DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

REAL-TIME AUDIO SIGNAL PROCESSING FOR SPEECH ENHANCEMENT

by
Fikret ALİM

August, 2011
İZMİR

REAL-TIME AUDIO SIGNAL PROCESSING FOR SPEECH ENHANCEMENT

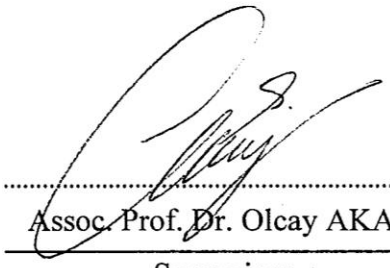
A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering

by
Fikret ALİM

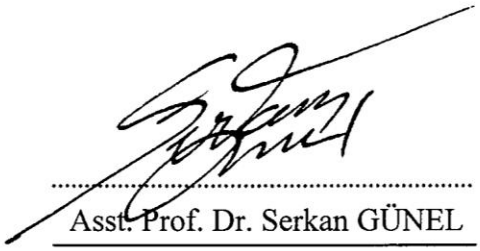
August, 2011
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"REAL-TIME AUDIO SIGNAL PROCESSING FOR SPEECH ENHANCEMENT"** completed by **FİKRET ALİM** under supervision of **ASSOC. PROF. DR. OLCAY AKAY** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Dr. Olcay AKAY
Supervisor



Asst. Prof. Dr. Serkan GÜNEL
Jury Member

Asst. Prof. Dr. Barış BOZKURT
Jury Member

Prof. Dr. Mustafa SABUNCU
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Assoc. Prof. Dr. Olcay AKAY for his continuous support of my M.Sc. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in performing this research and in writing this thesis. I also would like to thank Asst. Prof. Dr. Serkan GÜNEL and Asst. Prof. Dr. M. Alper SELVER for their support, crucial advice, and valuable contributions.

I gratefully acknowledge Mr. Onur DİNÇER, the Manager of SW Design Team at Arçelik A.Ş. Electronics plant, for his valuable help in determination of thesis topic and the procurement of the evaluation module that we have used in our studies. I also would like to acknowledge Mr. Özgür ÇELEBİCAN, Head of Audio Group at Arçelik A.Ş., and my colleague Gürcan ALTINKÖK for their valuable contributions and recommendations during my thesis study.

I also wish to express my sincere appreciation to my friend Sibel BARDAKÇI for the valuable insights she has provided.

Last but not the least, I would like to thank my family for supporting me spiritually throughout my life.

Fikret ALİM

REAL-TIME AUDIO SIGNAL PROCESSING FOR SPEECH ENHANCEMENT

ABSTRACT

In most applications, the purpose of speech enhancement is to enhance the quality and intelligibility of speech degraded by noise. Speech enhancement algorithms reduce or suppress the background noise to some extent and are used for various purposes at cellular telephone systems, air-ground communication, and hearing aids, etc. A typical speech enhancement study must include at least three main steps; enhancement, noise estimation, and evaluation.

In this thesis, we are focused on real-time speech enhancement by using a digital signal processing (DSP) evaluation module. Our purpose is to apply one of the speech enhancement algorithms in the literature together with a noise estimation algorithm by using real-time audio signal processing techniques. Besides, we are also focused on developing a new fusion noise estimation algorithm and utilizing this new method for our purposes. During this thesis study, many speech enhancement and noise estimation algorithms have been applied on the evaluation module and the performance of these algorithms have been evaluated by both subjective and objective measures. In addition to that, several additional signal processing techniques such as Infinite Impulse Response (IIR) filtering and some other enhancements related to software have been used in order to increase the real-time performance of the system. By this way, we have developed the final algorithm which includes the speech enhancement algorithm, our newly developed fusion noise estimation algorithm, and some other audio signal processing techniques combined to obtain the maximum performance at real-time.

Keywords: audio, speech, speech enhancement, noise reduction, noise estimation, real-time audio signal processing, digital signal processing (DSP), digital signal processor.

KONUŞMA SESLERİNİN İYİLEŞTİRİLMESİ İÇİN GERÇEK ZAMANLI OLARAK SES SİNYALLERİNİN İŞLENMESİ

ÖZ

Pek çok uygulamada konuşmanın iyileştirilmesinin amacı gürültüye maruz kalan konuşma verilerinin kalite ve anlaşılabilirliğini artırmaktır. Arka plan gürültüsünü belirli bir ölçüde azaltan ya da bastıran konuşma iyileştirme algoritmaları cep telefonu sistemleri, hava-yer irtibatı ve işitme cihazları gibi kullanım alanlarında çeşitli amaçlarla kullanılmaktadır. Tipik bir konuşma iyileştirme çalışması en azından üç temel adımdan oluşmaktadır; iyileştirme, gürültü tahmini ve değerlendirme.

Bu tezde bir sayısal sinyal işleme (DSP) geliştirme platformu kullanarak gerçek zamanda konuşmanın iyileştirilmesi üzerinde çalışmaktayız. Amacımız gerçek zamanlı ses sinyali işleme tekniklerini kullanarak literatürde varolan bir konuşma iyileştirme algoritmasıyla bir gürültü tahmin algoritmasının birlikte uygulanmasıdır. Bunun yanısıra, yeni bir füzyon gürültü tahmin algoritması geliştirmeye odaklanılmakta ve bu yeni yöntem amacımıza uygun olarak çalışmamıza dahil edilmektedir. Çalışma boyunca pek çok konuşma iyileştirme ve gürültü tahmin algoritması sayısal sinyal işleme geliştirme platformu üzerinde uygulanmıştır ve bu algoritmalar hem objektif hem de sübjektif kriterler ile değerlendirilmiştir. Buna ilave olarak, sistemin gerçek zamanda performansını artırmak maksadıyla sonsuz dürtü yanıtlı (IIR) filtre uygulaması gibi çeşitli sinyal işleme teknikleri ve yazılım yapısıyla ilgili diğer iyileştirmeler kullanılmıştır. Böylece, gerçek zamanda en iyi performansı elde etmek için konuşma iyileştirme algoritması, yeni geliştirilen füzyon gürültü tahmin algoritması ve diğer sinyal işleme tekniklerini içeren bir tümleşik algoritma geliştirilmiştir.

Anahtar Kelimeler: ses, konuşma, konuşma iyileştirme, gürültü azaltma, gürültü tahmini, gerçek zamanlı ses sinyal işleme, sayısal sinyal işleme, sayısal sinyal işlemcisi.

CONTENTS

Page

CHAPTER ONE
INTRODUCTION

1.1 Introduction

In most applications, the aim of speech enhancement is to improve the quality and intelligibility of speech degraded by noise.

Speech enhancement algorithms reduce or suppress the background noise to some extent and are sometimes referred to as noise suppression algorithms.

Figure 1.1 shows the speech enhancement process. The clean speech signal $s(n)$ is degraded by additive noise $d(n)$ and noisy speech signal $y(n)$ is obtained as shown in Equation (1.1). Ideally, our purpose is to obtain the clean speech signal $s(n)$ again by using a speech enhancement algorithm.

$$y(n) = s(n) + d(n) \tag{1.1}$$

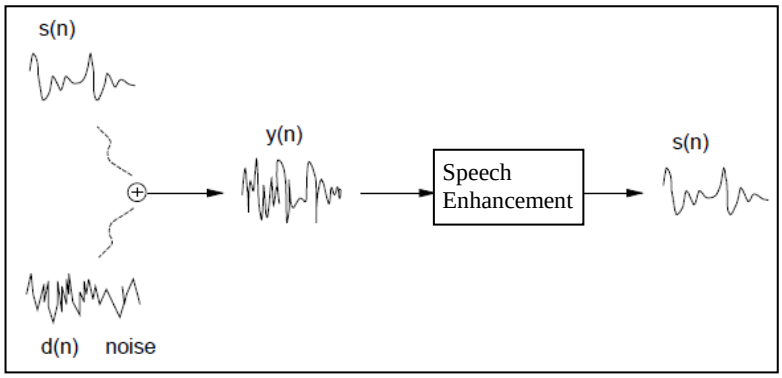


Figure 1.1 Basic speech enhancement process.

The aims of speech cleaning vary according to the application. Cellular telephone systems typically suffer from background noise present in the car, etc. In an air-ground communication, it is critical to eliminate high levels of cockpit noise. In this, as well as in similar communication systems used by military, it is more desirable to enhance the intelligibility rather than quality.

Hearing-impaired listeners wearing hearing aids (Figure 1.2) experience extreme difficulty communicating in noisy conditions. Speech enhancement can be used to clean the noisy signal before amplification.

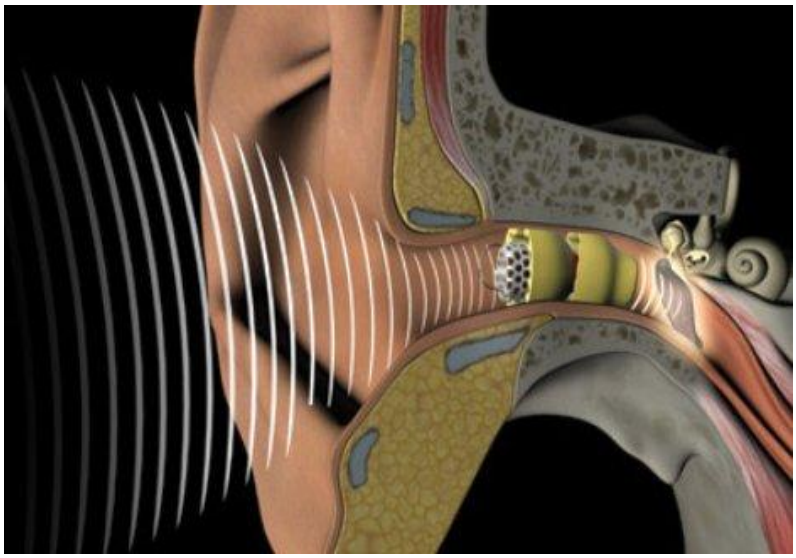


Figure 1.2 Hearing aid worn by hearing-impaired listeners.

These examples show that the goal of speech enhancement varies depending on the application at hand. Ideally, we would like to improve both quality and intelligibility. In practice, however, most enhancement algorithms improve only the quality of speech. It is possible to reduce the background noise, but at the expense of introducing speech distortion, which in turn may impair speech intelligibility. Hence, the main challenge is to design effective speech enhancement algorithms.

1.2 Typical Solution Steps for a Speech Enhancement Problem

Basically, there are three steps that should be performed for the solution of a speech enhancement problem. These three steps are; enhancement, noise estimation, and evaluation. While enhancement and noise estimation algorithms are used for development, evaluation algorithms are used to assess the performance of the developed system.

The enhancement algorithm deals with improving some perceptual aspects of speech. This is the algorithm that performs the main improvement operation. The enhancement algorithm requires a noise estimation algorithm in order to determine the required information about noise. This information is necessary, because we assume that we know nothing about the noise source. The accuracy of noise estimation and tracking algorithm is very important in terms of the performance of the enhancement algorithm. Because of this, an algorithm that is most suitable for our purposes should be used.

The evaluation algorithms are necessary for testing the performance of the developed system. These algorithms can be subjective or objective. We prefer objective measures in order to obtain quantitative results about the performance of the developed system. There are different objective measures for evaluating intelligibility and quality.

There exist many studies in the literature related to speech enhancement and noise estimation (Berouti, M., Schwartz, M. & Makhoul, J. (1979), Hu, Y. & Loizou, P. (2004), Jabloun, F. & Champagne, B. (2003), Martin, R. (2001), Hirsch, H. & Ehrlicher, C. (1995)). This thesis study considers Scalart, P. & Filho, J. (1996) and Ephraim, Y. & Malah, D. (1985) for speech enhancement and Lin, L., Holmes W. H. & Ambikairajah, E. (2003) and Hirsch, H. & Ehrlicher, C. (1995) for noise estimation as starting points, because these studies produced seemingly better results and are more appropriate for real-time application purposes as compared to others. Some other techniques such as subspace algorithms and the Karhunen-Loéve transform (KLT) based method proposed in Hu, Y. and Loizou, P. (2003) have also good performance, but since these algorithms require heavy numerical calculations such as eigenvalue decompositions and matrix inverses, they are not suitable for employing in a real-time system. In this thesis, it is aimed to obtain a novel real-time system that produces satisfactory and competitive results as compared to non real-time algorithms. It is also aimed to develop a fusion noise estimation algorithm by using Lin, L., Holmes W. H. & Ambikairajah, E. (2003) and Ramirez, J., Segura, J.

C., Benitez, C., Torre A. & Rubio, A. (2003) and to utilize this noise estimation algorithm together with a speech enhancement algorithm.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows. In Chapter Two, fundementals of speech and speech enhancement are given. Chapter Three presents the used hardware platform; the evaluation module of Texas Instruments for developing the software. Chapter Four outlines our thesis work related to speech enhancement algorithms and noise estimation algorithms performed by using the evaluation module. The novel fusion noise estimation algorithm is described in Chapter Five. The evaluation results of the developed algorithms are interpreted in Chapter Six. In Chapter Seven, the results of the thesis are discussed and our conclusions are given.

CHAPTER TWO
FUNDEMENTALS OF SPEECH ENHANCEMENT

2.1 Speech Production

The speech signal is a highly nonstationary signal in that its second-order statistics (power spectrum) change over time. When examined closely, however, over a sufficiently short period of time (10-30 msec), its spectral characteristics are fairly stationary.

Speech segments can be separated into three categories:

- Periodic
- Noiselike
- Silence

Speech production involves a number of organs and muscles and includes the lungs, the larynx, and the vocal tract. The lungs provide the main source of excitation in speech production. Figure 2.1 shows these organs and muscles and Figure 2.2 shows the glottal pulses generated by vocal tracts.

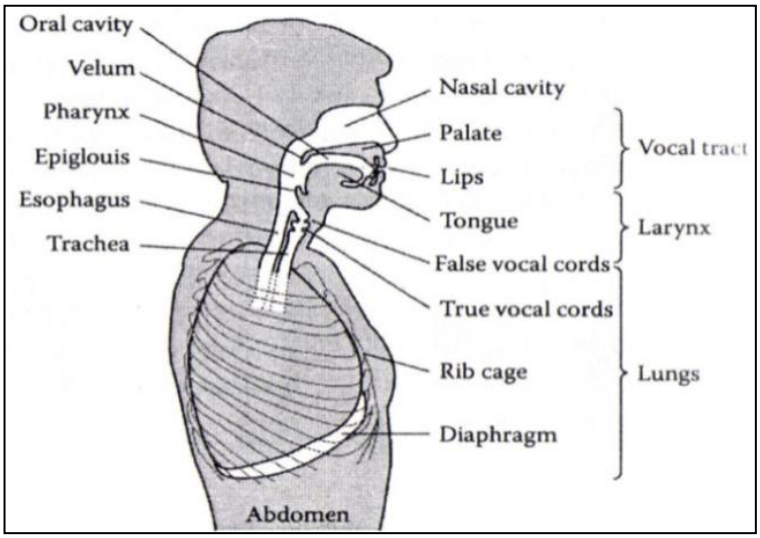


Figure 2.1 A cross section of the anatomy of speech production (Loizou, P. C., 2007).

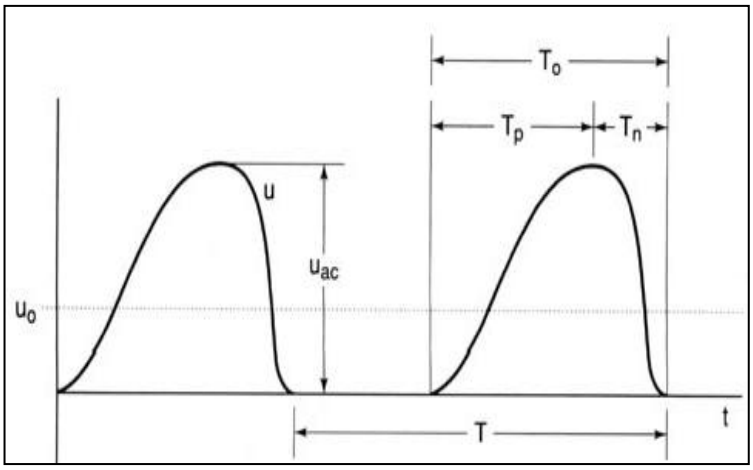


Figure 2.2 Glottal pulses generated by vocal tracts (NCVS Tutorials – Voice Production, www.ncvs.org).

The time duration of one glottal cycle is known as the pitch period which is labeled with *T* in Figure 2.2. The reciprocal of the pitch period is known as the fundamental frequency. Males typically have a lower fundamental frequency than females, because their vocal folds are longer and more massive. The fundamental frequency is 60-150 Hz for males and 200-400 Hz for females and children.

The vocal tract acts as a physical linear filter that spectrally shapes the input wave to produce distinctly different sounds.

2.2 Engineering Model of Speech Production

Figure 2.3 shows the engineering model of speech production (Loizou, P. C., 2007). In this model, the vocal tract is represented by a quasi-linear system that is excited by either a periodic or an aperiodic source, depending on the state of vocal folds. The output of this model is the speech signal that we can measure accurately. Vocal folds are modeled as a switch having two states if we ignore the breathing state.

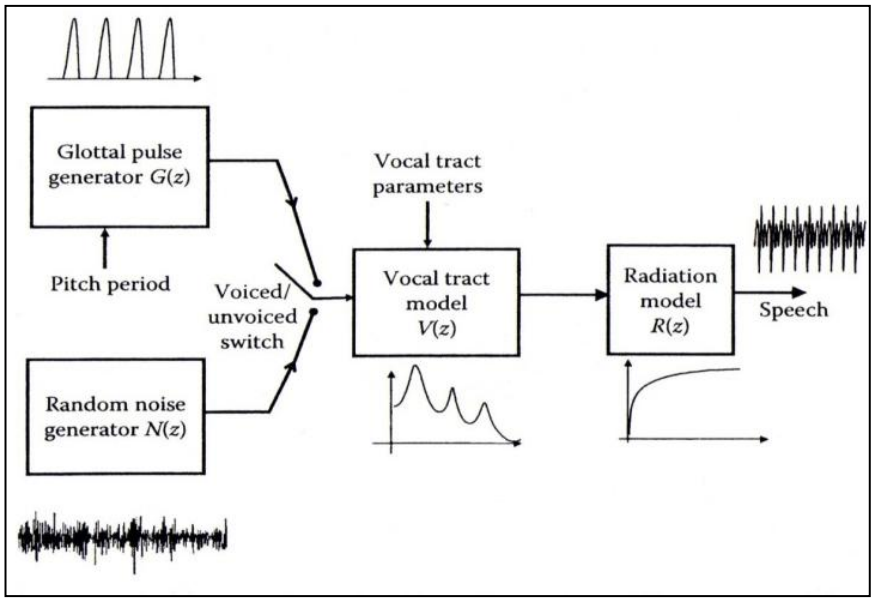


Figure 2.3 Engineering model of speech production (Loizou, P. C., 2007).

2.3 The Effects Degrading the Speech Signal

The principal degradations that we can be concerned with are:

- **Additive acoustic noise** – such as the noise added to the speech signal when recorded in an environment with noticeable background noise, like in an aircraft cockpit.
- **Acoustic reverberation** – results from the additive effect of multiple reflections of an acoustic signal.
- **Convolutive channel effects** – results when the communication channel is not modelled effectively for the channel equalizer to remove the channel impulse response.
- **Non-linear distortion arising such as due to clipping** – for example, when inappropriate gain is applied at the signal input stage.
- **Electrical interference**
- **Codec distortion** – distortion caused by the coding algorithm due to compression.
- **Distortion introduced by recording apparatus** – poor response of microphone.

2.4 Noise Sources

Noise appears in different shapes and forms in daily life. It can be stationary, such as fan noise from a PC or it can be non-stationary such as in a restaurant where sounds of multiple people speaking in the background are mixed with the sound coming from the kitchen. Figures 2.4 through 2.7 show time domain noise waveforms for a car and a train and their power spectral density estimates, all plotted using MATLAB.

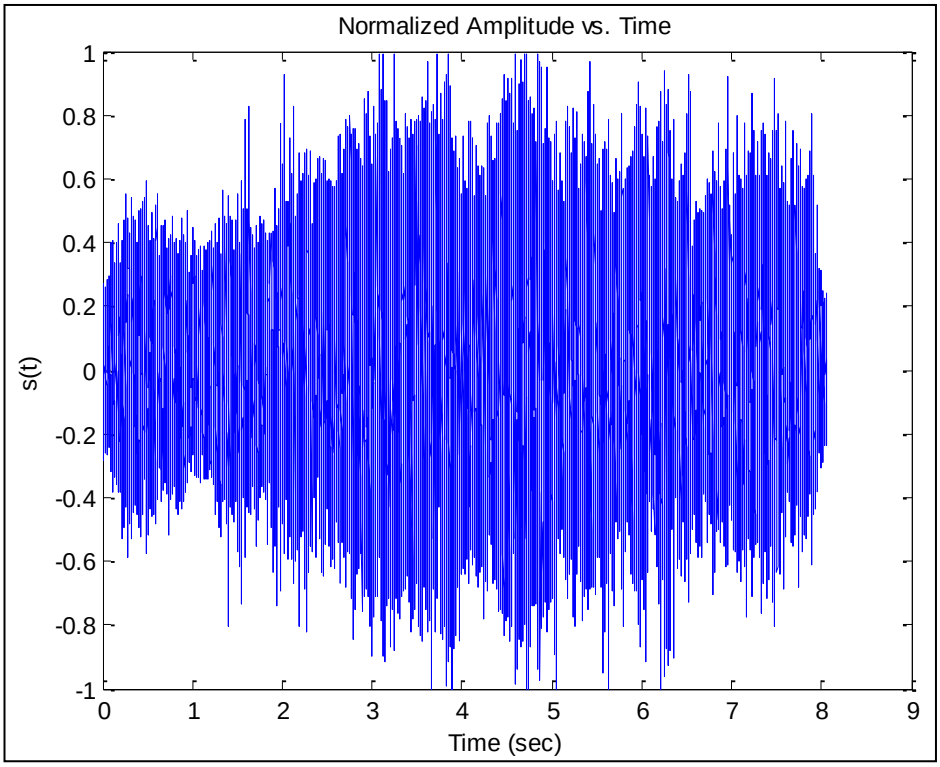


Figure 2.4 Example noise waveform for a car; s(t) is normalized amplitude.

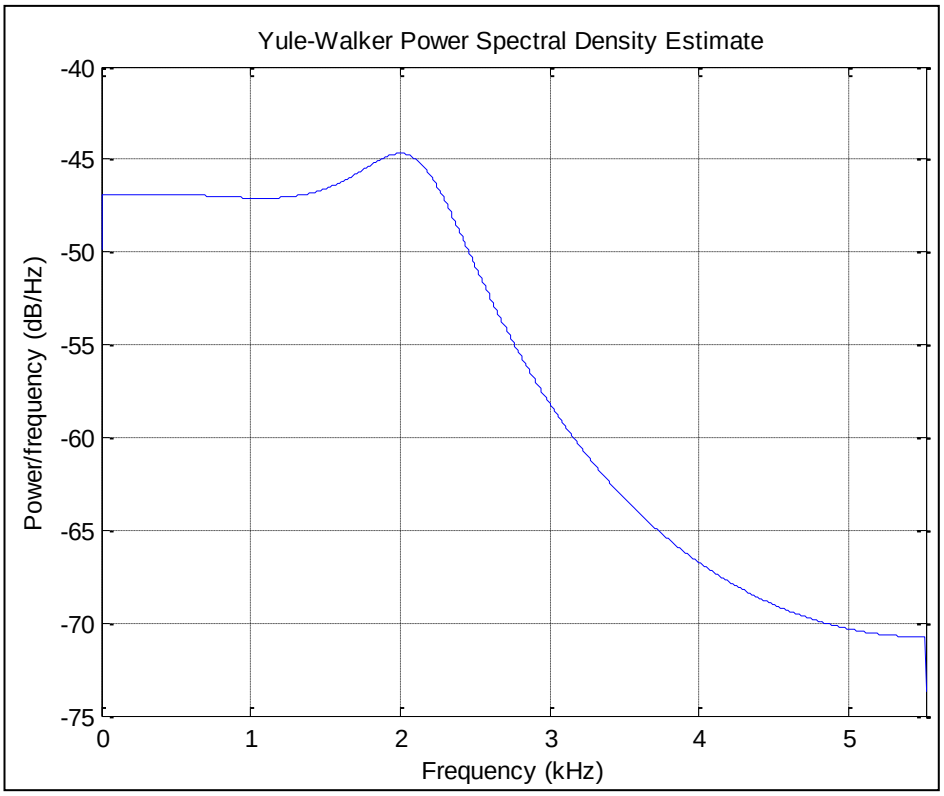


Figure 2.5 PSD estimate of the waveform in Figure 2.4.

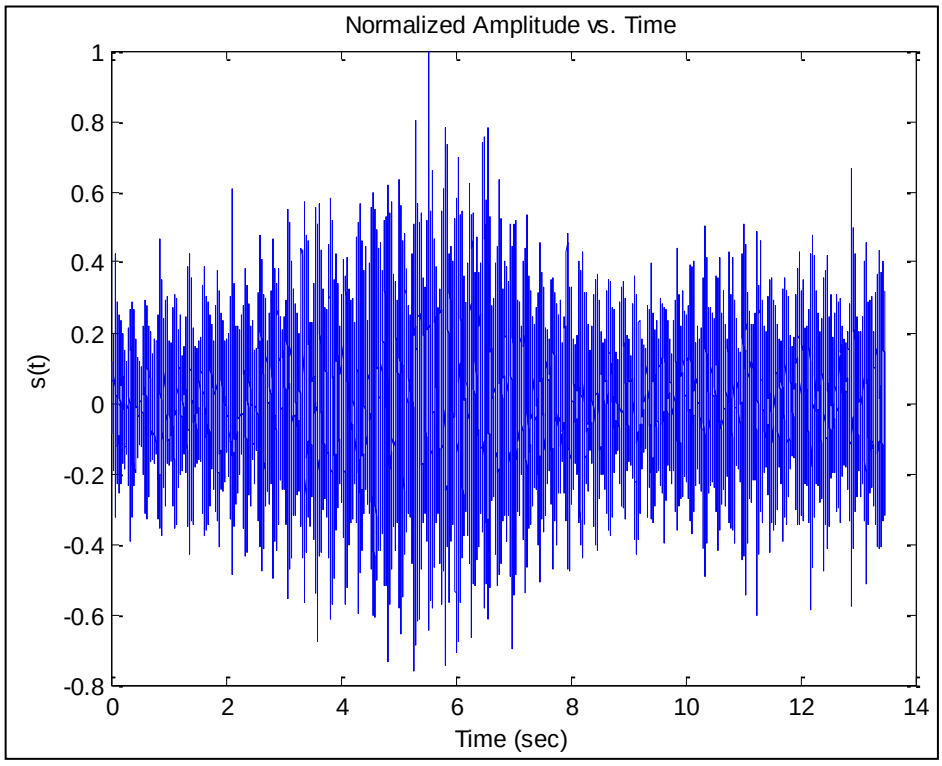


Figure 2.6 Example noise waveform for a train; s(t) is normalized amplitude.

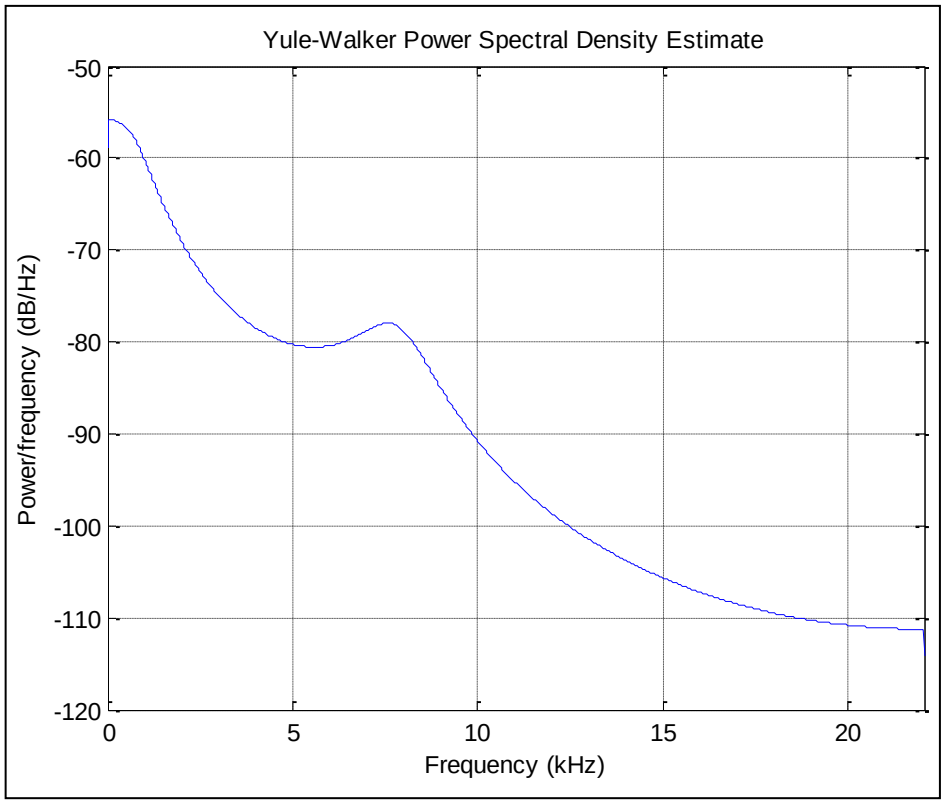


Figure 2.7 PSD estimate of the waveform in Figure 2.6.

2.5 Classes of Speech Enhancement Algorithms

A number of algorithms have been proposed in the literature for speech enhancement. These algorithms can be divided into three main classes:

- Spectral Subtractive Algorithms
- Statistical Model Based Algorithms
- Subspace Algorithms

2.5.1 Spectral Subtractive Algorithms

Assuming additive noise, the spectrum of the clean signal can be estimated by subtracting an estimate of the noise spectrum from the noisy speech spectrum. The noise spectrum can be estimated and updated during periods when the signal is absent. The assumption made is that noise is stationary or a slowly varying process, and that the noise spectrum does not change significantly between the updating periods.

The enhanced signal is obtained by computing the inverse discrete Fourier transform (DFT) of the estimated signal using the phase information of the noisy signal. The algorithm is computationally simple as it only involves a forward and an inverse Fourier transform.

This operation does not need to be performed on magnitude spectrum; it can also be performed on power spectrum or higher order spectrums. Figure 2.8 (Loizou, P. C., 2007) shows the general form of the spectral subtraction algorithm.

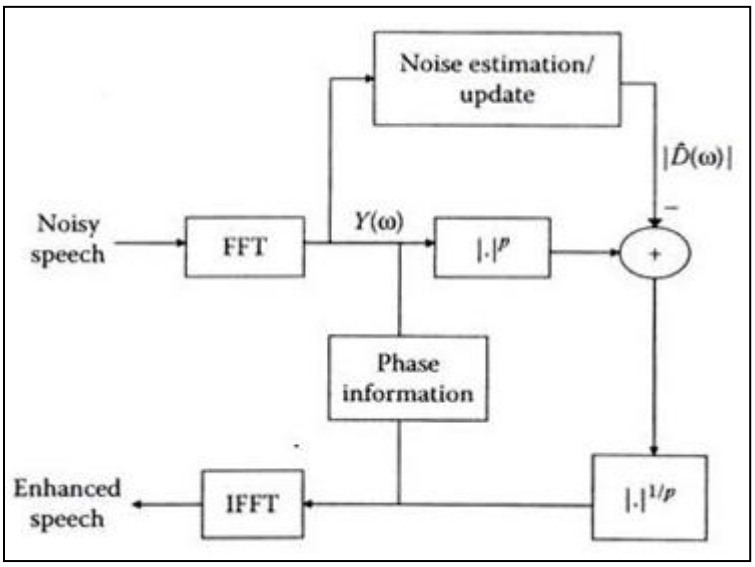


Figure 2.8 General form of the spectral subtraction algorithm (Loizou, P. C., 2007).

Subtracting the expected noise spectrum rather than its instantaneous value causes two problems:

- There is residual broad-band noise after processing.
- Individual narrow band spectral spikes remain and generate tonal noise often referred to as musical noise.

A number of improvements have been proposed to circumvent these problems including the introduction of a gain floor and over-subtraction (Berouti et. al., 1979, Vary, P. & Martin R., 2006). Many variants of the spectral subtraction method exist, each with its own strengths and weaknesses (Loizou, P. C., 2007).

2.5.2 Statistical Model Based Methods – Wiener Filtering

Wiener filtering approach derives the enhanced signal by optimizing a mathematically tractable error criterion, the mean-square error. The system is designed in such a way that the output signal is as close to the desired signal as possible. The estimation error is computed and made as small as possible. The optimal filter that minimizes the estimation error is called the *Wiener filter*. Figure 2.9 (Loizou, P. C., 2007) shows the block diagram of the Wiener filtering approach.

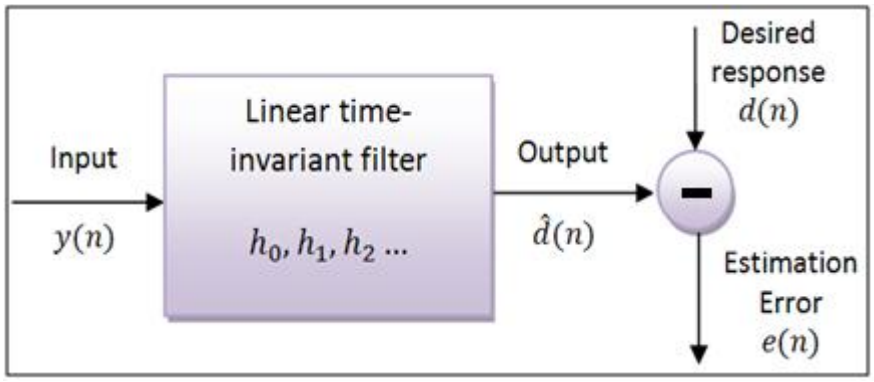


Figure 2.9 Block diagram of the statistical filtering problem (Loizou, P. C., 2007).

It should be noted that one of the constraints placed on the filter is that it is linear, thus making the analysis easy to handle. In principle, the filter could be finite impulse response (FIR) or infinite impulse response (IIR), but often FIR filters are used because:

- They are inherently stable.
- The resulting solution is linear and computationally easy to evaluate.

Assuming an FIR filter, we have;

$$\hat{d}(n) = \sum_{k=0}^{M-1} h_k y(n-k) \qquad n = 0, 1, 2, \ldots \tag{2.1}$$

The mean square of the estimation error is commonly used as a criterion for minimization, and the optimal filter coefficients can be derived in time or frequency domain (Loizou, P. C., 2007).

2.5.3 Statistical Model Based Methods – Nonlinear Estimators

Nonlinear estimators take the probability density function (PDF) of noise and the speech DFT coefficients explicitly into account. They are generally used with a soft-decision mechanism that takes the probability of speech presence into account. Here, the speech enhancement problem is given in a statistical estimation framework. When a set of measurements that depend on an unknown parameter are given, we wish to find a nonlinear estimator of the parameter of interest.

In our application, the measurements correspond to the set of DFT coefficients of the noisy signal (noisy spectrum) and the parameters of interest are the set of DFT coefficients of the clean signal (clean signal spectrum).

Various techniques exist in the estimation theory literature for deriving these nonlinear estimators such as maximum likelihood estimators and Bayesian estimators.

2.5.3.1 *Maximum Likelihood Estimators*

Maximum likelihood is the most popular approach in statistical estimation theory for deriving practical estimators. They are often used even for the most complicated estimation problems.

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, p(y;\theta) \tag{2.2}$$

In speech enhancement terminology, y can be the observed data set, θ might be the clean speech magnitude and $\hat{\theta}_{ML}$ is the maximum likelihood estimate of θ**.** $p(y;\theta)$ is the likelihood function (McAulay, R. J. & Malpass, M. L., 1980).

2.5.3.2 *Bayesian Estimators*

In maximum likelihood, we assumed that θ was deterministic but unknown. In Bayesian philosophy, θ is assumed as a random variable and the realization of it is to be estimated. This approach takes its name from Bayes' theorem. The Bayesian

estimators typically perform better than the maximum likelihood estimators, as they make use of prior knowledge (Lotter, T. & Vary, P., 2005, Wolfe, P. & Godsill, S., 2000, Wolfe, P. & Godsill, S., 2001).

2.5.3.3 *MMSE Estimator*

The Wiener estimator is considered to be the optimal (in the mean-square-error sense) complex spectral estimator, but it is not the optimal spectral magnitude estimator. Short-time spectral amplitude (STSA) is very important with respect to intelligibility and quality (Loizou, P. C., 2007). Hence, several authors have proposed optimal methods for estimating spectral magnitude, such as;

- Maximum likelihood method (Ephraim, Y. & Malah, D., 1985),
- Decision-directed approach (Cohen, I., 2005, Hasan, M., Salahuddin, S. & Khan, M., 2004, Cappe, O., 1994).

2.5.4 Subspace Algorithms

The speech enhancement algorithms described so far were based on signal processing and statistical estimation. Subspace algorithms are largely based on linear algebra theory. These algorithms are based on the principle that the clean signal might be confined to a subspace of the noisy Euclidean space. We could estimate the clean signal simply by nulling the components of the noisy vector residing in the noise subspace.

For the implementation of subspace algorithms, singular value decomposition (SVD) (Moor, B., 1993) and Karhunen-Loève transform (KLT) (Mittal, U. & Phamdo, N., 2000) based algorithms are used in speech enhancement. The majority of the subspace algorithms were originally formulated under the assumption that noise is white. Some extensions to handle colored noise also exist (Wilkinson, J. H., 1999, Rezayee, A. & Gazor, S., 2001). The implementation of subspace algorithms requires a high computational load as SVD or eigenvalue decomposition (EVD) needs to be performed in every frame (Loizou, P. C., 2007).

2.6 Noise Estimation Algorithms

For the speech enhancement algorithms given up to now, it is assumed that an estimate of the noise spectrum is available. Such an estimate is critical for the performance of the algorithms such as for evaluating Wiener filter, for estimating a priori SNR in the MMSE, or for estimating covariance matrix in the subspace algorithms.

The noise estimate has a major impact on the quality of the enhanced signal. If the noise estimate is too low with respect to true noise level, annoying residual noise will be audible. If the noise estimate is too high, speech will be distorted, possibly resulting in intelligibility loss.

2.6.1 Voice Activity Detection (VAD)

Voice activity detection (VAD) is the process of discriminating between voice activity and silence. It is the simplest approach which estimates and updates the noise spectrum during the silent segments of the signal. Although such an approach might work satisfactorily for stationary noise (e.g., white noise), it will not work well in more realistic enviroments (e.g., in a restaurant), where the spectral characteristics of noise change constantly.

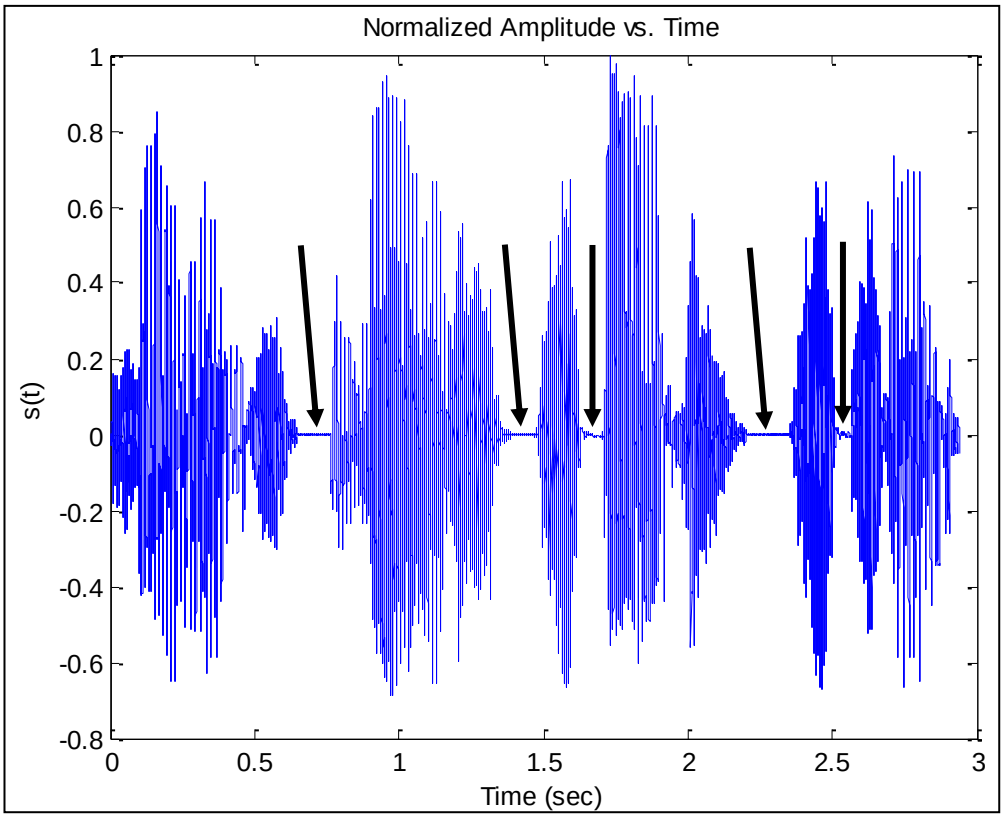


Figure 2.10 An example speech signal; arrows indicate the silent intervals.

Figure 2.10 shows the silent intervals in a sentence when a voice activity detector is active.

An accurate noise estimate is required at all times, even during the speech activity. Noise estimation algorithms continuously track the noise spectrum for nonstationary scenarios. There are three main classes of noise estimation algorithms.

2.6.2 Minimal-tracking Algorithms

These algorithms track the minimum of the noisy speech power in each frequency band and estimate the noise level in that band. Examples can be found in Martin, R. (1993), Martin, R. (1994), Martin, R. (2001) and Doblinger, G. (1995).

2.6.3 Time-recursive Averaging Algorithms

The principle of these algorithms is that we can estimate and update the noise spectrum of that band whenever the effective SNR at a particular frequency band is extremely low. Examples can be found in Sohn, J. & Sung, W. (1998), Cohen, I. (2002) and Lin, L., Holmes W. H. & Ambikairajah, E. (2003).

2.6.4 Histogram-based Algorithms

These algorithms assume that the noise levels correspond to the maximum of the histogram of energy values. Examples can be found in Hirsch, H. & Ehrlicher, C. (1995) and Ris, C. & Dupont, S. (2001).

2.7 Evaluating the Performance of Speech Enhancement Algorithms

The main evaluation methods for speech enhancement systems look at the effect of the system on the intelligibility of the speech signal and the improvement in the overall quality of the signal.

There are two groups of methods for assessing the intelligibility or quality of speech: Subjective and objective methods.

2.7.1 Subjective Methods

These methods require the judgement of human listeners. These can be further divided into two categories:

- Absolute scoring methods - a single stimulus is tested,
- Preference methods - multiple signals are compared.

2.7.2 Objective Methods

The judgment is predicted with some analysis of the system. Objective methods can be further divided into two categories:

- Intrusive - require access to both original and processed signals,
- Non-intrusive - only require access to the processed signal.

2.7.3 Quality and Intelligibility Evaluation Techniques

Quality assessment can be done using subjective listening tests or objective quality measures. Subjective evaluation involves comparisons of original and processed speech signals by a group of listeners who rate the quality. Objective evaluation involves mathematical comparison of the original and processed speech signals. Objective measures quantify quality by measuring the numerical "distance" between the original and processed signals. Clearly, for an objective measure to be valid, it needs to correlate well with the subjective listening tests.

Quality and intelligibility are different attributes, and the two are not equivalent. For that reason, different assessment methods are used to evaluate the quality and intelligibility. Quality is highly subjective in nature and is difficult to evaluate reliably. Quality measures includes attributes such as "natural", "raspy", "hoarse", "scratchy", and so on. Intelligibility measures assess "what" the speaker said; the meaning or the content of the spoken words. Unlike quality, intelligibility is not subjective and can be easily measured by presenting the speech material to a group of listeners and by asking them to identify the words spoken.

There are many methods available for evaluating both intelligibility and quality in the literature. We will explain the ones that we use for the purpose of evaluating the algorithms in the following sections.

2.7.3.1 *Perceptual Evaluation of Speech Quality*

PESQ, perceptual evaluation of speech quality, is a family of standards comprising a test methodology for automated assessment of the speech quality as experienced by a user of a telephony system. It is standardised as ITU-T recommendation P.862, (2001). Today, PESQ is a worldwide applied industry

standard for objective voice quality testing used by phone manufacturers, network equipment vendors, and telecom operators.

PESQ was particularly developed to model subjective tests commonly used in telecommunications (e.g. ITU-T P.800, 1996) to assess the voice quality by human listeners. Consequently, PESQ employs true voice samples as test signals. In order to characterize the listening quality as perceived by users, it is of paramount importance to load modern telecom equipment with speech-like signals. Many systems are optimized for speech and would respond in an unpredictable way to non-speech signals (e.g. tones, noise). Guidelines for proper applications of voice test samples are defined in the PESQ application guide ITU-T P.862.3.

The block diagram of the PESQ measure is shown in Figure 2.11 below. The original (clean) and degraded signals are first level-equalized to a standard listening level, and filtered by a filter with response similar to a standard telephone handset. The signals are aligned in time to correct for time delays, and then processed through an auditory transform to obtain the loudness spectra. The difference, termed disturbance, between the two loudness spectra is computed and averaged over time and frequency to produce the prediction of subjective mean opinion score (MOS) (Loizou, P. C. (2007)).

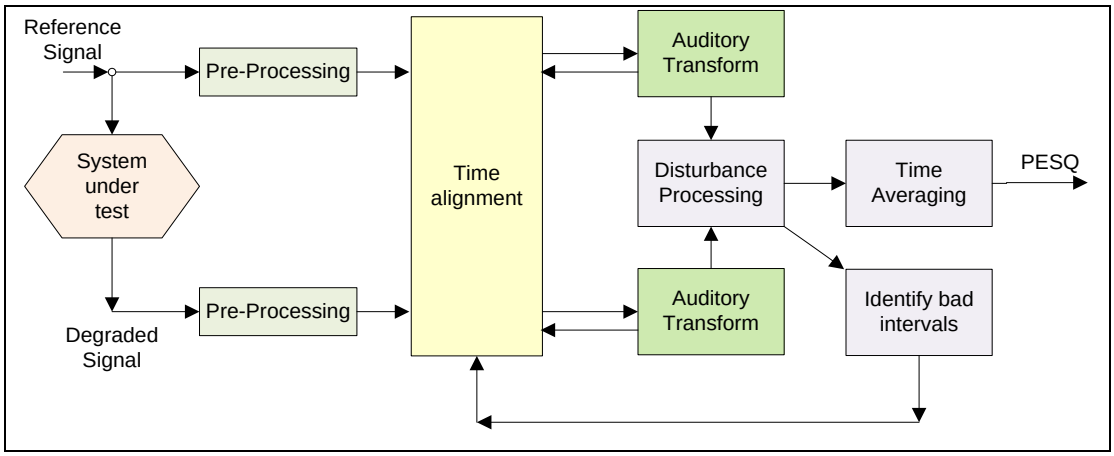


Figure 2.11 Block diagram of PESQ measure computation.

2.7.3.2 *Spectral Distance Measures Based on LPC*

Several objective measures were proposed based on the dissimilarity between all-pole models of the clean and enhanced speech signals (Quackenbush, S., Barnwell, T. & Clements, M., 1988). These measures assume that over short time intervals, speech can be represented by a p^{th} order all-pole model of the form

$$x(n) = \sum_{i=1}^{p} a_x(i)x(n-i) + G_x u(n) \tag{2.3}$$

where $a_x(i)$ are the coefficients of the all-pole filter (determined using linear prediction techniques), G_x is the filter gain, and $u(n)$ is a unit variance white noise excitation. Log-likelihood ratio is one of the most common all-pole-measures used to evaluate speech-enhancement algorithms. Cepstral distance measures derived from the linear predictive coding (LPC) coefficients are also used.

2.7.3.2.1 LLR Objective Speech Quality Measure. The log-likelihood ratio (LLR) measure is defined as:

$$d_{LLR}(a_x, \bar{a}_{\hat{x}}) = \log \frac{\bar{a}_{\hat{x}}^T R_x \bar{a}_{\hat{x}}}{a_x^T R_x a_x} \tag{2.4}$$

where $a_x^T = [1, -\alpha_x(1), -\alpha_x(2), \dots, -\alpha_x(p)]$ are the LPC coefficients of the clean signal, $\bar{a}_{\hat{x}}^T = [1, -\alpha_{\hat{x}}(1), -\alpha_{\hat{x}}(2), \dots, -\alpha_{\hat{x}}(p)]$ are the coefficients of the enhanced signal, and R_x is the *(p+1)x(p+1)* Toeplitz autocorrelation matrix of the clean signal.

2.7.3.2.2 Cepstrum Distance Objective Speech Quality Measure. The LPC coefficients can also be used to derive a distance measure based on cepstrum coefficients. This distance provides an estimate of the log-spectral distance between two spectra. The cepstrum coefficients can be obtained recursively from the LPC coefficients $\{a_j\}$ using the following expression (Rabiner, L. & Schafer, R., 1978)

$$c(m) = a_m + \sum_{k=1}^{m-1} \frac{k}{m} c(k) a_{m-k} \qquad 1 \le m \le p \tag{2.5}$$

where p, which is also given in Equation (2.3), is the order of the LPC analysis. A measure based on cepstrum coefficients can be computed as follows (Kitawaki, N., Nagabuchi, H., & Itoh, K., 1988)

$$d_{cep}(c_x, c_{\hat{x}}) = \frac{10}{\log_e 10} \sqrt{2 \sum_{k=1}^{p} [c_x(k) - c_{\hat{x}}(k)]^2} \tag{2.6}$$

where $c_x(k)$ and $c_{\hat{x}}(k)$ are the cepstrum coefficients of the clean and enhanced signals, respectively.

2.7.3.3 *Composite Objective Speech Quality Measure*

Composite measures are formed by combining multiple objective measures. The rationale behind the use of composite measures is that different objective measures capture different characteristics of the distorted signal, and therefore combining them in a linear or nonlinear fashion can potentially yield significant gains in correlations. One possibility is to use the following linear regression mode

$$y_i = f(x) + \varepsilon_i = \alpha_0 + \sum_{j=1}^{P} \alpha_j x_{ij} + \varepsilon_i \quad (2.7)$$

where $f(x)$ is the mapping function presumed to be linear, p is the number of objective measures involved, $\{y_i\}_{i=1}^{N}$ are the dependent variables corresponding to the subjective ratings of N samples of degraded speech, x_{ij} is the independent (predictor) variable corresponding to the *j*th objective measures computed for the i^{th} observation (degraded file), and ε_i is a random error associated with each observation. The regression coefficients α_i can be estimated to provide the best fit with the data using a least-squares approach which is described in Quackenbush, S., Barnwell, T. & Clements, M. (1988). The p objective measures considered in Equation (2.7) may include, among other measures, the LPC-based measures (e.g., LLR), segmental SNR measures (e.g., SNRseg), or the PESQ measure. The selection of objective measures to include in the composite measures is not straightforward and, in some cases, it is based solely on experimental evidence (trial and error) and intuition. Ideally, we would like to include objective measures that capture complementary information about the underlying distortions present in the degraded signal.

We have used the composite method described in Hu, Y. and Loizou, P. (2006) as one of our evaluation methods which involves LLR, SNRseg, weighted spectral slope (WSS), and PESQ measures.

2.7.3.4 *MARS Frequency-Variant fwSNRseg Measure*

A linear function $f(x)$ was assumed in Equation (2.7) for mapping p objective measures to the observed subjective ratings, $\{y_i\}_{i=1}^{N}$. Such a model is accurate only when the true form of the underlying function is linear. If it is not, then the modeling error will likely be large and the fit will be poor. Nonparametric models, which make no assumptions about the form of the mapping function, can alternatively be used. More specifically, models based on multivariate adaptive regression splines (MARS) have been found to yield better performance for arbitrary data sets (Friedman, J., 1991). Unlike linear and polynomial regression analysis, the MARS modeling technique is data driven and derives its functional form from the data. The basic idea of the MARS modeling technique is to recursively partition the domain into smaller subregions and use spline functions to locally fit the data in each region. The number of splines used in each subregion is automatically determined from the data. The MARS model has the following form

$$y_i = \alpha_0 + \sum_{j=1}^{M} \alpha_j B_j(x) + \varepsilon_i \tag{2.8}$$

where $B_j(x)$ are the basis functions, and M is the number of the basis functions that are automatically determined from the data (note that M could be larger than the number of objective measures, p). Spline basis funcitons of the following form were proposed in Friedman, J. (1991)

$$B_j(x) = \prod_{K=1}^{K_j} s_{kj} \max(0, x_{kj} - t_{kj}) \tag{2.9}$$

where x_{kj} are the predictor variables (values of the objective measures), t_{kj} are the split points (knots) determined from a recursive algorithm that partitions the domain into smaller subregions, K_j is the number of splits involved in the computation of the j^{th} basis function, and $s_{kj} = \pm 1$. One of the most powerful features of MARS modeling is that it allows for possible interactions between the predictor variables so that a better fit can be found for the target variable.

2.7.3.5 *Frequency Weighted SNRseg Objective Measure*

This method is a segmental SNR measure which is evaluated in frequency domain. For this measure to be meaningful, it is important that the original and

processed signals be aligned in time and that any phase errors present be corrected. The equation of the frequency domain segmental SNR can be given as follows

$$fwSNRseg = \frac{10}{M}\sum_{m=0}^{M-1}\frac{\sum_{j=1}^{K} B_j \log_{10}\left[F^2(m,j)/(F(m,j)-\hat{F}(m,j))^2\right]}{\sum_{j=1}^{K} B_j} \quad (2.10)$$

where B_j is the weight placed on the *j*th frequency band, K is the number of bands, M is the total number of frames in the signal, $F(m,j)$ is the filterbank amplitude of the clean signal in the j^{th} frequency band at the m^{th} frame, and $\hat{F}(m,j)$ is the filterbank amplitude of the enhanced signal in the same band. The main advantage of using the frequency-based segmental SNR over the time-domain SNRseg is the added flexibility of assigning different weights for different frequency bands of the spectrum. There is also the flexibility of choosing perceptually motivated frequency spacing such as critical-band spacing.

One potential problem with the estimation of SNRseg is that the signal energy during intervals of silence in the speech signal (which are abundant in conversational speech) will be very small, resulting in large negative SNRseg values, which will bias the overall measure. One way to remedy this is to exclude the silent frames from the sum in Equation (2.10) by comparing short-time energy values against a threshold.

2.7.3.6 *Frequency-Variant fwSNRseg Objective Measure*

As an alternative to the frequency domain fwSNRseg measure, the weights for each frequency band can be obtained using regression analysis, producing the so called *frequency-variant* objective measures (Quackenbush, S. et. al., 1988). This way, the weights can be chosen to give maximum correlation between the objective and subjective measures. For these measures, a total of K (one for each band) different objective measures, D_j, are computed for each file, where D_j is given as

$$D_j = \frac{1}{M}\sum_{m=1}^{M} 10\log_{10}\left[F^2(m,j)/(F(m,j)-\hat{F}(m,j))^2\right] \quad j = 1,2,\ldots,K \quad (2.11)$$

The optimal weights for each objective measure D_j of each band are obtained using K^{th}-order linear regression analysis, yielding the following frequency-variant objective measure

$$fwVar = a_0 + \sum_{j=1}^{K} a_j D_j \quad (2.12)$$

where $\{a_j\}$ are the regression coefficients, D_j is given in Equation (2.11), and K is the number of bands ($K = 6$ in Quackenbush, S. et. al., 1988). Nonlinear regression analysis can alternatively bu used to derive the frequency-variant objective measures.

2.7.3.7 *Speech Recognition*

Speech recognition converts spoken words to text. Recognizing the speaker can simplify the task of translating speech. Speech recognition applications include voice user interfaces such as voice dialing, call routing, voice control of home appliances, search, simple data entry, preparation of structured documents, speech-to-text processing (e.g., word processors or emails).

Both acoustic modeling and language modeling are important parts of modern statistics based speech recognition algorithms. Hidden Markov models (HMMs) are widely used in many systems. Language modeling has many other applications such as smart keyboard and document classification.

There are several speech-to-text algorithms which transcribe the spoken words simultaneously. We use "Dragon NaturallySpeaking" software, version 10.0, from Nuance Communications, Inc. for evaluating the intelligibility performance of systems that we develop. Methods other than speech recognition can only be used to evaluate the quality. However, speech recognition is directly related with the intelligibility. This evaluation is performed as follows:

- Firstly, it is necessary to train the speech-to-text software. A clean speech file is used to train the software.
- After the software is trained, the evaluation is performed using another speech file of the same speaker; that is, if an audio book is used, the first

chapter of the book can be used to train the algorithm and the second chapter of the same book can be used to evaluate the algorithm.

- In order to evaluate the algorithms, noise files including different types of noise (white noise, pink noise, speech noise, etc.) are added to the speech file reserved for evaluation, separately.
- These noisy speech files are used as input files to the evaluation module. The audio output of the evaluation module is connected to the audio input of the computer and the recordings are saved. Each noisy file is recorded with and without application of the enhancement algorithm. We do not use the noisy files directly. We record them from the output of the evaluation module without any processing in order to eliminate the effect of analog recording, evaluation module, cables, etc.
- Both the noisy speech files and the enhanced speech files are applied to the speech-to-text software as inputs. If the number of correctly identified words is greater in the enhanced file, it means that the intelligibility increases. Otherwise, if the number of correctly identified words in the noisy speech file is greater than the one in the enhanced file, then it means that the intelligibility decreases.
- This operation is performed for each noise type that we want to evaluate and according to all the results, the intelligibility performance of the system is obtained.

CHAPTER THREE
OMAP – L137 EVALUATION MODULE

The OMAP-L137 EVM is a standalone development platform that enables users to evaluate and develop applications for the OMAP-L137 processor. Note that the OMAP-L137 and C6747 are the same devices except the fact that C6747 does not include the ARM9 core. Detailed information concerning the OMAP L-137 processor can be found in Appendix A and OMAP-L137 Processor datasheet, 2008.

Figures 3.1 and 3.2 (OMAP-L137 EVM Tech. Ref., 2008) show the image of OMAP-L137 EVM and its block diagram, respectively.

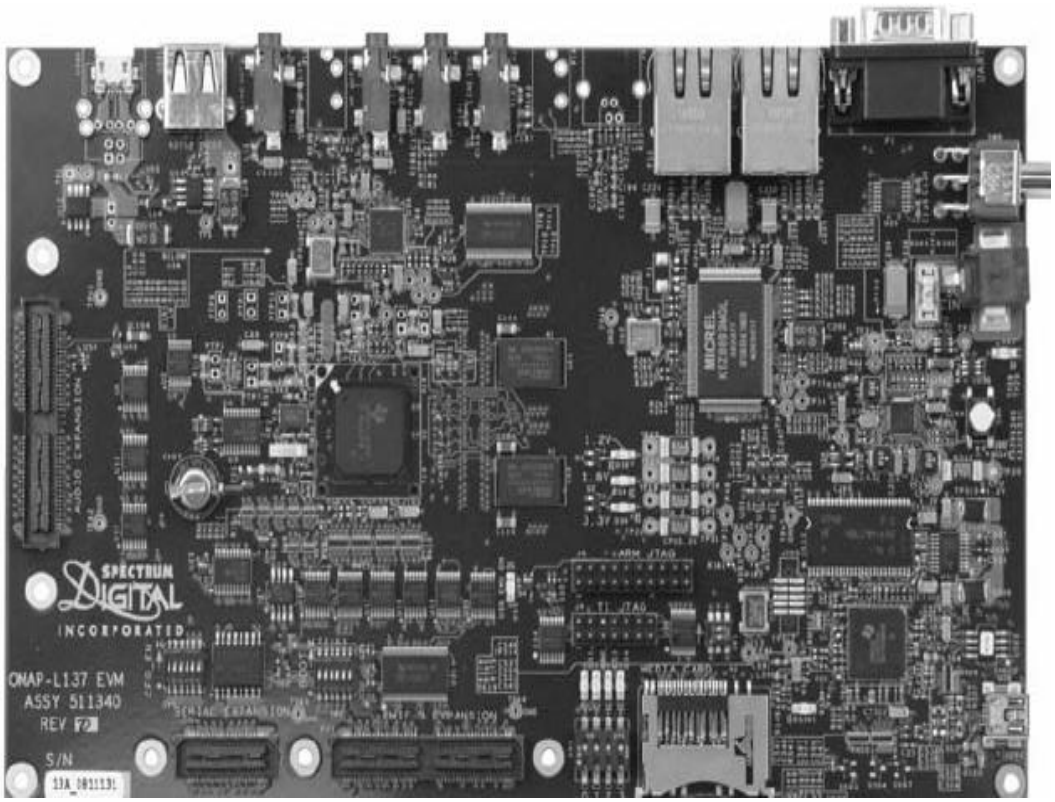


Figure 3.1 OMAP-L137 EVM.

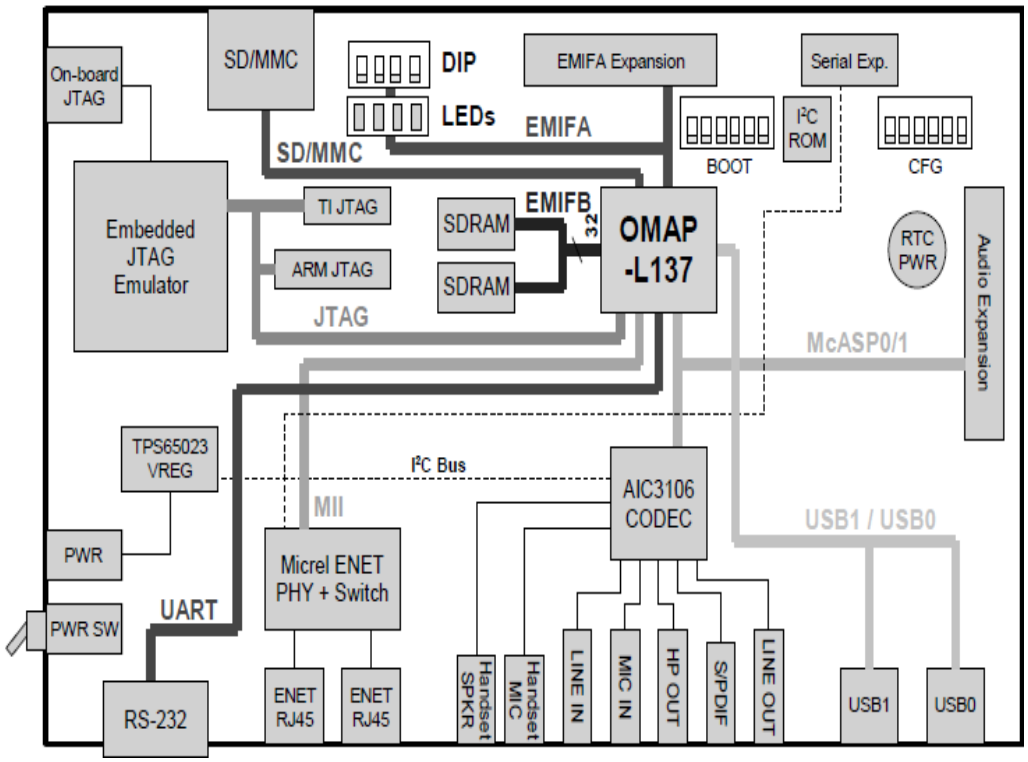


Figure 3.2 Block diagram of OMAP-L137 EVM.

OMAP-L137 is selected for this thesis study because of its advantages in terms of performance and flexibility such as being a floating point processor, including external SDRAM, and having an operating frequency of 300 MHz. Key features of the EVM are;

- A Texas Instruments OMAP-L137 device with a C674x VLIW DSP floating point processor and an ARM926EJ-S processor operating up to 300 MHz,
- 64 Megabytes SDRAM,
- SPI Boot EEPROM,
- Port Ethernet Phy/switch,
- SD/MMC/MMC Plus media card interfaces,
- TLV320AIC3106 Stereo Codec,
- USB 1.1 USB2 2.0 interfaces,
- RS-232 interface,
- On chip real time clock,
- User LEDs/4 position user DIP switch,
- Expansion connectors for daughter card use and embedded JTAG emulation.

Figure 3.3 (SPRT529B, 2009) shows the software structure of the OMAP-L137 EVM. Both processors inside the OMAP-L137 processor can be used or only one of them can be in use based on the application at hand. If only a digital signal processing task is to be performed, then it makes more sense to use only the DSP processor. However, if one wants to develop a media player, for example, then both ARM and DSP processors can be used.

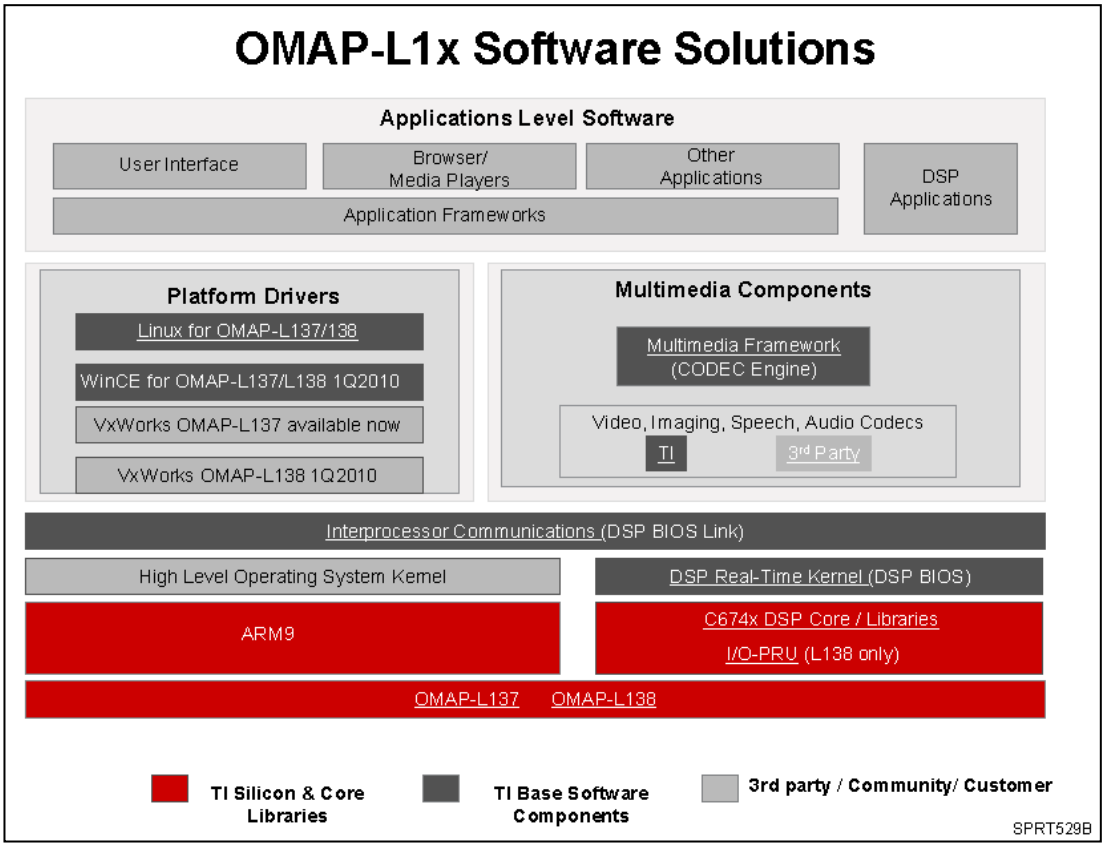


Figure 3.3 OMAP-L137 Software Solutions.

3.1 TI C67x DSP Library

The TI C67x DSPLIB is an optimized floating-point DSP Function Library for C programmers using TMS320C67x devices. It includes C-callable, assembly-optimized general-purpose signal-processing routines. These routines are typically used in computationally intensive real-time applications where optimal execution speed is critical. In addition, by providing ready-to-use DSP functions, TI DSPLIB can significantly shorten DSP application development time. Some example functions are given as follows;

- DSPF_sp_autocor (autocorrelation)
- DSPF_sp_biquad (biquad filter)
- DSPF_sp_convol (convolution)
- DSPF_sp_fftSPxSP (mixed radix forward fast Fourier transform (FFT) with bit reversal)

3.2 Direct Memory Access (DMA)

Direct memory access (DMA) is a feature of modern computers and microprocessors allowing certain hardware subsystems within the computer to access system memory for reading and/or writing independent of the central processing unit. Many hardware systems use DMA including disk drive controllers, graphics cards, network cards, and sound cards.

Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without a DMA channel. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time and allowing computation and data transfer concurrency.

Without DMA, using programmed input/output (PIO) mode for communication with peripheral devices, or load/store instructions in the case of multicore chips, the CPU is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other tasks. With DMA, the CPU would initiate the transfer, do other operations while the transfer is in progress, and receive an interrupt from the DMA controller once the operation has been done. This is especially useful in real-time computing applications where not stalling behind concurrent operations is critical. Another related application area is various forms of stream processing where it is essential to have data processing and transfer in parallel in order to achieve sufficient throughput.

CHAPTER FOUR
THE EXPERIMENTAL STUDY

In this chapter of the thesis, we outline the applied algorithms for the speech enhancement problem. First, a feasibility study for the studied algorithms is given and then the flowchart of the proposed algorithm is presented. Then, the equations and explanations of the applied speech enhancement, noise estimation, and evaluation algorithms are provided.

4.1 Hardware Configuration

We have used the following hardware configuration shown in Figure 4.1 during our studies. The computer is used for uploading the software to the evaluation module and for inputting audio files to the EVM. All the processing operations are performed on the EVM and the processed audio is given from the audio output of the EVM to the speakers or headphone.

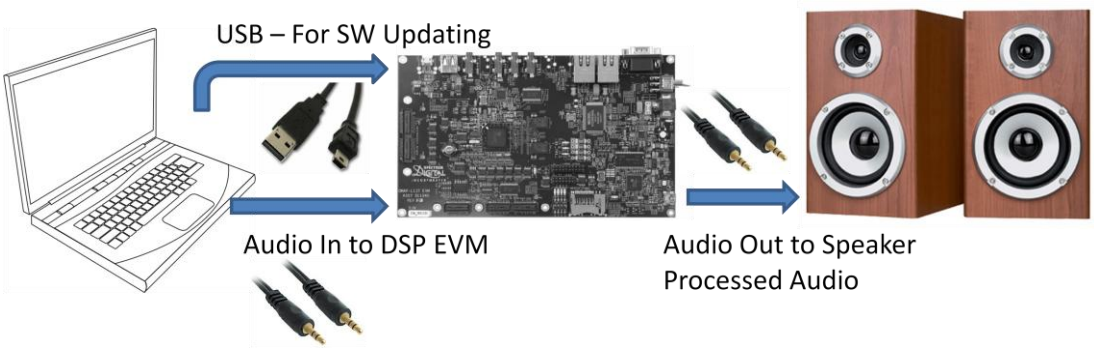


Figure 4.1 The hardware configuration used during this thesis study.

The configuration shown in Figure 4.1 is used for the subjective evaluation of the developed algorithms whereas we have used the configuration in Figure 4.2 for the objective evaluation of the developed algorithms. In this configuration, the only difference from Figure 4.1 is the use of a sound card for recording the processed speech data. By this way, the recorded files can be evaluated via the evaluation algorithms on the computer.

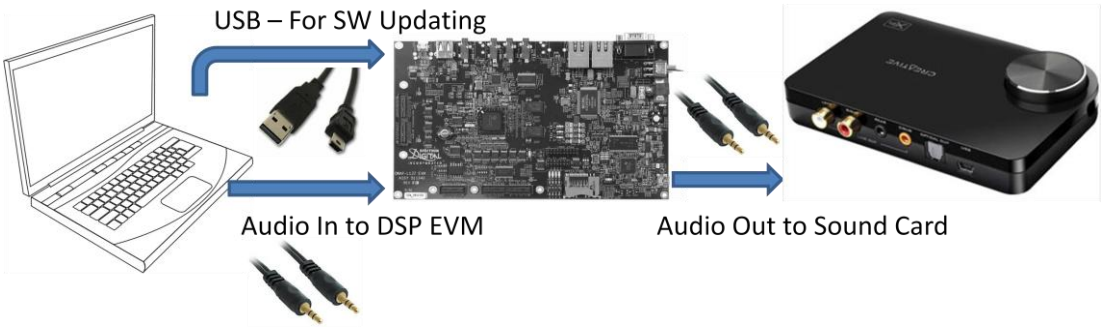


Figure 4.2 The hardware configuration used for objective evaluation.

4.2 The Feasibility Study for the Algorithms

The important points for developing our enhancement algorithm can be given as follows.

- The enhancement algorithm should include a noise estimation algorithm to be suitable for real-time processing, because the noise source is not considered as stationary. This operation is mandatory.
- The enhancement algorithm might include a block that determines whether the frame is speech or not. This step is optional.
- A decision mechanism might be necessary to decide whether a considered speech frame is intelligible enough. If it is not intelligible, then it requires enhancement. This step is also optional.
- The enhancement algorithm should not introduce either speech distortion or musical noise as much as possible. This is quite critical in terms of the performance of the developed algorithm.
- In order for the algorithm to be successful, it should at least enhance either the quality or intelligibility of the speech. If it enhances only one of them, it should not degrade the other. For example, if the algorithm enhances only the quality, it is desired that it does not degrade the intelligibility, or at least the intelligibility should stay the same.
- The whole process should not be too long for real-time processing. The algorithm at hand should finish its job until the next frame is taken. This issue is very critical, because we do not have a chance to store the data. All the

operations on one frame should be completed in approximately up to 30 msec at most.

4.3 The Basic Algorithm

In Figure 4.3, the flowchart of the basic algorithm is given.

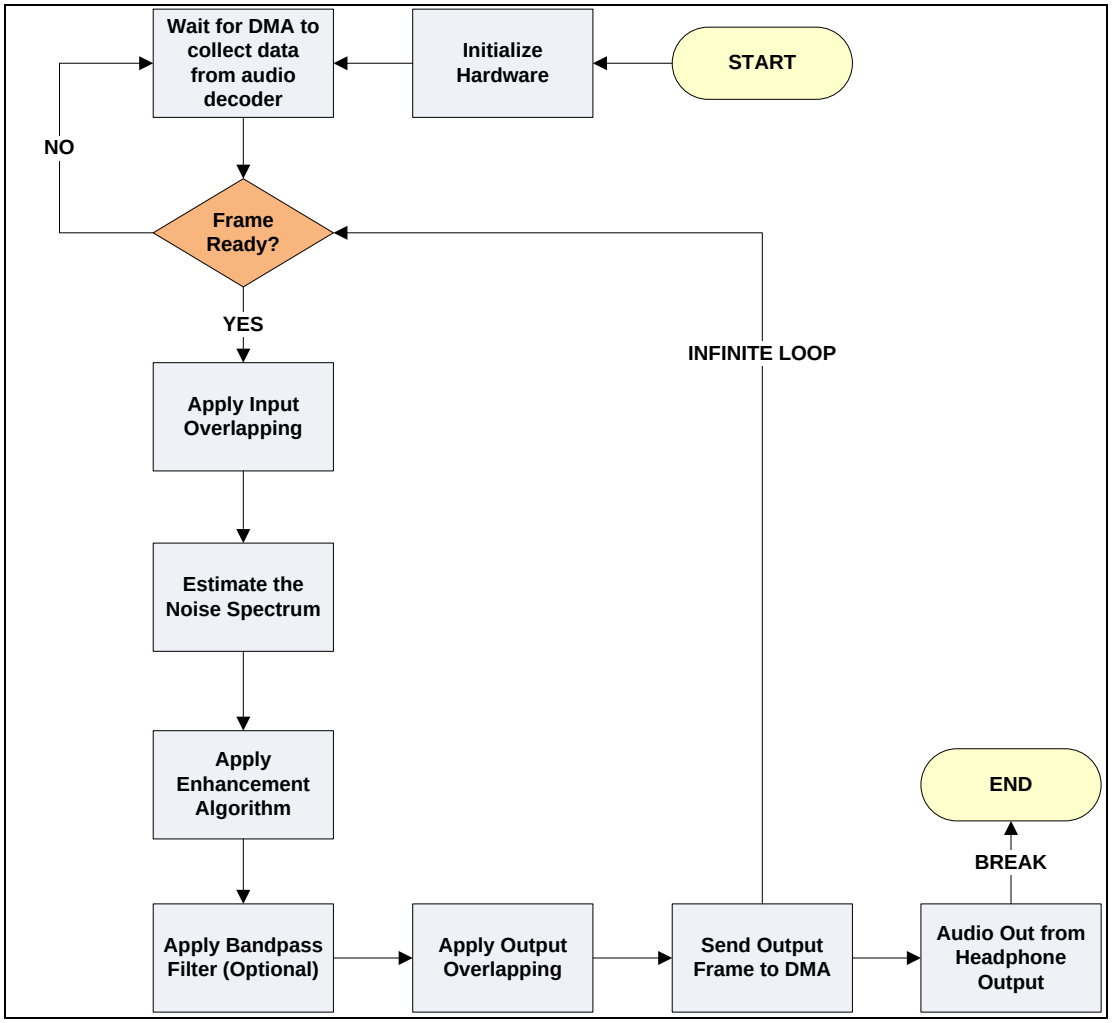


Figure 4.3 The flowchart of the basic algorithm.

After the software is executed, the first operation performed here is to initialize the hardware. During this process, the audio driver and the audio codec, the McASP port which is used to carry digital audio data, and the DMA block are initialized.

After the initialization process, the analog audio data are collected from the line-in audio input which exists on the EVM. This audio data correspond to the noisy speech

signal. The DMA block collects this data in a previously allocated buffer. When the audio frame is ready, the required operations on the collected data are performed. The first operation to be performed is the application of overlapping algorithm to the input data. Then, the noise spectrum of the input data should be estimated, because it is most likely that the noise spectrum is non-stationary. Thus, it is necessary to update the noise spectrum constantly. For the current frame, the noise spectrum is estimated and updated, and then the speech enhancement algorithm is applied. It is possible to increase the number of algorithms to be applied in this step. For example, a bandpass filter for filtering out the undesired frequencies other than speech frequencies could be used. After the processing operations are ended, an overlapping operation on the output data is performed. Next, the enhanced audio frame is transferred to the output DMA buffer and the DMA block sends these data to the codec. The last operation to be performed here is the digital-to-analog conversion of the audio data which is performed by the controller automatically and the audio output is given from the headphone output.

The steps explained above are for one audio frame. As shown in the flowchart, these steps are applied successively to every input audio frame until the user halts the software.

The overlapping, speech enhancement, noise estimation, and filtering operations shown in the flowchart are explained in the following sections.

4.4 Application of the Algorithms on EVM

As described in Section 1.2, there are three main steps to be performed for a typical speech enhancement study. In Figure 4.4, there is a summary of these steps.

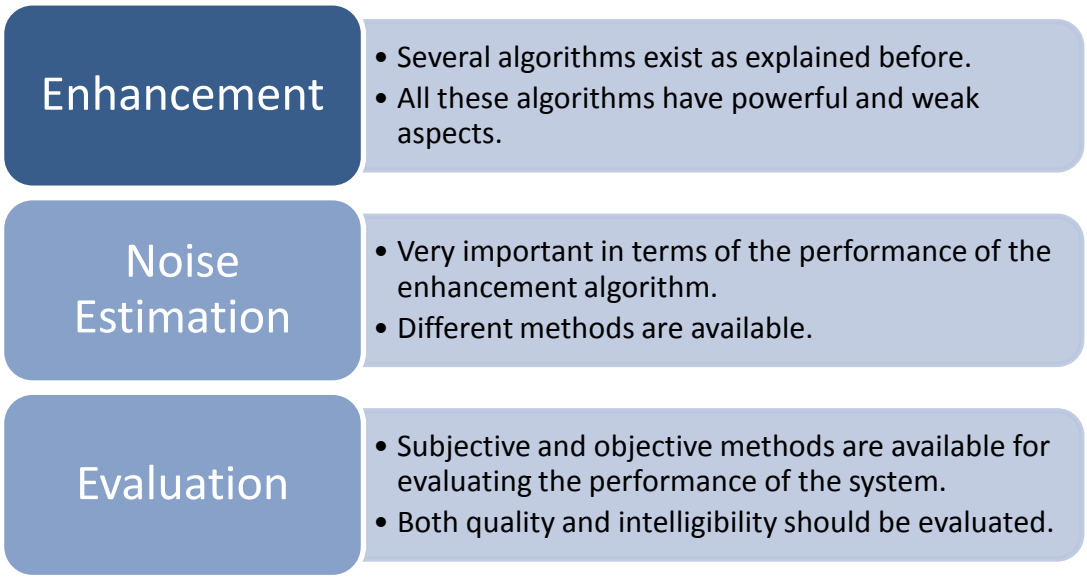


Figure 4.4 Summary of typical steps for the solution of a speech enhancement problem.

We explain speech enhancement and noise estimation algorithms applied on the EVM. As mentioned in Chapter Two, there are a variety of algorithms for both speech enhancement and noise estimation in the literature. Obviously, it is not practical to try all these algorithms. The purpose here is to develop an efficient system which is proper for a real-time application and has good performance. The main challenges are performance and execution time of the algorithm. Some algorithms such as subspace algorithms can have better performance in terms of intelligibility, but they cannot be used in a real-time system, because they require longer execution time.

In light of these facts, "Log-spectral amplitude estimator (LOG-MMSE)" proposed in Ephraim, Y. & Malah, D. (1985) and "Wiener filtering algorithm based on a priori SNR estimation (Wiener-SNR)" described in Scalart, P. & Filho, J. (1996) are selected as the enhancement algorithms, because these two algorithms are known to perform better (Loizou, P. C. (2007)).

We have tried several noise estimation algorithms together with the aforementioned enhancement algorithms. Since the noise estimation algorithm directly affects the performance of the system, different results are obtained by application of different noise estimation algorithms. We have tried voice activity

detection (VAD), “Weighted-spectral averaging”, “SNR-dependent recursive averaging algorithms” and “Histogram-based algorithms” as explained in (Loizou, P. C. (2007)) together with the enhancement algorithms.

For the application of these algorithms, one more step called overlapping is necessary. Overlapping operation is not only related with our algorithm, but also it is mandatory for a real-time audio application in order to prevent the discontinuities which cause a “popping” sound.

The last step performed is the filtering operation after the application of the enhancement algorithm, because the speech related frequencies do not occupy the whole audio frequency range and filtering out the undesired frequency bands increases the performance of the system.

The block diagram given in Figure 4.5 shows the essential steps of the whole algorithm applied for speech enhancement. In this section, we will explain implementation details at each step.

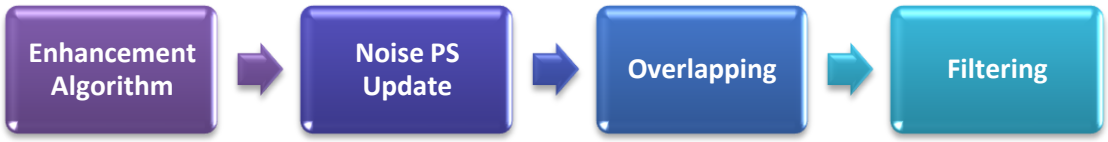


Figure 4.5 Main steps of the developed system.

4.4.1 Overlapping Method

As shown in Figure 4.3, after the audio frame is taken from the DMA buffer, the overlapping should be performed on the data. Overlapping is used to prevent discontinuities between successive frames. There are some methods such as overlap-add and overlap-save for the implementation of overlapping.

Our input and output buffers include 768 samples and the processing buffer includes 1024 samples, because we have used 25% overlapping rate. Since our input and output buffers include 768 samples and the sampling frequency is 32 kHz, the

time duration for processing one data packet can be at most 24 msec. As shown in Figure 4.6, the first 256 elements of the processing buffer is the last 256 elements of the previous input buffer and the remaining 768 elements of the processing buffer are the elements of the current input buffer. The last 256 elements of each input buffer are stored for the next processing buffer.

All the signal processing operations are performed on the processing buffer. For our application, these tasks could be obtaining the power spectral density estimate of noise, taking FFT, inverse FFT, and executing enhancement algorithm. After the processing tasks are done, the output buffer that is sent to the DMA for audio output should be obtained. This operation is illustrated in Figure 4.7.

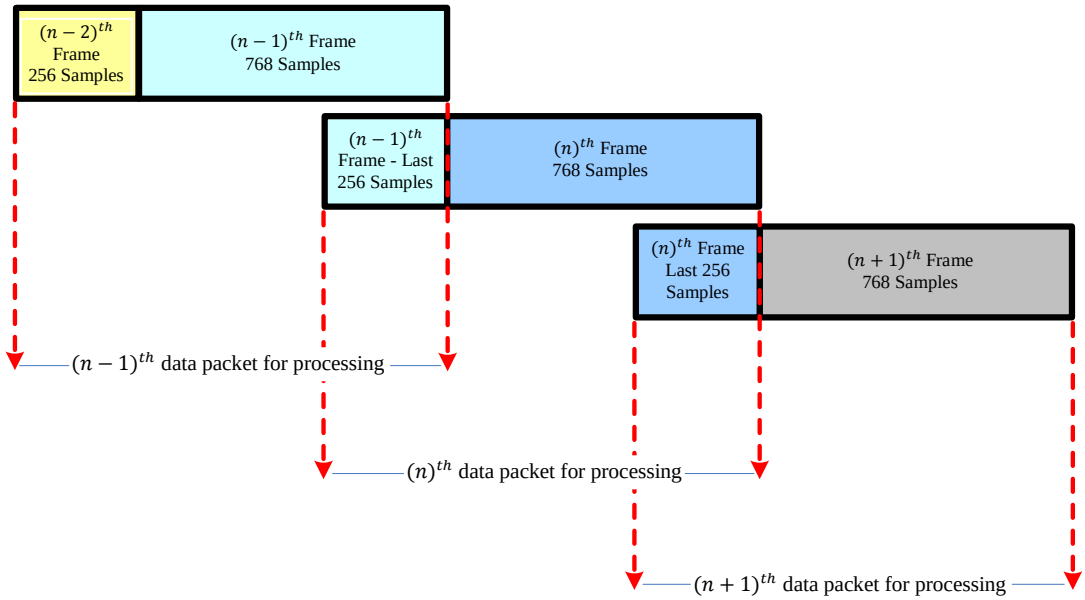


Figure 4.6 Demonstration of the overlapping on the input buffer.

As shown in Figure 4.7, the processed frames have 1024 samples, but the output buffer has only 768 samples. Due to overlapping, the last 256 samples of each processed frame are stored to be used with the next frame. As shown in Figure 4.7, the last 256 samples of the $(n-1)^{th}$ frame are multiplied with a decreasing linear ramp function and the first 256 samples of the n^{th} frame are multiplied with an increasing ramp function. Then, these two groups of multiplied samples are summed up.

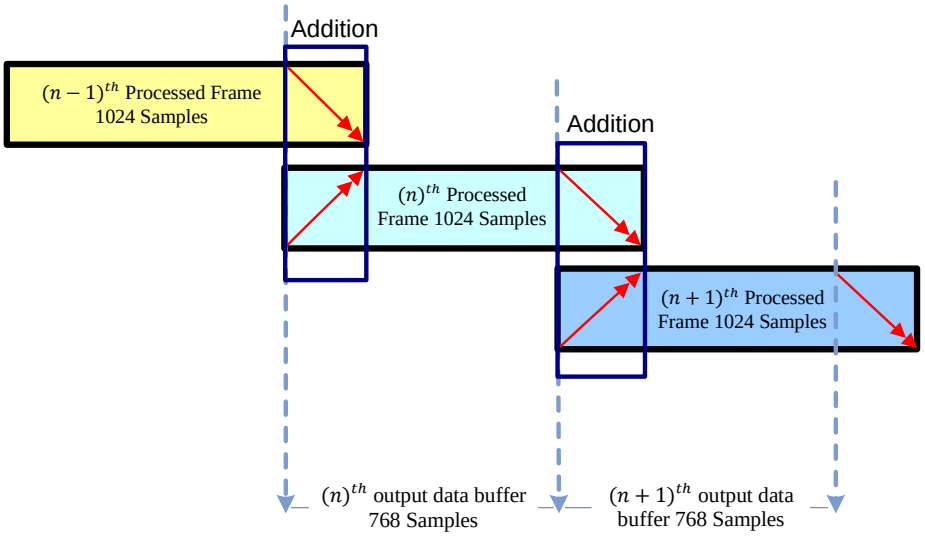


Figure 4.7 Demonstration of the overlapping on the output buffer.

Figure 4.8 shows this operation in a different way. Note that the last 256 samples of the n^{th} frame are not used with the n^{th} frame. They are stored to be used with the $(n+1)^{th}$ frame. As explained before, the last 256 samples of the $(n-1)^{th}$ frame are multiplied with a decreasing linear ramp function and added to the first 256 samples of the n^{th} frame which are multiplied with an increasing linear ramp function. By this way, the n^{th} output buffer containing 768 samples is created.

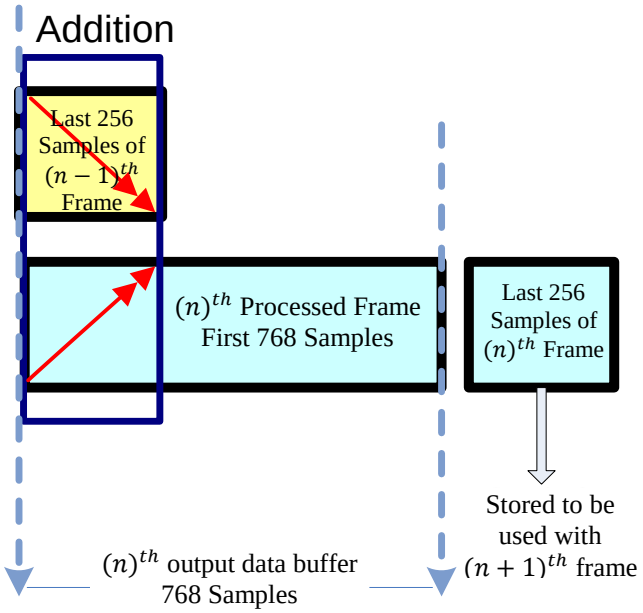


Figure 4.8 Demonstration of the overlapping process on the output buffer (illustration on a single frame).

4.4.2 The Applied Enhancement Algorithms

In this section, we explain the two enhancement algorithms that we have used during our studies; LOG-MMSE and Wiener-SNR algorithms.

4.4.2.1 *Wiener-SNR Algorithm*

This algorithm is based on the Wiener filtering approach. In speech enhancement applications, the input signal $y(n)$ expressed below is the noisy speech signal

$$y(n) = s(n) + d(n) \tag{4.1}$$

where $s(n)$ is the clean speech signal and $d(n)$ is the noise signal. The objective of the Wiener filter is therefore to produce an estimate of the clean signal $s(n)$.

The corresponding Wiener filters can be derived in time or frequency domain. Here, we do not give the derivation which can be found in Loizou, P. C. (2007). After the frequency domain derivation, the Wiener filter can be obtained as

$$H(w_k) = \frac{P_{ss}(w_k)}{P_{ss}(w_k)+P_{dd}(w_k)} \tag{4.2}$$

where P_{ss} is the power spectrum of the clean speech signal and P_{dd} is the power spectrum of the noise signal. As we can easily see from Equation (4.2), this filter is not realizable, because we do not know the clean speech signal (it is the signal to be obtained). If we define

$$\xi_k \triangleq \frac{P_{ss}(w_k)}{P_{dd}(w_k)} \tag{4.3}$$

as the *a priori* SNR at frequency w_k, we can also express the Wiener filter as follows

$$H(w_k) = \frac{\xi_k}{\xi_k+1} \tag{4.4}$$

Note that $0 \leq H(w_k) \leq 1$, and $H(w_k) \approx 0$ when $\xi_k \to 0$ (at extremely low-SNR regions) and $H(w_k) \approx 1$ when $\xi_k \to \infty$ (at extremely high-SNR regions). According to Equation (4.4), the Wiener filter emphasizes portions of the spectrum where SNR is high and attenuates portions of the spectrum where the SNR is low.

There are many written articles and many methods are available to obtain an estimate of *a priori* SNR ξ_k (Hu, Y. & Loizou, P., 2004, Scalart, P. And Filho, J., 1996, Cappe, O., 1994, Ephraim, Y. And Malah, D., 1984). For our algorithm, we have used a noniterative approach used in Scalart, P. & Filho, J. (1996) for estimating the Wiener gain function which can be expressed in terms of ξ_k

$$g(k) = \frac{\xi_k}{\xi_k + \mu} \tag{4.5}$$

where μ is the Lagrangian multiplier which is an adjustable parameter. The focus was on getting a good (low-variance) estimate of the *a priori* SNR ξ_k needed in the Wiener gain function, $g(k)$, because it is known that a low-variance estimate of ξ_k can eliminate musical noise. In the method we used, the *a priori* SNR ξ_k was estimated using the decision-directed method (Scalart, P. & Filho, J., 1996). More specifically, ξ_k was estimated as a weighted combination of the past and present estimates of ξ_k. At frame m, $\hat{\xi}_k(m)$ was estimated as

$$\hat{\xi}_k(m) = \alpha * \frac{|\hat{X}_k(m-1)|^2}{|D_k(m-1)|^2} + (1-\alpha) * max\left(\frac{|Y_k(m)|^2}{|D_k(m)|^2} - 1{,}0\right) \tag{4.6}$$

with α being a smoothing constant ($\alpha = 0.98$ in Scalart, P. & Filho, J., 1996). $\hat{X}_k(m-1)$ denotes the enhanced signal spectrum obtained at frame *m-1*, and $Y_k(m), D_k(m)$ denote the noisy speech and noise spectra, respectively. $\hat{\xi}_k(m)$ can be approximated as

$$\hat{\xi}_k(m) = \alpha * \hat{\xi}_k(m-1) + (1-\alpha) * \tilde{\xi}_k(m) \tag{4.7}$$

where $\tilde{\xi}_k(m)$ denotes the current frame estimate of ξ_k. This recursion provides smoothness in the estimate of ξ_k, and consequently can eliminate the musical noise.

The flowchart given in Figure 4.9 shows the steps of the Wiener-SNR algorithm used in the software and Figure 4.10 shows the related equations that are applied at each step.

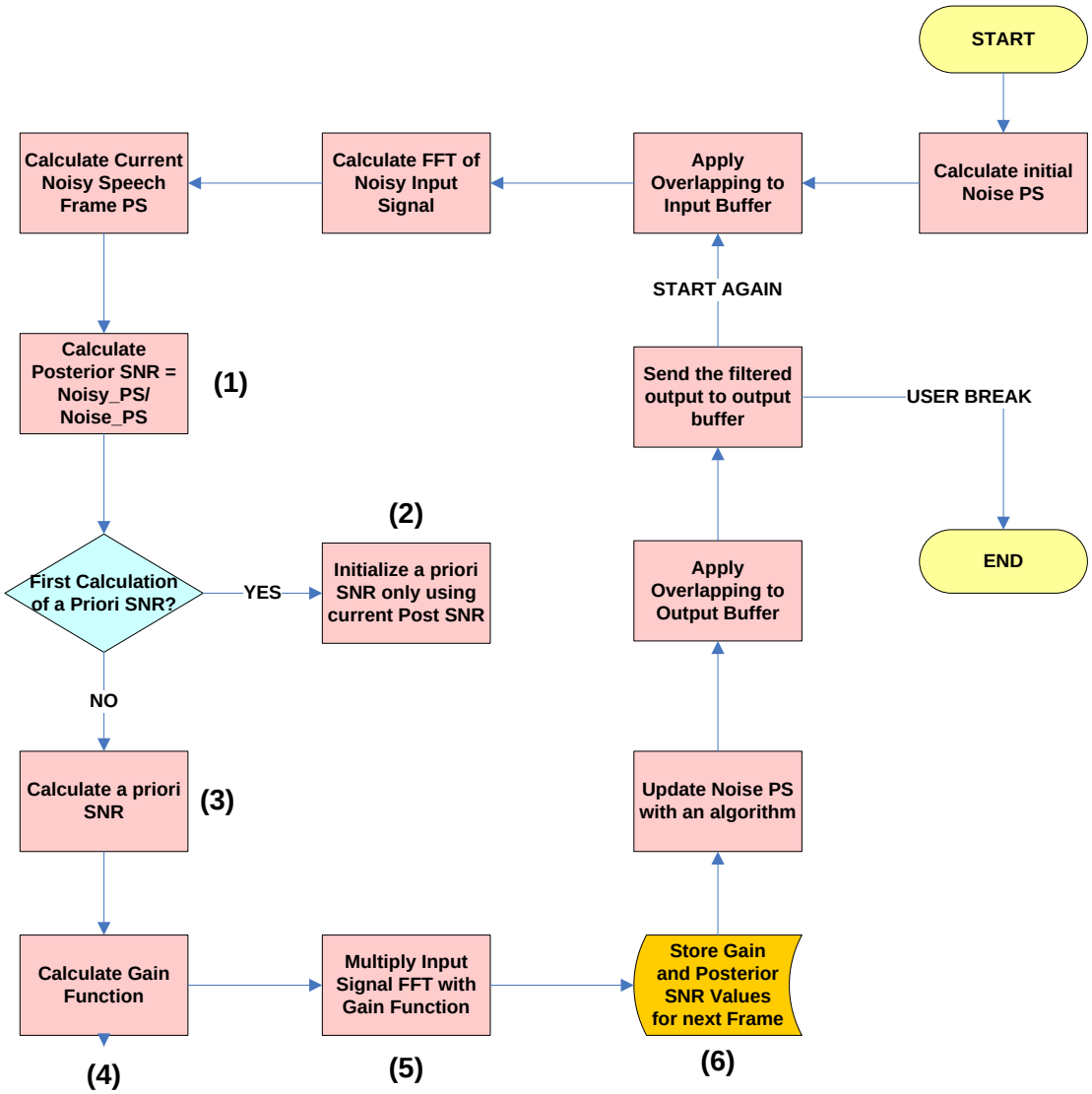


Figure 4.9 Flowchart of Wiener-SNR algorithm for speech enhancement.

$$posterior_{SNR} = \frac{Noisy\ Speech\ PS}{Noise\ PS} = \frac{P_{yy}}{P_{dd}} \quad (1)$$

$$a\ priori\ SNR = \xi_k = \alpha + (1-\alpha) * (posterior_{SNR} - 1) \quad (2)$$

$$a\ priori\ SNR = \xi_k = \alpha * \sqrt{G_{prev}} * posterior_{prev} + (1-\alpha) * (posterior_{SNR} - 1) \quad (3)$$

$$Gain = G = \sqrt{\frac{\xi_k}{\xi_k + 1}} \quad (4)$$

$$Output_{FFT} = Input_{FFT} * G \quad (5)$$

$$G_{prev} = G \ \ and \ \ posteri_{prev} = posteri_{SNR} \quad (6)$$

Figure 4.10 Defining equations of Wiener-SNR algorithm.

4.4.3 LOG-MMSE Estimator

The Wiener filtering approach given in the previous section assumes a linear relationship between the clean spectrum and the true spectrum. Acknowledging the importance of the short-time spectral amplitude (STSA) on speech intelligibility and quality, several authors have proposed optimal methods for obtaining the spectral amplitudes from noisy observations (Ephraim, Y. & Malah, D., 1985, Cohen, I., 2005, Hasan, M., Salahuddin, S. & Khan, M., 2004 and Cappe, O., 1994).

The MMSE estimator, unlike the Wiener estimator, does not assume the existence of a linear relationship between the observed data and the estimator, but it does require knowledge about the probability distributions of the speech and noise DFT coefficients.

Although a metric based on the squared error of the magnitude spectra as given in Loizou, P. C. (2007) is mathematically tractable, it may not be subjectively meaningful. It has been suggested that a metric based on the squared error of the log-magnitude spectra may be more suitable for speech processing. In Ephraim, Y. & Malah, D. (1985), the derivation of an estimator that minimizes the mean-square error of the log-magnitude spectra

$$E\{\left(\log X_k - \log \hat{X}_k\right)^2\} \tag{4.8}$$

is given. In Equation (4.8), $E\{\}$ is the expectation operator. The optimal LOG-MMSE estimator can be obtained by evaluating the conditional mean of the $\log X_k$ as

$$\log \hat{X}_k = E\{\log X_k \,|Y(w_k)\} \tag{4.9}$$

from which we can solve for $\hat{X}_k$ as

$$\hat{X}_k = \exp\left(E\{\log X_k \,|Y(w_k)\}\right) \tag{4.10}$$

Lengthy derivations are omitted here. However, they can be found in Ephraim, Y. & Malah, D. (1985). As a result, the optimal LOG-MMSE estimator is obtained as

$$\hat{X}_k = \frac{\xi_k}{\xi_k+1} * exp\left(\frac{1}{2}\int_{v_k}^{\infty}\frac{e^{-t}}{t}dt\right) * Y_k \triangleq G_{LSA}(\xi_k, v_k) * Y_k \tag{4.11}$$

where ξ_k is the *a priori* SNR, and $G_{LSA}(\xi_k, v_k)$ is the gain function of the LOG-MMSE estimator. The integral in Equation (4.11) is known as the exponential

integral and can be evaluated numerically. The exponential integral, *Ei(x)*, can be approximated as follows

$$Ei(x) = \int_x^{\infty} \frac{e^{-x}}{x} dx \approx \frac{e^x}{x} \sum_k \frac{k!}{x^k} \tag{4.12}$$

Some other approximations can also be found in the literature (Ephraim, Y. & Cohen, I., 2006).

Figure 4.11 shows the flowchart of the LOG-MMSE algorithm including the steps of the algorithm and Figure 4.12 shows the equations used in these steps.

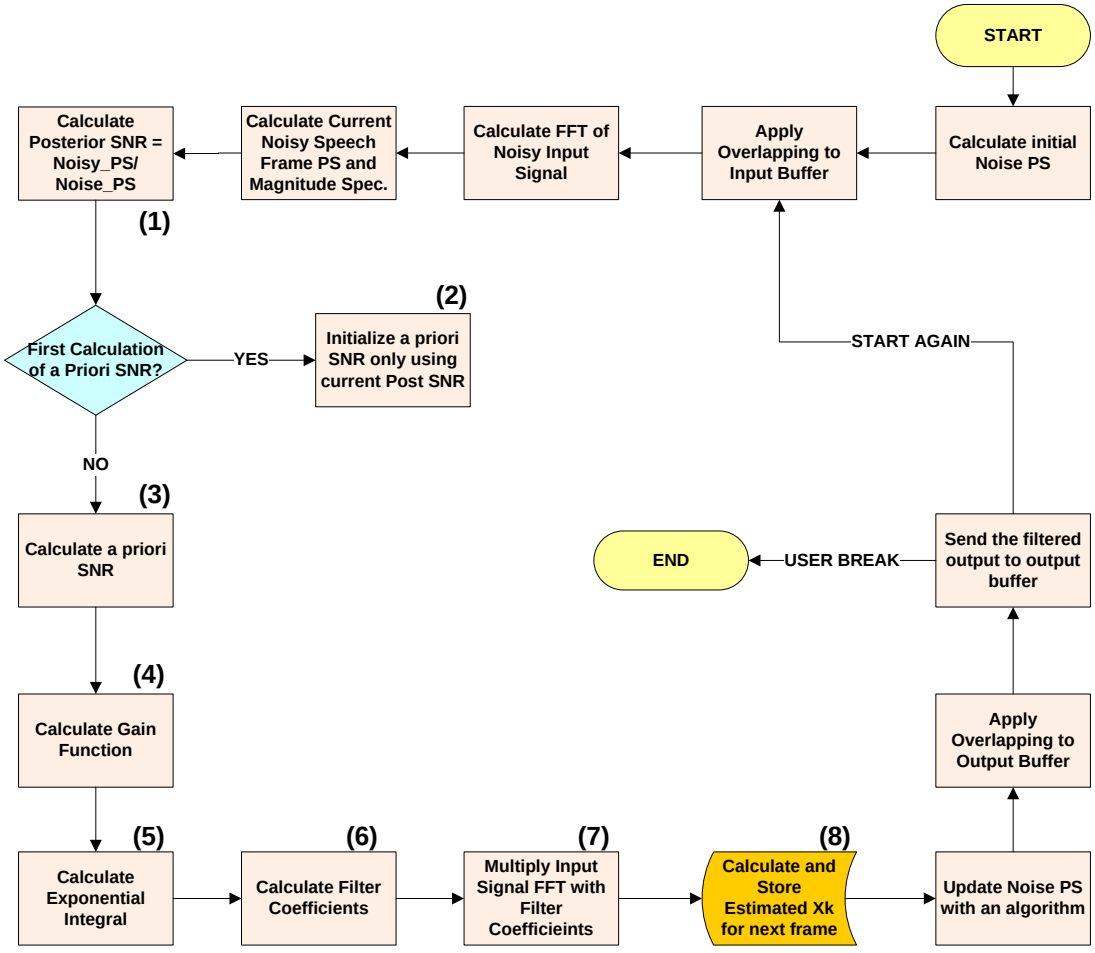


Figure 4.11 Flowchart of LOG-MMSE algorithm for speech enhancement.

$$posterior_{SNR} = \gamma_k = \frac{Noisy\ Speech\ PS}{Noise\ PS} = \frac{P_{yy}}{P_{dd}} \quad \textbf{(1)}$$

$$a\ priori\ SNR = \xi_k = \alpha + (1-\alpha) * (\gamma_k - 1) \quad \textbf{(2)}$$

$$a\ priori\ SNR = \xi_k = \alpha * \frac{X_{k\_prev}}{noise\_ps} * posterior_{prev} + (1-\alpha) * (\gamma_k - 1) \quad \textbf{(3)}$$

$$Gain = G = \frac{\xi_k}{\xi_k + 1} \quad \textbf{(4)}$$

$$Exp\ Integral = Ei_{vk} = \int_{v_k}^{\infty} \frac{e^{v_k}}{v_k} dv_k \approx \frac{e^{v_k}}{v_k} \sum_k \frac{k!}{{v_k}^k}\ , where\ v_k = G * \gamma_k \quad \textbf{(5)}$$

$$hw = G * e^{Ei\_vk} \quad \textbf{(6)}$$

$$Output_{FFT} = Input_{FFT} * hw \quad \textbf{(7)}$$

$$\widehat{X}_{k\_prev} = \left(\widehat{Sig}_{mag}\right)^2\ , where\ \widehat{Sig}_{mag} = |Input_{FFT}| * hw \quad \textbf{(8)}$$

Figure 4.12 Equations of LOG-MMSE algorithm.

4.4.4 Voice-Activity Detection (VAD)

As explained in the introduction section, voice activity detection (VAD) is the process of discriminating between voice activity and silence. This is a binary decision algorithm and is executed by comparing the calculated variable for voice activity decision with a constant value.

We use VAD at the beginning for updating the noise spectrum. However, we have to note that it is not very successful, especially for non-stationary noise case.

Detailed information about different VAD algorithms can be found in Loizou, P. C. (2007). The flowchart of the VAD algorithm used in this thesis study is given in Figure 4.13 and the related equations are given in Figure 4.14.

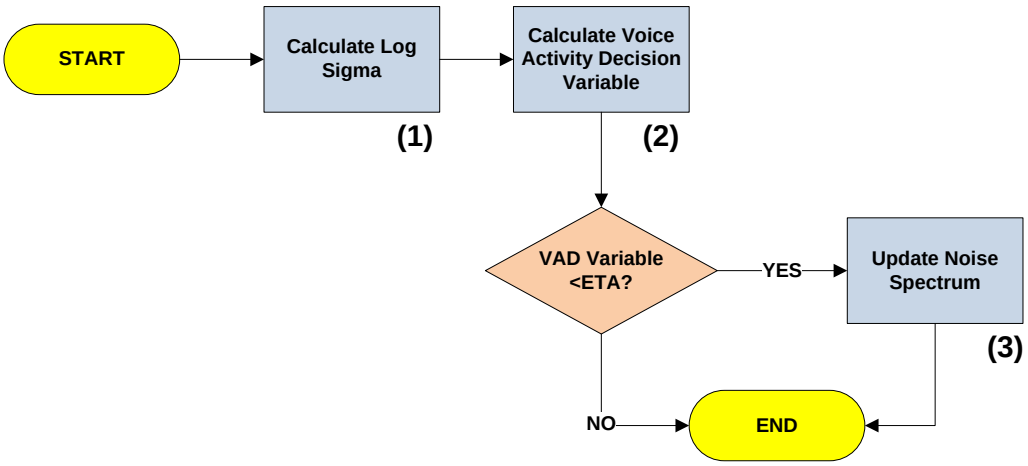


Figure 4.13 Flowchart of VAD for noise power spectrum estimation.

$$log_{sigma} = \gamma_k * \frac{\xi_k}{\xi_k + 1} - \log(1 + \xi_k) \quad \textbf{(1)}$$

$$VAD_{dec} = VAD_{dec} + log_{sigma} \quad \textbf{(2)}$$

$$noise_{ps} = P_{dd} = \mu * noise_{ps} + (1 - \mu) * noisy_{ps} \quad \textbf{(3)}$$

Figure 4.14 Equations of VAD method.

4.4.5 Weighted Spectral Averaging

Weighted spectral averaging is a simple approach for noise estimation via recursive averaging proposed in Hirsch, H. & Ehrlicher, C. (1995). This method is described as follows:

$$if \ \frac{|Y(\lambda,k)|}{\hat{\sigma}_d(\lambda-1,k)} < \beta \ then \quad (4.13)$$

$$\hat{\sigma}_d(\lambda,k) = \alpha\hat{\sigma}_d(\lambda-1,k) + (1-\alpha)|Y(\lambda,k)| \quad (4.14)$$

$$otherwise, \hat{\sigma}_d(\lambda,k) = \hat{\sigma}_d(\lambda-1,k) \quad (4.15)$$

where $\hat{\sigma}_d(\lambda,k)$ indicates the magnitude spectrum of the noise, α denotes the smoothing factor which is fixed, and β denotes the threshold.

In this approach, the smoothing factor, α, is fixed, but a different method is used to control the update of the noise spectrum. More specifically, the decision as to whether the noise spectrum should be updated or not is based on the comparison of the estimated *posterior* SNR to a threshold. If the *posterior* SNR is found to be

smaller than a specified threshold, suggesting absence of speech, then the noise spectrum is updated. Conversely, if the *posterior* SNR is found to be larger than the threshold, suggesting presence of speech, then the noise spectrum update is postponed.

In Figures 4.15 and 4.16, the flowchart and the equations of the weighted spectral averaging technique are given, respectively.

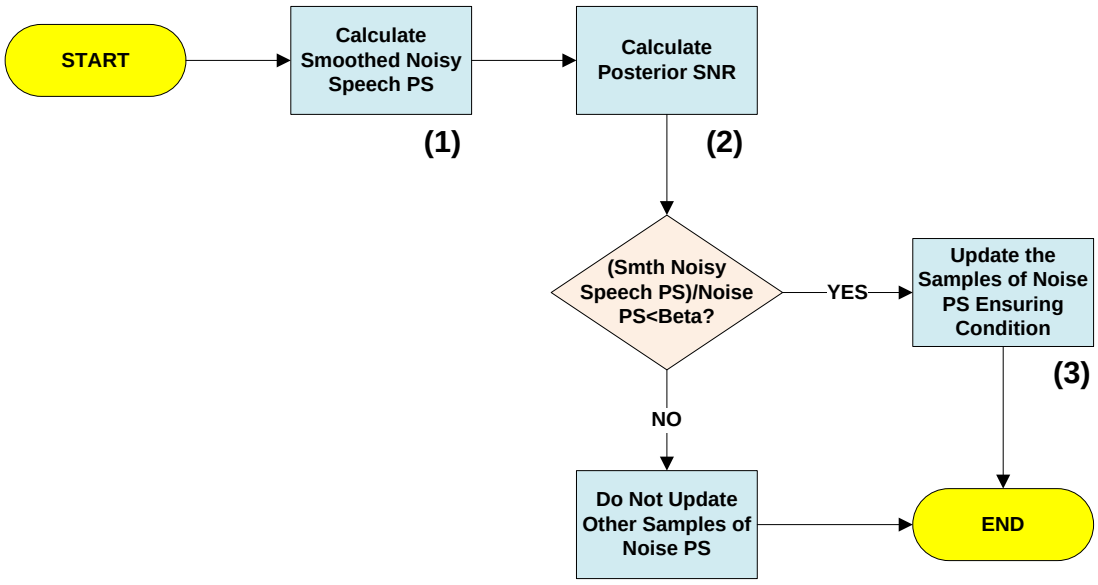


Figure 4.15 Flowchart of weighted spectral averaging algorithm for noise power spectral estimation.

$$Smoothed\ noisy\ PS = P_{smt} = \alpha * P_{smt\_prev} + (1-\alpha) * noisy_{ps} \quad (1)$$

$$\gamma_k = \frac{P_{smt}}{P_{dd}}, where\ P_{dd}\ is\ noise\ ps \quad (2)$$

$$P_{dd}[i] = \alpha * P_{dd}[i] + (1-\alpha) * P_{smt}[i] \quad (3)$$

Figure 4.16 Equations of weighted spectral averaging algorithm.

In this method, the threshold value β can have a significant effect on the noise spectrum estimation. If β is chosen too small, then the noise spectrum is not updated often enough and is underestimated. On the other hand, if β is chosen too large, then the noise spectrum is overestimated. In this approach, the β parameter is determined experimentally according to the input data at hand.

4.4.6 Histogram-Based Noise Estimation

The histogram-based noise estimation algorithms are motivated by the observation that the most frequent value of energy values in individual frequency bands correspond to the noise level of the specified frequency band; that is, the noise level corresponds to the maximum of the histogram of energy values.

In its most basic formulation, the noise estimate is obtained based on the histogram of past power spectrum values (Hirsch, H. & Ehrlicher, C., 1995); that is, for each incoming frame, we first construct a histogram of power spectrum values spanning a window of several hundreds of milliseconds, and take as an estimate of the noise spectrum the value corresponding to the maximum of the histogram values. This is done separately for each individual frequency bin. A first-order recursive smoothing may also be performed on the noise spectrum estimate to smooth out any outliers in the estimate.

Figures 4.17 and 4.18 show a simple example of finding the noise level estimate at a frequency component. Figure 4.17 shows the noisy signal power spectrum levels during an interval of 150 msec and Figure 4.18 is the histogram of this signal calculated with 40 bins. The histogram based noise estimation algorithm detects the maximum of the histogram and sets that level as the noise level at that frequency. For this example, the noise level at frequency 1000 Hz is set as -8.66 dB.

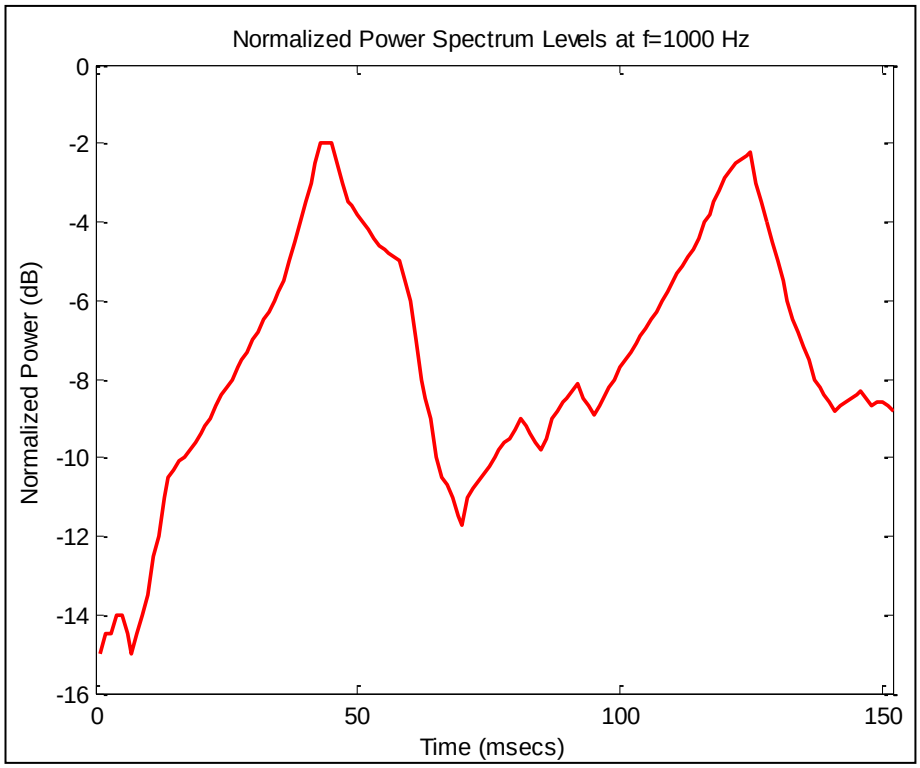


Figure 4.17 Normalized power spectrum levels for the noisy signal at f=1000 Hz.

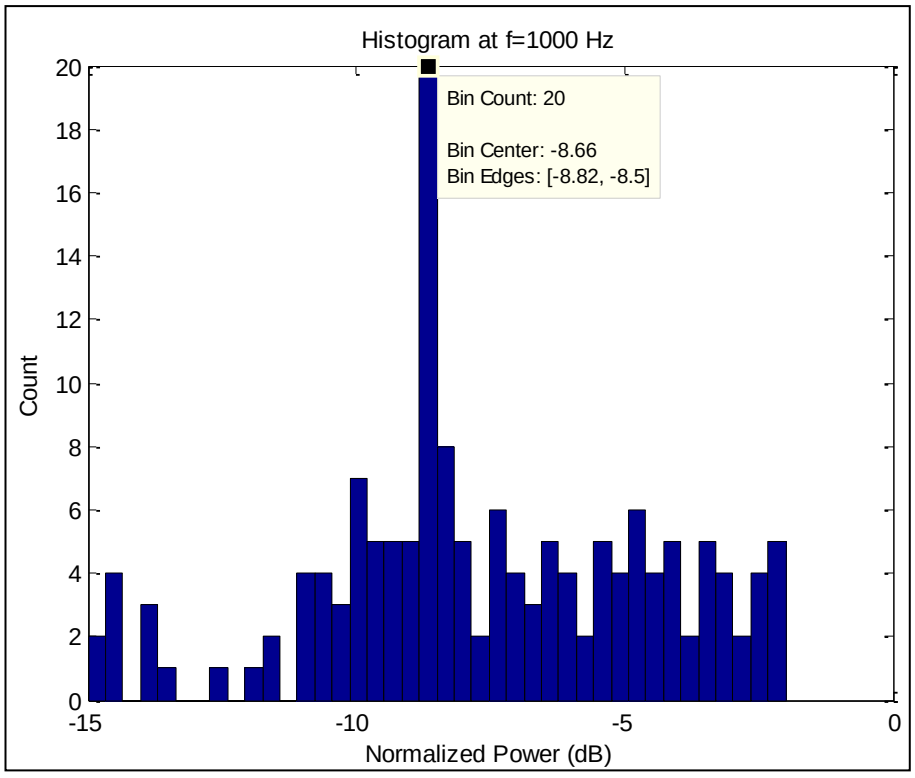


Figure 4.18 Histogram of normalized power spectrum levels for the noisy signal at f=1000 Hz

The histogram method described in Loizou, P. C. (2007), can be summarized as follows:

For each frame λ;

(1) Compute the noisy speech power spectrum $|Y(\lambda, \mathrm{k})|^2$.

(2) Smooth the noisy power spectral density (PSD) using first-order recursion

$$S(\lambda, \mathrm{k}) = \alpha * S(\lambda - 1, \mathrm{k}) + (1 - \alpha) * |Y(\lambda, \mathrm{k})|^2 \quad (4.16)$$

where α is a smoothing constant.

(3) Compute the histogram of D past PSD estimates $S(\lambda, \mathrm{k})$ $\{S(\lambda - 1, \mathrm{k}), S(\lambda - 2, \mathrm{k}), \ldots\ldots, S(\lambda - \mathrm{D}, \mathrm{k})\}$ using, say, 40 bins.

(4) Let $\boldsymbol{c} = [c_1, c_2, \ldots\ldots, c_{40}]$ be the counts in each of the 40 bins in the histogram and $\boldsymbol{s} = [s_1, s_2, \ldots\ldots, s_{40}]$ denote the corresponding centers of the histogram bins. Let c_{max} be the index of the maximum count, i.e;

$$c_{max} = \underset{1 \le i \le 40}{\operatorname{argmax}}\, c_i \quad (4.17)$$

Then, determine the estimate of the noise PSD (denoted by $H_{max}(\lambda, \mathrm{k})$) as the value corresponding to the maximum of the histogram, i.e., $H_{max}(\lambda, \mathrm{k}) = \mathrm{s}(c_{max})$.

(5) Smooth the noise estimate $H_{max}(\lambda, \mathrm{k})$ using first order recursion

$$\hat{\sigma}_d^2(\lambda, \mathrm{k}) = \alpha_\mathrm{m} * \hat{\sigma}_d^2(\lambda - 1, \mathrm{k}) + (1 - \alpha_\mathrm{m}) * H_{max}(\lambda, \mathrm{k}) \quad (4.18)$$

where $\hat{\sigma}_d^2(\lambda, \mathrm{k})$ is the smoothed estimate of the noise PSD, and α_m is the smoothing constant.

Figure 4.19 shows the flowchart of the histogram-based method.

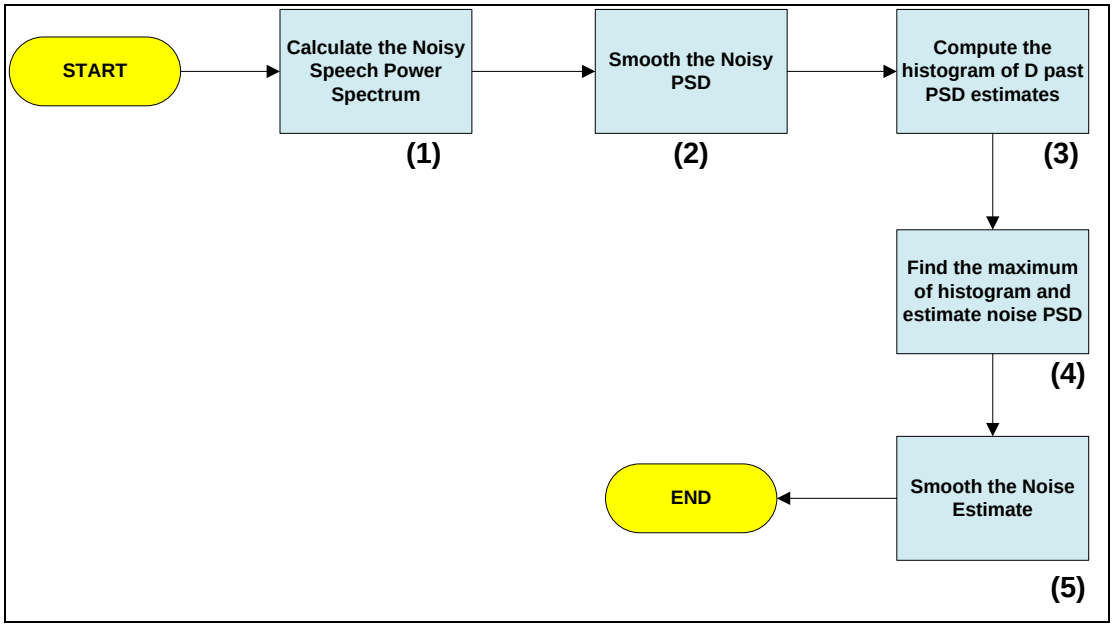


Figure 4.19 Flowchart of histogram-based noise estimation algorithm.

Because these steps are performed for each frequency bin (1024 frequency bins for our case), it takes too much time for a real-time processing task. Hence, we have performed an optimization in order to implement this algorithm in software. Especially, Steps 3 and 4 require too much processing power.

In order to use this algorithm as our noise estimation algorithm, we first calculate the power spectrum of each noisy speech frame and smooth them. We have generally used 80 frames for creating a noise estimate (80 frames takes approximately 2 seconds for 32 kHz sampling rate). These 80 smoothed noisy power spectrums are stored in an 80x1024 array. During the collection of the data, no other operation is performed. After all these smoothed noisy frames are collected, the histogram of each frequency component is calculated. Since we use 1024-point FFT, there must be 1024 histogram calculations. This operation needs too much processing power to perform during a single period (24 msec). In order to solve this problem, the histogram calculation operation is divided into 8 parts and completed during these 8 periods. Moreover, we have not calculated 1024 histograms. Instead, we have calculated 512 histograms for 512 frequency components, because the Fourier transform output is symmetrical. Therefore, we calculate the noise PSD by using 512 frequency components and duplicate these components in order to perform 1024-

point noise PSD. For the histogram calculation, we have used 40 bins. After this operation, the maximum of each histogram must be found. Again, this operation needs too much processing power (it is completed during 4 cycles). The maximum of each histogram is used to estimate the noise PSD. The last operation is the smoothing of the noise PSD with a smoothing constant.

4.4.7 SNR-Dependent Recursive Averaging Noise Estimation

The time-recursive averaging algorithms exploit the observation that the noise signal typically has a nonuniform effect on the spectrum of speech, in that some regions of the spectrum are affected by noise more than others. Put differently, each spectral component will typically have a different effective SNR. Consequently, we can estimate and update individual frequency bands of the noise spectrum whenever the effective SNR at a particular frequency band is extremely low. Equivalently, we can update the individual frequency bands of the noise spectrum whenever the probability of speech being present at a particular frequency band is extremely low. This observation led to the recursive-averaging type of algorithms in which the noise spectrum is estimated as a weighted average of past noise estimates and the present noisy speech spectrum. The weights change adaptively depending either on the effective SNR of each frequency bin or on the speech-presence probability.

All time-recursive algorithms have the following general form

$$\hat{\sigma}_d^2(\lambda, \mathrm{k}) = \alpha(\lambda, \mathrm{k}) * \hat{\sigma}_d^2(\lambda - 1, \mathrm{k}) + \big(1 - \alpha(\lambda, \mathrm{k})\big) * |Y(\lambda, \mathrm{k})|^2 \qquad (4.19)$$

where $|Y(\lambda, \mathrm{k})|^2$ is the noisy speech magnitude spectrum squared (periodogram), $\hat{\sigma}_d^2(\lambda, \mathrm{k})$ denotes the estimate of the noise PSD at frame λ and frequency k, and $\alpha(\lambda, \mathrm{k})$ is the smoothing factor, which is time and frequency dependent. Different algorithms were developed depending on the selection of the smoothing factor $\alpha(\lambda, \mathrm{k})$.

In the recursive averaging technique proposed in Lin, L., Holmes W.H. & Ambikairajah, E. (2003), the smoothing factor $\alpha(\lambda, \mathrm{k})$ in Equation (4.19) is chosen to be a sigmoid function of the *posterior* SNR $\gamma_k(\lambda)$

$$\alpha(\lambda, k) = \frac{1}{1+e^{-\beta(\gamma_k(\lambda)-1.5)}} \quad (4.20)$$

β is a parameter with values in the range $15 \leq \beta \leq 30$ and $\gamma_k(\lambda)$ is an approximation to the *posterior* SNR given by

$$\gamma_k(\lambda) = \frac{|Y(\lambda,k)|^2}{\frac{1}{10}\sum_{m=1}^{10} \hat{\sigma}_d^2(\lambda-1,k)} \quad (4.21)$$

The denominator in Equation (4.21) gives the average of the estimated noise PSD for the past 10 frames.

The recursive algorithm can be explained as follows: If speech is present, the *posterior* estimate $\gamma_k(\lambda)$ will be large and therefore $\alpha(\lambda, k) \approx 1$. Consequently, because $\alpha(\lambda, k) \approx 1$, we will have $\hat{\sigma}_d^2(\lambda, k) \approx \hat{\sigma}_d^2(\lambda - 1, k)$; that is, the noise update will cease and the noise estimate will remain the same as the previous frame's estimate. Conversely, if speech is absent, the *posterior* estimate $\gamma_k(\lambda)$ will be small and therefore $\alpha(\lambda, k) \approx 0$. As a result, $\hat{\sigma}_d^2(\lambda, k) \approx |Y(\lambda, k)|^2$; that is, the noise estimate will follow the PSD of the noisy spectrum in the absence of speech. The main advantage of using the time smoothing factors, as opposed to using a fixed value for $\alpha(\lambda, k)$, is that these factors are time and frequency dependent. This means that the noise PSD will be adapted differently and at different rates in the various frequency bins, depending on the estimate of *posterior* SNR, $\gamma_k(\lambda)$, in that bin. This is particularly suited in situations in which the noise is colored.

4.4.8 Filtering as Post Processing

The sounds of a normal speaking voice contain fundamental frequencies between 100 and 300 Hz. The overtones contained in these sounds extend the range of frequencies to approximately 5000 Hz. Voices of different individuals vary in their frequency content. Men usually have voices with lower fundamental and harmonic frequencies than those of women and children. The range of fundamental frequencies of the singing voice is greater than that of the speaking voice; it varies from about 80 Hz for a deep bass to about 1200 Hz for a high soprano. The overtones contained in the sounds of the singing voice reach as high as 10000 Hz. For purposes of

comparison, the frequency range of the instruments of a symphony orchestra includes fundamental of about 16 to 4000 Hz with overtones ranging to 12000 Hz or higher.

Thus, the speech information is not included inside the entire audio spectrum between 0 - 20 kHz. The telephony systems use 300 Hz - 3400 Hz for speech transmission, but decreasing this interval can worsen the speech quality. Generally, the lowest frequency band that carries speech information can be regarded as 200 Hz, while the highest frequency that carries speech frequency can be thought approximately between 5 - 10 kHz depending on the harmonics.

By considering this information, we have employed a fourth order IIR filter after the speech enhancement operation. This filter is a bandpass filter having cut-off frequencies of 200 Hz and 8000 Hz.

The filter is designed using MATLAB “fdatool” as shown in Figure 4.20 and applied to the DSP platform software.

After exporting the coefficients of the generated filter to MATLAB workspace, there is one more thing that must be done in order to use the filter in the DSP platform. MATLAB generates “SOS” and “G” default coefficients which represent the second-order section of a given digital filter. These coefficients should be converted to the equivalent transfer function representation (“A and B” coefficients) using “sos2tf” command of MATLAB. The MATLAB command line “[b,a] = sos2tf(sos,g)” returns the transfer function that describes a discrete-time system given by SOS in second-order section form with gain G.

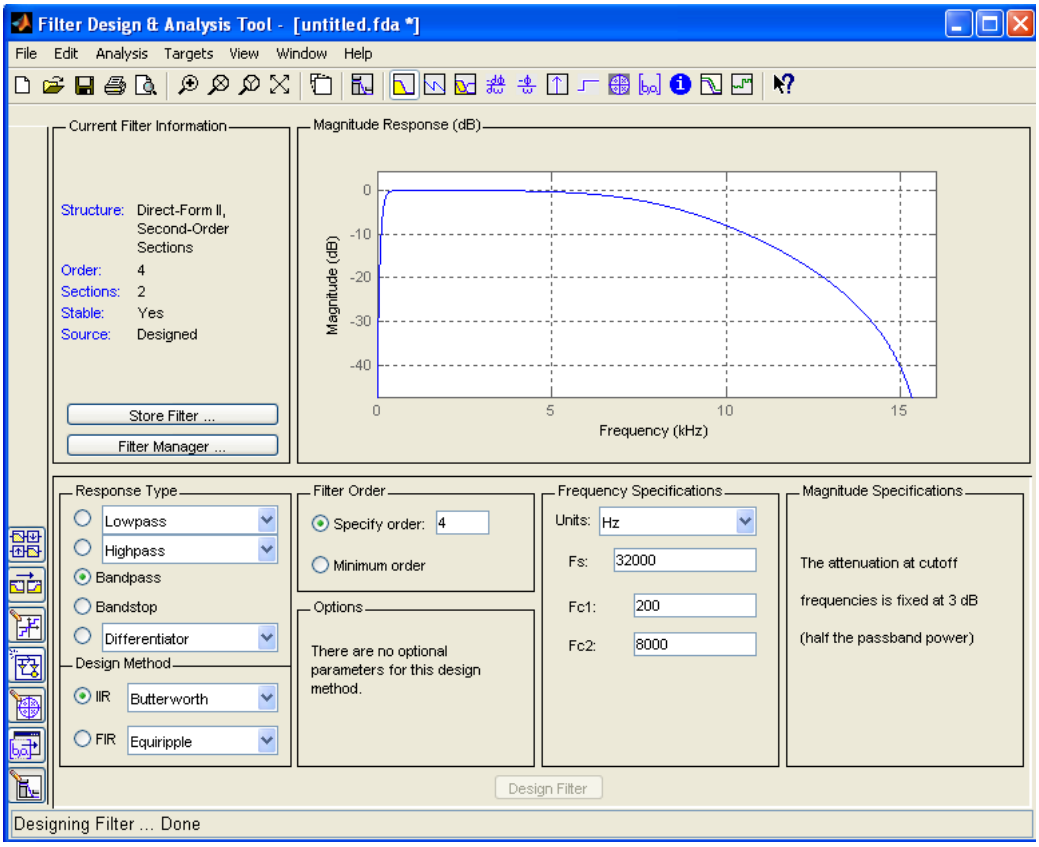


Figure 4.20 MATLAB "fdatool" used for designing digital filters.

These generated coefficients are used in DSP evaluation module software. We can also check the generated coefficients by plotting the magnitude and phase responses as in Figure 4.21.

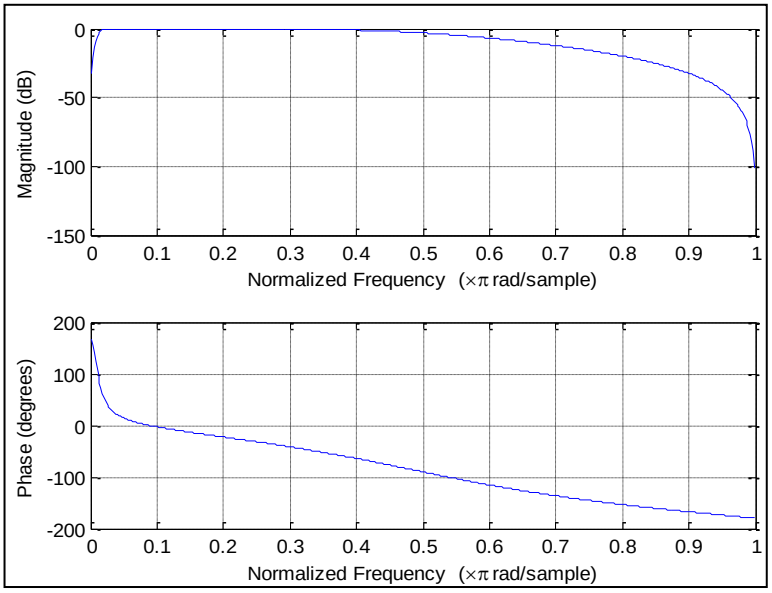


Figure 4.21 Magnitude and phase responses of the applied filter.

CHAPTER FIVE
A NOVEL FUSION NOISE ESTIMATION ALGORITHM

In this chapter, we explain the details of our newly developed fusion noise estimation algorithm. This algorithm is a combination of two methods related to noise estimation introduced in Lin, L., Holmes W. H. & Ambikairajah, E. (2003) and Ramirez, J. et. al. (2003). We have proposed two new algorithms called the "reset algorithm" and the "β parameter estimation algorithm" inspired by the existing methods proposed in Lin, L., Holmes W. H. & Ambikairajah, E. (2003) and Ramirez, J. et. al. (2003). The reset algorithm is used to reset the noise estimation algorithm when the input data change suddenly. The β parameter estimation algorithm deals with dynamically updating the β parameter that is used in the SNR dependent recursive averaging noise estimation algorithm introduced in Section 4.4.7. Both algorithms require computation of long-term spectral divergence (LTSD) value described in Ramirez J. et. al. (2003).

5.1 Long-term Spectral Divergence (LTSD)

In Ramirez, J. et. al. (2003), it is stated that the proposed speech/non-speech detection algorithm assumes that the most significant information for detecting voice activity on a noisy speech signal remains on the time-varying signal spectrum magnitude. It uses a long-term speech window instead of instantaneous values of the spectrum to track the spectral envelope and is based on the estimation of the so-called long-term spectral envelope (LTSE). The decision rule is then formulated in terms of the long-term spectral divergence (LTSD) between speech and noise.

Following Ramirez, J. et. al. (2003), we also utilize LTSD values in our proposed algorithm. In order to calculate LTSD, the following procedure is used. Let $x(n)$ be a noisy speech signal that is segmented into overlapped frames and, $X(k,l)$ be its amplitude spectrum for the k^{th} band and at frame l. The N-order long-term spectral envelope is defined as

$$LTSE_N(k,l) = max\{X(k,l+j)\}_{j=-N}^{j=+N} \quad (5.1)$$

Note that the LTSE is calculated for the interval between $-N$ and $+N$. However, since our study is at real-time we do not have access to the future samples. Hence, we have slightly modified this definition as follows:

$$LTSE_N(k,l) = max\{X(k,l+j)\}_{j=-2N}^{j=0} \tag{5.2}$$

The N-order long-term spectral divergence between speech and noise is defined as the deviation of the LTSE with respect to the average noise spectrum magnitude $N(k)$ for the k^{th} band with *k = 0; 1; . . . ;NFFT-1*. It is given by;

$$LTSD_N(l) = 10\log_{10}\left(\frac{1}{NFFT}\sum_{k=0}^{NFFT-1}\frac{LTSE^2(k,l)}{N^2(k)}\right) \tag{5.3}$$

In Ramirez, J. et. al. (2003), it is proved that the optimal window length, N, should be around 5 or 6. Hence, we also employ these values in the experiments of our study.

5.2 Noise Estimation Reset Algorithm

In this algorithm, our purpose is to reset the noise estimation algorithm in case the input data change immediately. At sudden changes, the noise estimation algorithm does not operate correctly because of the continuous interval of silence during the input switching. Thus, it is necessary to reset the noise estimation algorithm for the system to operate correctly.

A case of sudden increase is illustrated in Figure 5.1. Without the reset algorithm, it is seen that the noise estimation algorithm does not operate correctly after the sudden increase. Hence, the noise estimation algorithm must be restarted. After the application of the reset algorithm proposed in this section, the noise estimation algorithm continues to operate as desired. Figure 5.2 demonstrates the operation of the reset algorithm. It is seen that the noise estimation algorithm is reset as marked in Figure 5.2 and continues to operate correctly after the reset occurs.

Figure 5.1 Calculated LTSD values after a change in the input signal (without application of the reset algorithm).

Figure 5.2 Calculated LTSD values after a change in the input signal (with application of the reset algorithm).

The flowchart given in Figure 5.3 shows the steps of the noise estimation reset algorithm. The operation of the algorithm can be summarized as follows.

If the algorithm runs at first time, the previous LTSD vector is filled with ±0.25 of current LTSD value. That is; the odd samples of the vector are filled with current LTSD value plus 0.25, while the even samples of the vector are filled with current LTSD value minus 0.25. This is necessary for the operation of one of the reset cases.

There are two reset cases for this algorithm. The first case checks if there is a sudden increase from the previous LTSD values to the current LTSD value. If the current LTSD value is greater than any of the previous LTSD values times LIM1, the reset case occurs. The LIM1 value is selected as 1.4 (40% difference). The second reset case checks the rate of change of LTSD. If it is too stable, then it is experimentally determined that a reset is necessary. In our system, the stability limit is ±1%, CNT_LIMIT is assigned as 38 and the total size of previous LTSD vector is 40. When one of the reset cases occurs, the noise estimation parameters are reset.

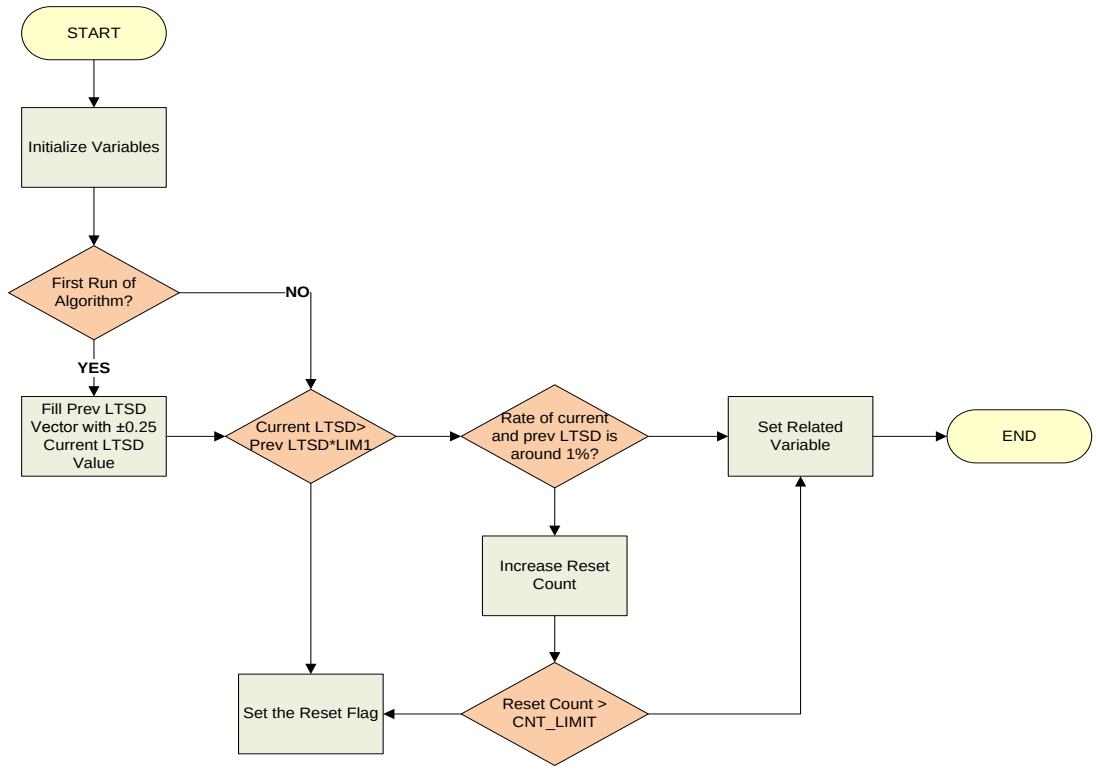


Figure 5.3 The flowchart of the noise estimation reset algorithm.

5.3 β Parameter Estimation Algorithm

It is stated in Lin, L., Holmes W. H. & Ambikairajah, E. (2003) that the β parameter is important for the rate of noise updates: larger values of β lead to slower noise updates, whereas smaller values of β give faster noise updates at a risk of over-estimation. It is also stated that this value is selected as a constant value before the operation.

Because our operation is at real-time and the input data are completely random, we do not have a chance to select the β parameter based on the input data. Additionally, a selected constant value will not be appropriate for all input signals. Thus, we need to estimate and update this parameter at run-time when necessary.

In order to update the β parameter dynamically, we need some feature data. These data must have distinctive properties for input signals including different noise types and levels. For this purpose, we have experimented with several techniques and selected to use long-term spectral divergence (LTSD) which is used for speech/non-speech analysis of the input data (Ramirez, J. et. al., 2003).

The purpose of the developed algorithm is the selection of different β parameters based on the calculated LTSD value. Following our experiments with different signal and noise types, we have decided to use four different β values as shown in Figure 5.4.

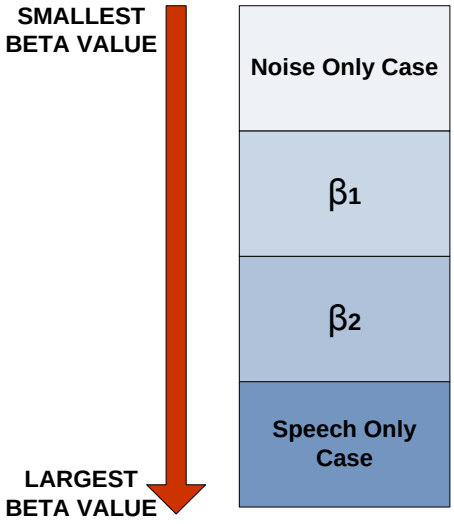


Figure 5.4 Use of different β values.

If the input data are determined as just noise by the algorithm, a small β value is assigned. This value is selected as 0.1. If the input data are determined as only speech, which means that there is no or negligible level of noise, then a large β value is selected. This value is determined as 1.5 in our algorithm. "β_1" and "β_2" are the β values between noise only and speech only cases. The rationale behind the selection of β_1 and β_2 is different from the other two β values. For determining the appropriate β value, we check whether the pre-determined number of LTSD samples is outside the limit of ±10% of the average LTSD value. If this condition is satisfied, then the β value is set to β_2. However, if this condition is not satisfied, then the β value is set as equal to β_1. This is because, if the amount of LTSD variation with respect to the average LTSD value is sufficient in order to perform the speech/non-speech discrimination, then a larger β should be used. However, if the amount of LTSD variation is not sufficient to perform this discrimination, then a smaller β value should be used. β_1 and β_2 values have been selected as 0.45 and 0.6, respectively, after experimenting with different input signals.

The calculated LTSD values obtained by using a constant β value (β = 0.6) for different input signals are given in figures below. The calculated LTSD values for noise-only input are shown in Figure 5.5. It is seen that the LTSD values are small and around 20. Thus, a threshold can be used for detection of noise-only input. This

threshold is set as 25 based on our experiments. That is, if the calculated LTSD value is under 25 for some interval, the input data are decided as noise-only and the β value for noise-only case is used.

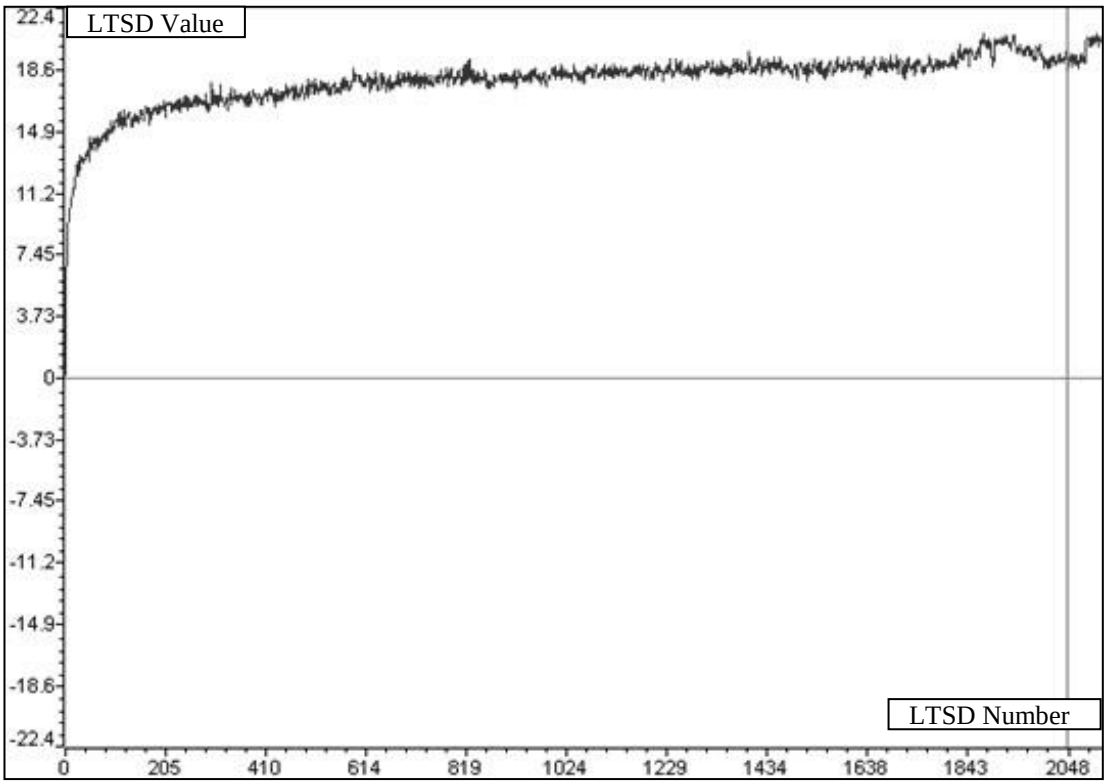


Figure 5.5 Calculated LTSD values for noise only input data.

In Figure 5.6, the calculated LTSD values for speech-only input case are plotted. It is seen that the values are quite high as compared to the noise-only case. Again, a threshold should be assigned. This threshold is determined to be 70. Hence, if the calculated LTSD values are greater than the determined threshold value, the input data are decided as the speech-only and the pre-determined β value for this case is set.

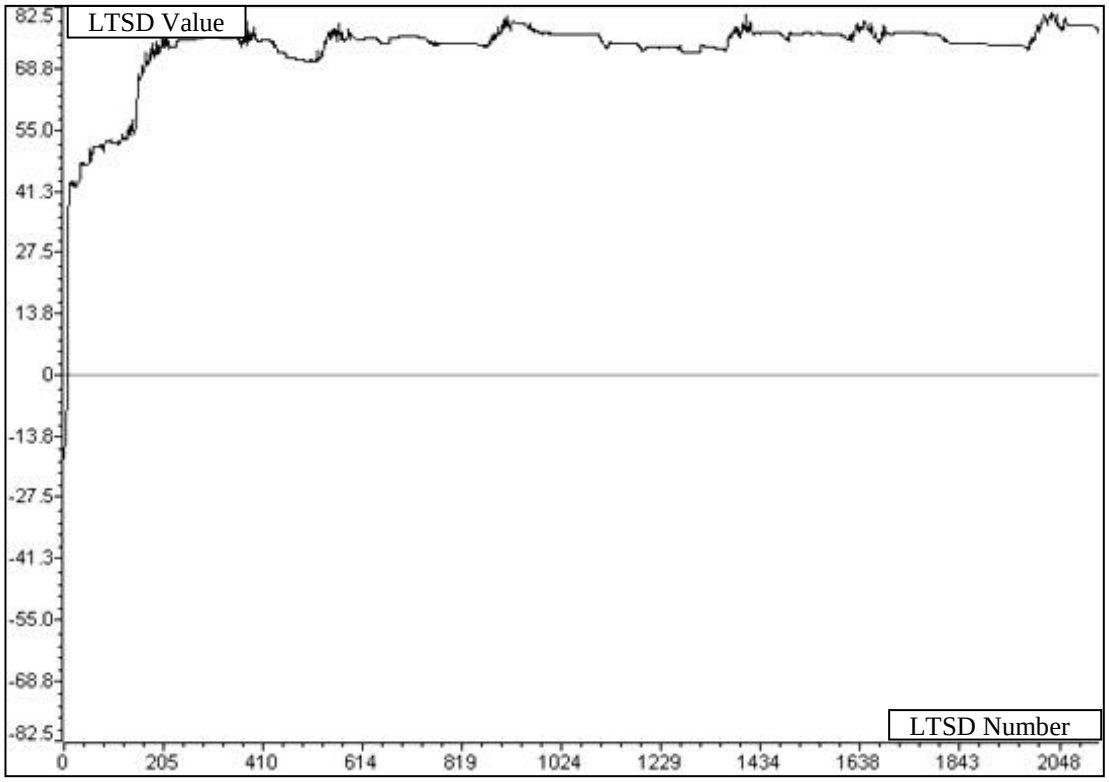


Figure 5.6 Calculated LTSD values for speech only input data.

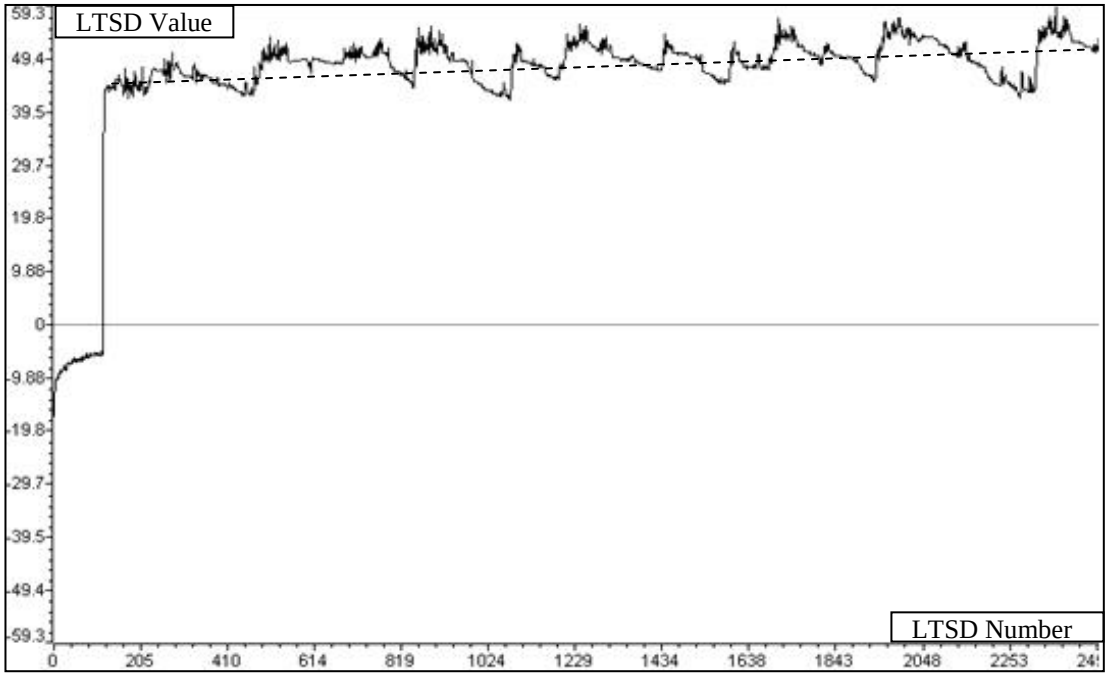


Figure 5.7 Calculated LTSD values for a signal having a clear speech/non-speech discrimination.

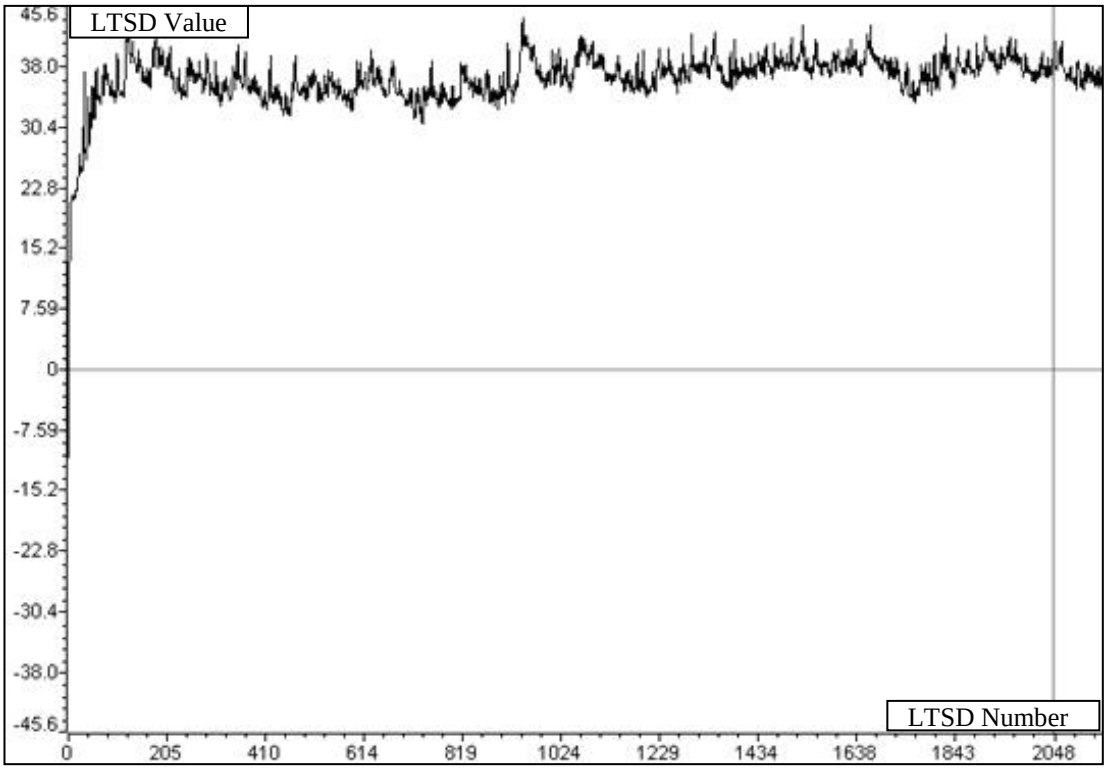


Figure 5.8 Calculated LTSD values for a signal not having a clear speech/non-speech discrimination.

The calculated LTSD values for two different signals are shown in Figures 5.7 and 5.8. We can determine a threshold value as shown in Figure 5.7 with a dashed line. By this way, we can easily perform the speech/non-speech discrimination for this input signal. If we want to perform VAD for the data in Figure 5.7, the values above the threshold would be decided as speech regions and the values under the threshold as non-speech regions. However, our purpose here is not to perform VAD. It is sufficient for us only to perform the speech/non-speech discrimination. Thus, the β value of 0.6 is used for such signals.

If the calculated LTSD values in Figure 5.8 are analyzed, it is seen that the speech/non-speech discrimination is not clear as in the previous case. If we try to set some threshold values, it is hard to determine speech and non-speech regions for these data. Hence, we set the β value as 0.45 if this type of an input signal is encountered.

5.3.1 The Flowchart of the Algorithm

In this section, we give the flowchart of the β parameter estimation algorithm and explain its steps.

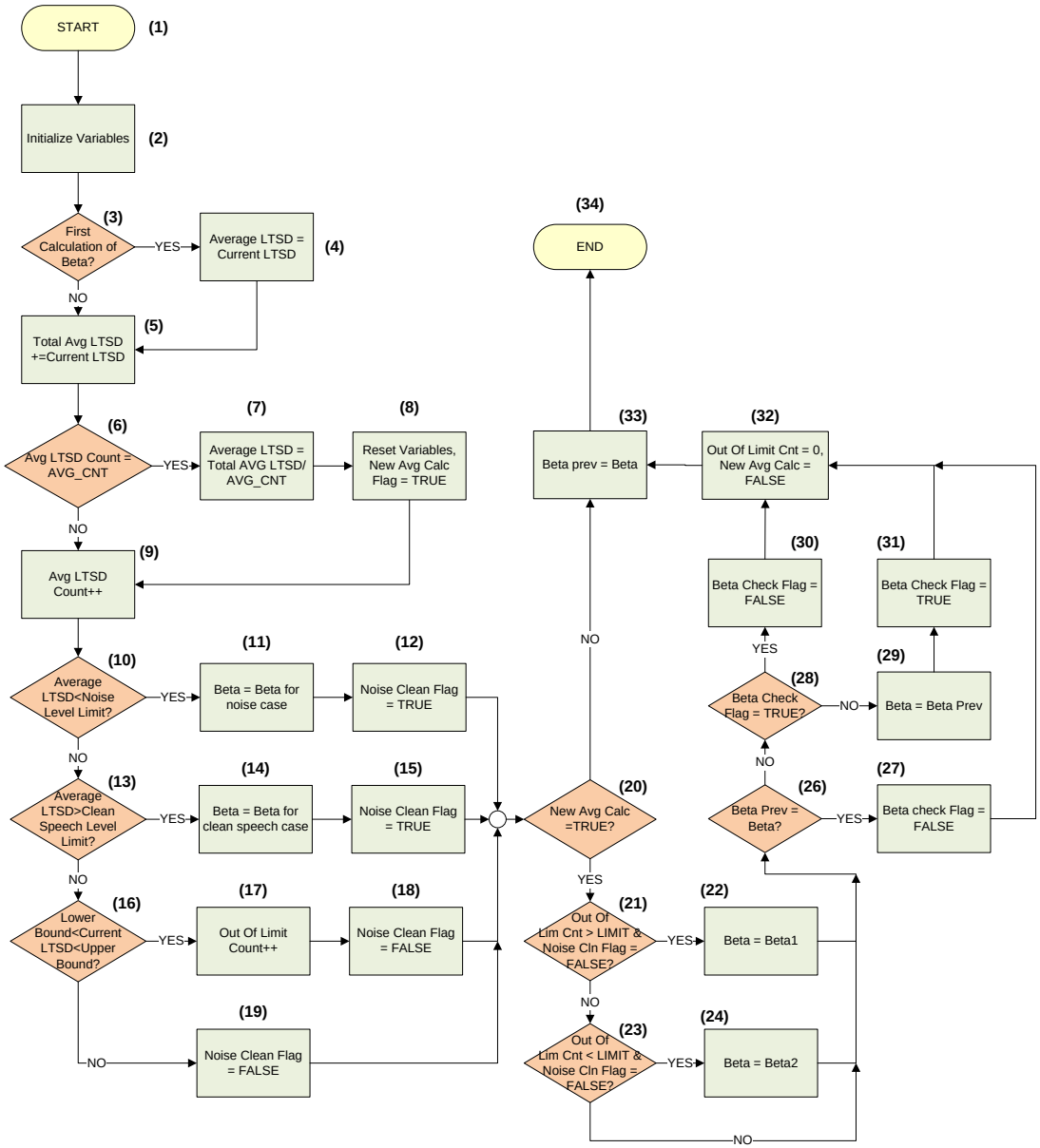


Figure 5.9 The flowchart of β parameter estimation algorithm.

The flowchart given in Figure 5.9 shows the steps of the β parameter estimation algorithm. Because the flowchart looks somewhat complicated, we explain the blocks one by one as follows.

(1), (33) Start and End blocks of the algorithm

(2) Initialization of the required variables for the algorithm.

(3), (4) If the algorithm runs at first time, then we do not have an "average LTSD" value. Thus, it is equalized to the "current LTSD" value at first run of the algorithm.

(5) For a pre-determined number (AVG_CNT) of LTSD samples, the average LTSD is computed. For this purpose, the current LTSD values are summed until the average LTSD value is calculated. The AVG_CNT is defined as 250. That is, for every 250 LTSD samples, the average LTSD value should be computed.

(6), (7), (8) If the number of calculated LTSD values reaches to AVG_CNT, then the algorithm computes the average LTSD value by dividing the summation of calculated LTSD values which are obtained in step **(5)** by the AVG_CNT. After the average LTSD value is obtained, then some variables are reset for new average calculation and a flag called "New Avg Calc" showing the average LTSD value is calculated in this cycle.

(9) If the number of summed LTSD values does not reach the desired AVG_CNT value, then the average LTSD value should be calculated in this cycle. In this case, the counter that keeps the number of calculated LTSD values for average calculation should be increased by one.

(10), (11), (12) If the average LTSD value is smaller than the pre-defined "Noise Level Limit" value, then it is decided by the algorithm that the current input does not include speech data, but only includes noise data. Therefore, the pre-defined β value for the noise case is set as the current β value. The "Noise Clean Flag" is used to keep the noise only/speech only cases. That is, if one of the β parameters is selected for noise only or speech only case, this flag is set to TRUE. This is

necessary because, if β is different than the noise only or speech only case, extra operations are needed to be performed.

(13), (14), (15) If the average LTSD value is larger than the pre-defined "Clean Speech Level Limit" value, then it is decided by the algorithm that the current input includes speech data completely. Thus, the pre-defined β value for the speech only case is set as the current β value. The "Noise Clean Flag" value is again set to TRUE.

(16), (17), (18), (19) If the β value is not set to speech only or noise only β values, then it is necessary to determine if the β is set to β_1 or β_2. For this operation, the number of LTSD samples that are outside the range should be counted. As already mentioned above, the range is ±10% of the average LTSD value. For each LTSD sample, this operation should be performed and the "Out of Limit Count" should be increased by one provided that the condition at issue is satisfied. The "Noise Clean Flag" should be set to FALSE even if the condition in question is satisfied or not.

(20), (32) In this step, "New Avg Calc" flag is checked, because we should change the value of β to β_1 or β_2 based on a counting operation. If "New Avg Calc" flag is FALSE, the β value is kept unchanged, (the previous β value should be valid).

(21), (22), (23), (24) If the "Out of Limit Count" is greater than the pre-defined "LIMIT" value and "Noise Clean Flag" is FALSE, then the β value should be set as β_2. If the "Out of Limit Count" is smaller than the pre-defined "LIMIT" value and "Noise Clean Flag" is again FALSE, then the β value should be set as β_1. The "LIMIT" value is set 40 based on our experiments. That is, if 40 of the LTSD samples out of the total 250 samples provide outside limit condition, then β_2 value should be used, otherwise β_1 is assigned. If the "Noise Clean Flag" is TRUE, we do not perform any operations in this step, because the value of β has already been determined for noise only or speech only cases.

(25), (26), (27), (28), (29), (30) There is one more operation to be performed before updating the β parameter. Because the input data are completely random, there can be some sudden increases or decreases in a certain period of time. In order to eliminate these sudden effects, a control case is added. This case checks if the following β parameters are obtained as the same value. If two sequential β parameters are the same, then we can update the β parameter. If not, the previous β value is accepted as valid, namely we do not update the β parameter in this case.

(31) After the β parameter is updated based on the calculated average LTSD value, some initial values should be set for next calculations. The "Out of Limit Count" is set 0, because this count should be created from the new data set. The "New Avg Calc" flag is set as FALSE in order to start the next calculation operation.

CHAPTER SIX
EVALUATION RESULTS

In this chapter, the evaluation results for quality and intelligibility measures are given.

6.1 Evaluation Results for Quality Measure

The results are obtained for different input signals by using different speech enhancement and noise estimation algorithms. For all the signals, all possible combinations are evaluated. Two tables are created based on the use of a bandpass filter. Table 6.1 shows the evaluation results when the bandpass filter is not in use and Table 6.2 shows the evaluation results when the bandpass filter is in use.

The first lines of the tables show the optimal values of each quality measure. For instance, according to the PESQ measure, the speech quality is very high if the calculated value is near 4.5 and the speech quality is very low if the calculated value is near 0. The abbreviations used in these tables are given below.

- Comp FW Mars - Frequency-variant fwSNRseg measure based on MARS analysis (Section 2.7.3.4).
- Comp FW Seg Variant - Frequency-variant fwSNRseg measure (Section 2.7.3.6).
- Composite - Composite objective measure (Section 2.7.3.3).
  Each of the three speech quality measures given above produces three ratings:
  - SIG - predicted rating of speech distortion (between 1 and 5).
  - BAK - predicted rating of noise distortion (between 1 and 5).
  - OVL - predicted rating of overall quality (between 1 and 5).
- LLR - Log likelihood ratio measure (Section 2.7.3.2.1).
- CEP - Cepstrum distance measure (Section 2.7.3.2.2).
- Comp FWSEG - frequency-weighted SNRseg measure using a different weighting function, the clean spectrum (Section 2.7.3.5).
- PESQ - PESQ measure based on the ITU standard P.862 (Section 2.7.3.1)

Different noise types are used in our experiments. The PSD estimates of these noise types are given in Figures 6.1 through 6.5.

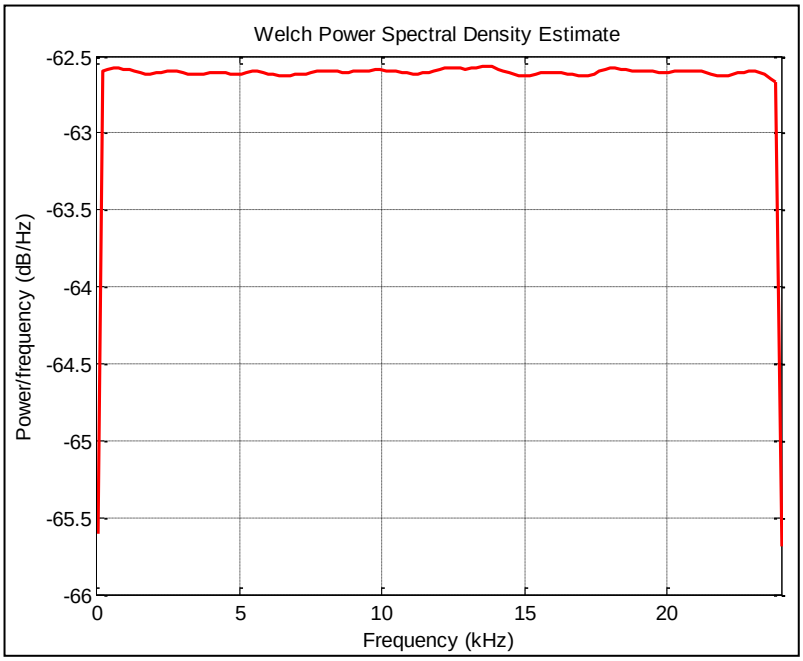


Figure 6.1 PSD estimate of white noise.

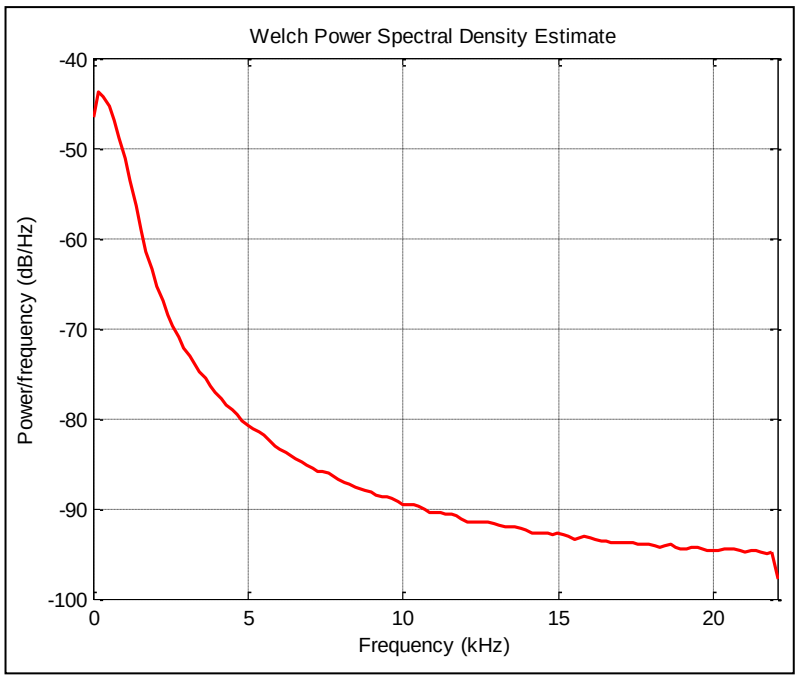


Figure 6.2 PSD estimate of speech noise.

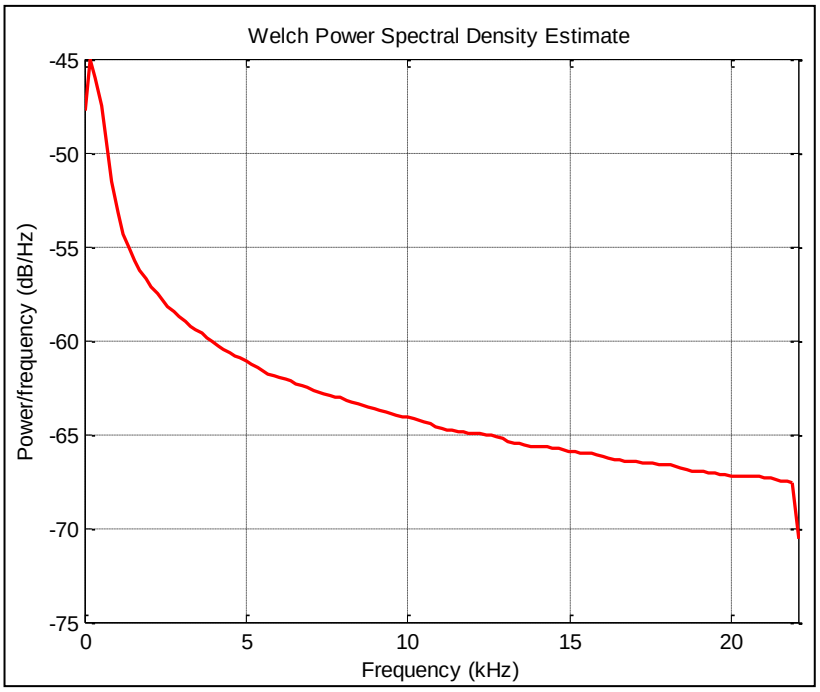


Figure 6.3 PSD estimate of PN pink noise.

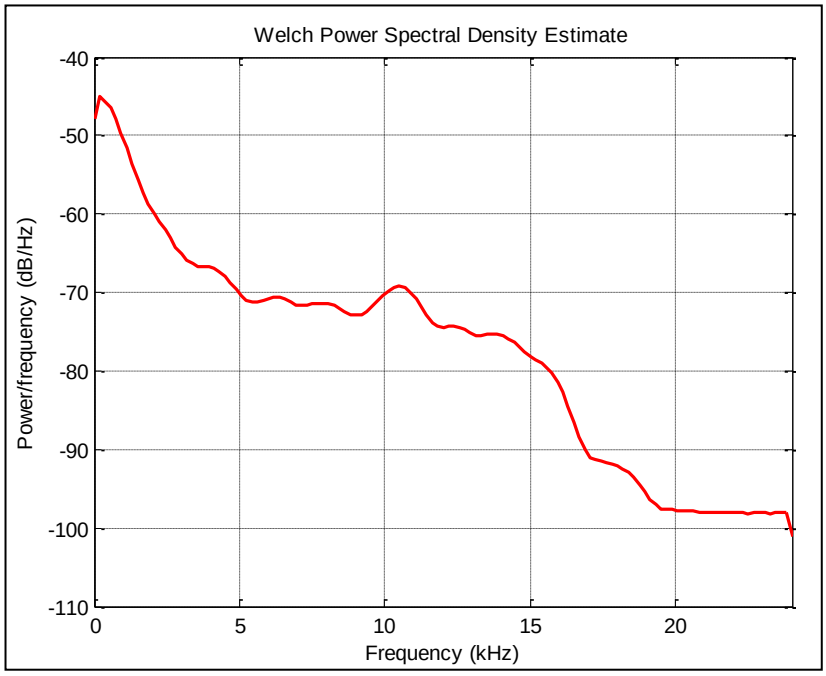


Figure 6.4 PSD estimate of restaurant ambiance noise.

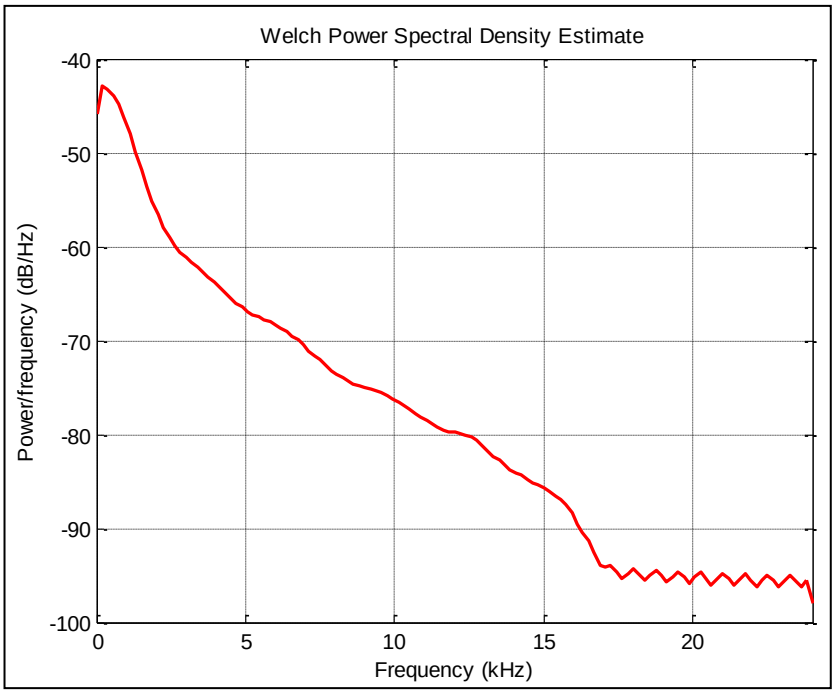


Figure 6.5 PSD estimate of airport noise.

If we analyze the results in Tables 6.1 and 6.2, we see that the results when the bandpass filter is active are not better than the ones when the bandpass filter is not in use. This can be due to the comparison of clean speech signal and the filtered output, because the original speech data include the whole frequency range, but the processed speech data do not include the whole frequency range due to filtering operation. Therefore, the mathematical comparison may not match well with the application in this case. Either using subjective evaluation techniques or filtering the original speech data as pre-processing might solve this problem.

If we compare the speech enhancement and noise estimation algorithms based on the evaluation results given in Tables 6.1 and 6.2, we see that Wiener filtering approach together with the recursive-averaging noise estimation algorithm has produced favorable results for all input signals. Hence, this combination is selected for use in our further studies.

Table 6.1 The objective evaluation results for different input signals when the IIR filter is not in use.

				Comp FW Mars			Comp FW Seg Variant			Composite						
Content	**Noise Type**	**Enh. Algorithm**	**Noise Est. Algorithm**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**LLR**	**CEP**	**Comp FWSEG**	**PESQ**
Opt. Values	**No**	**No**	**No**	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	0,00	0,00	35,00	4,50
G. Hitabe	**White**	**No**	**No**	**3,30**	**2,80**	**3,57**	**2,62**	**3,08**	**3,83**	**2,32**	**2,65**	**2,72**	**1,36**	**7,13**	**9,93**	**2,58**
G. Hitabe	**White**	**LOG-MMSE**	**Hist. Based**	3,11	2,91	3,86	2,35	2,78	3,52	2,45	2,92	3,27	1,03	5,88	9,06	2,60
G. Hitabe	**White**	**W-SNR**	**Hist. Based**	**3,57**	**3,20**	**3,88**	**2,48**	**3,07**	**3,92**	**2,40**	**2,60**	**2,56**	**1,24**	**7,09**	**10,07**	**2,70**
G. Hitabe	**White**	**LOG-MMSE**	**Rec. Avg.**	3,00	2,92	3,87	2,38	2,87	3,64	2,48	2,98	3,21	1,11	6,14	9,45	2,78
G. Hitabe	**White**	**W-SNR**	**Rec. Avg.**	**3,18**	**3,05**	**3,82**	**2,38**	**2,93**	**3,75**	**2,43**	**2,88**	**2,98**	**1,23**	**6,62**	**9,71**	**2,83**
News	**Rest Amb**	**No**	**No**	1,25	2,09	3,37	1,66	2,35	3,24	1,99	2,44	3,04	0,82	5,53	7,24	1,95
News	**Rest Amb**	**LOG-MMSE**	**Hist. Based**	**1,58**	**2,03**	**3,20**	**1,66**	**2,07**	**2,80**	**1,95**	**2,07**	**2,62**	**0,91**	**5,81**	**6,28**	**1,69**
News	**Rest Amb**	**W-SNR**	**Hist. Based**	1,49	2,32	3,61	1,80	2,40	3,24	1,96	2,28	2,84	0,82	5,60	7,19	1,91
News	**Rest Amb**	**LOG-MMSE**	**Rec. Avg.**	**1,62**	**2,36**	**3,54**	**1,84**	**2,38**	**3,19**	**1,99**	**2,25**	**2,84**	**0,80**	**5,40**	**7,11**	**1,84**
News	**Rest Amb**	**W-SNR**	**Rec. Avg.**	1,70	2,40	3,66	1,85	2,60	3,53	2,06	2,47	3,08	0,74	5,24	8,03	1,99
Old Ages	**Speech**	**No**	**No**	**1,14**	**1,64**	**1,70**	**1,39**	**1,43**	**1,86**	**1,40**	**1,67**	**1,97**	**1,21**	**6,91**	**1,45**	**1,71**
Old Ages	**Speech**	**LOG-MMSE**	**Hist. Based**	1,52	1,70	2,16	1,55	1,64	2,11	1,81	2,13	2,49	1,01	6,71	3,14	2,11
Old Ages	**Speech**	**W-SNR**	**Hist. Based**	**1,62**	**1,48**	**1,67**	**1,57**	**1,57**	**1,99**	**1,68**	**1,86**	**2,00**	**1,36**	**8,07**	**2,43**	**2,08**
Old Ages	**Speech**	**LOG-MMSE**	**Rec. Avg.**	1,39	1,68	2,05	1,55	1,60	2,08	1,71	2,02	2,34	1,04	6,58	2,96	2,07
Old Ages	**Speech**	**W-SNR**	**Rec. Avg.**	**1,28**	**1,71**	**1,85**	**1,51**	**1,54**	**1,98**	**1,62**	**1,94**	**2,24**	**1,12**	**6,77**	**2,26**	**2,00**
G. Hitabe	**PN Pink**	**No**	**No**	3,32	3,22	3,70	2,70	3,54	4,52	2,41	3,19	3,71	0,75	4,79	11,68	2,69
G. Hitabe	**PN Pink**	**LOG-MMSE**	**Hist. Based**	**2,42**	**2,67**	**3,53**	**2,04**	**2,49**	**3,24**	**2,44**	**3,06**	**3,57**	**0,70**	**4,69**	**7,92**	**2,64**
G. Hitabe	**PN Pink**	**W-SNR**	**Hist. Based**	3,24	3,19	3,63	2,47	3,23	4,17	2,52	3,22	3,61	0,85	5,03	10,68	2,88
G. Hitabe	**PN Pink**	**LOG-MMSE**	**Rec. Avg.**	**2,85**	**3,02**	**3,80**	**2,27**	**2,90**	**3,79**	**2,49**	**3,23**	**3,75**	**0,66**	**4,52**	**9,46**	**2,78**
G. Hitabe	**PN Pink**	**W-SNR**	**Rec. Avg.**	3,27	3,18	3,74	2,59	3,47	4,51	2,52	3,39	3,89	0,67	4,38	11,69	2,93

Table 6.2 The objective evaluation results for different input signals when the IIR filter is active.

				Comp FW Mars			Comp FW Seg Variant			Composite						
Content	**Noise Type**	**Enh. Algorithm**	**Noise Est. Algorithm**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**LLR**	**CEP**	**Comp FWSEG**	**PESQ**
Opt. Values	**No**	**No**	**No**	**5,00**	**5,00**	**5,00**	**5,00**	**5,00**	**5,00**	**5,00**	**5,00**	**5,00**	**0,00**	**0,00**	**35,00**	**4,50**
G. Hitabe	**White**	**LOG-MMSE**	**Hist. Based**	2,88	2,68	3,66	2,50	2,78	3,39	2,45	2,81	3,09	1,11	5,83	7,95	2,60
G. Hitabe	**White**	**W-SNR**	**Hist. Based**	**3,24**	**3,13**	**3,70**	**2,66**	**2,92**	**3,52**	**2,44**	**2,49**	**2,34**	**1,28**	**7,02**	**8,59**	**2,73**
G. Hitabe	**White**	**LOG-MMSE**	**Rec. Avg.**	2,81	2,68	3,70	2,52	2,89	3,56	2,53	2,87	3,05	1,12	6,00	8,31	2,77
G. Hitabe	**White**	**W-SNR**	**Rec. Avg.**	**3,13**	**2,98**	**3,74**	**2,71**	**3,03**	**3,65**	**2,47**	**2,80**	**2,81**	**1,25**	**6,51**	**8,99**	**2,85**
News	**Rest Amb**	**LOG-MMSE**	**Hist. Based**	1,65	1,83	2,60	1,89	1,86	2,43	1,81	1,74	2,02	1,16	6,61	4,73	1,75
News	**Rest Amb**	**W-SNR**	**Hist. Based**	**1,59**	**1,97**	**2,93**	**1,77**	**2,08**	**2,74**	**1,83**	**1,99**	**2,39**	**1,03**	**6,28**	**5,16**	**1,87**
News	**Rest Amb**	**LOG-MMSE**	**Rec. Avg.**	1,97	2,20	3,07	1,93	2,22	2,84	1,94	2,11	2,58	0,92	5,68	5,78	1,87
News	**Rest Amb**	**W-SNR**	**Rec. Avg.**	**1,92**	**2,25**	**3,26**	**2,04**	**2,56**	**3,29**	**2,06**	**2,41**	**2,97**	**0,79**	**5,28**	**6,91**	**2,01**
Old Ages	**Speech**	**LOG-MMSE**	**Hist. Based**	1,18	1,73	2,50	1,32	1,57	2,18	1,74	1,96	2,31	1,08	6,62	2,99	1,94
Old Ages	**Speech**	**W-SNR**	**Hist. Based**	**1,24**	**1,35**	**1,92**	**1,19**	**1,34**	**1,90**	**1,68**	**1,83**	**1,95**	**1,37**	**7,98**	**2,19**	**2,08**
Old Ages	**Speech**	**LOG-MMSE**	**Rec. Avg.**	1,17	1,72	2,39	1,35	1,54	2,12	1,75	2,05	2,38	1,04	6,48	3,00	2,08
Old Ages	**Speech**	**W-SNR**	**Rec. Avg.**	**1,00**	**1,55**	**2,17**	**1,16**	**1,36**	**1,95**	**1,64**	**1,91**	**2,18**	**1,17**	**6,91**	**2,03**	**2,00**
G. Hitabe	**PN Pink**	**LOG-MMSE**	**Hist. Based**	2,58	2,54	3,32	2,20	2,55	3,21	2,45	2,99	3,41	0,83	4,75	7,21	2,68
G. Hitabe	**PN Pink**	**W-SNR**	**Hist. Based**	**3,29**	**3,02**	**3,50**	**2,54**	**2,92**	**3,59**	**2,55**	**2,99**	**3,19**	**1,08**	**5,44**	**8,39**	**2,90**
G. Hitabe	**PN Pink**	**LOG-MMSE**	**Rec. Avg.**	2,70	2,80	3,54	2,31	2,72	3,46	2,51	3,11	3,52	0,85	4,79	7,93	2,79
G. Hitabe	**PN Pink**	**W-SNR**	**Rec. Avg.**	**3,29**	**3,25**	**3,66**	**2,68**	**3,29**	**4,09**	**2,54**	**3,27**	**3,65**	**0,86**	**4,57**	**9,90**	**2,95**

Table 6.3 The objective evaluation results for the new fusion noise estimation algorithm.

			Comp FW Mars			Comp FW Seg Variant			Composite						
Content	**Noise Type**	**β**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**BAK**	**OVL**	**SIG**	**LLR**	**CEP**	**Comp FWSEG**	**PESQ**
Opt. Values	**No**	**-**	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	5,00	0,00	0,00	35,00	4,50
News	**Airport**	**0,60**	**1,83**	**2,24**	**3,20**	**1,94**	**2,54**	**3,32**	**2,12**	**2,79**	**3,39**	**0,57**	**4,46**	**7,31**	**2,38**
News	**Airport**	**0,45**	1,92	2,33	3,25	1,95	2,43	3,16	2,11	2,72	3,26	0,65	4,82	6,90	2,40
News	**Airport**	**Auto**	**1,92**	**2,31**	**3,21**	**1,95**	**2,44**	**3,18**	**2,07**	**2,72**	**3,26**	**0,65**	**4,79**	**6,85**	**2,40**
Syn. Abst.	**PN Pink**	**0,60**	2,14	2,13	2,84	1,94	2,11	2,70	1,99	2,00	2,23	1,55	7,50	6,13	1,87
Syn. Abst.	**PN Pink**	**0,45**	**2,20**	**2,22**	**2,93**	**1,98**	**2,17**	**2,77**	**NA**	**NA**	**NA**	**1,36**	**7,44**	**6,44**	**NA**
Syn. Abst.	**PN Pink**	**Auto**	2,18	2,22	2,92	1,98	2,18	2,78	1,99	2,01	2,24	1,53	7,43	6,33	1,90
G. Hitabe	**White**	**0,60**	**3,79**	**4,48**	**3,02**	**2,78**	**3,99**	**5,00**	**2,87**	**3,77**	**4,08**	**0,87**	**4,67**	**14,30**	**3,44**
G. Hitabe	**White**	**0,45**	3,70	4,33	3,36	2,73	3,93	5,00	2,83	3,63	3,87	0,97	5,10	9,84	3,44
G. Hitabe	**White**	**Auto**	**3,80**	**4,52**	**3,08**	**2,75**	**3,94**	**5,00**	**2,90**	**3,80**	**4,12**	**0,86**	**4,63**	**14,88**	**3,46**

The objective evaluation results for the new fusion noise estimation algorithm are given in Table 6.3. Looking at these results, we can infer that better results are obtained when β value is assigned as 0.45 for "News with airport noise" and "Syn. Abst. with PN pink noise" input signals while better results are obtained for "G. Hitabe with white noise" input signal when β equals to 0.60.

In Table 6.3, the β values labeled with "Auto" means that the β value is determined by the novel fusion noise estimation algorithm. As can be seen from the results, the new algorithm has produced favorable results for all three cases. If β had been selected as a constant value such as 0.45, it would have been appropriate for the first and second input signals. However, this would have not been the best selection for the last input signal. Thus, using our fusion noise estimation algorithm is a good way to determine the β parameter automatically.

Besides, the fusion noise estimation algorithm also works for "noise-only" and "speech-only" cases that a constant β value can not handle. We do not give any objective results for these cases, but it is obvious that it is not necessary to apply the enhancement algorithm when the input data do not include noise. In the same way, if the input data includes only noise, we should suppress the input as much as possible, because it does not carry any information.

6.2 Evaluation Results for Intelligibility Measure

In this section, the evaluation results for intelligibility measure are given by using the procedure explained in Section 2.7.3.7. The clean speech file used in this test has not been used in the training phase of the speech-to-text algorithm for a more correct evaluation. Three types of noise are added to this file; white noise, speech noise, and PN pink noise. We obtain the results for each noise type and as corresponding to the application of different noise estimation and enhancement algorithms and filtering operation. The evaluation results obtained by using the speech-to-text software is given in Table 6.4.

Table 6.4 The number of correctly identified words by the speech-to-text software for different algorithm applications and noise types.

Noise Type	Enhancement Algorithm	Noise Estimation Algorithm	IIR Filter	Number of Correctly Identified Words	% Word Error Rate
No	**No**	**No**	**No**	749	6 (Org. File. Cmp.)
White Noise	**No**	**No**	**No**	**7**	**99,07**
White Noise	**Log-MMSE**	**Hist. Based**	**OFF**	15	98,0
White Noise	**Log-MMSE**	**Hist. Based**	**ON**	**26**	**96,53**
White Noise	**Log-MMSE**	**Rec. Avg.**	**OFF**	77	89,7
White Noise	**Log-MMSE**	**Rec. Avg.**	**ON**	**124**	**83,4**
White Noise	**W-SNR**	**Hist. Based**	**OFF**	172	77,0
White Noise	**W-SNR**	**Hist. Based**	**ON**	**173**	**76,9**
White Noise	**W-SNR**	**Rec. Avg.**	**OFF**	176	76,5
White Noise	**W-SNR**	**Rec. Avg.**	**ON**	**328**	**56,2**
Speech Noise	**No**	**No**	**No**	114	84,78
Speech Noise	**Log-MMSE**	**Hist. Based**	**OFF**	**6**	**99,2**
Speech Noise	**Log-MMSE**	**Hist. Based**	**ON**	13	98,27
Speech Noise	**Log-MMSE**	**Rec. Avg.**	**OFF**	**14**	**98,13**
Speech Noise	**Log-MMSE**	**Rec. Avg.**	**ON**	22	97,06
Speech Noise	**W-SNR**	**Hist. Based**	**OFF**	**35**	**95,33**
Speech Noise	**W-SNR**	**Hist. Based**	**ON**	32	95,73
Speech Noise	**W-SNR**	**Rec. Avg.**	**OFF**	**129**	**82,78**
Speech Noise	**W-SNR**	**Rec. Avg.**	**ON**	138	81,57
PN Pink Noise	**No**	**No**	**No**	**155**	**79,3**
PN Pink Noise	**Log-MMSE**	**Hist. Based**	**OFF**	58	92,26
PN Pink Noise	**Log-MMSE**	**Hist. Based**	**ON**	**107**	**85,71**
PN Pink Noise	**Log-MMSE**	**Rec. Avg.**	**OFF**	199	73,43
PN Pink Noise	**Log-MMSE**	**Rec. Avg.**	**ON**	**192**	**74,37**
PN Pink Noise	**W-SNR**	**Hist. Based**	**OFF**	297	60,35
PN Pink Noise	**W-SNR**	**Hist. Based**	**ON**	**358**	**52,2**
PN Pink Noise	**W-SNR**	**Rec. Avg.**	**OFF**	338	54,87
PN Pink Noise	**W-SNR**	**Rec. Avg.**	**ON**	**377**	**49,67**

In this table, “No” means that no algorithm is applied to the noisy speech file; that is, the noisy speech file is directly used as an input to the speech-to-text software.

The original speech file used for intelligibility evalulation includes 797 words. When the clean speech file is applied to the speech-to-text software, 749 words are correctly identified; that is, the word error rate of the speech-to-text software is 6%. The number of correctly identified words for noisy signals are calculated with

respect to the output of speech-to-text software for clean speech file, not the original text. However, since the word error rate of the speech-to-text software is 6%, this issue does not have an important effect on the scores. In Table 6.4, the word error rates of the noisy speech signals are given with respect to the output of the speech-to-text software for clean speech file as mentioned. The first word error rate value given in this table (6%) is calculated with respect to the original text.

As can be seen from Table 6.4, application of the algorithms generally increases the number of correctly identified words. For the white noise case, if no algorithms are applied, the number of correctly identified words is only 7. As shown in Figure 6.6, application of algorithms always increases intelligibility for the white noise case. However, the maximum intelligibility is obtained when Wiener-SNR algorithm is applied together with recursive-averaging noise estimation algorithm and IIR filter. When the speech noise is applied to the clean speech data, this time some algorithms reduce the intelligibility as shown in Figure 6.7. In this case, again only the combinations of Wiener-SNR and recursive-averaging noise estimation algorithms increase the intelligibility while the others decrease it. For the PN pink noise, only the combinations of LOG-MMSE and histogram-based noise estimation algorithms decrease the intelligibility while the others increase it. Again, the best performance is obtained with the application of Wiener-SNR and recursive averaging noise estimation algorithms as shown in Figure 6.8.

If the number of correctly identified words is compared with the total number of words in the clean speech file, it follows that the word error rate is 49,67% which means that only half of the total words are correctly identified at the best case. The word error rate is 99,07% for the white noise case, 84,78% for the speech noise case and 79,3% for the PN pink noise case without the application of any enhancement algorithms. If the values in Table 6.4 are analyzed, it can be seen that Wiener-SNR algorithm together with recursive-averaging noise estimation algorithm and IIR filter has decreased the word error rate from 99,07% to 56,2% for the white noise case, from 84,78% to 81,57% for the speech noise case and from 79,3% to 49,67% for the

PN pink noise case. Thus, we can state that application of the enhancement algorithm generally decreases the word error rate a great deal.

In conclusion, Wiener-SNR algorithm together with recursive-averaging noise estimation algorithm has produced the best results among others for all noise types that are experimented with. It has also been seen that application of IIR filtering increases the number of correctly identified words considerably.

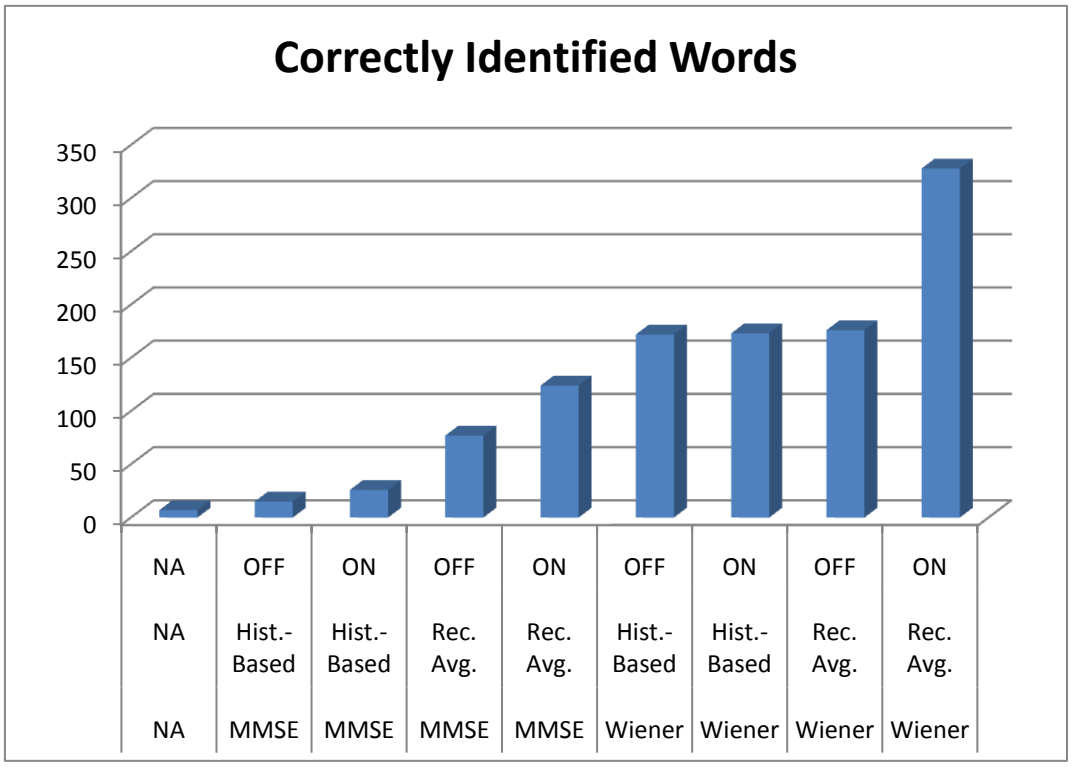


Figure 6.6 Graphical comparisons of speech-to-text algorithm outputs for white noise.

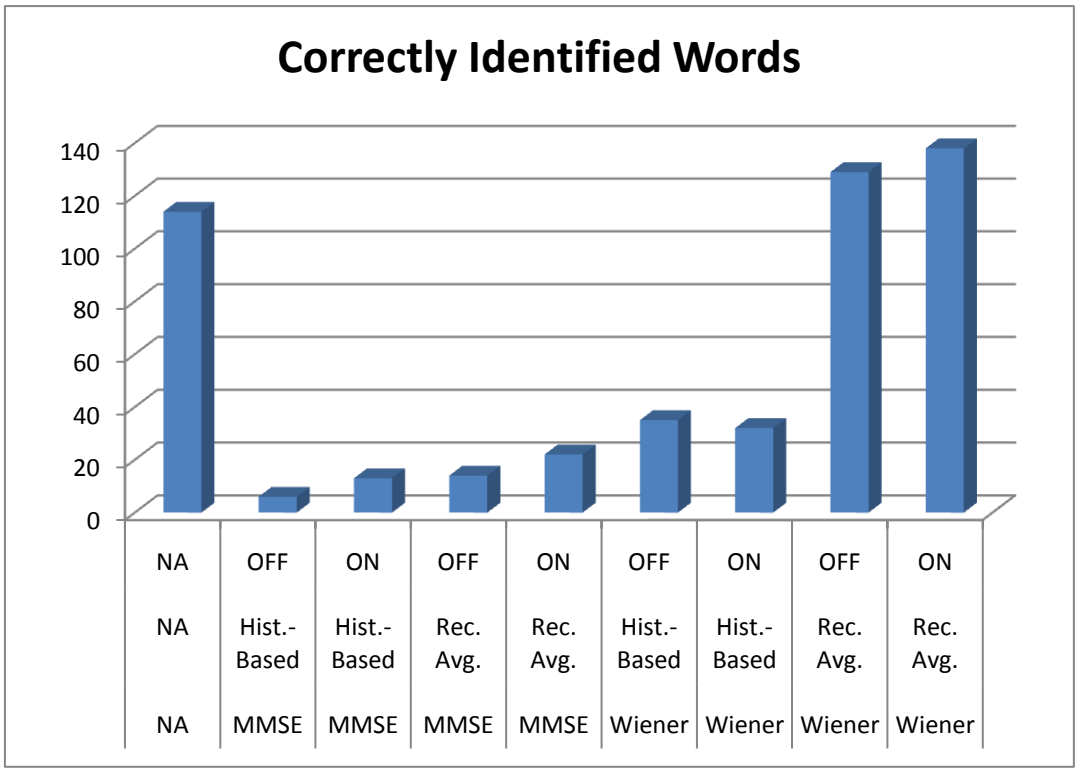


Figure 6.7 Graphical comparisons of speech-to-text algorithm outputs for speech noise.

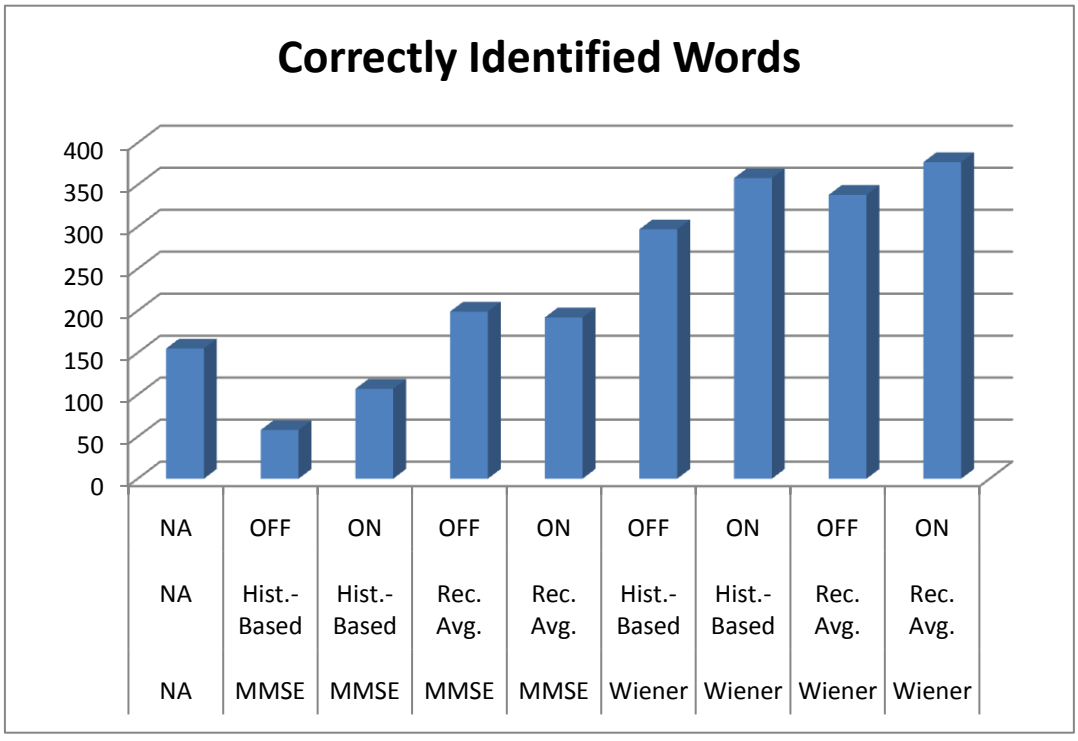


Figure 6.8 Graphical comparisons of speech-to-text algorithm outputs for PN pink noise.

CHAPTER SEVEN
CONCLUSIONS

Speech enhancement aims to improve speech quality by using various algorithms. The objective of enhancement is improvement in intelligibility and/or overall perceptual quality of degraded speech signal using audio signal processing techniques.

Throughout this thesis work, a real-time system that performs speech enhancement task is designed and implemented on the evaluation module of OMAP-L137 digital signal processor of Texas Instruments. Unlike the algorithms proposed in the literature, we have implemented the algorithm at real-time. Therefore, one of the most important challenges for us is the speed of the process. This means that the operations on the currently processed data must be completed until a new audio packet arrives. In addition to this requirement, the developed system needs to improve speech quality and/or speech intelligibility.

We have implemented several algorithms by combining different speech enhancement, noise estimation algorithms, and filtering operations on the evaluation module. In order to implement some algorithms such as histogram-based noise estimation and LOG-MMSE speech enhancement algorithm on the evaluation module, we have applied several optimizations related to software and the architecture of the DSP. For instance, the memory management is very important in terms of the performance of the algorithm; the internal and external RAM blocks must be used correctly for the maximum performance. Inaccurate memory management causes delays in the process which leads to nonexecution of the desired algorithms.

In addition to the implementation of the speech enhancement and noise estimation algorithms in the literature, we have also proposed a novel fusion noise estimation algorithm as outlined in Chapter Five. The proposed algorithm aims to increase the performance of the SNR-dependent recursive averaging noise estimation algorithm

described in Lin, L., Holmes W.H. & Ambikairajah, E. (2003) by updating the β parameter, which determines the rate of the noise updates based on the input signal. We have used the decision rule called LTSD which is proposed in Ramirez, J. et. al. (2003) in order to estimate the β parameter. By this way, we have proposed a new method that updates the β parameter based on the input data as opposed to the use of a constant β parameter as in Lin, L., Holmes W.H. & Ambikairajah, E. (2003).

Different speech and noise signals are used during the studies: Atatürk's address to the Turkish youth, several news contents, a speech related to old ages, soccer games with heavy “Vuvuzela” noise, sounds of pilots in an airplane cockpit, white noise, PN pink noise, speech noise, restaurant ambiance noise, and airport noise, etc. The developed algorithms are tested by using all these noisy speech files and the processed data are evaluated by using both subjective and objective methods.

We have evaluated several combinations of algorithms by using objective measures for both quality and intelligibility as described in Chapter Six. We have used eight different quality measures in order to evaluate the speech quality. According to the obtained results, the Wiener filtering approach together with recursive averaging noise estimation algorithm has produced favorable results in all cases and has generally provided the best results in terms of all quality measures. As already mentioned in Section 6.1, the use of IIR filter has not improved the quality noticeably. This is probably because of not filtering of the clean speech signal that is used by the objective measures for comparison.

It is seen in Table 6.3 that the use of different β parameter values for different input types increases the performance and a constant β parameter value is not approprite for all cases. Hence, the β parameter should be updated based on the input signal type for a better performance. As shown in Table 6.3, the novel fusion noise estimation algorithm succeeds in automatically updating the β parameter based on the input signal type and produces acceptably good results for all cases. As described in Chapter Five, the β parameter value is determined as one of the four different pre-defined values. Further study can be performed in order to derive a procedure or

formula that can be used to compute the β parameter value by using some kind of audio features instead of using experimentally-determined β values.

We have also evaluated the implemented algorithms in terms of speech intelligibility by using a speech recognition technique given in Section 2.7.3.7. Generally, most of the algorithms increases the intelligibility, but some of them decreases it for some input signals. According to this evaluation technique, again the Wiener filtering approach together with recursive averaging noise estimation algorithm and IIR filtering is the combination that provides the best performance in terms of intelligibility. That combination has decreased the word error rate from 99,07% to 56,2% for the white noise case, from 84,78% to 81,57% for the speech noise case and from 79,3% to 49,67% for the PN pink noise case. Thus, we can state that application of the enhancement algorithm generally decreases the word error rate and increases the performance in terms of speech intelligibility a great deal.

We have also used “vuvuzela” noise in our studies, but we have not applied objective evaluation methods for that noise type. With respect to subjective evaluation measures, it is seen that the enhancement algorithms, especially the Wiener filtering approach together with recursive-averaging noise estimation algorithm, suppresses the “vuvuzela” noise considerably. This combination has increased the speech quality without decreasing the speech intelligibility for the input signal containing “vuvuzela” noise.

In summary, the combination of Wiener filtering approach and SNR-dependent recursive averaging noise estimation algorithm together with the application of IIR filtering has improved both speech quality and intelligibility in most cases. The novel fusion noise estimation algorithm has increased the performance of the noise estimation algorithm in terms of speech quality.

REFERENCES

Berouti, M., Schwartz, M. & Makhoul, J. (1979). Enhancement of speech corrupted by acoustic noise. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 208-211.*

Cappe, O. (1994). Elimination of the musical noise phenomenon with the Ephraim and Malah noise suppressor. *IEEE Trans. Speech Audio Process., 2(2), 346-349.*

Cohen, I. (2002). Noise estimation by minima controlled recursive averaging for robust speech enhancement. *IEEE Signal Process. Lett., 9(1), 12-15.*

Cohen, I. (2005). Relaxed statistical model for speech enhancement and a priori SNR estimation. *IEEE Trans. Speech Audio Process., 13(5), 870-881.*

Doblinger, G. (1995). Computationally efficient speech enhancement by spectral minima tracking in subbands. Proc. Eurospeech, pp. 1513-1516.

Ephraim, Y. & Malah, D. (1985). Speech enhancement using a minimum mean-square error log-spectral amplitude estimator. *IEEE Trans. Acoust., Speech, Signal Process., ASSP-23(2), 443-445.*

Hasan, M., Salahuddin, S. & Khan, M. (2004). A modified a priori SNR for speech enhancement using spectral subtraction rules. *IEEE Signal Process. Lett., 11(4), 450-453.*

Hirsch, H. & Ehrlicher, C. (1995). Noise estimation techniques for robust speech recognition. *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 153-156.*

Hu, Y. & Loizou, P. (2003). A generalized subspace approach for enhancing speech corrupted by colored noise. *IEEE Trans. on Speech and Audio Processing, 11, 334-341.*

Hu, Y. & Loizou, P. (2004). Speech enhancement based on wavelet thresholding the multitaper spectrum. *IEEE Trans. on Speech and Audio Processing, 12(1), 59-67.*

Hu, Y. & Loizou, P. (2006). Evaluation of objective measures for speech enhancement. *Proc. Interspeech, Pittsburg, PA.*

ITU-T International Telecommunication Union, Telecommunication Standardization Sector of ITU, P.862 (2001). Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *Series P: Telephone Transmission Quality, Telephone Installations, Local Line Networks.*

ITU-T International Telecommunication Union, Telecommunication Standardization Sector of ITU, P.862 (1996). Methods for subjective determinatıon of transmission quality. *Series P: Telephone Transmission Quality.*

Jabloun, F. & Champagne, B. (2003). Incorporating the human hearing properties in the signal subspace approach for speech enhancement. *IEEE Trans. on Speech and Audio Processing, 11(6), 700-708.*

Lin, L., Holmes W. H. & Ambikairajah, E. (2003). Adaptive noise estimation algorithm for speech enhancement. *Electronics Lett., 39(9), 754-755.*

Loizou, P. C. (2007). Speech Enhancement: Theory and Practice, *CRC Press.*

Lotter, T. & Vary, P. (2005). Speech enhancement by maximum a posteriori spectral amplitude estimation using a supergaussian speech model. *EURASIP J. Appl. Signal Process, 2005 (7), 1110-1126.*

Martin, R. (1993). An efficient algorithm to estimate the instantaneous SNR of speech signals. *Proc. Eurospeech, pp. 1093-1096.*

Martin, R. (1994). Spectral subtraction based on minimum statistics. Proc. Euro. Signal Process., pp. 1182-1185.

Martin, R. (2001). Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on Speech and Audio Processing, 9(5), 504-512.*

McAulay, R. J. & Malpass, M. L. (1980). Speech enhancement using a soft-decision noise suppression filter. *IEEE Trans. Acoust. Speech Signal Process., 28, 137-145.*

Mittal, U. & Phamdo, N. (2000). Signal/noise KLT based approach for enhancing speech degraded by noise. *IEEE Trans. Speech Audio Process., 8(2), 159-167.*

Moor, B. (1993). The singular value decomposition and long and short spaces of noisy matrices. *IEEE Trans. Signal Process., 41(9), 2826-2838.*

NCVS Tutorials – Voice Production. *National Center for Voice and Speech, http://www.ncvs.org/ncvs/tutorials/voiceprod/tutorial/graphing.html*

OMAP-L137 Evaluation Module, Technical Reference, 511345-0001 Rev. A, (2008). *Spectrum Digital, Inc., www.spectrumdigital.com*

OMAP-L137 Low-Power Applications Processor Datasheet, SPRS563D, (2008). *Texas Instruments, Inc., www.ti.com*

Rabiner, L. & Schafer, R. (1978), Digital Signal Processing of Speech Signals, *Englewood Cliffs, NJ: Prentice Hall.*

Ramirez, J., Segura, J. C., Benitez, C., Torre A. & Rubio, A. (2003). Efficient voice activity detection algorithms using long-term speech information. *Speech Communication, 42, (2004), 271–287. www.elsevier.com/locate/specom*

Rezayee, A. & Gazor, S. (2001). An adaptive KLT approach for enhancing speech degraded by noise. *IEEE Trans. Speech Audio Process., 8(2), 159-167.*

Ris, C. & Dupont, S. (2001). Assessing local noise level estimation methods: Application to noise robust ASR. *Speech Communication, 34, 141-158.*

Quackenbush, S., Barnwell, T. & Clements, M. (1988). Objective Measures of Speech Quality, *Englewood Cliffs, NJ: Prentice Hall.*

Scalart, P. & Filho, J. (1996). Speech enhancement based on a priori signal to noise estimation. *Proc. IEEE Int. Conf. Acoust, Speech, Signal Processing, 629-632.*

Sohn, J. & Sung, W. (1998). A voice activity detector employing soft decision based noise spectrum adaptation. *Proc. IEEE Int. Conf. Acoust. Speech Signal Process., pp. 365-368.*

SPRT529B, (2009). OMAP-L1x Software Solutions. *Texas Instruments, www.ti.com*

Vary, P. & Martin, R. (2006). Digital speech transmission: Enhancement, coding and error concealment. *Chichester, England: John Wiley & Sons.*

Wilkinson, J. H. (1999). The algebraic eigenvalue problem. *New York: Oxford University Press.*

Wolfe, P. & Godsill, S. (2000). Towards a perceptually optimal spectral amplitude estimator for audio signal enhancement. *Proc. IEEE Int. Conf. Acoust. Speech Signal Process., 2, pp. 821-824.*

Wolfe, P. & Godsill, S. (2001). Simple alternatives to the Ephraim and Malah suppression rule for speech enhancement. *Proc. IEEE Int. Conf. Acoust. Speech Signal Process., pp. 496-499.*

APPENDIX A

OMAP-L137 PROCESSOR

A.1 Advanced Information

OMAP-L137 Low-Power Applications Processor

Check for Samples: OMAP-L137

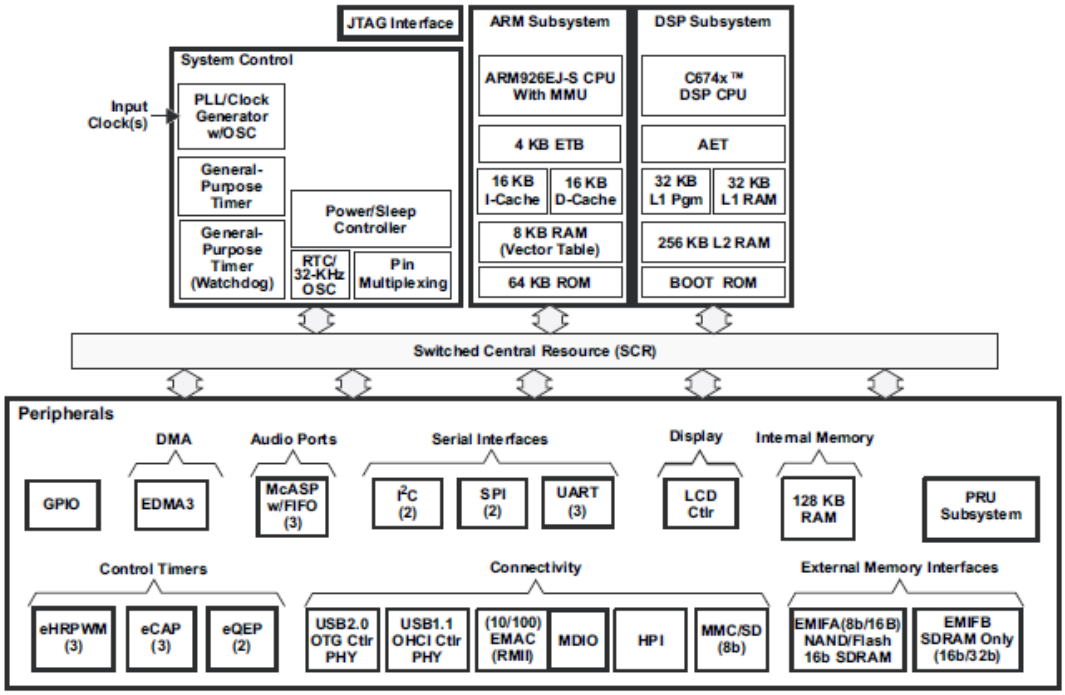
1 OMAP-L137 Low-Power Applications Processor

1.1 Features

- Highlights
  - Dual Core SoC
    - 375/456-MHz ARM926EJ-S™ RISC MPU
    - 375/456-MHz C674x VLIW DSP
  - TMS320C674x Fixed/Floating-Point VLIW DSP Core
  - Enhanced Direct-Memory-Access Controller 3 (EDMA3)
  - 128K-Byte RAM Shared Memory
  - Two External Memory Interfaces
  - Three Configurable 16550 type UART Modules
  - LCD Controller
  - Two Serial Peripheral Interfaces (SPI)
  - Multimedia Card (MMC)/Secure Digital (SD)
  - Two Master/Slave Inter-Integrated Circuit
  - One Host-Port Interface (HPI)
  - USB 1.1 OHCI (Host) With Integrated PHY (USB1)
- Applications
  - Industrial Diagnostics
  - Test and measurement
  - Military Sonar/ Radar
  - Medical measurement
  - Professional Audio
- Software Support
  - TI DSP/BIOS™
  - Chip Support Library and DSP Library
- Dual Core SoC
  - 375/456-MHz ARM926EJ-S™ RISC MPU
  - 375/456-MHz C674x VLIW DSP
- ARM926EJ-S Core
  - 32-Bit and 16-Bit (Thumb®) Instructions
  - DSP Instruction Extensions
  - Single Cycle MAC
  - ARM® Jazelle® Technology
  - EmbeddedICE-RT™ for Real-Time Debug
- ARM9 Memory Architecture
- C674x Instruction Set Features
  - Superset of the C67x+™ and C64x+™ ISAs
  - Up to 3648/2736 C674x MIPS/MFLOPS
  - Byte-Addressable (8-/16-/32-/64-Bit Data)
  - 8-Bit Overflow Protection
  - Bit-Field Extract, Set, Clear
  - Normalization, Saturation, Bit-Counting
  - Compact 16-Bit Instructions
- C674x Two Level Cache Memory Architecture
  - 32K-Byte L1P Program RAM/Cache
  - 32K-Byte L1D Data RAM/Cache
  - 256K-Byte L2 Unified Mapped RAM/Cache
  - Flexible RAM/Cache Partition (L1 and L2)
- Enhanced Direct-Memory-Access Controller 3 (EDMA3):
  - 2 Transfer Controllers
  - 32 Independent DMA Channels
  - 8 Quick DMA Channels
  - Programmable Transfer Burst Size
- TMS320C674x Fixed/Floating-Point VLIW DSP Core
  - Load-Store Architecture With Non-Aligned Support
  - 64 General-Purpose Registers (32 Bit)
  - Six ALU (32-/40-Bit) Functional Units
    - Supports 32-Bit Integer, SP (IEEE Single Precision/32-Bit) and DP (IEEE Double Precision/64-Bit) Floating Point
    - Supports up to Four SP Additions Per Clock, Four DP Additions Every 2 Clocks
    - Supports up to Two Floating Point (SP or DP) Reciprocal Approximation (RCPxP) and Square-Root Reciprocal Approximation (RSQRxP) Operations Per Cycle
  - Two Multiply Functional Units
    - Mixed-Precision IEEE Floating Point Multiply Supported up to:
      - 2 SP x SP -> SP Per Clock
      - 2 SP x SP -> DP Every Two Clocks
      - 2 SP x DP -> DP Every Three Clocks
      - 2 DP x DP -> DP Every Four Clocks
    - Fixed Point Multiply Supports Two 32 x

32-Bit Multiplies, Four 16 x 16-Bit Multiplies, or Eight 8 x 8-Bit Multiplies per Clock Cycle, and Complex Multiples
  - Instruction Packing Reduces Code Size
  - All Instructions Conditional
  - Hardware Support for Modulo Loop Operation
  - Protected Mode Operation
  - Exceptions Support for Error Detection and Program Redirection
- 128K-Byte RAM Shared Memory
- 3.3V LVCMOS IOs (except for USB interfaces)
- Two External Memory Interfaces:
  - EMIFA
    - NOR (8-/16-Bit-Wide Data)
    - NAND (8-/16-Bit-Wide Data)
    - 16-Bit SDRAM With 128MB Address Space
  - EMIFB
    - 32-Bit or 16-Bit SDRAM With 256MB Address Space
- Three Configurable 16550 type UART Modules:
  - UART0 With Modem Control Signals
  - Autoflow control signals (CTS, RTS) on UART0 only
  - 16-byte FIFO
  - 16x or 13x Oversampling Option
- LCD Controller
- Two Serial Peripheral Interfaces (SPI) Each With One Chip-Select
- Multimedia Card (MMC)/Secure Digital (SD) Card Interface with Secure Data I/O (SDIO)
- Two Master/Slave Inter-Integrated Circuit (I^2C Bus™)
- One Host-Port Interface (HPI) With 16-Bit-Wide Muxed Address/Data Bus For High Bandwidth
- Programmable Real-Time Unit Subsystem (PRUSS)
  - Two Independent Programmable Realtime Unit (PRU) Cores
    - 32-Bit Load/Store RISC architecture
    - 4K Byte instruction RAM per core
    - 512 Bytes data RAM per core
    - PRU Subsystem (PRUSS) can be disabled via software to save power
  - Standard power management mechanism
    - Clock gating
    - Entire subsystem under a single PSC clock gating domain
  - Dedicated interrupt controller
  - Dedicated switched central resource
- USB 1.1 OHCI (Host) With Integrated PHY (USB1)
- USB 2.0 OTG Port With Integrated PHY (USB0)
  - USB 2.0 High-/Full-Speed Client
  - USB 2.0 High-/Full-/Low-Speed Host
  - End Point 0 (Control)
  - End Points 1,2,3,4 (Control, Bulk, Interrupt or ISOC) Rx and Tx
- Three Multichannel Audio Serial Ports:
  - Six Clock Zones and 28 Serial Data Pins
  - Supports TDM, I2S, and Similar Formats
  - DIT-Capable (McASP2)
  - FIFO buffers for Transmit and Receive
- 10/100 Mb/s Ethernet MAC (EMAC):
  - IEEE 802.3 Compliant (3.3-V I/O Only)
  - RMII Media Independent Interface
  - Management Data I/O (MDIO) Module
- Real-Time Clock With 32 KHz Oscillator and Separate Power Rail
- One 64-Bit General-Purpose Timer (Configurable as Two 32-Bit Timers)
- One 64-bit General-Purpose/Watchdog Timer (Configurable as Two 32-bit General-Purpose Timers)
- Three Enhanced Pulse Width Modulators (eHRPWM):
  - Dedicated 16-Bit Time-Base Counter With Period And Frequency Control
  - 6 Single Edge, 6 Dual Edge Symmetric or 3 Dual Edge Asymmetric Outputs
  - Dead-Band Generation
  - PWM Chopping by High-Frequency Carrier
  - Trip Zone Input
- Three 32-Bit Enhanced Capture Modules (eCAP):
  - Configurable as 3 Capture Inputs or 3 Auxiliary Pulse Width Modulator (APWM) outputs
  - Single Shot Capture of up to Four Event Time-Stamps
- Two 32-Bit Enhanced Quadrature Encoder Pulse Modules (eQEP)
- 256-Ball Pb-Free Plastic Ball Grid Array (PBGA) [ZKB Suffix], 1.0-mm Ball Pitch
- Commercial, Industrial, Extended, or Automotive Temperature
- Community Resources
  - TI E2E Community
  - TI Embedded Processors Wiki

A.2 Functional Block Diagram



Note: Not all peripherals are available at the same time due to multiplexing.