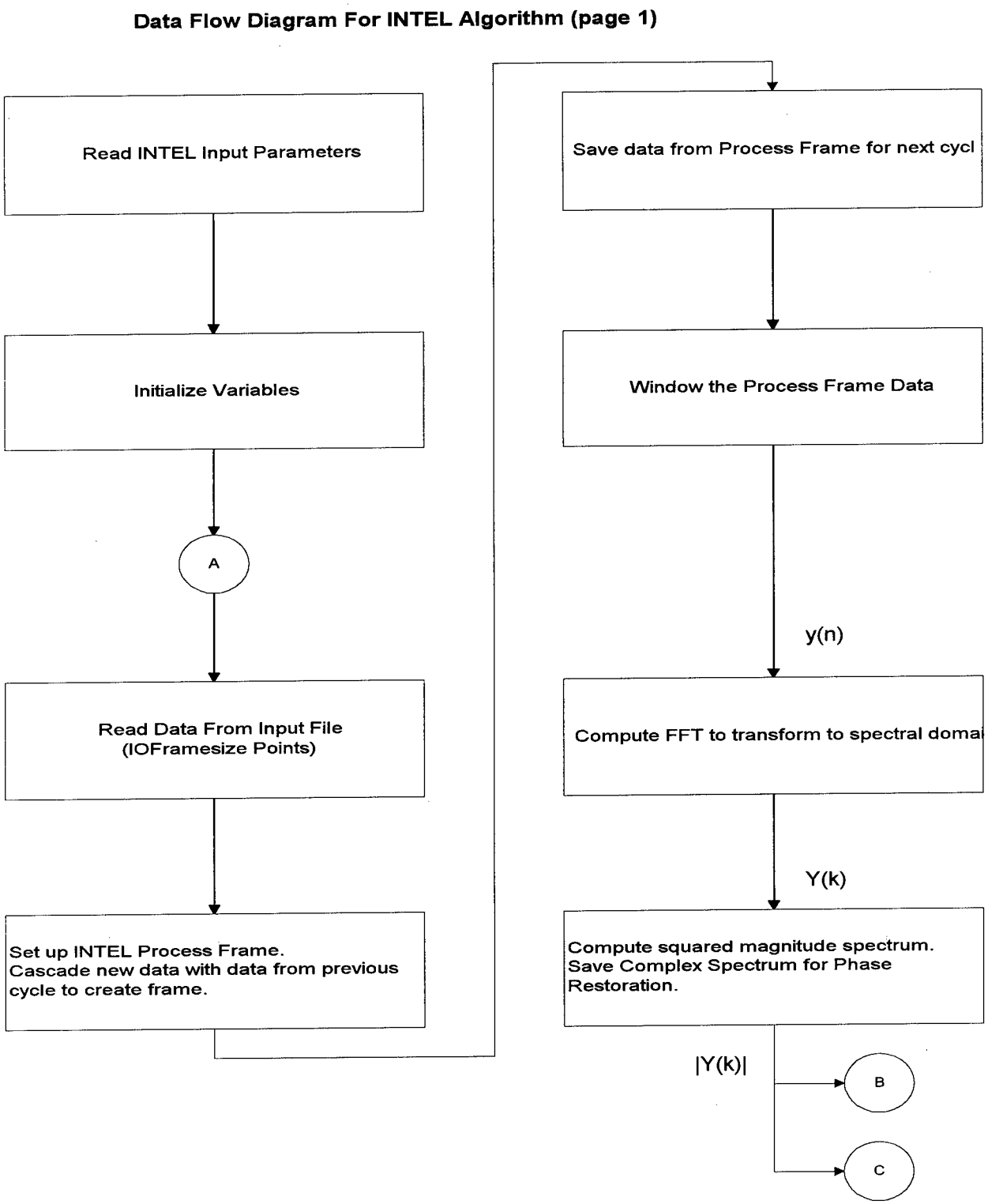


Figure 3-1 Schematic diagram of the INTEL algorithm.

Data Flow Diagram For INTEL Algorithm (page 2)

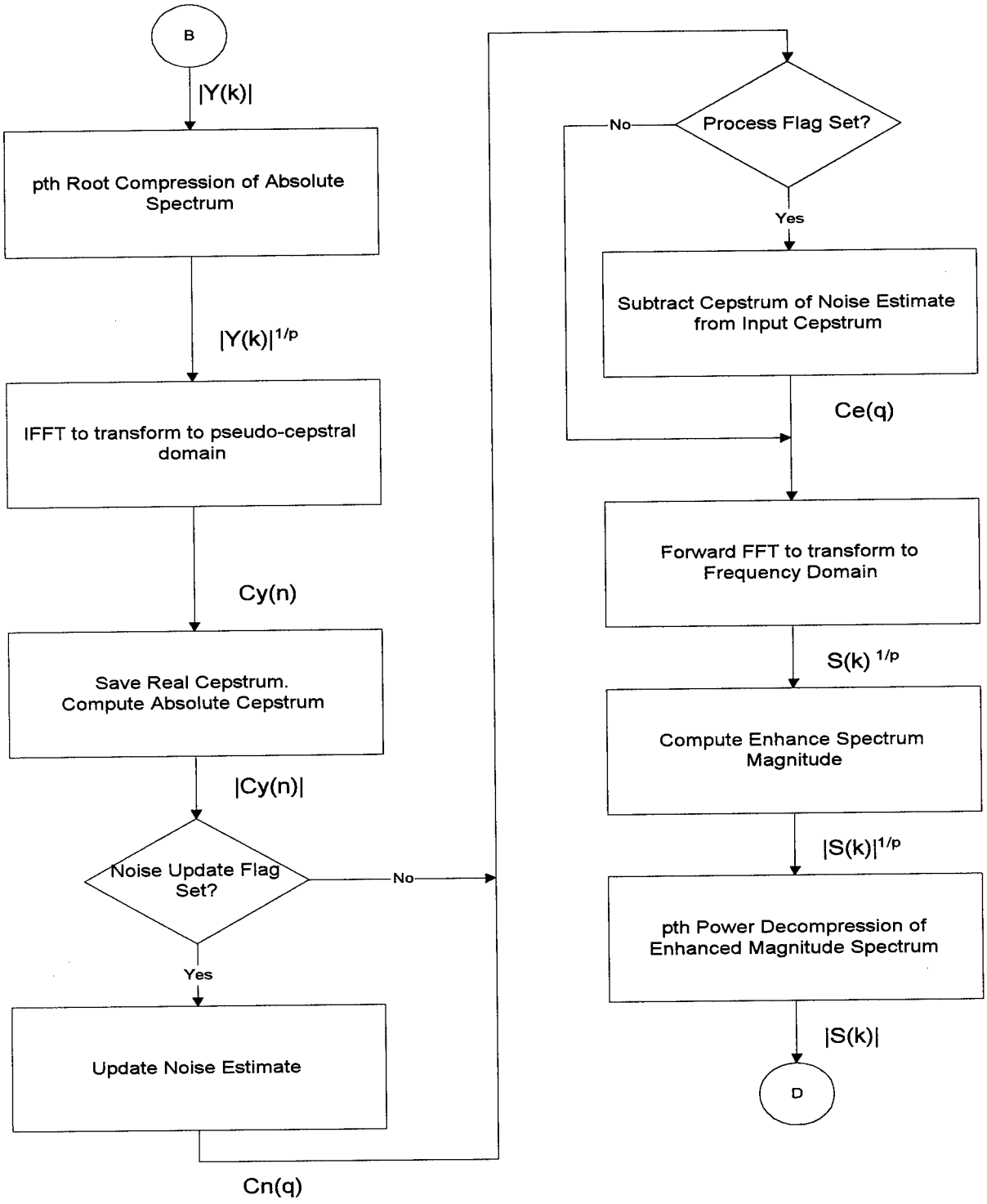


Figure 3-1 (continued)

Data Flow Diagram For INTEL Algorithm (page 3)

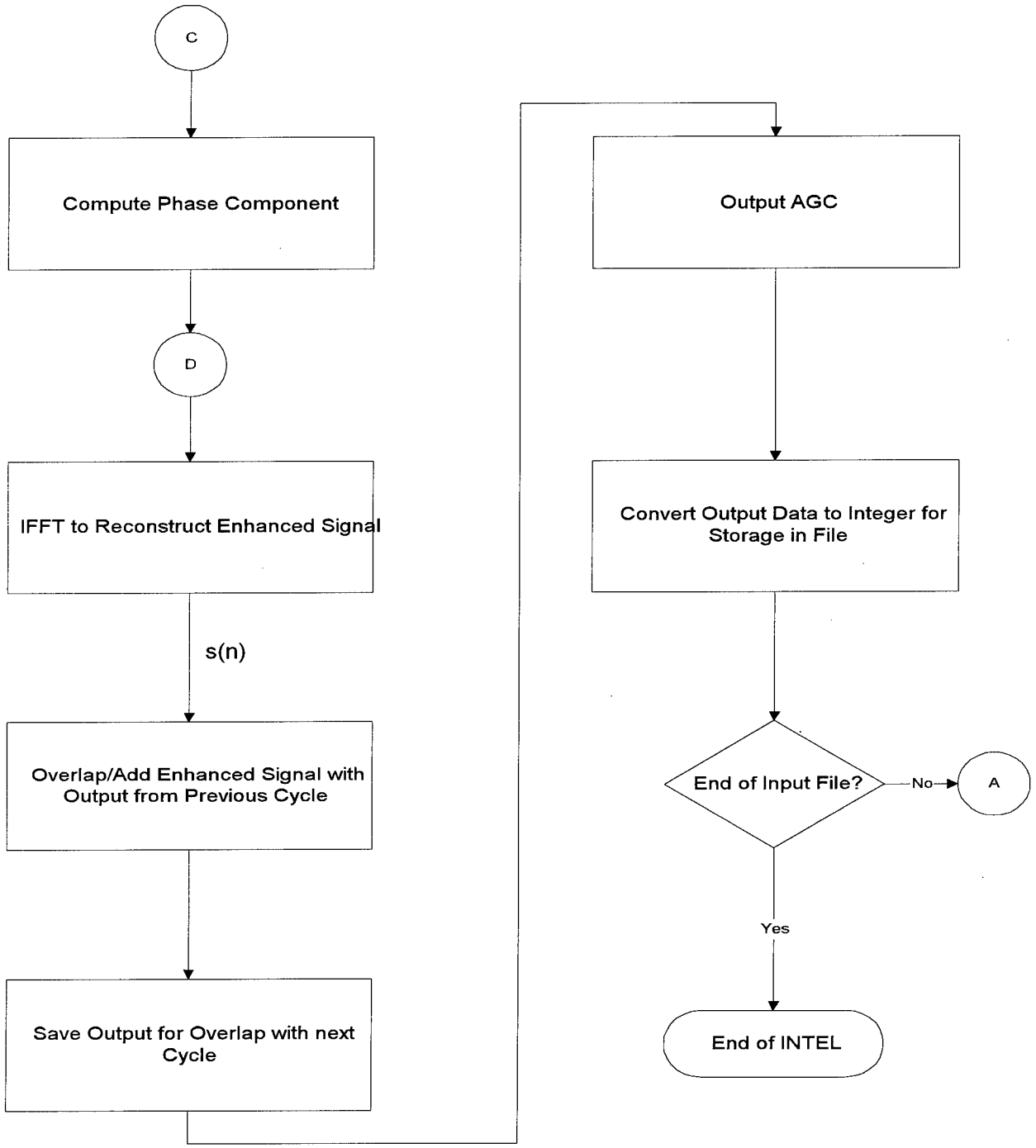


Figure 3-1 (continued)

Table 2.2-3 Frequency bands and attenuation gate width

Band	Frequency Range in KHz	Attenuation Gate Width
1	0.000 to 0.793	11
2	0.780 to 1.859	11
3	1.833 to 2.743	11
4	2.730 to 3.640	13
5	3.627 to 5.200	13
6	5.070 to 6.500	15
7	6.370 to 8.000	15

Within the attenuation region, narrow band filter weights are set to attenuate/extract the tone and tonal peak and the skirts in the spectrum are zeroed out. Hitones operation allows for either tone attenuation or tone extraction. If tone attenuation is desired, the real and imaginary spectrum within the set attenuation gates are multiplied by 0.001 to accomplish the desired 10 dB attenuation and four frequency points to the left and right of the attenuation gates are multiplied by 0.03, 0.1, 0.2 and 0.4 to smooth out the discontinuities in the spectrum around the attenuated tone. If tone extraction is desired, the real and imaginary spectrum within the attenuation gates are multiplied by 1.0 and four frequency points to the left and right of the attenuation gates are multiplied by 0.4, 0.2, 0.1 and 0.003 to effect complete tone extraction.

3. INTEL algorithm

INTEL is the algorithm that measures and removes broadband noise from noisy speech signals. The INTEL algorithm is based on homomorphic filtering and subtraction in the pseudo-cepstral domain. Experiments with the INTEL algorithm have shown improvement in noise removal capability when compared to simple spectral subtraction. The pseudo-cepstrum is defined as the inverse FFT of the nth root of the FFT of a signal. The true cepstrum uses the logarithm in place of the nth root. In the cepstral domain, broad band noise is compressed around the origin and above a quefrency of 0.5 msec, the noise energy falls off quickly while speech energy is still present at the pitch period and its harmonics. Therefore, if a noise-only cepstrum can be estimated, subtracting it from the speech plus noise cepstrum can efficiently reduce the broad band noise. The significance of noise reduction depends on the estimation of the noise cepstrum. To accomplish this, a moving average of the noise cepstrum is formed, and the noise cepstrum is then able to follow changes in the noise distribution. An optimal set of weights for the moving average filters are chosen and the noise cepstrum is updated for each processing cycle whether the incoming frame contains speech or only noise. Previous versions of INTEL utilized a speech

noise classifier in an attempt to estimate noise cepstra using noise frames alone. The new version of INTEL (sometimes referred to as INTEL95) simply averages all frames into the noise estimate, without making a voiced/unvoiced decision. The estimated noise cepstrum is then subtracted from the input cepstrum for noise removal. Figure 3-1 gives a schematic view of INTEL.

INTEL operates on data frames that can be either 512 or 1024 points long, windowed with either triangular or cosine functions, and analyzed with either 50% or 25% overlap. Table 3-1 shows the processing rate for various combinations of frame rate and overlap, assuming 16 Khz data sampling.

Table 3-1. Processing rate of INTEL (msec/cycle) versus Frame size and overlap.

Frame Size \ Overlap	25%	50%
512	24	16
1024	48	32

An INTEL process frame is set up with the input data from the previous and current processing cycle. The frame is formed by cascading a segment from the previous processing cycle with the data read in during the current processing cycle. The windowed frames are transformed into the spectral domain with forward FFT operations. The p^{th} root (p is generally either 2 or 4) of the spectral magnitude and phase are computed for each spectral point. The spectral magnitude is compressed by raising to 1/p and is transformed to the pseudo-cepstral domain by applying an inverse FFT on the compressed spectral amplitude. A weighted estimate of the noise pseudo-cepstrum is then updated and subtracted from the input pseudo-cepstrum. The noise reduced signal is regenerated by performing the inverse of pseudo-cepstral transformation on the results of cepstral subtraction. An inverse pseudo-cepstral transformation performs the following operations: taking a FFT back to the frequency domain, decompressing the spectral amplitude by raising to the p^{th} power, re-inserting the saved phase components, and taking an inverse FFT back to the time domain. A block diagram of the INTEL algorithm is shown in Figure 3.1. Four major functions of INTEL algorithms that will be described in the following sections are: (1) Pseudo-cepstral transformation, (2) Noise pseudo-cepstrum estimate and subtraction, (3) Inverse pseudo-cepstral transformation, and (4) Output automatic gain control (AGC) process.

3.1 Pseudo-cepstral transformation

The input noisy speech signal y(n) is composed of a clean speech component, s(n) and an additive broad band noise component, z(n) as described in Equation 3.1

$$y(n) = s(n) + z(n) \tag{3.1}$$

where n is the discrete time index. The noisy input signal is windowed and the windowed time domain data are then transformed to the spectral domain by performing a fast Fourier transform operation. Thus Equation 3.1 becomes

$$Y(k) = S(k) + Z(k) \tag{3.2}$$

where Y(k) is the complex spectrum of the input signal y(n), and k is the discrete frequency index. The magnitude spectrum of Y(k) is then obtained by squaring and summing the complex terms and taking the square root to give

$$| Y(k) | = | S(k) | + | Z(k) | \tag{3.3}$$

The complex spectrum is retained for phase recovery in final signal synthesis after noise cepstral subtraction. The magnitude spectrum $|Y(k)|$, is then transformed into the root-cepstral domain by taking the p^{th} root compression followed by an inverse FFT (IFFT). The pseudo-cepstrum of the input signal $C_y(n)$ is given by Equation 3.4 as

$$C_y(n) = \text{IFFT} \{ |Y(k)|^{1/p} \} \tag{3.4}$$

The compression factor, p, can be chosen to be 2 (second root) or 4 (fourth root). A fourth root cepstrum causes a greater compression of the noise component toward the origin in the cepstral domain. This permits the noise to be subtracted more efficiently and hence yield a greater improvement in SNR.

3.2 Noise Pseudo-cepstrum Estimate And Subtraction

The noise pseudo-cepstrum estimate is computed for all signal frames, whether speech or non speech. For the incoming signal frame, an update noise pseudo-cepstrum estimate is updated by means of Equation 3.5:

$$C_n(q) = \beta_m(q) * |C_y(q)| + (1-\tau(q)) * C_p(q)] \tag{3.5}$$

where
- $C_n(q)$ is the new updated noise pseudo-cepstrum estimate.
- $C_p(q)$ is the most recent noise pseudo-cepstrum estimate.
- $C_y(q)$ is the pseudo-cepstrum of incoming signal frame.
- τ (q) is the noise threshold update time constant vector.
- β_m (q) is the weighting vector that combines noise threshold update time constant τ (q) with noise category row m of noise threshold multipliers vector γ_m
- q is the index in the pseudo-cepstral domain.

The weighting vector β_m (q) is computed using Equation 3.6 as

$$\beta_m(q) = \begin{cases} \gamma_m(0) * \tau(q) & q=0 \\ \gamma_m(1) * \tau(q) & q=1,5 \\ \gamma_m(2) * \tau(q) & q=6,N-1 \end{cases} \quad (3.6)$$

where
- τ (q) is noise threshold update time constant vector.
- γ_m is noise threshold multiplier factors for desired noise category m
- m is noise category index and can be set to 0,1,2. This selects scale factors for a particular SNR. Three SNR conditions are specified corresponding to low, high and adjustable
- N is the size of the pseudo-cepstrum

The noise category m refers to the expected SNR of the incoming signal, and determines how much noise will be subtracted out. Each noise category is associated with a vector of γ factors. In each γ vector of three numbers, the first number is the multiplier for the first cepstral coefficient, the second is the multiplier for the next five coefficients, and the third is the multiplier for the remaining coefficients. The γ factors can be adjusted individually through the GUI.

Since the average noise cepstrum is built up over time, any long sections of silence (1 sec or more) drives the average noise cepstrum to zero. When the signal reappears, a brief noise burst results until the noise cepstrum can be reestablished. The choice of update time constant weights τ(q) depends on the speed with which the system tracks changes in the noise characteristics.

Noise removal occurs through pseudo-cepstrum subtraction of the noise estimate C_n(q) from the input pseudo-cepstrum C_y(q) in the pseudo-cepstral domain. Pseudo-cepstrum subtraction is performed according to the relation where the enhanced pseudo-cepstrum C_e(q) retains the same polarity of the input cepstrum C_y(q) as shown in Equation 3.8.

$$C_e(q) = SIGN[C_y(q)] * MAX<[\ |C_y(q)| - C_n(q)],\ 0> \quad (3.8)$$

where
- C_n(q) is the new noise pseudo-cepstrum estimate.
- C_y(q) is the pseudo-cepstrum of input signal.
- q is the index in the pseudo-cepstral domain.

3.3 Inverse Pseudo-cepstral Transformation

The enhanced cepstrum $C_e(q)$ is transformed back to the frequency domain using a forward FFT operation and the spectral amplitude is decompressed by raising the spectral magnitude components to the p^{th} power. The decompressed spectral amplitude is combined with the saved phase components and an inverse FFT is performed to obtain the noise removed speech s(n) in the time domain. Final INTEL output is obtained by overlap adding the output of current processing window with the output from previous processing cycle over the overlap segment of data .

3.4 Output automatic gain control (AGC) process

Output automatic gain control (AGC) process is a digital operation that optimizes the level of the digital data for D/A conversion and minimizes sudden changes in the volume of output analog signal. The output AGC process consists of a dual-threshold system that compares either the frame average root mean square value (RMS AGC) or the largest peak value in the frame (peak AGC) to a high threshold and a low threshold to regulate the output gain value. For a 16-bit D/A converter using RMS AGC, the high RMS threshold value is set equal to 10000, and the low RMS threshold value is set equal to 750. For peak AGC, the high peak threshold value is set equal to 14336, and the low RMS threshold value is set equal to 560. Output AGC processes INTEL output over the non overlapping segment of data. Frame root mean square (RMS) is computed and low passed filtered as shown in Equation 3.9.

$$RMS = \sqrt{\frac{\sum_{i=1}^{N} s(i)}{N}} \qquad (3.9)$$

$$RMS_{av} = 0.95 * RMS_{av} + 0.05 * RMS$$

where s(i) is the INTEL processed signal output,
RMS_{av} is the filtered average RMS value.

The product of the updated average root mean square value and previous gain is computed and compared to the high and low thresholds. If the computed product exceeds the high threshold, gain value is reduced by a 0.6 factor. If the computed product is below the low threshold, gain value is increased by a 1.2 factor. If the computed product is between the high and low thresholds, gain value stays unchanged. Maximum gain value is limited to 50 times minimum gain and minimum gain value is set by the user as manual gain. Gain value is updated only once per frame.

3.5 Data output process

The AGC output is converted from floating point number to 16-bit fixed point integer and written to an output data file.

4 Graphical User Interface for the SEU Algorithms

An integrated Graphical User Interface was developed using the XVP Development Package running under MOTIF. The GUI will run the SEU algorithms on any UNIX platform equipped with X-Windows. The package was expanded by the creation of new widgets required to allow the program to display speech signals on the screen. Figure 4-1 shows a sample screen from the GUI, with the INTEL parameter window opened. Directions for using the GUI are given in Appendix III.

The GUI requires MOTIF to be resident on a machine on which the program will be compiled. If the target machine does not have MOTIF, then a static, pre-compiled executable version of the package must be used.

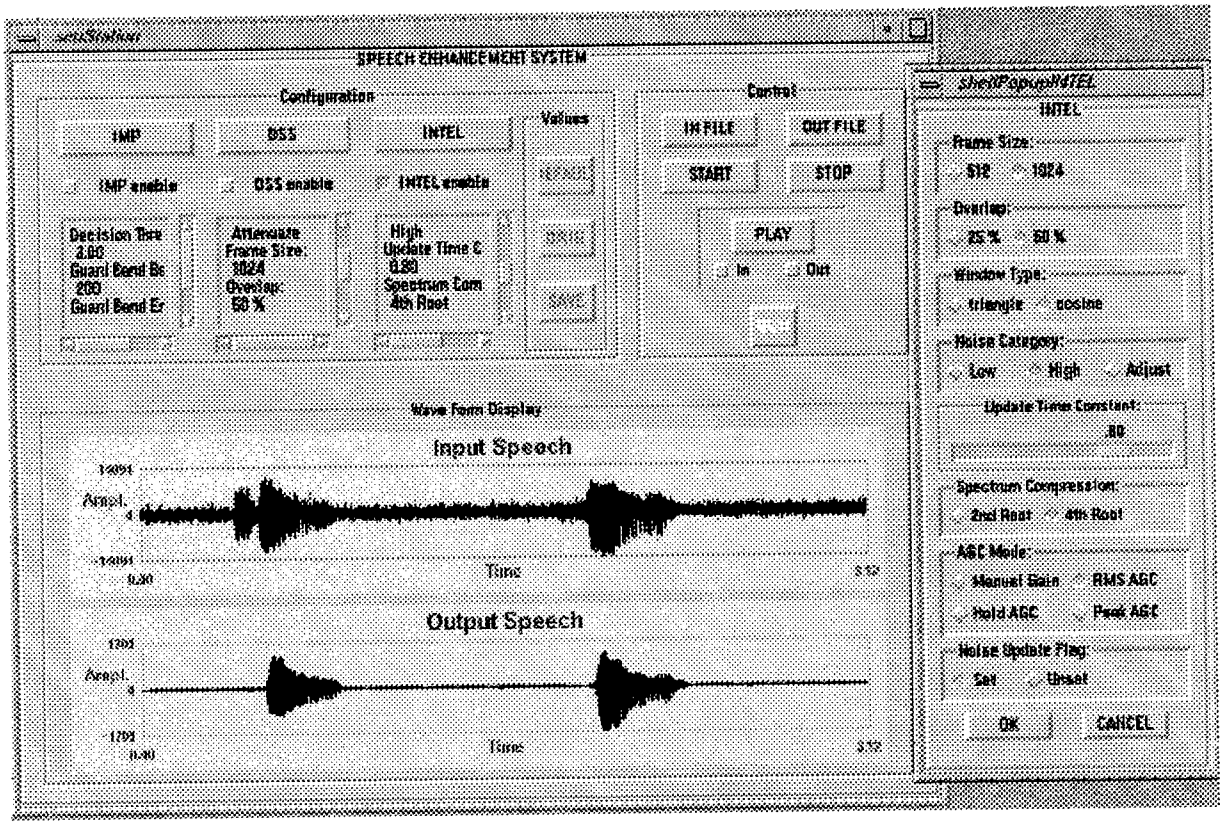


Figure 4-1 Sample screen from the Graphical User Interface

Test and Evaluation of the SEU Algorithms

In this section, the results of a variety of tests designed to measure the performance of the SEU algorithms are given. For each test, a summary of the test description from the approved testplan is included, followed by the test results and comments.

1 IMP Evaluation

1.1 Impulse Amplitude Range

Test Description

There is no set amplitude threshold above which signal segments are identified as impulses and are removed. Instead, an impulse is defined as a multiple of the average absolute value of the frame. For stochastic signals, such as speech or white noise, the average can only be approximated or measured after signal generation from the signal statistics. For the sinusoid, however, it can be predicted with reasonable accuracy. The long-term average absolute amplitude of the signal $\mathbf{Y = Asin(\omega t)}$ is $\mathbf{Avg = A/2\pi}$. For the sinusoid defined above, that average computes to a value of 1000. The presence of impulses within a data frame will increase the computed average above this value. Further, leakage (the fact that a data frame is not an integer multiple of the sinusoid half-period) will also have some effect on the average calculation. Nonetheless, the value of $\mathbf{A/2\pi}$ should serve as a close approximation for the test purposes.

Assuming stationarity, the average absolute amplitude of a Gaussian Noise signal is σ, or the standard deviation of the original signal.

For both of these signals, a train of pulses, one data point wide, and occurring at a frequency of 62.5 impulses per second, were added. The frame rate of the IMP process is also 62.5 frames/second, so we are ensuring that there will be one impulse per frame. The threshold was set at 2 times the frame average, and it is expected that impulses above the amplitude of 2000 will be removed from both waveforms, while those below 2000 will be left in.

Test Results

1.1a

Test Input:

Impulses mixed with 1000 Hz sinusoid, amplitude 6283.

Impulse Description :

Impulse width: 1 msec

No of Impulses per frame: 1 (62.5/sec)

Impulse amplitude: 500, 1000, 1500, 2000, 2500, 3000

IMP parameters:

IMP decision threshold:	2
Rise Time:	200 μsecs
Fall Time:	700 μsecs

Observation: Impulses above the amplitude of 2000 are removed from the waveform, while those below 2000 are left in.

1.1b

Test Input: Impulses mixed with white noise of standard deviation 1000.

Impulse Description:	Impulse width	1 msec
	No of Impulses per frame	1 (62.5/sec)
	Impulse amplitude:	500, 1000, 1500, 2000, 2500, 3000
IMP parameters:	IMP decision threshold:	2
	Rise Time:	200 μsecs
	Fall Time:	700 μsecs

Observation: Impulses above the amplitude of 1000 are removed from the waveform, while those below 1000 remain.

1.2 Impulse Frequency

The effect of increasing the frequency of impulse occurrence is to increase the average value of each frame, making each individual tone stand out less. For example, in a silent frame (all zeros) into which a single impulse of amplitude **I** is introduced, the frame average will be computed as **I**/256, and the impulse itself will be computed as 256 times the average. On the other hand, if an impulse appeared at every other point in the frame, the frame average will be **I**/2, and each impulse will only appear as twice the average.

The test of the effect of impulse frequency will have the same test conditions as the amplitude test described in Section 1.1 above. The amplitude of the impulses will initially be set to 4000. The frequency of the impulses will be increased in multiples of 62.5/sec, so that the number of impulses per frame will increase by one with each iteration. The frequency at which the impulses are no longer removed will be noted, and the average will be recomputed to ensure that the threshold criteria for removal are met when an impulse is detected.

1.2a

Test Input: Impulse mixed with sinusoid of frequency 1000 Hz and amplitude 6283
Length of sinusoid: 2 seconds

Impulse Description:
Impulse width: 1 msec
No. of Impulses per frame: 1, 3, 5, 7, 9, 11, 13, 15
Impulse amplitude: 4000

IMP parameters:
IMP decision threshold: 2
Rise Time: 200 μsecs
Fall Time: 700 μsecs

Observation: The maximum amplitude at the impulse locations before and after processing are given in Table 1.2-1 below. For impulse frequencies of 1 and 3 (62.5 and 187.5/sec), the impulse was removed completely so that the output sinusoid has an amplitude of 6404 (the input amplitude of the impulse itself is 6283). For the rest of the frequencies (5-15), impulses are removed, but not completely, which could be attributed to the interpolation values encountered as the number of impulses are increased.

Table 1.2-1 Impulse amplitudes for different impulse rates

No. Impulses Per Frame	No. Impulses Per Second	Input Amplitude	Output Amplitude
1	62.5	10283	6404
3	187.5	10283	6404
5	312.5	10283	8442
7	437.5	10283	8442
9	562.5	10283	8442
11	687.5	10283	8442
13	812.5	10283	8442
15	937.5	10283	8442

1.2b

Test Input: Impulses mixed with white noise of standard deviation 1000.
Length of file: 2 seconds

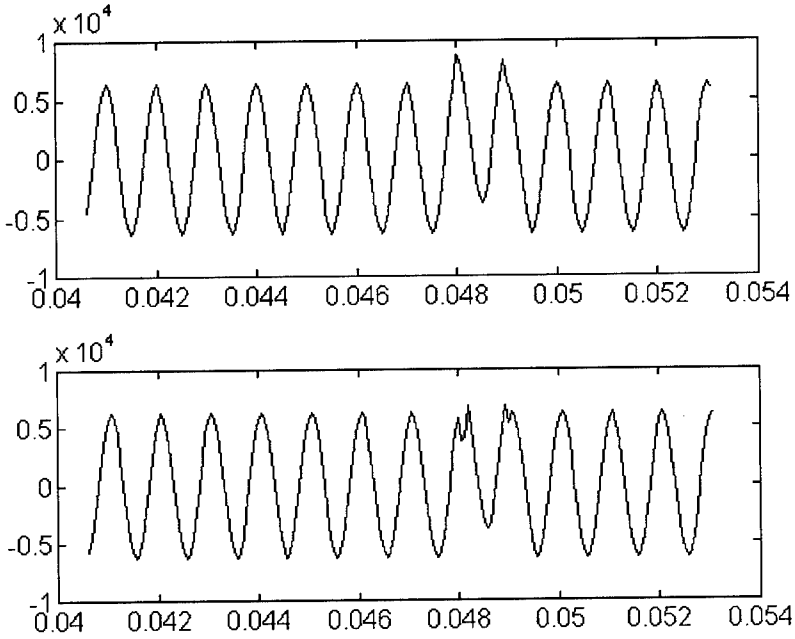


Figure 1.1-1 Removal of impulses (amplitude 2500) from sinusoid. Top: Input signal with 2 impulses. Bottom: Signal after IMP processing. Deformations are artefacts of averaging process.

Impulse Description:	Impulse width:	1 msec
	No. of Impulses per frame:	1, 3, 5, 7, 9, 11, 13, 15
	Impulse amplitude:	4000
IMP parameters:	IMP decision threshold:	2
	Rise Time:	200 μsecs
	Fall Time:	700 μsecs

Observation: The results are given in Table 1.2-2. Again, the IMP algorithm works well at frequencies up to approximately 200Hz, above which it begins to break down.

Table 1.2-2 IMP results for impulses in white noise.

Impulse Frequency (Hz)	Observation
62.5	Impulses Removed
187.5	Impulses Removed
312.5	Impulses Partially Removed

437.5	Impulses Partially Removed
562.5	Impulses Not Removed
687.5	Impulses Not Removed
812.5	Impulses Not Removed
937.5	Impulses Not Removed

1.3 Impulse Duration

The test described above was repeated, but, instead of increasing the frequency of single-point duration impulses, the duration of pulses fixed in frequency at 62.5/sec was increased.

1.3a.

Test Input: Impulses mixed with sinusoid of frequency 1000 Hz and amplitude 6283
Length of sinusoid: 2 seconds

Impulse Description:
Impulse width: 1-8 msec
No. of Impulses per frame: 1 (62.5/sec)
Impulse amplitude: 4000

IMP parameters:
IMP decision threshold: 2
Rise Time: 200 μsecs
Fall Time: 700 μsecs

Observation: The algorithm successfully impulses at all widths used in the test.

1.3b.

Test Input: Impulses mixed with white noise of standard deviation 1000.
Length of files: 2 seconds

Impulse Description:
Impulse width: 1-8 msec
No. of Impulses per frame: 1 (62.5/sec)
Impulse amplitude: 4000

IMP parameters:	IMP decision threshold:	2
	Rise Time:	200 μsecs
	Fall Time:	700 μsecs

Observation: Impulse widths up to 4 milliseconds are removed.

Summary of IMP Test Results

The tests of the IMP algorithm show that it works in the manner predicted by the algorithm theory and descriptions. These results were found when the input impulse disturbance was varied by amplitude, frequency, and pulse width.

2 Hitones (DSS) Evaluation

An important aspect of the Hitones test and evaluation is the comparison between the threshold method of tone identification used in older versions of the algorithm with the histogram method, introduced in the latest versions. The current implementation of Hitones allows the methods to be switched on or off independently, depending on the ambient signal conditions.

2.1 Multiple Tones

This test evaluates the performance of Hitones in the presence of multiple narrowband disturbances.

A group of signal files were developed containing a combination of white noise and pure tones. The tones were placed so that one tone falls into each frequency band of Hitones, as given in Table 2.1-1. The amplitude of the tones were adjusted to achieve a signal to noise ratio (when compared to the white noise) of 10 dB for each tone.

2.1.1 Multiple Tones Across Bands

Test Input: White noise mixed with tones that fall into each frequency band of Hitones. The amplitude of the tones were adjusted to achieve a signal to noise ratio of 10dB. Table 2.1.1-1 shows the band ranges and tone input frequencies for this test. This test was repeated using DSS, the precursor algorithm of Hitones. DSS uses five bands, which are shown in Table 2.1.1-1a.

Table 2.1.1-1 Hitones Frequency bands and test tones

Band	Frequency Range (KHz)	Tone Frequency (KHz)
1	0.000 to 0.793	0.5
2	0.780 to 1.859	1.0
3	1.833 to 2.743	2.0
4	2.730 to 3.640	3.0
5	3.627 to 5.200	4.0
6	5.070 to 6.500	6.0
7	6.370 to 8.000	7.0

Table 2.1.1-1a DSS Frequency bands and test tones

Band	Frequency Range (KHz)	Tone Frequency (KHz)
1	0.000 to 0.750	0.5
2	0.750 to 1.500	1.0
3	1.500 to 2.250	2.0
4	2.250 to 3.000	3.0
5	3.000 to 5.000	4.0

Hitones Parameters:
Frame Size: 1024
Overlap: 50 %
Window: Cosine window. Files were analyzed with a 4096 point FFT .
AGC Mode: OFF

Observation: The Histogram method successfully identifies (thereby removing) all the tones while the Threshold method did not. If the tone amplitude is increased to 20 dB (SNR with noise), the threshold method also removes all the tones. Tables 2.1.1-2 and 2.1.1-3 show the output amplitudes of the tones using different tone identification combinations. Figure 2.1.1-1 shows input and output spectra for the 10 dB case.

Table 2.1.1-2. Tone frequencies and degree of attenuation 10 dB SNR (Hitones)

Band	Tone Freq	Amplitude (dB)	Output Amp: Threshold Method	Output Amp: Histogram Method	Output Amp: Hist/Thresh Method
1	500	113.5	113.5	42	44
2	1000	114	111	50	41
3	2000	114	112	47	44
4	3000	115	107	46	44
5	4000	114.5	107	45	40
6	6000	114.5	107	40	40
7	7000	114	111	41	42

Table 2.1.1-2a. Tone frequencies and degree of attenuation 10 dB SNR (DSS)

Band	Tone Freq	Amplitude (dB)	Output Amplitude (dB)
1	500	118	105
2	1000	115	100
3	2000	118	68
4	3000	115	73
5	4000	115	102

Table 2.1.1-3. Tone frequencies and degree of attenuation 20 dB SNR (Hitones)

Band	Tone Freq	Amplitude (dB)	Output Amp: Threshold Method	Output Amp: Histogram Method	Output Amp: Hist/Thresh Method
1	500	124	< measurable	< measurable	< measurable
2	1000	124	61	60	61
3	2000	124	< measurable	< measurable	< measurable
4	3000	124	< measurable	< measurable	< measurable
5	4000	124	< measurable	< measurable	< measurable
6	6000	124	< measurable	< measurable	< measurable
7	7000	124	< measurable	< measurable	< measurable

Table 2.1.1-2a. Tone frequencies and degree of attenuation 20 dB SNR (DSS)

Band	Tone Freq	Amplitude (dB)	Output Amplitude (dB)
1	500	126	78
2	1000	126	78
3	2000	126	78
4	3000	126	78
5	4000	126	78

2.1.2 Multiple Tones in a single band

In this test, the tones were narrowly spaced so that they fall in a single operational band. Again, different combinations of tone detection methods were employed.

Test Input: White noise mixed with pure tones that falls in a single frequency band (Band 3) of Hitones. The amplitude of the tones are adjusted to achieve a signal to noise ratio of 10 dB.

Observation: It was found that the histogram approach generally outperforms the threshold method when closely spaced multiple tones are present. The results for Hitones and for DSS are given in Table 2.1.2-1. (The DSS results are for the same tone spacing, though fewer tones fit in a single band.)

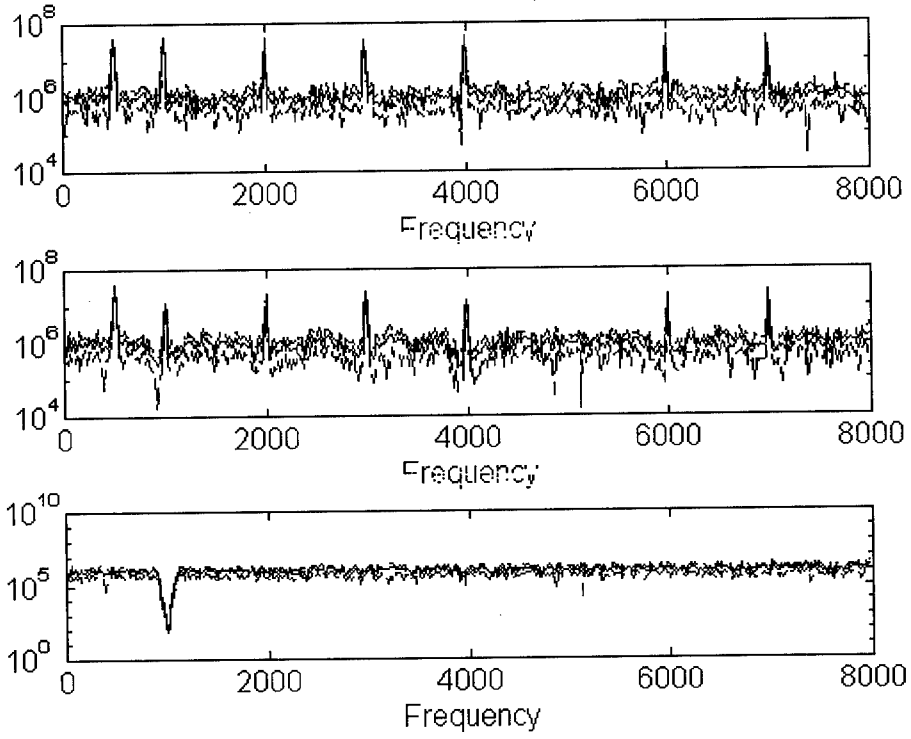


Figure 2.1.1-1 Hitones performance on multiple tones in noise, SNR = 10dB. Top: Spectrum of Input Signal. Middle: Results using Threshold method. Bottom: Results using Histogram method.

Table 2.1.2-1 Hitones response to multiple tones within a single band.

Number of Tones	Tone Spacing (Hz)	Output Amp: Threshold Method	Output Amp: Histogram Method	Output Amp: Hist/Thresh Method	DSS with same tone spacing
3	300	Removes some tones	Removes all tones	Removes all tones	Removes all tones
5	200	Fails to remove tones	Removes all tones	Removes some tones	Removes some tones
9	100	Fails to remove tones	Fails to remove tones	Fails to remove tones	Fails to remove tones
18	50	Fails to remove tones	Fails to remove tones	Fails to remove tones	Fails to remove tones

2.2 Amplitude Range

To measure the effective amplitude range in the various frequency bands, each of the tones identified in Table 2.1.1-1 was varied in amplitude to produce SNR's of -15, -10, -5, -3, 0, 3, 5, 10, 15, and 20 dB. The level at which tone identification breaks down was noted.

Test Input: White noise mixed with pure tone of 1000 Hz. The amplitude of the tones are varied to achieve a signal to noise ratio of -15, -10, -5, -3, 0, 3, 5, 10, 15, 20 dB.

Length of the test input:	3 seconds
No of Tones:	1
Tone Frequency:	1000 Hz
Tone Amplitude:	SNR with noise -15, -10, -5, -3, 0, 3, 5, 10, 15, 20 dB
Tone Frequency:	1000

Hitones Parameters:

Frame Size:	1024
Overlap:	50 %
Window:	Cosine window
AGC Mode:	OFF

Observation: The Histogram method successfully removed a single tone for all amplitudes tested. The threshold method failed when the SNR was -15 or -10 dB. A partial removal occurred in the case of - 5dB. For the rest of

the amplitudes tones were removed but at certain instants of the signal the 1000 Hz tone reappears and vanishes. Table 2.2-1 summarizes the results of the test.

Table 2.2-1 Removal of a single tone at different SNR values by Hitones and DSS.

SNR (dB)	Output Amp: Threshold Method	Output Amp: Histogram Method	Output Amp: Hist/Thresh Method	DSS Results
20	Tone removed	Tone removed	Tone removed	Tone removed
15	Tone removed	Tone removed	Tone removed	Tone removed
10	Tone removed	Tone removed	Tone removed	Tone removed
5	Tone removed	Tone removed	Tone removed	Tone removed
3	Tone removed	Tone removed	Tone removed	Tone removed
0	Tone removed	Tone removed	Tone removed	Tone removed
-3	Tone removed	Tone removed	Tone removed	Fails to remove tones
-5	Partially removed	Tone removed	Tone removed	Fails to remove tones
-10	Fails to remove tones	Tone removed	Tone removed	Fails to remove tones
-15	Fails to remove tones	Tone removed	Tone removed	Fails to remove tones

2.3 Bandwidth

To test the effective bandwidth of Hitones, test signals were created by passing a white noise signal through different narrow band FIR filter to produce band-limited signals. The test procedure was as follows:

- A gaussian noise signal was produced digitally
- 512 tap FIR bandpass filters were constructed with Q factors (frequency/bandwidth) of 0.05, 0.1, 0.2, 0.3, 0.4, and 0.5 for frequencies of 500, 1000, and 2000 Hz.
- The noise signal was processed with the filters to produce band limited (colored) noise signals.
- These signals were processed by Hitones to determine if a tone is detected within the noise.

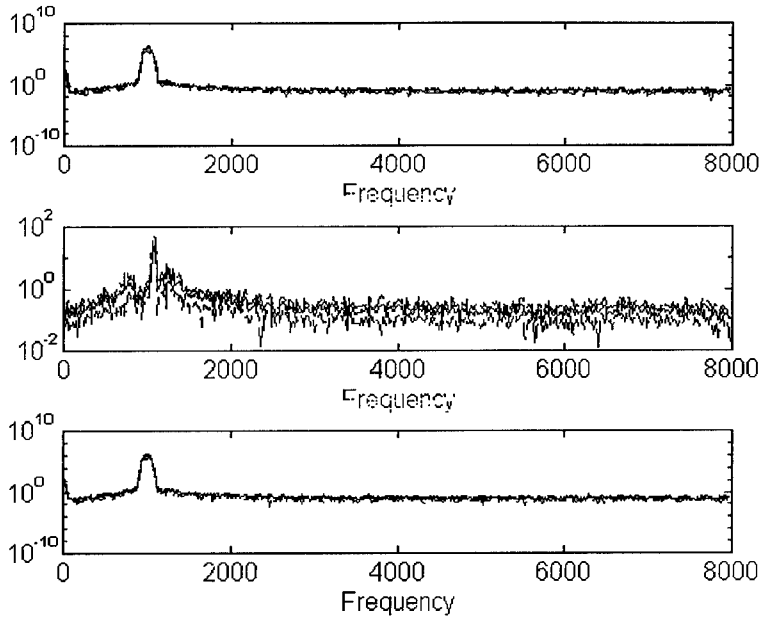


Figure 2.3-1 Removal of bandlimited noise. Bandwidth =100 Hz. Center Frequency = 100 Hz. Top: Input Spectrum. Middle: Hitones using Threshold method (note lower magnitude). Bottom: Hitones using Histogram method.

Test Input: White noise signal processed by FIR with Q factors 0.05, 0.1, 0.2, 0.3, 0.4, and 0.5 at frequencies 500, 1000, 2000 Hz

Length of the test input:	5 seconds
Standard Dev. of Noise:	1000
Tone Frequencies:	500, 1000, 2000
FIR Q factors:	0.05, 0.1, 0.2, 0.3, 0.4, and 0.5

Hitones Parameters:
Frame Size: 1024
Overlap: 50 %
Window: Cosine window
AGC Mode: OFF

Results:

This was a case where the threshold method far outperformed the histogram method of identifying tones. In bandlimited noise, the threshold method identified tones with a bandwidth of up to 200 Hz, while the histogram method failed at a

bandwidth of 100 Hz. Table 2.3-1 summarizes the results of the test, and Figure 2.3-1 shows input-output examples.

Table 2.3-1 Removal of Bandlimited noise by Hitones and DSS

Tone Center Freq.	Bandwidth (Hz)	Hitones - Threshold	Hitones - Histogram	DSS
500	25	Tone Removed	Tone Removed	Tone Removed
500	50	Tone Removed	Tone Removed	Tone Removed
500	100	Tone Partially Removed	Tone Not Removed	Tone Removed
500	150	Tone Not Removed	Tone Not Removed	Tone Partially Removed
500	200	Tone Not Removed	Tone Not Removed	Tone Not Removed
500	250	Tone Not Removed	Tone Not Removed	Tone Not Removed
1000	50	Tone Removed	Tone Not Removed	Tone Removed
1000	100	Tone Removed	Tone Not Removed	Tone Removed
1000	200	Tone Removed	Tone Not Removed	Tone Partially Removed
1000	300	Tone Not Removed	Tone Not Removed	Tone Not Removed
1000	400	Tone Not Removed	Tone Not Removed	Tone Not Removed
1000	500	Tone Not Removed	Tone Not Removed	Tone Not Removed
2000	100	Tone Removed	Tone Not Removed	Tone Removed
2000	200	Tone Partially Removed	Tone Not Removed	Tone Removed
2000	400	Tone Not Removed	Tone Not Removed	Tone Not Removed
2000	600	Tone Not Removed	Tone Not Removed	Tone Not Removed
2000	800	Tone Not Removed	Tone Not Removed	Tone Not Removed
2000	1000	Tone Not Removed	Tone Not Removed	Tone Not Removed

2.4 Stability

The stability limitations of Hitones were tested by producing tones whose frequencies increase at a constant rate. The rate was changed until the tone is no longer tracked. The input signal is of the form $y = A \sin(\omega t)$, where $\omega = \omega_0 + rt$, where r is the rate of frequency increase.

Test Input: Tones whose frequencies increase at a constant rate

Initial Tone Frequency:	500 Hz
Tone Amplitude:	1000.0
Rate of change of freq:	100, 200, 300, 400, 500 Hz/second

Hitones Parameters:

Frame Size:	1024
Overlap:	50 %
Window:	Cosine window
AGC Mode:	OFF

Observation:

With the sinusoidal ramp signal, the histogram method fails in all cases. The threshold method works well, with some intermittent failures occurring at the higher sweep rates as the signal sweeps across frequency bin boundaries. This is another case in which the threshold method outperforms the histogram method, reinforcing the desirability of having both methods available to the system operator. DSS, which uses a threshold method of tone identification, also performed well at all ramp rates.

2.5 Frequency Range

The effective frequency range of Hitones was determined by finding the lowest frequency tone that can be attenuated, and the highest frequency tone.

Test Input: White noise mixed with Tones at 20, 40, 80, 100, 7920, 7940, 7960, 7980 Hz
Tone Amplitude: 20 dB (SNR with noise)

Hitones Parameters:

Frame Size:	1024
Overlap:	50 %
Window:	Cosine window
AGC Mode:	OFF

Observation: For the Threshold method, the frequency range was 20 - 7900 Hz. The range for the Histogram method was 120 - 7920 Hz, and the range for the Threshold/Histogram method was 20 - 7900 Hz. The range for DSS was found to be 20 - 4580 Hz.

2.6 Degree of Attenuation

The degree of attenuation was checked for one tone in each of the seven frequency bands, by noting the amplitude of the power spectrum of the signal before and after attenuation. The results are given in the table in Section 2.1.1.

2.7 Attack Time

The attack time (latency, in data points, between tone onset and tone attenuation) was recorded for both the threshold and histogram methods. A tone pulse 2 seconds in duration was mixed with white noise (SNR=15dB) and presented to Hitones. For each method, the latency between tone onset and attenuation was noted.

Test Input: Tone Pulse 2 seconds in duration, mixed with noise
Length of the test input: 3 seconds
Sine Amplitude: 1000
Sine Frequency: 1000
Pulse width: 2 seconds
White Noise of std deviation 20 added to signal.

Hitones Parameters:
Frame Size: 1024
Overlap: 50 %
Window: Cosine window
AGC Mode: OFF

Results:

The results of the test are summarized in Table 2.7-1. Using the threshold method of tone identification, Hitones was able to react within 6.25 milliseconds, while the histogram method required far more time. Figure 2.7-1 shows sample waveforms illustrating the attack time.

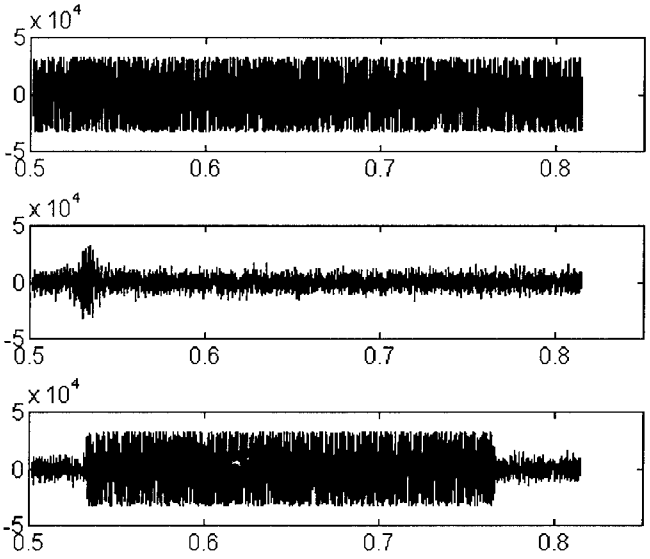


Figure 2.7-1 Attack time for Hitones. Top: Input signal (Tone with noise). Middle: Threshold method of tone identification. Bottom: Histogram method.

Table 2.7-1 Attack time for Hitones and DSS

Test Observations	Attack Time (msec)
Threshold	6.25
Histogram	220.00
Threshold/Histogram	6.25
DSS	6.00

An interesting observation was made during this test concerning the response of the Histogram method of tone identification to sudden onsets of tonal noise. At certain values of Histlimfr, the number of frames involved in the Histogram calculation, a sudden "blip" occurred in the output. This can be seen in Figure 2.6-2. A careful analysis of the algorithm uncovered the reason for the anomoly.

As a frame of data comes in, Hitones performs the following tasks, as fully described earlier in this report (this synopsis is greatly simplified, with all detail not pertaining to the blip removed):

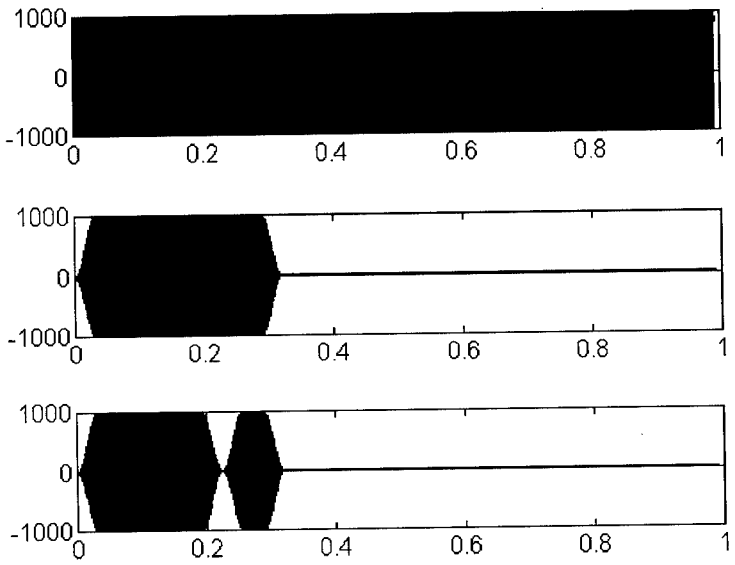


Figure 2.7-2 Illustration of the "blip" anomoly in the Histogram tone identification method. Top: Input sinusoid. Middle: Output with Histlimfr = 6, yielding no anomoly. Bottom: Histlimfr = 7, showing the anomoly.

1. Find First Order Peaks (FOP)
2. Find Second Order Peaks (SOP)
3. Update SOP histogram
4. If the framecount > histlimfr, normalize the histogram by multiplying all values by 0.7. This is equivalent to infinite noise averaging in INTEL.
5. Check if the histogram value is above the band threshold. If it is, mark the presence of a tone, and proceed to remove it. Then go to the next frame.

Take first the simple case of a pure, single tone. If histlimfr is set to 7, as it is in the default case, then the normalization will take place every 8 frames (note the strict > sign in step 4). The band thresholds are fixed, and assume the threshold for the tone in question is 7 (the value for the first 2 bands).

At powerup, the histogram starts at zero. As each frame is read, a peak is found (since we have a steady tone), and the histogram value is incremented by 1. At frame #7, the histogram reaches the band threshold (step 5), and the tone is detected. The output will reflect the absence of the tone as it is removed.

Now let's look at frame 8. The histogram is updated to a value of 8 (step 3). In step 4, however, the normalization takes place, and the histogram is multiplied by .7, yielding a value

of 5.6. This is below the band threshold, and the tone will reappear in the output. It will be 2 frames before the histogram again exceeds 7 and the tone is again removed. Once it is removed the second time, the histogram value will remain high enough to prevent a second blip.

Similar analysis for different values of histlimfr show that the blip will appear for values of 3, 4, 7, and 8. This exactly matches the experimental results.

This problem, while very minor and of little practical significance, can occur at powerup, and at the onset of a new tone later on. Changing the histlimfr--band_threshold combination cannot guarantee that the problem won't occur, since a tone need not begin on the boundary of a normalization block. Any time the framecount reaches a multiple of histlimfr+1, and the histogram value is above threshold but less than threshold/0.7, a similar blip will occur. The effects on speech quality of such a short blip is probably minimal, but is still can occur.

A few other facts fell out of our analysis of the problem. We were wondering if the histogram value would grow unbounded in the presence of a long duration tone. In fact, it does not. If the histogram is incremented by 1 every frame, and normalized every A frames by multiplying the histogram by B ($B < 1$), then the upper limit of the histogram value is $(AB/(1-B))+A-1$. For the default values of 8 and .7, this comes out to 25.67 (It will reach 26.67, but will then be immediately normalized back to 18.67).

This value has implications on how long the tone threshold will be exceeded after the input tone ceases to be present. For the default values cited above, it will take three "normalizations", or 24 frames, for the histogram value to fall below threshold after input tone cessation.

2.8 Speech Distortion

Objective measures of speech distortion were used to determine the extent to which Hitones alters the speech signals from which it removes narrowband noise. Although the Itakura-Saito measure was the method originally planned, it was later decided jointly by the Government and the contractor to use the log-likelihood measure instead. Multiple tones at 100, 200, 500, and 1000 Hz were added to a speech signal, and were removed using Hitones. The log-likelihood metric was used to estimate the amount of degradation the process caused to the original speech.

Test Input:	Speech mixed with multiple tones	
	Length of the test input:	3.34 seconds
	Amplitude of Tones: 1000	
	Freq of Tones:	100, 200, 500, 1000 Hz

Hitones Parameters:

Frame Size:	1024
Overlap:	50 %
Window:	Cosine
Operation Type:	Threshold, Histogram

Observation :

When the log-likelihood measure was used to compare clean speech with speech contaminated with the tones described above, a distortion value of 0.796 was measured. When Hitones was used to remove the noise, a distortion value of 0.593 was measured when the Threshold method was employed, and a distortion value of 0.494 was measured when the Histogram method was employed.

3 INTEL Evaluation

The purpose behind the evaluation of INTEL was to determine the amount of noise reduction it obtains, the time course of the noise reduction, and to measure the distortion it causes in a speech signal when it removes noise.

3.1 Adaptation Time Determination

Adaptation time refers to the period between onset of noise or a change in ambient noise characteristics and the effective removal of noise by INTEL. Adaptation time will be quantized as two numbers: the amount of time from noise onset to the start of noise removal (judged visually by viewing output profiles), and the amount of time until the system "settles down" to 90% of its stable residual noise level (i.e. $Amp_{adapt} = 0.1*Amp_{Peak} + 0.9*Amp_{SteadyState}$).

Two noise files were used for the test:

- A clean (no signal) region followed by the onset of white noise. This tests the sudden onset or increase in noise level
- Five seconds of colored noise, bandlimited between 500 and 1500 Hz, followed immediately by five seconds of noise bandlimited between 1500 and 2500 Hz. This tests INTEL's ability to adapt to sudden changes in noise characteristics.

The tests were repeated for update time constant (UTC) values of 0.1, 0.5, and 1.0.
Adaptation time T_{adapt} is defined as the time from signal onset to the time the signal amplitude reaches Amp_{adapt} defined above.

a. Noise onset test

Test Input: A clean region of length 0.5 seconds followed by onset of white noise.

Duration of white noise:	2.5 seconds
Standard deviation of Noise:	1000

INTEL Parameters:	Processing Frame Size:	512
	Percent overlap:	50.0
	Window Type:	Cosine
	Spectrum compression:	fourth root
	Noise category:	High
	Update Time Constant:	0.1, 0.5, and 1.0
	AGC:	off

Results: Table 3.1-1 shows the adaptation time for different update time constants. Note that in the case of UTC=1, there is no adaptation time, as the algorithm is merely removing a set percentage of the input signal without noise averaging. Figure 3.1-1 shows input and output waveforms for the case of UTC = 0.5.

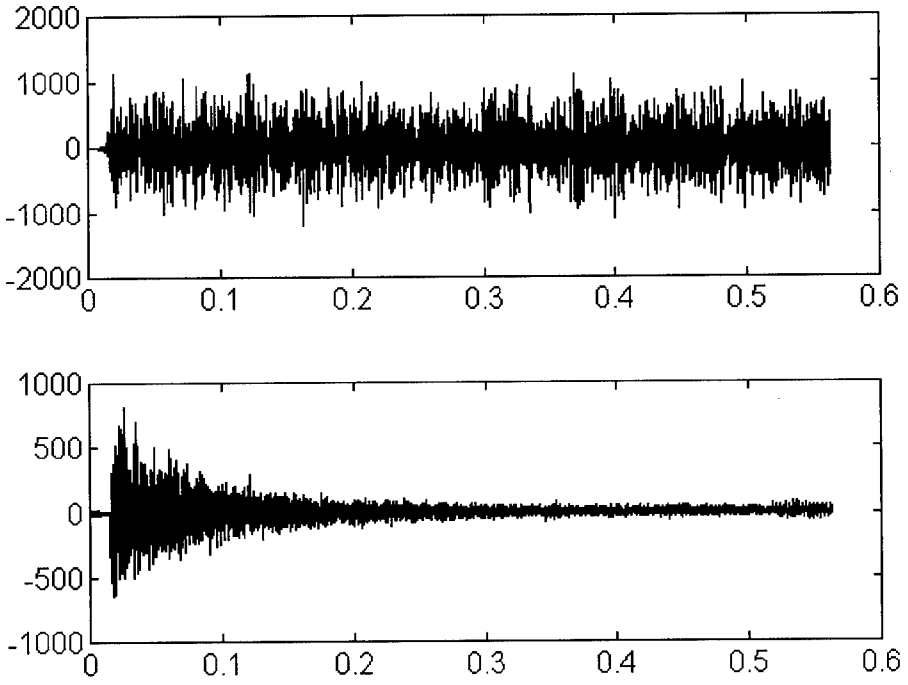


Figure 3.1-1 A white noise signal (top), and the time response of the INTEL algorithm (bottom) for UTC = 0.5.

Table 3.1-1 Adaptation times for INTEL at different UTC values.

Update Time Constant	Adaptation Time (msec)
0.1	172
0.5	60
1.0	0*
Original Version of INTEL	179

* at UTC = 1, algorithm simply takes out a percentage of the current frame

b. 1.5 seconds colored noise bandlimited between 500 and 1500 Hz followed by 1.5 seconds noise bandlimited between 1500 and 2500 Hz. The standard deviation of the noise is 1000. $T_{adapt}(1)$ is defined as the adaptation time from signal onset of the 500-1500 Hz signal and $T_{adapt}(2)$ is defined as the adaptation time when the noise switches to the 1500-2500 Hz band.

Intel Parameters:

Processing Frame Size:	512
Percent overlap:	50.0
Window Type:	Cosine
Spectrum compression:	fourth root
Noise category:	High
Update Time Constant:	0.1, 0.5, and 1.0
AGC:	off

Table 3.1-2 gives the adaptation times for two different UTC values. In each case, there is an increase in the adaptation time as the first noise band is "flushed out" of the noise buffer. Figure 3.1-2 shows a waveform illustrating the adaptation time. An interesting aspect of this process is that the noise level at band switching does not jump to the full input amplitude, as one would expect with a linear process such as spectral subtraction. This shows that the pseudo-cepstral measure used by INTEL is a fairly complex one, and seemingly orthogonal signals such as the ones used here do reflect one another in their representations in the noise average.

Table 3.1-2 Adaptation times for INTEL with bandlimited noise.

Update Time Constant	$T_{adapt}(1)$ (msec)	$T_{adapt}(2)$ (msec)
0.1	176	316
0.5	48	60
Original Version of INTEL	410	256

3.2 Residual Noise

The amount of noise that remains after the INTEL process is the residual. This was measured by comparing the average energy level of noise frames of speech input files both before and after INTEL. Representative spectra were collected of before and after frames. The test were conducted for INTEL in both the 2nd and 4th root modes, and for the noise category set to low, high, and adjust.

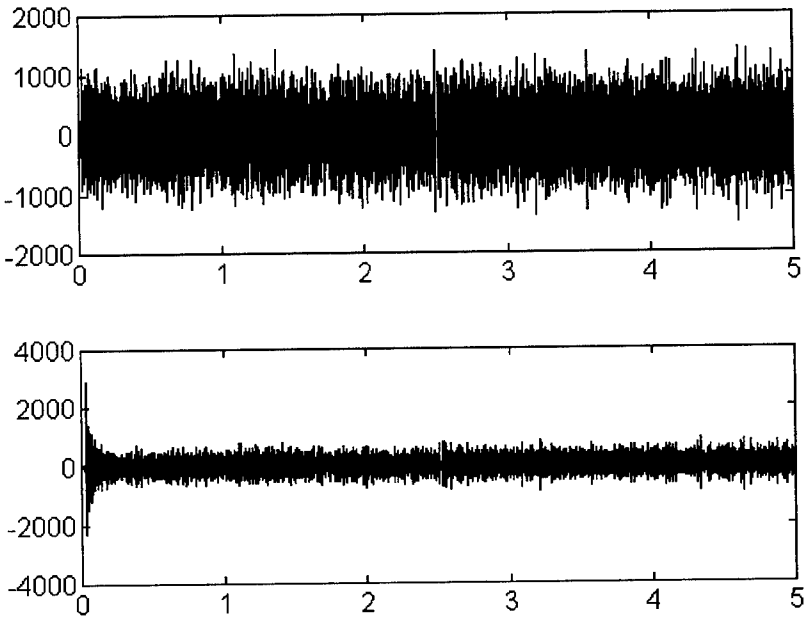


Figure 3.1-2 INTEL response to bandlimited noise. Top: Input noise, consisting of 2.5 seconds limited to 500-1500 Hz followed by 2.5 seconds at 1500-2500 Hz. Bottom: INTEL output, UTC = 0.5.

Test Input: 3 seconds white noise, standard deviation of 1000 (60 dB)

Intel Parameters:

Processing Frame Size	512
Percent overlap	50.0
Window Type	Cosine
Spectrum compression	Second, Fourth root
Noise category	Low, High, Adjust
Update Time Constant	0.3
AGC	off

Table 3.2-1 shows the noise residual, in dB, for different INTEL settings.

Table 3.2-1 Residual Energy in dB

Noise Category	2nd root	4th root
Low	51.79	43.46
High	50.47	40.89
Adjustable	45.89	33.02

Noise Category	2nd root	4th root
Original Version of INTEL	51.93	51.86

3.3 Amount of Noise Reduction (SNR Improvement)

The test to measure SNR improvement used the following procedure:

- Mix clean speech with white noise to a predetermined SNR.
- Process with INTEL
- Measure the average energy of the resultant speech frames (SF').
- Measure the average energy of the residual noise frames (N').
- Subtract N' from SF' to compensate for the residual noise still in the speech frames. This yields the residual speech signal S'.
- Find the new SNR by comparing S' with N'.

This test was repeated for 2nd and 4th root compression modes, and for the noise category set to low, high, and adjust. In addition to measuring the amount of noise reduction, the relative distortion introduced into the speech waveform was estimated using the log-likelihood measure between the original clean speech file and the processed files.

Test Input: Speech mixed with white noise, SNR = 10, 5, 0 dB

Intel Parameters:

Processing Frame Size:	512
Percent overlap:	50.0
Window Type:	Cosine
Spectrum compression:	Second, Fourth root
Noise category:	Low, High, Adjust
Update Time Constant:	0.3
AGC:	off

Results: Tables 3.3-1 and 3.3-2 show the signal/noise ratio improvements and the estimated distortion for various combinations of INTEL parameters. The noise category labelled as "Adjust" reflects a typical set of values used in a previous version of INTEL, and, while considered a good choice for the type of signal used, may not be truly optimal. Table 3.3-3 shows the results using the older version of INTEL that uses a speech/noise classifier. In general, the classifier leads to a greater improvement in SNR, at the cost of a somewhat higher distortion level. Note that when speech was manually labelled, the SNR improvement was even greater, with a moderate distortion. This illustrates the importance of accurate classification in algorithms that make use of speech and noise differentiation.

Table 3.3-1 Test Results for UTC = 0.3

Input SNR	Root	Noise Cat	SNR	Log-likelihood Distortion
10	2	Low	10.699	0.351
10	2	High	10.985	0.342
10	2	Adjust	12.063	0.336
10	4	Low	11.130	0.332
10	4	High	11.526	0.328
10	4	Adj	13.080	0.304
5	2	Low	6.330	0.549
5	2	High	6.666	0.547
5	2	Adjust	7.986	0.532
5	4	Low	6.765	0.534
5	4	High	7.213	0.535
5	4	Adj	9.031	0.503
0	2	Low	1.369	0.674
0	2	High	1.684	0.672
0	2	Adjust	2.936	0.660
0	4	Low	1.755	0.670
0	4	High	2.166	0.664
0	4	Adj	3.879	0.639

Table 3.3-2 Test Results for UTC = 0.1

Input SNR	Root	Noise Cat	SNR	Log-likelihood Distortion

10	2	Low	12.607	0.329
10	2	High	13.204	0.328
10	2	Adjust	15.326	0.312
10	4	Low	13.578	0.300
10	4	High	14.410	0.310
10	4	Adj	17.522	0.282
5	2	Low	8.058	0.518
5	2	High	8.691	0.532
5	2	Adjust	11.066	0.515
5	4	Low	8.944	0.520
5	4	High	9.799	0.518
5	4	Adj	13.138	0.483
0	2	Low	2.786	0.658
0	2	High	3.342	0.656
0	2	Adjust	5.459	0.635
0	4	Low	3.491	0.650
0	4	High	4.227	0.639
0	4	Adj	7.174	0.612

Table 3.3-3 Test Results for the original version of INTEL

Input SNR	Root	SNR	Log-likelihood Distortion
10	2	13.52	0.630
10	4	13.31	0.741
5	2	12.81	1.066
5	4	14.26	1.048
0	2	9.42	1.346

Input SNR	Root	SNR	Log-likelihood Distortion
0	4	11.56	1.322
5	4	20.65	0.890

3.4 INTEL AGC Test

This test was conducted to check the operation of the AGC module supplied as part of the SEU algorithms.

Test Input: Speech mixed with white noise.
SNR 5
Scale the input file with a scale factor of 0.1

Intel Parameters:

Processing Frame Size:	512
Percent overlap:	50.0
Window Type:	Cosine
Spectrum compression:	Second, Fourth root
Noise category:	Adjust
Update Time Constant:	0.1
AGC:	OFF, Manual, Hold, RMS, Peak

The AGC module was found to operate in a manner similar to those on previous platforms.

Automatic Gain Control

A goal of this program was to develop experimental routines that could lead to improved performance by the automatic gain control (AGC) mechanisms of the SEU algorithms. The experimental routines were developed in the Matlab programming language, which has a large acceptance in the realm of scientific programming. The routines can later be converted to C for inclusion in the SEU suite of algorithms.

A simple AGC routine, which works reasonably well, is already part of the SEU suite of algorithms. However, certain problems with the AGC were discovered during operational use that warranted an investigation into possible improvements. The algorithms developed in this program are a first step in the long-term development of an improved SEU AGC capability.

Three routines were developed. The first is an input AGC, which is an intelligent gain control for the A/D converter through which speech data is digitized. The second is an output AGC, which controls listening levels for the user. Finally, a spectral tilt compensator, which adjusts for spectral tilts caused by channel characteristics, was developed.

The following sections describe the three routines.

Input AGC

This effort called for the development of an algorithm to perform input gain control to help maintain an appropriate dynamic range of input signals. The need for this algorithm arises when the level of an analog audio signal does not properly match the voltage range of the input A/D converter. If the analog signal is too large, much or all of the signal information is corrupted by clipping, when the instantaneous amplitude of the signal exceeds the limit of the ADC. If the signal is too small in dynamic range, the digitization will take place over a reduced number of bits, leading to a loss of signal resolution. The AGC algorithm is designed to adjust the gain of an analog input amplifier that modifies the signal before it reaches the ADC. The instantaneous gain value, in a prototype system, will be saved for gain compensation later in the signal processing stage. Current input AGC requirements are met with analog means. This program is designed to provide a digital solution as well.

Because this is a new algorithm, it was decided that, in order to increase the facility of changing and adding features to the algorithm, it would be developed in the Matlab language. Matlab is an interpreted language specially designed for numeric computation, signal processing, and array manipulation. Versions of Matlab are available for both PC platforms (used for the development), and the SUN workstation family. The use of Matlab also facilitates the later

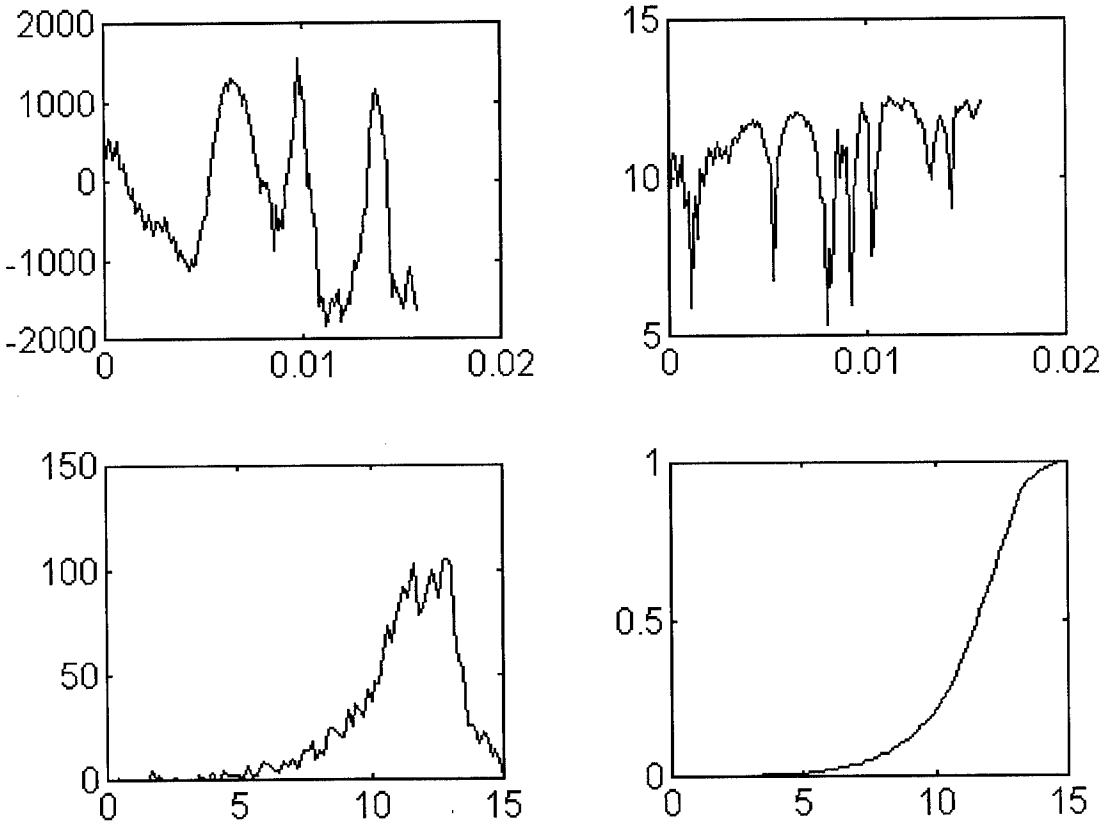


Figure AGC-1. Diagram of the Input AGC algorithm.

migration of the algorithm to a real-time system using the algorithm to control hardware gain values.

The algorithm developed here creates a software simulation of the AGC using input data from a recorded audio file. The computed gain is applied immediately to the data upon retrieval from the recorded file, simulating the real-world case in which the gain is applied to the analog data before conversion.

Figure AGC-1 gives an overview of the AGC procedure. The computation of the appropriate gain is based upon the energy histogram computed from a recent window of input data. As each frame of data is read, it is added to the window, while the oldest frame in the window is discarded. The data is squared to compensate for negative values, and the logarithm (base 2) is computed. Using the logarithm will give us a histogram whose bins represent bits of information.

Speech is a dynamic time signal, whose momentary amplitude changes rapidly. Therefore, rather than specifying a target for the maximum dynamic range, it is more realistic to target a more central value. For this reason, the cumulative histogram of the log data is computed. Data points on the cumulative histogram represents the probability that the value of incoming data is less than a certain value. Figure AGC-2 shows the steps involved in computing the cumulative histogram of a buffer of speech data. We can set a target probability and a target dynamic range. For example, we can say that we want 80% of all income values to fall at or below the 10bit (60 dB) level. Depending on the nature of the input signal, that probability-energy profile can be adjusted by the user to optimize the results for a particular application (speech has a higher variability than pro forma signals, and would therefore call for a different probability setting).

When the signal level corresponding to the target probability is found, it is compared to the desired level range. If it is below the minimum range, the gain will be increased. If it above the range, the gain will be reduced. The speed at which the gain is incremented can be adjusted by the user. Similarly, the user can set an absolute maximum gain value, beyond which the dynamic range will not be compensated. If the signal level is found to be within the allowable range set by the user, the algorithm will slowly (again at a user-settable pace) try to migrate the gain toward unity.

A separate control mechanism is maintained to reduce the amount of clipping, regardless of the overall histogram profile of the data. This control is based on the analysis of the most recent data frame, and is therefore designed to react very rapidly to the saturation of the ADC. Saturation is defined by the bin in which data falls. A very strict definition of saturation uses

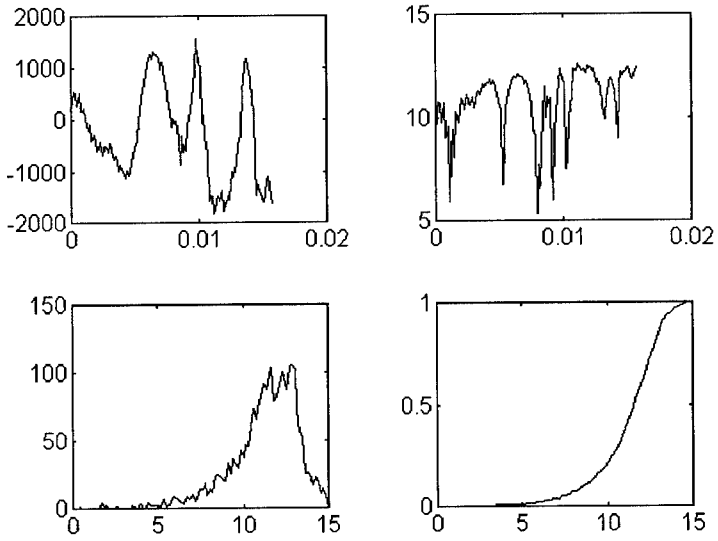


Figure AGC-2 Histogram computation. Top left: One speech frame vs. time. Top right: Magnitude of the frame expressed as power of 2. Bottom left: Log histogram of 20 speech frames. Bottom right: Cumulative histogram of 20 frames (prob. vs bits).

only the topmost bin, and corresponds to data that is equal to the maximum number resolvable by the ADC (32767 for a 16 bit device). The definition of saturation can be loosened to allow the algorithm to compensate for values that are below the absolute maximum but close enough to cause concern for the integrity of the signal. In addition, compensation begins only when a user-defined percentage of points are defined as clipped. This allows for the occasional spike that would not otherwise affect signal data.

Output AGC

The SOW calls for the development of an algorithm to perform output gain control to help maintain an appropriate dynamic range of output signals and a comfortable listening level. The need for this algorithm originates from the attenuation of signal amplitude inherent in the INTEL noise removal algorithm. Some output gain control has always been necessary to restore speech amplitudes to more audible levels. A problem often occurred, however, in the wide allowable dynamic range of the AGC. Weak signals are amplified to the low border of the allowable range, while signals that are too strong are attenuated to the high border. However, when a

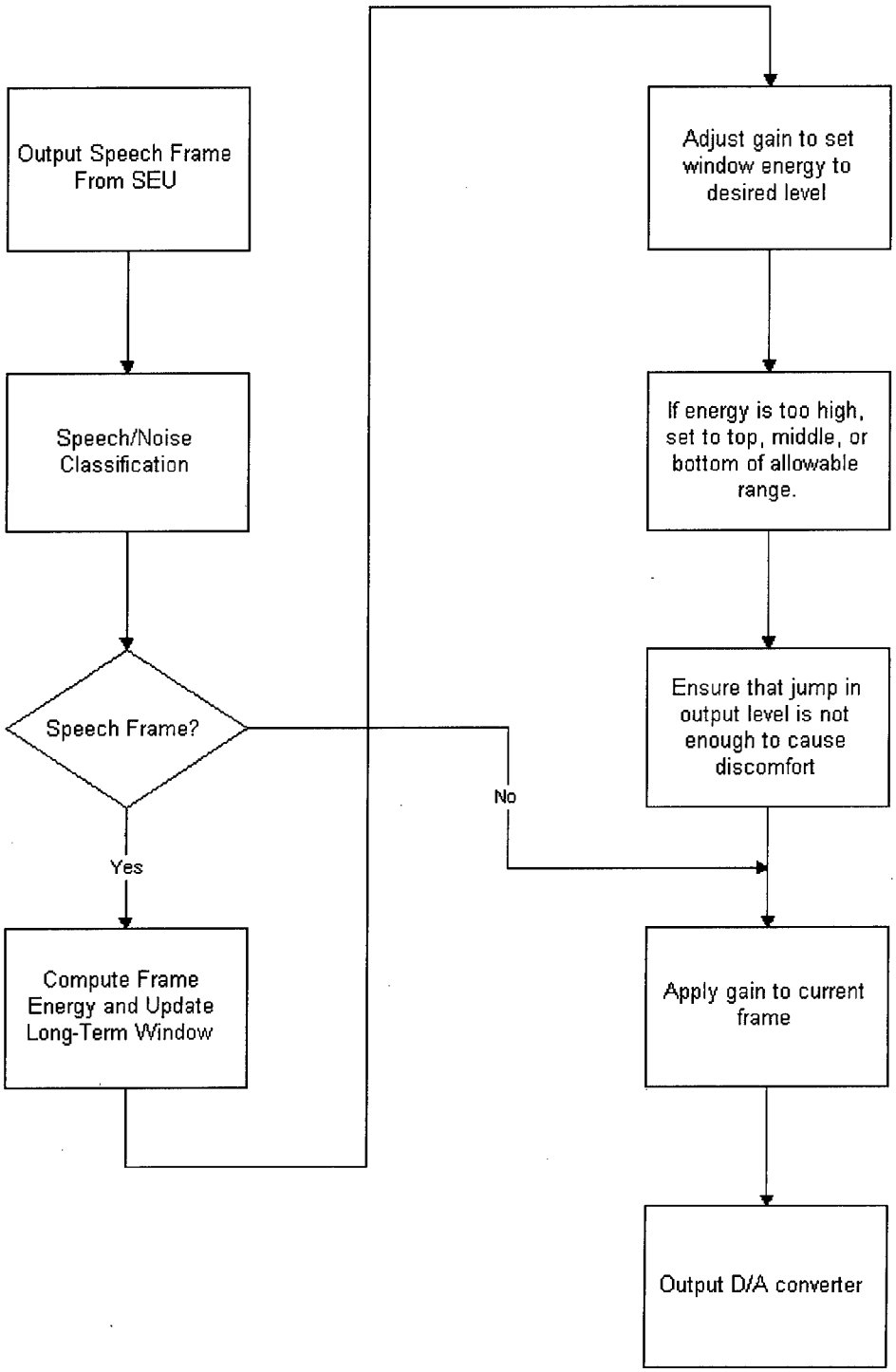


Figure AGC-3. Diagram of the Output AGC algorithm.

weak signal is followed by a strong one, the sudden change from one end of the range to the other is uncomfortable for the listener, and caused some users to disable the function entirely. This new AGC scheme seeks to alleviate the problem by measuring the energy of successive signal segments and restricting the magnitude of sudden increases in output amplitude.

Figure AGC-3 gives an overview of the Output AGC procedure. As in the case of the Input AGC algorithms, the computation of the appropriate gain is based upon the energy histogram computed from a recent window of input data. As each frame of data is read, it is added to the window, while the oldest frame in the window is discarded. The data is squared to compensate for negative values, and the logarithm (base 2) is computed. Using the logarithm will give us

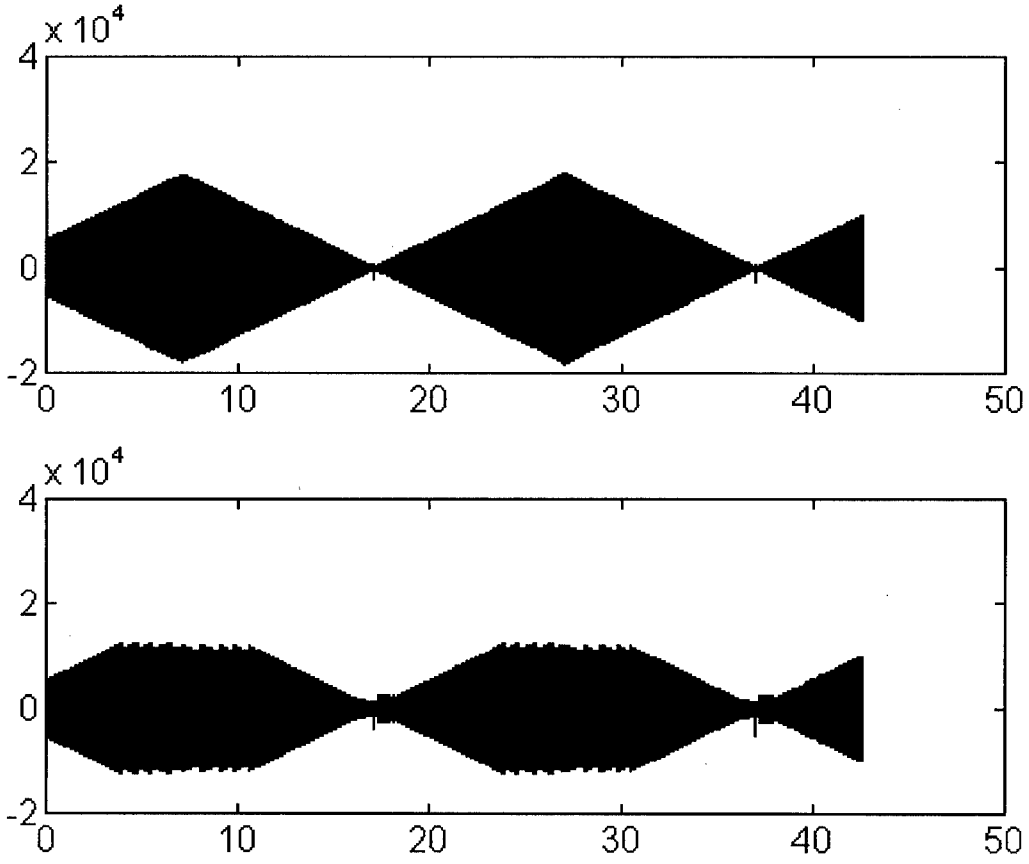


Figure AGC-4 Test input file before processing by Output AGC (top) and after processing (bottom).

a histogram whose bins represent bits of information. The size of the computation window is an adjustable parameter, as are all thresholds.

The output AGC developed here requires a differentiation to be made between speech and non-speech frames. While the computed gain is applied to all frames regardless of their content, only speech frames are used for the gain computation process. For this algorithm, a relatively simple pitch detector was invoked using an autocorrelation and center clip technique described by Rabiner and Schafer in 'Digital Processing of Speech Signals', Prentice Hall, 1978, pp.157. In a hardware realization of the AGC, the same method could be used, or the speech-noise classifier (if any) being used with INTEL could be used instead.

When the signal level corresponding to the target probability is found, it is compared to the desired level range. If it is below the minimum range, the gain will be increased. If it above the range, the gain will be reduced. The user can set an absolute maximum gain value, beyond which the dynamic range will not be compensated. Figure AGC-4 shows the effect of processing a test sinusoid with the AGC algorithm.

An important concept in this version of the AGC is that of the maximum instantaneous amplitude increase. When it is determined that application of the computed gain will result in an increase in speech amplitude that exceeds a maximum amount, the gain will be attenuated to prevent the discomfort of a sudden "blast" of speech in the user's earpiece. Another aspect is the option to bring the level of speech down to either the middle or the bottom of the allowable range whenever the input level goes above the top of the allowable range. This addition allows the operator to force a rapid decrease in amplitude, reducing the chances of discomfort caused by sudden bursts of channel energy.

Spectral Tilt Compensation Program

The spectral tilt compensation program was written in Matlab to provide a testbed for its operation. The program is designed to compensate for channel characteristics that tend to reduce the quality of speech by flattening the noise spectrum through FIR filtering. The algorithm updates the FIR filter when it detects noise, or non-speech frames. The algorithm then takes an average noise estimate, inverts it, and generates an FIR filter from the inverted spectral characteristics. The speech waveform is then filtered using the generated filter. Figure AGC-5 shows a block diagram of the program operation.

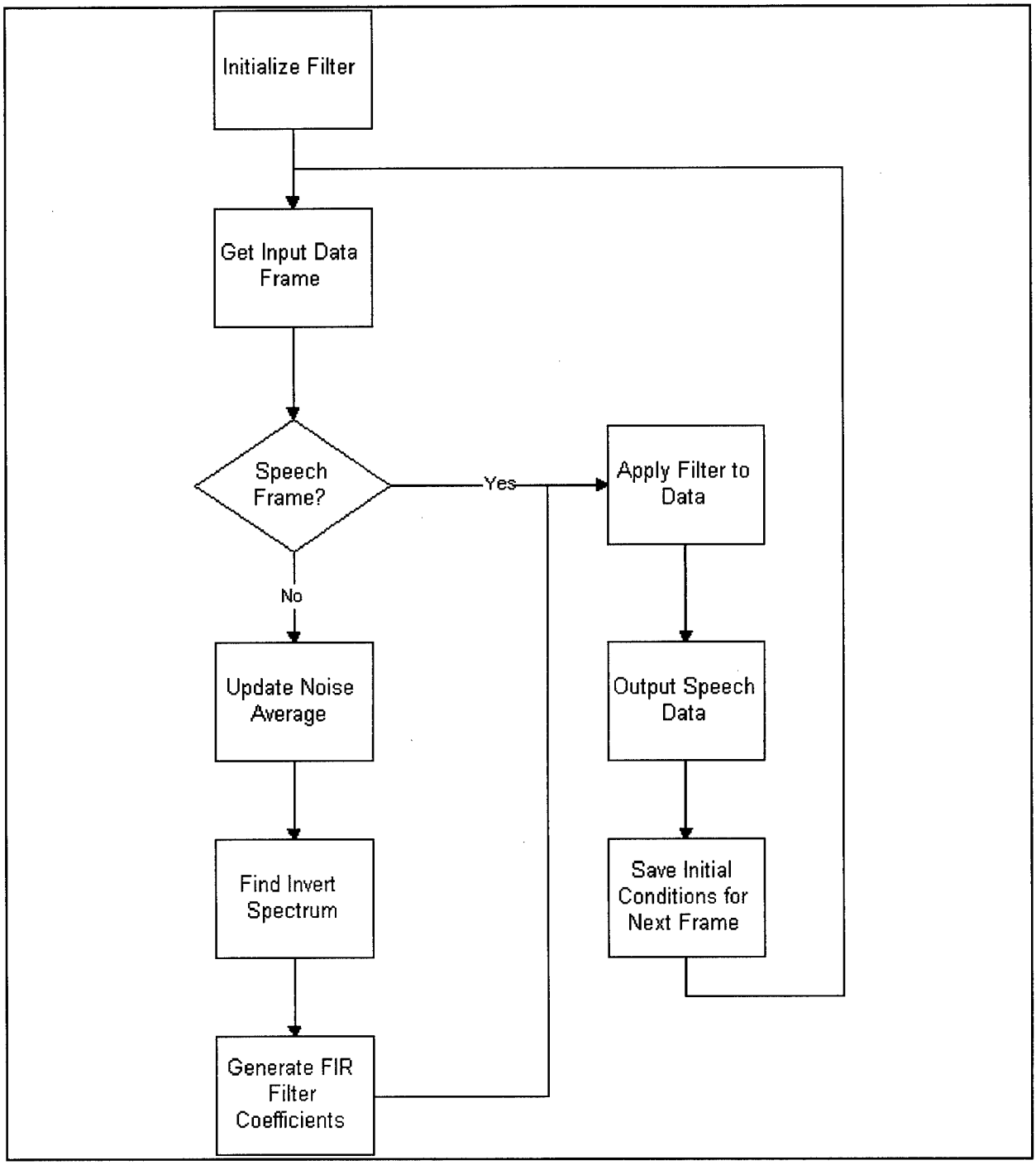


Figure AGC-5. Spectral Tilt Compensation Filter

☆U.S. GOVERNMENT PRINTING OFFICE: 1997-610-130-61086

MISSION OF ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Material Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.