INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS DE LA DIVISION ELECTRONICA, COMPUTACION, INFORMACION Y COMUNICACIONES

A WAVELET-BASED FRONT END FOR ROBUST ASR

T E S I S

PRESENTADA COMO REQUISITO PARCIAL PARA
OBTENER EL GRADO ACADEMICO DE
MAESTRO EN CIENCIAS
EN INGENIERIA ELECTRONICA CON ESPECIALIDAD
EN SISTEMAS ELECTRONICOS

PABLO HENNINGS YEOMANS

MAYO, 2002

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY
CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA,
COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

A WAVELET-BASED FRONT END FOR ROBUST ASR

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL
GRADO ACADÉMICO DE MAESTRO EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA CON ESPECIALIDAD EN
SISTEMAS ELECTRÓNICOS

PABLO HENNINGS YEOMANS

MAYO 2002

A WAVELET-BASED FRONT END FOR ROBUST ASR

TESIS

MAESTRÍA EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA CON ESPECIALIDAD EN
SISTEMAS ELECTRÓNICOS

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

POR

PABLO HENNINGS YEOMANS

MAYO DE 2002

A mis papás

AGRADECIMIENTOS

Con toda sinceridad y profundo cariño, fuera de todo formulismo, más allá de cualquier tradición o costumbrismo superficial, muchas gracias a ti Señor, mi Dios y mi Todo, Padre, rico en misericordia, lleno de ternura y compasión.

Con todo mi cariño y sincera admiración, a mis papás, Nora Leticia Yeomans de Hennings y Luis Ralph Paul Hennings Noriega, muchas gracias por todo su apoyo, sus sacrificios, por su visión y todo lo que me han enseñado.

A todos los Yeomans, a mi nana y a mi tata (que en paz descanse), a mis tías, a mis tíos y a todos mis primos. Prometo en la tesis doctoral explayarme en tantos agradecimientos que les debo.

Al padre Victor Manuel Chaveznava, por su amistad y su experiencia en estos años que estuve lejos de casa.

A todos mis amigos que están en todos lados. De todos me acuerdo, pero no terminaría de escribir sus nombres. Gracias a todos por su amistad.

Al Dr. Juan Arturo Nolazco Flores, muchas gracias por su paciencia, dedicación y esfuerzo que por mucho hicieron que todo esto fuera posible.

Al Dr. Ramón Rodríguez Cruz y al Dr. Arturo Galván Rodríguez por formar parte en el comité de tesis.

ABSTRACT

In this work we present results attained by noisy speech recognition experiments using wavelet analysis schemes. It is shown that under white noisy signals the wavelet parameters outperform the Mel-Frequency Cepstral Coefficients (MFCC). The main difference between the wavelet derived coefficients and the traditional MFCC consists in the computation of the spectrum, since the proposed parameters apply a wavelet packet transform instead a discrete Fourier transform.

The filters in the wavelet packet transforms used in this work are Daubechies 20, Beylkin 18 and Vaidyanathan 24. The Daubechies filters maximize the smoothness of the associated scaling function by maximizing the rate of decay of its Fourier transform [Daubechies]. The Beylkin's filter was designed by placing roots for the frequency response polynomial close to the Nyquist frequency on the real axis, thus concentrating power spectrum energy in the desired band. Vaidyanathan's filter was optimized for its length to satisfy standard requirements for effective speech coding [Wikerhauser].

The experimental work show that under a noisy continuous spoken digits task the word accuracy improves up to a 32% compared with the MFCC results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This work is about improving the overall word recognition accuracy (word error rate) of the Carnegie Mellon University Sphinx 3 automatic recognition system (ASR). Our efforts concentrate on extracting better features from speech signals using wavelet-based feature schemes recently proposed in the literature and schemes we suggest based on our own findings.

Although, the speech recognition problem is not a recent problem, improving accuracy of robust speech recognition systems remains a challenge and an active field of intense research [Acero]. Moreover, given the interdisciplinary nature of automatic speech recognition, a large variety of different approaches to the problem is found in the literature. This coalescing environment actually pervades ASR since the beginning; the merging of fields like signal processing, acoustic modeling, and language modeling (to name a few) in the same research framework has solved many of the quests thought lost more than twenty years ago, and becomes better illustrated in the different components of the speech recognizer we introduce in the next chapter.

On the other hand, a relatively recent merging of different areas, namely, subband coding, multiresolution theory, and other engineering fields (originally, for example, seismology) was brought by *wavelet theory* [Mallat] to support and impel contemporary signal processing applications. The tools provided by this new time-frequency (multiresolution) analysis has been shown very promising in the last two decades because its not so complex practical implementation and potential to attain better results in many engineering applications [Goswami] [Mallat] [Farooq].

As we had the opportunity to search and learn about wavelet theory in the application reports found in the literature and books we had at hand, and because our interest resides in the signal processing of speech signals (for ASR, now), we decided starting with the project

that motivated this work. First, we built a programming *workbench* that provided us with the basic needs for a testing stage. Then, we adapted our scripts to fit the Sphinx 3 codebase. Indeed unfortunately, this work and so much time-consuming stage cannot be seen in this report, but in the thousands of c- and perl-programming lines buried in our workstations, although, only the preprocessor of Sphinx 3 was modified.

It is fair to say that the author is not an expert in the field (yet), neither wavelet theory nor ASR, but as we have a global overview and research-avid intuition, we established the basic guidelines to accomplish the project this work is all about. Our main goal (or hypothesis) is to prove that a front end which transforms speech data into cunningly chosen wavelet coefficients, for the training and decoding stage of Sphinx 3, will be as good as the one providing the baseline coefficients (Mel-Frequency Cepstral Coefficients for ASR), and even significantly better.

Without luck, completely proving this might take more time than the one scheduled for this work. We propose a couple of classes of parameters or coefficients, and present the structure and conditions of the experiments we were able to accomplish by the time this report was required. With this, all components of the recognizer, but the front end are to remain unchanged with a previously defined configuration throughout our experiments. Even though, this self-imposed restriction might not be the best for all kind of speech features, we just had to leave with it in sake of a baseline reference for analyzing and discussing our results.

Beginning with chapter 2, as first framework, we give a brief description of speech recognition tools which environment supports our experiments. We present a mathematical formulation of the speech recognition problem and speech recognizer components as found in [Jelinek]. Then we outline the main algorithms involved in training and decoding speech recognition systems (explained in [Acero]), namely, Baum-Welch estimation process and Viterbi search algorithm, respectively. Finally, the Sphinx 3 system configuration and baseline environment are briefly mentioned.

Another framework dedicated to signal processing is in chapter 3; the speech processing fundamentals of the baseline and our designed experiments are outlined, this is, the cepstral analysis and wavelet transforms. Cepstral analysis is discussed following the approach by [Harrington] and introducing necessary definitions found in [Deller]. Wavelet transforms theory is explained more in an intuitive didactic way, rather then with a strict mathematical development.

Then, our experiments implementation and practical definitions are presented in chapter 4. This chapter includes our proposed front end with its configuration explained. It consists of three sections: the (MFCC) baseline coefficients, and the two different classes of proposed wavelet-based features.

The results of our experiments are presented in chapter 5 of this report in an executive-summary fashion, and a brief last conclusion and statement of future work is found in chapter 6.

The author encourages the reader to search in the references for a better explanation of the subjects presented. Furthermore, the experiments and results we provide are those that were implemented and attained by the time this report was required. A thorough description of our actual work can be found in the internet [http://juang.mty.itesm.mx].

CHAPTER 2

FRAMEWORK 1: A ROUGH INTRODUCTION TO AUTOMATIC SPEECH RECOGNITION

In this chapter we give a brief description of the main structure of the automatic speech recognition (ASR) environment which fully permeates the experiments of this work. There are several good books that explain all the material shown here in a thorough and better strictness than we do. Our purpose at this point is to fit the complacence of an outlined framework for a better understanding (by the novice reader) of the speech recognition tasks conveyed throughout this thesis.

As stated before, our work concentrates its efforts on improving the overall word recognition accuracy of the Sphinx 3 system by proposing and experimenting different wavelet-based front ends, but living the rest of the system configuration constant. These front ends are intended to yield a bundle of parameters that would provide with better information to the recognition system, and so improve the recognition task.

We begin with a mathematical formulation of the speech recognition problem which is extracted from [Jelinek] and then give a rough outline of the most important algorithms to solve the basic speech recognition modeling inquiries, as in [Acero]. The reader is referred to these sources for a complete description of the statistical point of view of ASR which pervades the most important tasks of the system we work with. Then, we describe the Sphinx recognition system and the speech corpus we used to standardize our results.

2.1 A Mathematical Description of the Recognition Problem

Restating Jelinek's statistical approach, let **D** denote the acoustic evidence (data features) from which the recognizer will somehow make its decision about which utterances were spoken. As we practically have digital data (because of digital computers), then we may thought that, in general, **D** is a sequence of symbols extracted from some *m*-length alphabet Δ:

$$\mathbf{D} = d_1, d_2, ..., d_m \; d_i \in \Delta \tag{1}$$

As indicates by the index i, the symbols d_i may be thought of as generated in time (which indeed they have). Let

$$\mathbf{W} = w_1, w_2, ..., w_n \quad w_i \in \Gamma \tag{2}$$

denote a sequence of n uttered words, each belonging to a known vocabulary Γ previously arranged.

If P(**W**|**D**) defines the probability that the word sequence **W** was spoken, given that the evidence data features **D** were extracted from it, then the recognizer should decide in favor of a utterance $\hat{\mathbf{W}}$ satisfying

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{D}) \tag{3}$$

This means, the recognizer will choose the most likely word sequence given the observed (acoustic) data features.

We can rewrite the right-hand side probability of (3) using Bayes' formula, as

$$P(\mathbf{W} \mid \mathbf{D}) = \frac{P(\mathbf{W})P(\mathbf{D} \mid \mathbf{W})}{P(\mathbf{D})} \tag{4}$$

where P(**W**) is the probability that the word sequence **W** will be spoken, P(**D**|**W**) is the probability that when the speaker says **W** the data features **D** will be observed, and P(**D**) is the average probability that **D** will be observed. Thit is,

$$P(\mathbf{D}) = \sum_{\mathbf{W}'} P(\mathbf{W}')P(\mathbf{D} \mid \mathbf{W}') \tag{5}$$

As we can see in (3), the maximization is achieved with the data features variable **D** fixed, that is there is no other acoustic features but the one we are given, it then follows from (3) and (4) that the recognizer's aim is to find the word sequence $\hat{\mathbf{W}}$ that maximizes the product P(**W**)P(**D**|**W**), i.e., it satisfies

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{W})P(\mathbf{D} \mid \mathbf{W}) \tag{6}$$

2.2 The Speech Recognizer

The different approaches to speech recognition [Rabiner] lead to different specific implementations, but we may abstract the basic components of a speech recognizer and present them as in [Jelinek]. The reader may refer to [Rabiner], [Acero] and [Jelinek] for a thorough discussion on this subject.

2.2.1 Acoustic Processing

In the first place, we need to furnish the acoustic features **D** that will be observed by the recognizer. This means we need to design the front end that will transform the sound's waveform into the parameters d_i the recognizer will deal with. To achieve this the front end will need a microphone (which produces an electric signal), a means of sampling the electric signal, and an algorithm for processing the (sampled) digital signal [Jelinek].

The signal processor converts the digital signal into feature vectors at hundred times per second. This feature vectors could be energy values of a filter bank's output applied to the sampled electric signal the microphone provided us.

We will discuss signal processing issues in chapter 3. By now, an intuitive approach to this preprocessing (front end) stage is enough.

2.2.2 Acoustic Modeling

In formula (6), we see that the recognizer will need to compute the value $P(\mathbf{D}|\mathbf{W})$ of the probability that when the speaker uttered the word sequence **W** the front end yielded the features **D**. This probability must be calculated at real-time since it needs to be available for all possible combinations of **W** and **D**.

As this later number of possible combinations is too large, we better compute this value $P(\mathbf{D}|\mathbf{W})$ through a statistical acoustic model of the speaker's interaction with the front end [Jelinek]. The complete process we are modeling here involves how the speaker *pronounces* the words of **W**, the ambience (e.g. noise, reverberation), the microphone placement, etc. and the speech signal processing achieved by the front end.

The acoustic model the Sphinx platform uses is the hidden Markov model (HMM), even though others models based on artificial neural networks or dynamic time warping are also possible. Let us first briefly define the Markov chain model and then present the HMM. We will not explain the concept of HMM, but state the formal components of the HMM definition as in [Acero].

The Markov Chain

The *Markov chain* models a class of random processes that incorporates a minimum amount of memory without being completely memoryless. We focus on the discrete-time Markov chain only, but the extension to the continuous case is also possible.

Let $\mathbf{X} = X_1, X_2, ..., X_t$ be a sequence of random variables from a finite discrete alphabet $\mathbf{O} = \{o_1, o_2, ..., o_M\}$. Using Bayes rule, the probability of the sequence is

$$P(X_1, X_2, ..., X_t) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_1^{i-1}) \tag{7}$$

where $X_1^{i-1} = X_1, X_2, ..., X_{i-1}$. The random variables X form a first-order Markov chain provided that

$$P(X_i \mid X_1^{i-1}) = P(X_i \mid X_{i-1}) \tag{8}$$

which is known as the *Markov assumption*. This assumption is said to use very little memory to model dynamic data sequences because the probability of the random variable at a given time depends only on the value at the preceding time. In this manner, Equation (7) becomes for the first-order Markov chain

$$P(X_1, X_2, ..., X_t) = P(X_1)\prod_{i=2}^{n} P(X_i \mid X_{i-1}) \tag{9}$$

Expanding our point view, the Markov chain can be used to model time-invariant (stationary) events if we discard the time index i, so that

$$P(X_i = s \mid X_{i-1} = s') = P(s \mid s'). \tag{10}$$

Furthermore, if we associate X_i to a state, the Markov chain can be represented by a finite state process with transition between states specified by the probability function $P(s|s')$. Using this finite state representation, the Markov assumption (8) may be restated as follows: the probability that the Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time [Acero].

Until now, we have described an *observable* Markov model because the output of the process is the set of states at each time instance t, where each state corresponds to an observable event X_t, i.e., there is one-to-one (*deterministic*) correspondence between the observable event sequence **X** and the Markov chain state sequence $\mathbf{S} = \{s_1, s_2, ..., s_n\}$.

The Hidden Markov Model

The hidden Markov model introduces a non-deterministic process that generates output observation symbols in any given state, so it may be seen as a natural extension to the Markov chain. In other words, a hidden Markov model is basically a Markov chain where the output observation is a random variable X generated according to an output probabilistic function associated with each state. To formally state the definition of the HMM we present the one found in [Acero]. In this manner, the HMM is defined by:

- $\mathbf{O} = \{o_1, o_2, ..., o_M\}$

An output observation alphabet. The observation symbols correspond to the physical output of the system being modeled. Even though we refer to a discrete output observation, it can be extended to the continuous case with a continuous probability density function.

- $\Omega = \{1, 2, ..., N\}$

A set of states representing the state space. Here s_t is denoted as the state at time t.

- $\mathbf{A} = \{a_{ij}\}$

A transition probability matrix, where a_{ij} is the probability of taking a transition from state i to state j, i.e.,

$$a_{ij} = P(s_t = j \mid s_{t-1} = i)$$

- $\mathbf{B} = \{b_i(k)\}$

An output probability matrix, where $b_i(k)$ is the probability of emitting symbol o_k when state i is entered. Let $\mathbf{X} = X_1, X_2, ..., X_t$ be the observed output of the HMM. The state sequence $S = s_1, s_2, ..., s_t$ is not observed (hidden) and $b_i(k)$ can be rewritten as

$$b_i(k) = P(X_t = o_k \mid s_t = i)$$

- $\pi = \{\pi_i\}$

An initial state distribution, where

$$\pi_i = P\{s_0 = i\} \qquad 1 \le i \le N$$

The following properties must hold for π_i, a_{ij}, $b_i(k)$ because of being probabilities.

$$a_{ij} \ge 0, b_i(k) \ge 0, \pi_i \ge 0 \qquad \forall\, i, j, k.$$

$$\sum_{j=1}^{N} a_{ij} = 1$$

$$\sum_{k=1}^{M} b_i(k) = 1$$

$$\sum_{i=1}^{N} \pi_i = 1$$

Summarizing, a complete specification of an HMM includes two constant-size parameters, N and M, representing the total number of states and the size of observation alphabets, observation alphabet **O**, and three sets (matrices) of probability measures **A**, **B**, π. At last, there are two assumptions in the first order HMM, the *Markov assumption* and the *output independence assumption*. The former was introduced in the previous subsection, and is restated here as

$$P(s_t \mid s_1^{t-1}) = P(s_t \mid s_{t-1}) \tag{11}$$

where $s_1^{t-1} = s_1, s_2, ..., s_{t-1}$. The *output independence assumption* states that the probability of generating a particular symbol at time t depends only in state s_t, but no other preceding state or past observations. This is written as

$$P(X_t \mid X_1^{t-1}, s_1^t) = P(X_t \mid s_t) \tag{12}$$

where $X_1^{t-1} = X_1, X_2, ..., X_{t-1}$.

HMM-Based Phonetic Acoustic Model

The following are the basic steps for building HMMs as a phonetic acoustic model for words (as presented in [Jelinek]).

1. A phonetic dictionary for the vocabulary to be recognized must be provided, i.e. we need to establish the correspondence between each word and a sequence of symbols representing the phonetic parts of the word. Here we must choose the phonetic alphabet to be used.
2. We assign a prototype HMM for each symbol of the phonetic alphabet. Figure 2.1 shows the transition structure of a HMM (without starting and ending states, which might be needed) that could be appropriate for each symbol.
3. A concatenation of the prototype HMMs specified by the sequence of symbols corresponding to a word is the HMM of that word. The final state of one HMM is connected to the starting state of the next HMM by a null transition.
4. The *composite* model for an utterance of some given speech data **D** is the indiviual words HMMs connected by the HMM of silence symbols and (possibly) end-of-word symbols.
5. To estimate the HMM parameters we use the Baum-Welch algorithm [Jelinek]. This process involves letting users read a prepared text **W**, observing the front end's output **D** and using the composite HMM corresponding to **W** as a model of the production mechanism that resulted in the observed **D** [Jelinek].

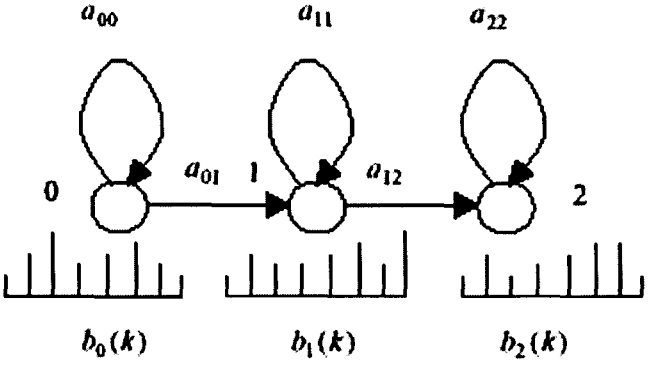


Figure 2.1. A typical hidden Markov model used to model phoneme. There are three states (0-2) and each state has an associated output probability distribution [Acero *et al.*].

A further description of this models is out of the scope of this work. The reader might found [Jelinek] an excellent book on this subject.

2.2.3 Language Modeling

The a priori probability P(**W**) that the speaker will utter **W** for every string **W** is also needed by formula (6). As stated in [Jelinek], Bayes' formula allows many decompositions of P(**W**), but because the recognizer "naturally" wishes to convey the text in the sequence in which it was spoken, we use the next decomposition

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i \mid w_1,\ldots, w_{i-1}) \tag{13}$$

As the number of arguments in the probability $P(w_i|w_1,\ldots, w_{i-1})$ is too large, we use equivalence classes $\Phi(w_1,\ldots,w_{i-1})$ instead. And formula (13) becomes

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i \mid \Phi(w_1,\ldots, w_{i-1})) \tag{14}$$

Then language modeling consists of determining the equivalence classification Φ and a method of estimating the probabilities $P(w_i|\Phi(w_1,\ldots, w_{i-1}))$ [Jelinek].

Typically, speech recognizers are based on trigram and bigram language models, as histories are the same if they end in the same two words or one word, respectively, as explained in the next subsection.

2.2.4 Hypothesis Search

At last, the speech recognizer needs to find the best matching transcription $\hat{\mathbf{W}}$ of the acoustic features **D** by formula (6) searching over all possible strings **W** to find the maximizing one [Jelinek]. As the number of words in **W** is extremely large, we cannot search on all of them one by one, but a much frugal hypothesis search is performed.

The Most Likely Word Sequence

Let us consider first the (simplest) language model in which all histories $w_1,w_2,\ldots,w_{i-1}$ are equivalent. Then

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i) \tag{15}$$

To find $\hat{\mathbf{W}}$ in this case we need to search the most likely path through our model schematically shown in Figure 2.2; it is not needed to know what happens inside the v_i's HMMs.

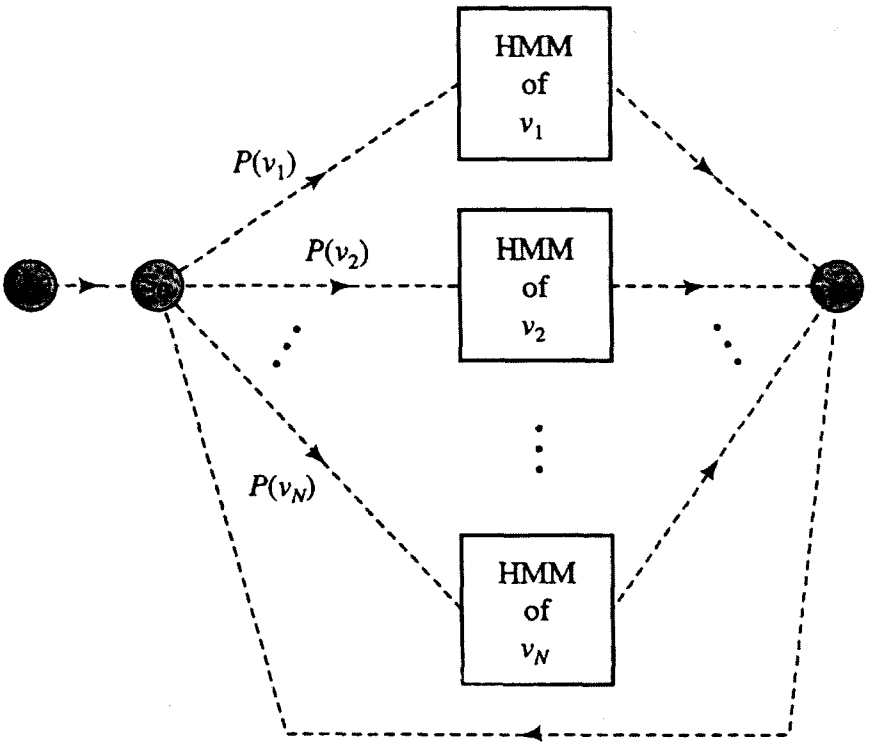


Figure 2.2. Composite model of speech generation when word generation is assumed to be memoryless [Jelinek]

The bigram language model assigns probabilities by the formula [Jelinek]

$$P(\mathbf{W}) = P(w_1)\prod_{i=2}^{n} P(w_i \mid w_{i-1}) \tag{16}$$

Now $\hat{\mathbf{W}}$ is given by the most likely path in a model where each word (HMM) depends only on the identity of the preceding word, as the one shown in Figure 2.3. It is important to note that in this case the number of acoustic models remains the same as in Figure 2.2 even when the graph in Figure 2.3 appears to be more cumbersome. Actually, the total number of states is proportional to the vocabulary size, which leads us to choose a bigram model (due to accuracy [Jelinek]) whenever possible, except for the cases where the conditional probabilities cannot be obtained.

Finally, for a trigram language model de probability is

$$P(\mathbf{W}) = P(w_1)P(w_2 \mid w_1)\prod_{i=3}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \tag{17}$$

Here the composite model is more complex; the number of states is proportional to $|m|^2$, where m is the vocabulary size [Jelinek].

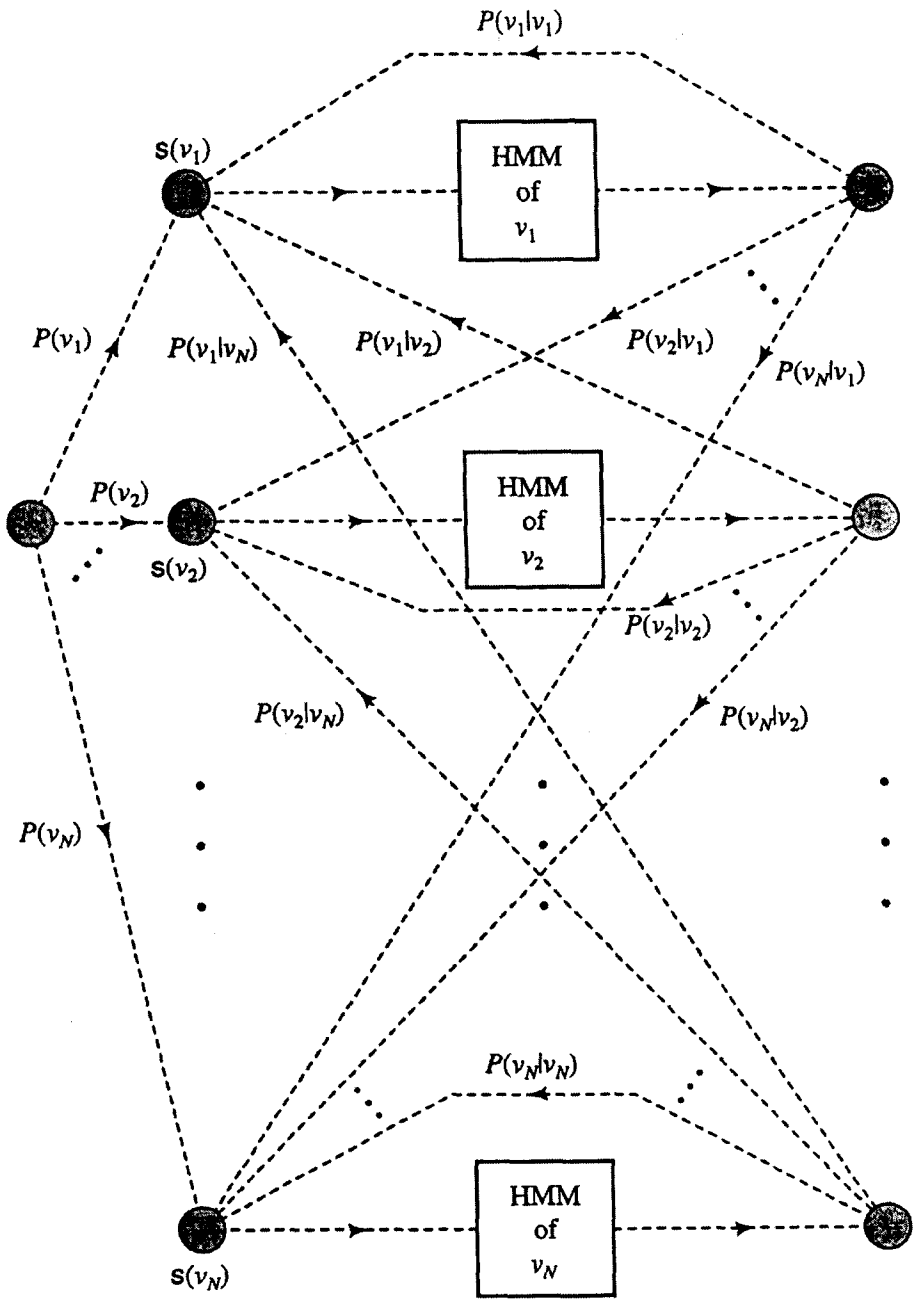


Figure 2.3. Composite model of speech generation when the generated words are assumed to depend only on the identity of the preceding word (bigram language model) [Jelinek].

As stated above, our language composite model is a huge HMM that *ties* together the word HMMs, as could be seen if we replace the boxes in the figures above for their corresponding HMMs. In this way, the Viterbi algorithm will find the most likely path through such huge HMMs. The recognized word string will be the specified by the sequence of word models yielded by the path Viterbi search ends with.

We present next, as in [Acero], the Forward Algorithm to better introduce Viterbi's, as the latter may be seen as a modification of the former.

The Forward Algorithm

The Forward Algorithm solves the problem of evaluating how well a given HMM matches a given observation sequence. Specifically, it aims to find the probability $P(\mathbf{X}|\Phi)$, given a model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, ..., X_T)$. In this way, it could be used in pattern recognition tasks, since this likelihood can be used to compute posterior probability $P(\Phi|\mathbf{X})$, and the HMM with highest posterior probability can be chosen to be the best pattern for the observation sequence.

Intuitively, the straightforward way of calculating $P(\mathbf{X}|\Phi)$ of the observation sequence $\mathbf{X} = (X_1, X_2, ..., X_T)$, given the HMM Φ, is to add the probabilities of all possible state sequences as

$$P(\mathbf{X}\,|\,\mathbf{\Phi}) = \sum_{all\ \mathbf{S}} P(\mathbf{S}\,|\,\mathbf{\Phi}) P(\mathbf{X}\,|\,\mathbf{S},\mathbf{\Phi}) \tag{18}$$

This means that we will need to list all state sequences **S** possible that produce observation sequence **X**, then we find the probability of each path **S** as the product of the likelihood of the state sequence (first factor in (18)) and the joint output probability (second factor) up the path, and then add all path probabilities [Acero].

The first factor in (18), the state sequence probability, can be expanded for a sequence $\mathbf{S} = (s_1, s_2, ..., s_T)$, where s_1 is the starting state, and T is the length of the sequence, as

$$P(\mathbf{S}\,|\,\mathbf{\Phi}) = P(s_1\,|\,\mathbf{\Phi}) \prod_{t=2}^{T} P(s_t\,|\,s_{t-1},\mathbf{\Phi}) = \pi_{s_1} a_{s_1 s_2} ... a_{s_{T-1} s_T} \tag{19}$$

which applies the Markov assumption.

In the same manner, the second factor, the joint output probability up the path can be expanded by applying the output independence probability as

$$\begin{aligned} P(\mathbf{X}\,|\,\mathbf{S},\mathbf{\Phi}) &= P(X_1^T\,|\,S_1^T,\mathbf{\Phi}) \\ &= \prod_{t=1}^{T} P(X_t\,|\,s_t,\mathbf{\Phi}) \\ &= b_{s_1}(X_1) b_{s_2}(X_2) ... b_{s_T}(X_T) \end{aligned} \tag{20}$$

Substituting (19) and (20) into (18), we have

$$\begin{aligned} P(\mathbf{X}\,|\,\mathbf{\Phi}) &= \sum_{all\ S} P(\mathbf{S}\,|\,\mathbf{\Phi}) P(\mathbf{X}\,|\,\mathbf{S},\mathbf{\Phi}) \\ &= \sum_{all\ S} \pi_{s_1} b_{s_1}(X_1) a_{s_1 s_2} b_{s_2}(X_2) ... a_{s_{T-1} s_T} b_{s_T}(X_T) \end{aligned} \tag{21}$$

which means that, after enumerating all possible state sequences with length $T+1$, for any given sequence, we start at state s_1 with probability π_{s1}, and step to the next state, in general,

from s_{t-1} to s_t generating observation X_t with probability $b_{s_t}(X_t)$ all up the path until the last transition.

But according to the interpretation above, evaluation of (21) will require enumeration of $O(N^T)$ possible state sequences, which indeed results in exponential computational complexity [Acero].

The artifice of the *forward algorithm* to avoid such an expense is to store intermediate results and reuse them in subsequent state-sequence calculations. As the HMM assumption involves only the actual observation X_t and state s_t and only one state before s_{t-1}, let us define the forward probability as

$$\alpha_t(i) = P(X_1^t, s_t = i \mid \Phi) \tag{22}$$

which is the probability that the HMM is in state i and has generated all partial observations from the starting state up to state t (namely, X_1, X_2, ..., X_t).

The following algorithm [Acero] shows how $\alpha_t(i)$ can be calculated inductively with a *time-synchronous* recursive fashion often illustrated in a *trellis* (see [Acero] for a thorough example and illustration). So, let us state the *forward algorithm*:

1. Initialization

$\alpha_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$

2. Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t) \qquad 2 \le t \le T;\ 1 \le j \le N \tag{23}$$

3. Termination

$P(\mathbf{X} \mid \Phi) = \sum_{i=1}^{N} \alpha_T(i)$ $\qquad P(\mathbf{X}|\Phi) = \alpha_T(s_F)$ if a final state is required.

Now, the complexity of the calculations reduces to $O(N^2T)$, because of the partially computed (forward) probabilities along the trellis.

The Viterbi Algorithm

In continuous speech recognition, as stated before, finding the best path or state sequence is the fundamental problem in the (properly called) recognition or decoding stage. This decoding problem [Acero][Jelinek] may be restated as follows: given a HMM model Φ and a sequence of observations $\mathbf{X} = (X_1, X_2, ..., X_T)$, what is the most likely state sequence $\mathbf{S} = (s_0, s_1, s_2, ..., s_T)$ in the model that caused the observations?

The forward algorithm, calculates the probability that an HMM generates an observation sequence by adding the probabilities of all possible paths without providing the best state sequence path. Solving the question above uncovers the hidden state sequence and is the task of the Viterbi search.

Since the state sequence is hidden in the HMM structure, we ought to find the state

sequence that has the highest probability of being tracked while producing the corresponding observations. This means that we are looking for the state sequence $\mathbf{S} = (s_0, s_1, s_2, ..., s_T)$ that maximizes $P(\mathbf{S},\mathbf{X}|\Phi)$.

Let us define the best path probability as

$$V_t(i) = P(X_1^t, S_1^{t-1}, s_t = i \,|\, \mathbf{\Phi}) \tag{24}$$

where $V_t(i)$ is the probability of the most likely state sequence at time t, which has produced the observation X_1^t (namely, $X_1, X_2, ..., X_t$) and ends in state i. Instead of summing up probabilities from different paths coming to the same destination state, as in the forward algorithm, the Viterbi algorithm chooses and remembers the best path. The Viterbi algorithm states an inductive process similar to the forward algorithm that achieves this best-sequence finding:

1. Initialization

$V_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$

$B_1(i) = 0$

2. Induction

$$V_t(j) = \underset{1\le i\le N}{Max}\left[V_{t-1}(i)a_{ij}\right] b_j(X_t) \qquad 2 \le t \le T;\ 1 \le j \le N \tag{25}$$

$$B_t(j) = \underset{1\le i\le N}{Arg\ max}\left[V_{t-1}(i)a_{ij}\right] \qquad 2 \le t \le T;\ 1 \le j \le N \tag{26}$$

3. Termination

The best score = $\underset{1\le i\le N}{Max}\left[V_t(i)\right]$

$$s_T^* = \underset{1\le i\le N}{Arg\ max}\left[B_T(i)\right]$$

4. Backtracking

$s_t^* = B_{t+1}(s_{t+1}^*) \qquad t = T\text{-}1, T\text{-}2, ..., 1$

$\mathbf{S}^* = (s_1^*, s_2^*, ..., s_T^*)$ is the best sequence

The complexity of this Viterbi algorithm is the same as that of the forward algorithm, $O(N^2T)$. The computations also have a time-synchronous recursive fashion from left to right, that can be illustrated better in a trellis (see, also, [Acero] for an example and illustration).

2.2.5 HMM Parameters Estimation

The Baum-Welch Algorithm

Finally we present a solution to the very important issue of estimating the HMM parameters $\Phi = (\mathbf{A},\mathbf{B},\pi)$ to accurately describe the observation sequences. We follow the mathematical development found in [Acero], and refer the reader to the source for further understanding of the algorithm schematically presented here.

Let us define the backward probability as

$$\beta_t(i) = P(X_{t+1}^T \mid s_t = i, \Phi) \tag{27}$$

where $\beta_t(i)$ is the probability, similar to the forward probability, of producing partial observation $X_{t+1}^T = (X_{t+1}, X_{t+2}, ..., X_T)$, from $t+1$ to the end, given that the HMM is in state i at time t. This backward probability can be calculated inductively as follows.

1. Initialization

$$\beta_T(i) = 1/N \qquad 1 \le i \le N$$

2. Induction

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right] \qquad t = T\text{-}1, T\text{-}2, ..., 1;\ 1 \le i \le N \tag{28}$$

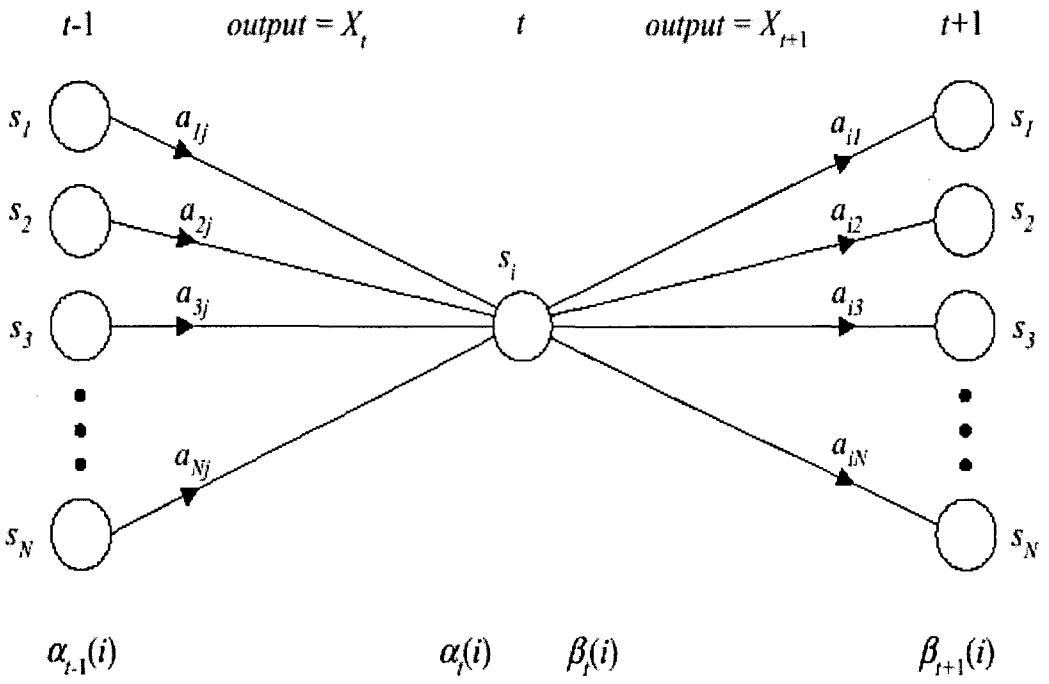


Figure 2.4. The relationship between the backward and forward probabilities at three different adjacent times in the Baum-Welch algorithm [Acero].

Figure 2.4 illustrates the relationship between the forward probability α and the backward probability β. α is calculated from left to right, and β from right to left, but both recursively.

Now, the probability of taking the transition from state i to state j at time t, given the model and observation sequence, is defined as

$$\begin{aligned}\gamma_t(i,j) &= P(s_{t-1}=i, s_t=j \mid X_1^T, \boldsymbol{\Phi}) \\ &= \frac{P(s_{t-1}=i, s_t=j, X_1^T \mid \boldsymbol{\Phi})}{P(X_1^T \mid \boldsymbol{\Phi})} \\ &= \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}\end{aligned} \tag{29}$$

which can be better illustrated in Figure 2.5.

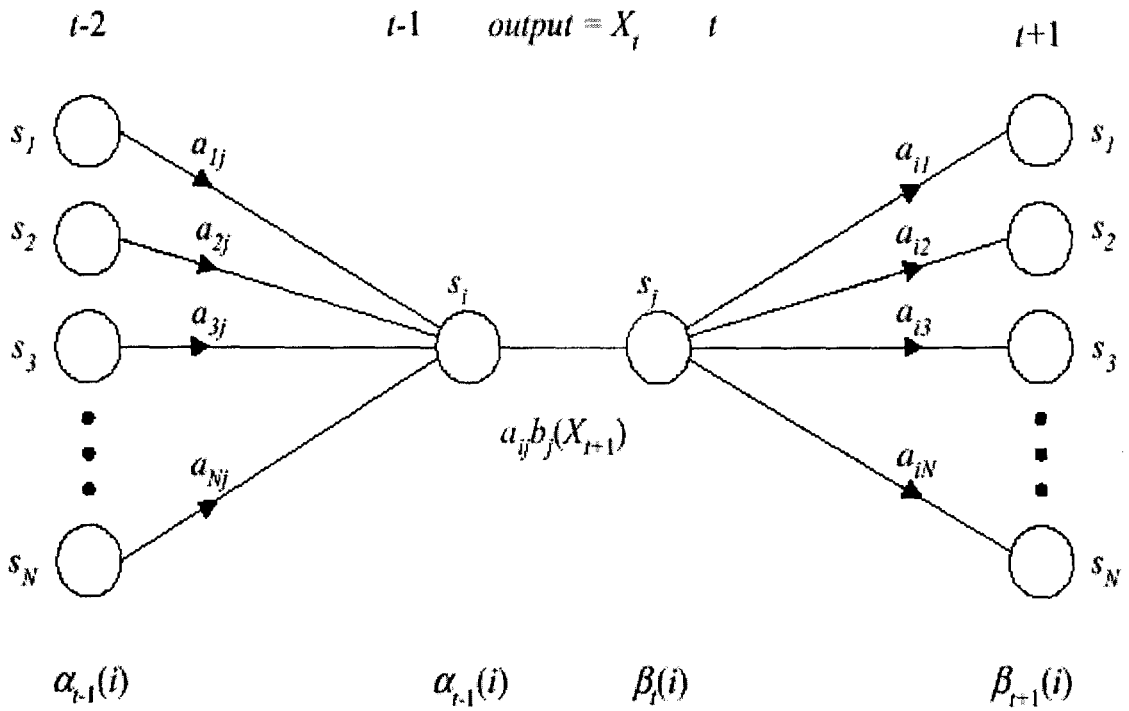


Figure 2.5. Illustration of the operations required for the computation of the probability of taking the transition from state i to state j at time t.

The HMM parameters $\Phi = (\mathbf{A},\mathbf{B},\pi)$ can be refined through iterations by maximizing the probability $P(\mathbf{X}|\Phi)$ for each iteration. Thus, $\boldsymbol{\Phi}$ will define the new HMM parameters provided from the HMM Φ in the previous iteration. In this way, the maximization process is equivalent to maximizing the Q-function (according to the EM algorithm found in [Acero]).

$$Q(\Phi,\hat{\Phi}) = \sum_{all\ S} \frac{P(\mathbf{X},\mathbf{S}\,|\,\Phi)}{P(\mathbf{X}\,|\,\Phi)} \log P(\mathbf{X},\mathbf{S}\,|\,\hat{\Phi}) \tag{30}$$

where $P(\mathbf{X},\mathbf{S}|\Phi)$ and $\log P(\mathbf{X},\mathbf{S}\,|\,\hat{\Phi})$ can be expressed as

$$P(\mathbf{X},\mathbf{S}\,|\,\Phi) = \prod_{t=1}^{T} a_{s_{t-1}s_t} b_{s_t}(X_t) \tag{31}$$

$$\log P(\mathbf{X},\mathbf{S}\,|\,\Phi) = \sum_{t=1}^{T} \log a_{s_{t-1}s_t} + \sum_{t=1}^{T} \log b_{s_t}(X_t) \tag{32}$$

and with this, (30) can be restated as

$$Q(\Phi,\hat{\Phi}) = Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{a}}_i) + Q_{\mathbf{b}_j}(\Phi,\hat{\mathbf{b}}_j) \tag{33}$$

where

$$Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{a}}_i) = \sum_i \sum_j \sum_t \frac{P(\mathbf{X}, s_{t-1}=i, s_t=j\,|\,\Phi)}{P(\mathbf{X}\,|\,\Phi)} \log \hat{a}_{ij} \tag{34}$$

$$Q_{\mathbf{a}_i}(\Phi,\hat{\mathbf{b}}_i) = \sum_i \sum_j \sum_{t\in X_t=o_k} \frac{P(\mathbf{X}, s_t=j\,|\,\Phi)}{P(\mathbf{X}\,|\,\Phi)} \log \hat{b}_j(k) \tag{35}$$

Thus, subject to probability constraints, the maximization procedure on $Q(\Phi,\hat{\Phi})$ can be achieved by maximizing the individual terms separately. The constraints are the following.

$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall\, i \tag{36}$$

$$\sum_{k=1}^{M} b_j(k) = 1 \qquad \forall\, j \tag{37}$$

Furthermore, the terms in (34) and (35) are of the form

$$F(x) = \sum_i y_i \log x_i \tag{38}$$

where $\sum_i x_i = 1$

Lagrange multipliers grant (38) to achieve maximum value at

$$x_i = \frac{y_i}{\sum_i y_i} \tag{39}$$

With all this, we use (38) and (39) to abstract the models estimates from (34) and (35) as

$$\hat{a}_{ij} = \frac{\frac{1}{P(\mathbf{X}|\mathbf{\Phi})}\sum_{t=1}^{T} P(\mathbf{X}, s_{t-1}=i, s_t=j|\mathbf{\Phi})}{\frac{1}{P(\mathbf{X}|\mathbf{\Phi})}\sum_{t=1}^{T} P(\mathbf{X}, s_{t-1}=i|\mathbf{\Phi})} = \frac{\sum_{t=1}^{T}\gamma_t(i,j)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\gamma_t(i,k)} \tag{40}$$

$$\hat{b}_j(k) = \frac{\frac{1}{P(\mathbf{X}|\mathbf{\Phi})}\sum_{t=1}^{T} P(\mathbf{X}, s_t=j|\mathbf{\Phi})\delta(X_t, o_k)}{\frac{1}{P(\mathbf{X}|\mathbf{\Phi})}\sum_{t=1}^{T} P(\mathbf{X}, s_t=j|\mathbf{\Phi})} = \frac{\sum_{t\in X_t=o_k}\sum_{i}\gamma_t(i,j)}{\sum_{t=1}^{T}\sum_{i}\gamma_t(i,k)} \tag{41}$$

The re-estimation formula (40) can be thought as the ratio between the expected number of transition from state i to state j and the expected number of transitions from state i. The output probability re-estimation formula (41) is the expected number of times the observation data emitted from state j with the observation symbol o_k, and the denominator is the expected number of times the observation data emitted from state j [Acero].

It is interesting to note that the initial probability $\hat{\pi}_i$ can be computed as a special case of the transition probability, even though is often fixed (to 1) for most speech recognition tasks.

We now present the algorithmic description of the Baum-Welch re-estimation process as found in [Acero].

1. Initialization: Choose an initial estimate Φ.
2. E-step: Compute auxiliary function $Q(\mathbf{\Phi},\hat{\mathbf{\Phi}})$ based on Φ (Eqs. (33), (34), (35)).
3. M-step: Compute $\hat{\mathbf{\Phi}}$ according to the re-estimation equations (40) and (41) to maximize the auxiliary Q-function.
4. Iteration: Set $\mathbf{\Phi}=\hat{\mathbf{\Phi}}$, repeat from step 2 until convergence.

Even though, the forward-backward (Baum-Welch) algorithm described refers to only one training sequence, it can be generalized to multiple training observations. See [Acero] for further extension and illustration of this algorithm.

2.2 Experimental Framework

2.2.1 The Carnegie Mellon University SPHINX-3 ASR System

CMU Sphinx is a large vocabulary, speaker independent speech recognition codebase and suite of tools [CMU Sphinx FAQ].

The language model describes the likelihood, probability, or penalty taken when a sequence or collection of words is seen. Sphinx 3 uses N-gram models, and usually N is 3, so they are trigram models, and these are sequences of three words. All the sequences of three words, two words, and one word are combined together using back-off weights in order to assign probabilities to sequences of words.

Acoustic models characterize how sound changes over time. Each phoneme or speech sound is modeled by a sequence of states and signal observation probabilities; distributions of sounds that you might hear (observe) in that state. Sphinx 3 is implemented using a 5-state phonetic model; each phone model has exactly five states. At run-time, frames of the input audio are compared to the distributions in the states to see which ones the sound could have come from. Acoustic models which perform best are those that matched to the conditions they will be used in. That is, for example, English acoustic models work best for English, and telephone models work best on the telephone.

Context-independent phones (CI-phones) are modeled using data from many different context, and triphones are phones that take into account left and right context in the modeling. Sphinx 3 uses tied states to reduce the total number of states in the system. The default English acoustic models contain 6,000 states, or senones in historic CMU nomenclature, that are shared among all the 5-state phone models. These states share their parameter weights with other states also. In our implementation we configured Sphinx to use 500 senones.

2.2.2 The Speech Corpus

The different front end proposals presented in this work will be evaluated by performing recognition experiments using two standard speech databases, namely TIMIT and CSDigit. TIMIT was designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of English language automatic speech recognition systems. It was prepared at the National Institute of Standards and Technology (NIST) with sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). CSDigit is a continuous digit database with utterances of more than 1500 Spanish language speakers. We explain briefly the contents and general characteristics of these corpora:

TIMIT

TIMIT contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States [TIMIT]. Appendix 1 provides a table describing each of these regions. The corpus has microphone quality, phonetically balanced, phonetically labeled, read utterances, and has been divided for training (4260

utterances) and testing (1680 utterances). Its lexicon adds to 6229 vocabulary words, and it was sampled at a 16 kHz rate as used in our experiments.

Continuous Spontaneous Digit Utterances (CSDigit)

This is a continuous digit utterances database of more than 1500 Spanish language speakers. It contains a small lexicon (about 75 vocabulary words) of numbers and quantities spontaneously concatenated. It may be divided in about 6000 utterances for training and 3000 utterances for testing, as we did. Its original sampling rate is 8 kHz which we kept untouched for our experiments.

CHAPTER 3

FRAMEWORK 2: A FEATURE TOUR OF SPEECH SIGNAL PROCESSING

The most widely used features in speech recognition tasks are linear prediction parameters and Mel-cepstral coefficients. Both provide information regarding the spectral analysis of the speech signal and derive their features from a smooth representation of the energy content of the signal [Rabiner]. Since Mel-cepstral coefficients have shown to be superior to LPC coefficients [Quatieri], and because our baseline front-end consists of an implementation for extracting such mel parameters, we will devote the first section of this chapter to a not so thorough but sufficient explanation of cepstral analysis. On the other hand, a linear prediction coding explanation is out of the scope of this work since it is not related neither to our baseline nor to our proposed front-end. The reader is encouraged to search further in the references. Actually, there are very good sources (see, for example, [Harrington], [Deller]) that present a complete and strict treatment of both techniques.

The second section of this chapter introduces wavelet analysis. We do not present an exhaustive treatment of all wavelet related issues, not even what may be considered cardinal, but restrict ourselves to an introductory view of the tools used in this work. [Quatieri] might be an excellent start for speech people, while [Wickerhauser] would be better for implementation-avid readers. A strict mathematic development of the theory is found in [Mallat], [Daubechies] and [Meyer]. A more signal processing point of view of the subject would be [Vetterli] and [Rioul], and very good enlightening applications are explained in [Goswami].

Finally, the last section restates the basic signal-to-noise theory (as introduced by [Deller]), which is used in this work as explained in the next chapter.

3.1 Cepstral Analysis of Speech Signals

3.1.1 Source Filter Model

That speech sounds are produced by the action of a filter (the vocal tract) on a sound source (either the glottis or some other constriction within the vocal tract) is the main idea in the source filter model. Fundamental to the model is the assumption that the filter and the source are independent, i.e. the properties of the filter can be modified without changing the properties of the source and viceversa. This assumption may not be strictly true in all cases but in practical terms provides us with a very useful and sufficiently accurate model of speech production [Harrington].

In voiced speech, the vibration of the vocal folds in response to airflow from the lungs corresponds to the source of the model. This vibration is periodic and would consist of a series of broad spikes if it could be examined independently of the properties of the vocal tract, which change it's spectral shape.

The spectrum of the glottal source is made up of a number of frequency spikes corresponding to the harmonics of the fundamental frequency of vibration of the vocal folds, which is inversely proportional to the pitch period. With increasing frequency, the spectrum decreases in amplitude at a rate of -12dB per octave approximately. In other words, for each doubling in frequency, the amplitude of the spectrum decreases by around 12dB.

When the frequency of vibration of the vocal folds is increased, the gap between the frequency spikes in the glottal source spectrum is widened, since these spikes occur at multiples of the fundamental frequency (or pitch).

The source frequency of an average male voice is around 100Hz, female and child voices are typically higher in pitch; around 200 Hz for an average female voice and 200 to 300 Hz for children [Harrington].

The sound source in unvoiced speech is not a cuasi-periodic vibration but rather vibrations are caused by turbulent airflow due to a constriction in the vocal tract. The various shapes which may take the vocal tract is a matter of speech sounds classification (which is presented in almost any speech book and not treated here).

The sound created via a constriction in the vocal tract may be described as a noise source which contains no dominating periodic component. This leads to a relatively flat spectrum meaning that every frequency component is represented (almost) equally.

In summary, the source filter model describes the process of speech production in terms of two independent contributions, the sound source and the vocal tract filter. As a model of production, this allows us to synthesize sounds, but the model can also be used to aid the analysis of speech sounds. The key is to notice the importance of the independence of the source and filter; this means that each makes a separate contribution to the characteristic features of the resulting speech sounds. The source, in voiced speech, is responsible for the pitch of the sound while the vocal tract filter is responsible for the location of the formants and the overall spectral shape. In a real speech spectrum the overall filter shape and the

location of the formants is often drowned out by the effects of the source spectrum. If it were possible to remove the source effects then the two spectra could be studied separately to give a more accurate picture of the precursors of the speech sound [Harrington].

Removing the effects of the source from a spectrum also has the effect of removing many sources of variability from the signal since many of these are realized in the source rather than in the filter. Hence removing the source can help make, for example, the vowel sounds more distinct from one another and could make vowel sounds from different speakers look more similar [Harrington]. An enlightening application example and discussion can be found in [Sundberg] where the acoustics of singing voice are explained.

3.1.2 Cepstral Analysis

Cepstral analysis relies on the observation that a logarithmic speech spectrum is made up from the source and filter spectra added together. To understand this, recall that filtering in the frequency domain is achieved by multiplying spectra together. Multiplication corresponds to adding logarithms and so a filtered spectrum can be derived by adding logarithmic (or dB) spectra. The procedure for cepstral analysis is to take the inverse Fourier transform of the dB spectrum, thus converting the signal back into the time domain. This time domain signal is not a regular acoustic signal, since it was derived from the logarithmic spectrum (somehow, for this reason it is called a cepstrum, kind of a *backward* spectrum). The important property of the cepstrum is that since the log-spectrum is the sum

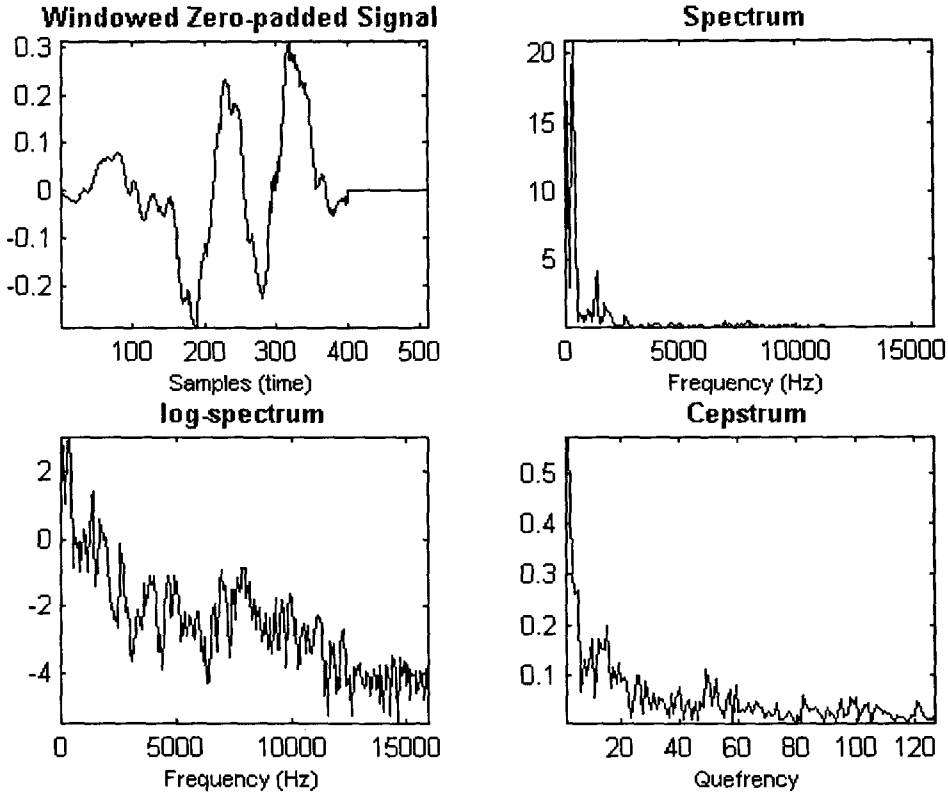


Figure 3.1. Cepstrum from a 25 ms window corresponding to an /e/ vowel.

of two spectra, so the cepstrum is the sum of two components corresponding to the source and the filter. Usefully the two components are separated: the lower end of the cepstrum corresponds to the filter while the higher end (or rather the middle of the reflected cepstrum) corresponds to the source. An illustration of the process of applying cepstral analysis is shown in Figure 3.1.

A further operation on the cepstrum is to remove the central part of the reflected cepstrum, the part that corresponds to the source, and perform a Fourier transform to again generate a frequency domain version of the signal. Since the effect of the source has been removed from this spectrum, it shows the characteristics of the vocal tract filter. This is manifested as a much smoother spectrum than the original; the degree of smoothing depends upon the number of cepstral coefficients removed prior to the final Fourier transform.

The technique of cepstral analysis is described in more detail in [Harrington]. The technique can be used to generate smoothed spectra which show the characteristics of the filter as described above. The cepstrum can also be used as a means of determining the fundamental frequency of voiced speech since the part of the cepstrum corresponding to the source is often manifested as a single spike. The location of this spike gives a measure of the frequency of the source signal.

This is just one of a family of techniques that seek to separate the source and filter in speech analysis. These techniques are useful in that they enable us to remove variability from the speech signal which is due to the source and concentrate our analysis on the vocal tract filter as implemented for speech recognition.

3.1.3 The Mel-Cepstrum

After Davis and Mermelstein [Davis], the cepstrum or cepstral coefficients became a better choice than linear prediction parameters because its ability to smooth the logarithmic spectrum of speech signals and its improvement of recognition performance. In this section we provide a sketch of the thorough explanation of the subject found in [Deller].

In studies of human auditory perception a *mel* is a unit of measure of perceived pitch or frequency of a tone. Instead of linearly corresponding to the physical frequency of a tone, it fits the human auditory system perception of that pitch. An approximation suggested by Fant (1959) is

$$F_{mel} = \frac{1000}{\log\ 2}\left[1+\frac{F_{Hz}}{1000}\right], \qquad (1)$$

in which F_{mel} is the perceived frequency in *mels* and F_{Hz} is the real frequency in Hz.

As stated by Deller *et al.*, it has been found that the perception of a particular frequency by the human auditory system is influenced by energy in a critical band of frequencies

around that particular frequency. The bandwidth of a critical band varies with frequency, beginning approximately at 100 Hz for frequencies below 1 kHz, and then increasing logarithmically above 1 kHz. So, instead of simply using the mel-distributed log magnitude frequency components to compute the (short-time real) cepstrum, it has been suggested using the log total energy in critical bands around the mel frequencies as inputs to the final IDFT block. The process is illustrated in Figure 3.2, were *Y(i)* denotes the log total energy in the *i*th critical band.

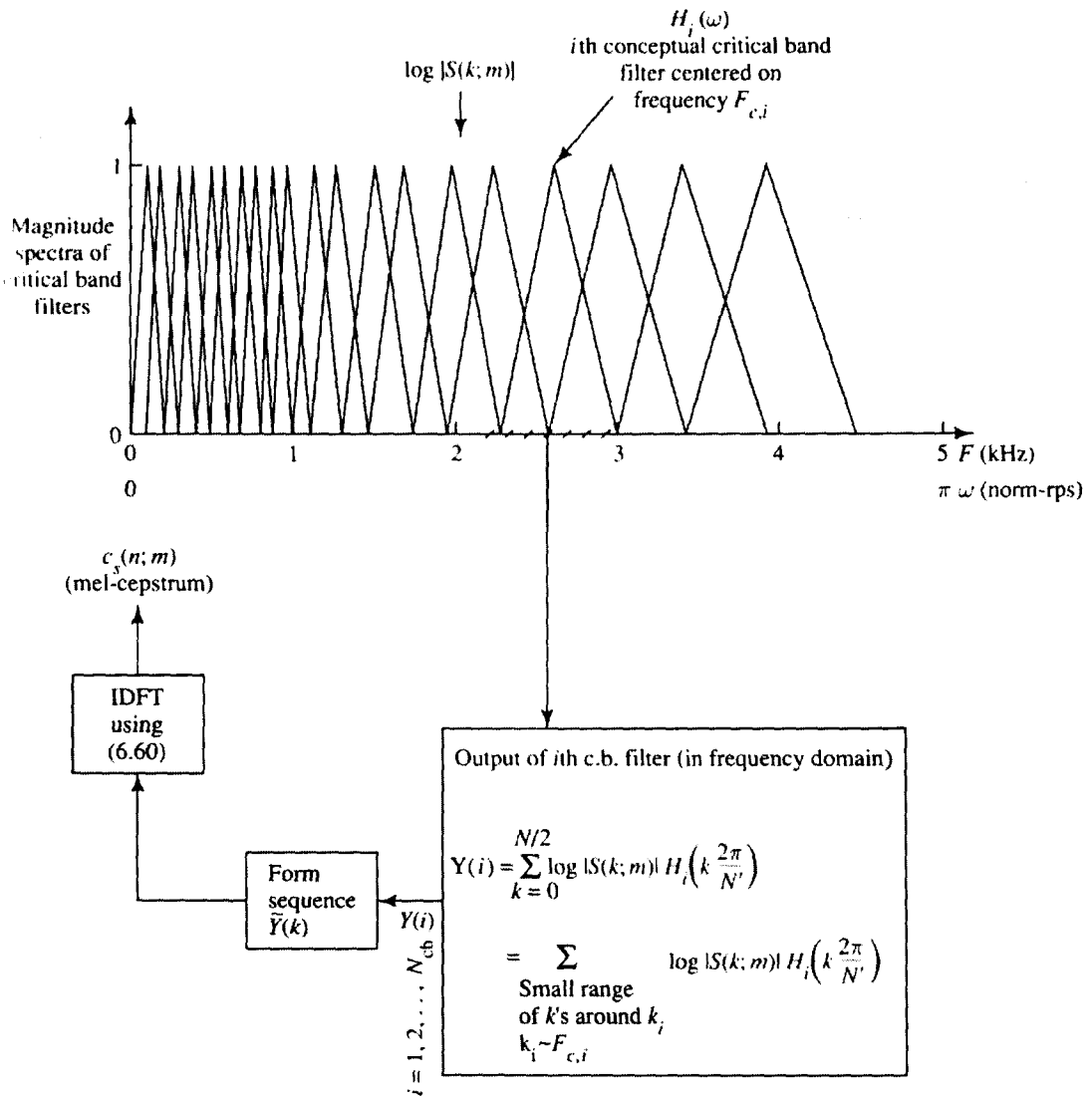


Figure 3.2. Use of critical band filters to compute the mel-cepstrum [Deller *et al.*].

A thorough discussion and critical analysis of the cepstrum can be found in [Deller]. Another excellent explanation of homomorphic analysis is presented by [Quatieri]. The specific structure and computation of the cepstral features used in our experiments is sketched in chapter 4.

3.2 Wavelets

3.2.1 An Intuitive Description of the Multiresolution Concept

The basic idea in multiresolution analysis is successive approximation. A signal may be decomposed as a coarse approximation plus a prediction error (that might be thought as the "detail" of the signal) which is the difference between the original signal and a prediction based on the coarse version. If we add back the prediction error (detail) to the prediction (approximation) immediate reconstruction of the signal is achieved. This scheme is intimately related to wavelet decomposition and might be iterated on the coarse approximation, as we will see, to perform a (dyadic tree) discrete wavelet transform.

3.2.2 The Uncertainty Principle

It is known that a limitation of the STFT is that of its window w(t) fixed duration. With a particular window w(t) we cannot accurately observe impulse like events and closely spaced long lasting tones [Quatieri]. The uncertainty principle limits time-frequency resolution such that the product of duration D(x) and bandwidth B(x) of a signal x(t) must exceed a constant value. This is,

$$D(x)B(x) \geq \frac{1}{4}$$

This formula quantitatively states the tradeoff peculiarity of the spectrogram (and other time-frequency transforms). And, if we deal with discriminating speech signal events, as closely spaced harmonic frequencies with time occurrence glottal pulses and short duration plosive bursts, this limitation gains particular importance [Quatieri]. A thorough treatment and definition of this principle can be found in [Mallat], [Quatieri], [Vetterli] and others. The search for minimizing the limitations of the uncertainty principle is found in the signal processing literature and many related papers (see, for example, [Mallat] or [Rioul]).

3.2.3 Wavelet Transform

Defining an "ideal" time-frequency transform has been a research subject since long ago and lately with great enthusiasm as the recent development of wavelet transforms found to be useful in many fields and practical applications.

In the remaining sections of this chapter we sketch the insightful approach presented by [Quatieri] sticking to the terminology found in [Rioul]. As our purpose is not to present a strict and thorough description of the subject, we refer the reader to [Mallat], [Rioul] and [Vetterli] for a more extensive treatment on wavelets.

So to speak, a way of going around the limitations imposed by the uncertainty principle is to compute many spectrograms with different analysis window durations. As illustrated in Figure 3.3, the wavelet transform can be thought of as a collage of pieces of spectrograms calculated using analysis windows of different duration and thus different time-frequency resolutions [Quatieri].

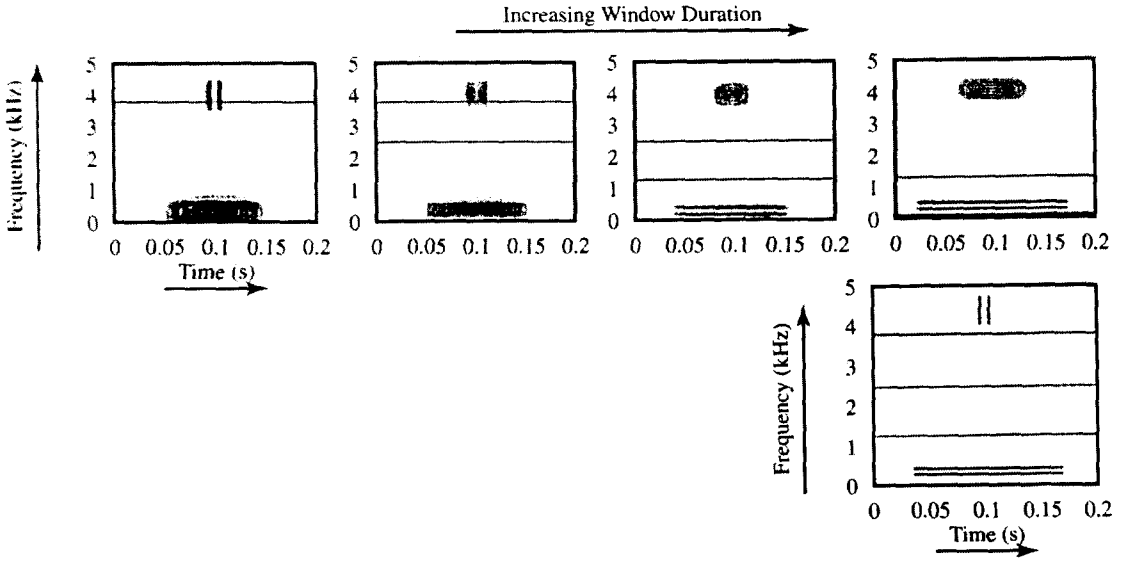


Figure 3.3. The wavelet transform as a collage of spectrograms. A short window at high frequency gives good time resolution, while a long window at low frequency gives good frequency resolution. The lower right panel is obtained by piecing together regions of the upper four panels [Quatieri].

As may be observed, short windows are used to calculate the (wide-band) spectrogram and then the high frequency piece is extracted from it as to keep the good time localization part yielded by it. In the same perspective, as we decrease in frequency longer windows are used to calculate the spectrogram and then the lower frequency piece is extracted from it as to keep the (better frequency resolution) part corresponding to this step. This ends when the longer window is used to calculate the (narrow-band) spectrogram and then we extract the good frequency resolution low frequency piece and put it all together as illustrated in Figure 3.3. The frequency changing time-frequency resolution cell in contrast to the fixed time-frequency resolution cell of the STFT is shown in Figure 3.4. From this approach, we see that we don't defeat the uncertainty principle because the area of the cell is fixed. Instead, we benefit of its changing dimensions when moving around the time-frequency plane. The mathematics of this concept are found in the multiresolution analysis literature developed in the 1980's by different fields including speech, image and seismic processing [Quatieri].

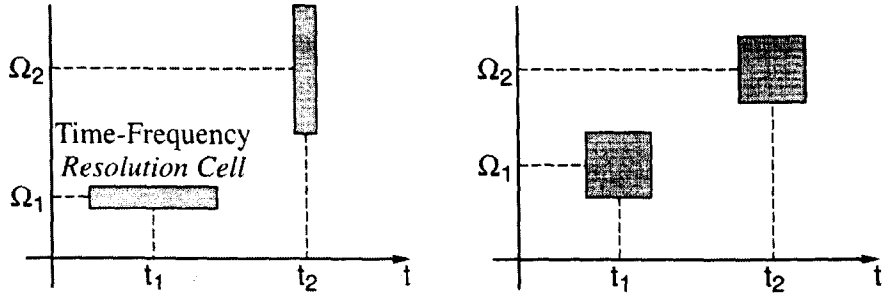


Figure 3.4. Adaptation of window size to frequency in the wavelet transform (left panel) in contrast to a fixed window in the STFT (right panel) [Quatieri].

The notion of adapting time resolution to frequency is formalized by the continuous wavelet transform (CWT). To present a brief mathematical approach, a set of functions that yield the time-scaled and shifted versions of a prototype $\psi(t)$, which can be thought of as a band-pass function, is defined as

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right), \tag{2}$$

where $\psi(t)$ is the basic (or mother) wavelet, $\psi_{a,b}(t)$ are the related wavelets, b is the time shift, an a is the scaling factor, as illustrated in Figure 3.5. The factor $|a|^{-\frac{1}{2}}$ is used to ensure energy preservation [Rioul].

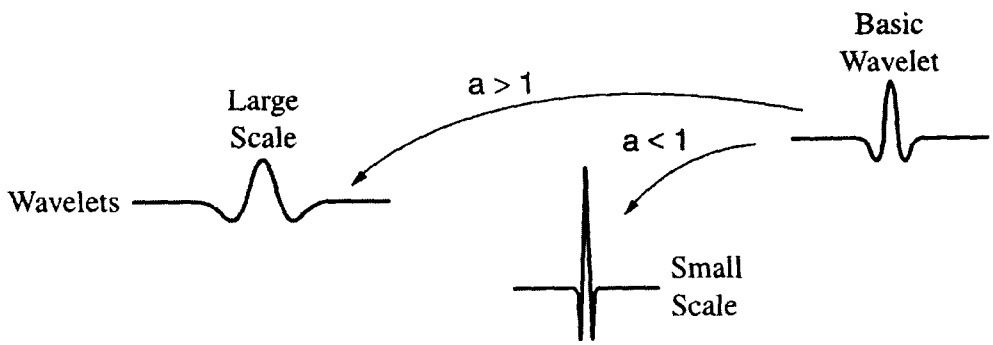


Figure 3.5. Schematic of a basic wavelet and its associated wavelets at different scales [Quatieri]

The continuous wavelet transform (CWT), as introduced originally by Goupillaud, Grossmann, and Morlet, is

$$CWT\{x(t);a,b\} = \int x(t)\psi^*_{a,b}(t)dt \tag{3}$$

where * stands for complex conjugation and the time t and the time-scale parameters (a, b) vary continuously. Equation (3) is a measure of "similarity" of $x(t)$ with $\psi(t)$ at different scales and time shifts [Quatieri]. We can see that with a scale factor $a < 1$, the basic wavelet is contracted and the resulting wavelet is convolved with $x(t)$. So, we may also see the wavelet transform CWT from a filtering point of view as

$$CWT\{x(t);a,b\} = \frac{1}{\sqrt{a}}\int x(t)\psi^*\left(\frac{t-b}{a}\right)dt = \frac{1}{\sqrt{a}}x(b) * \psi^*\left(-\frac{b}{a}\right) \tag{4}$$

where * denotes convolution. Here a smaller scale yields filters with a wider bandwidth. The quantity $|CWT|^2$ is referred as the scalogram.

If we rewrite (3) as

$$CWT\{x(t);a,b\} = \frac{1}{\sqrt{a}} \int x(at)\psi^*\left(t - \frac{b}{a}\right)dt \tag{5}$$

where we keep the (wavelet) filter $\psi(t)$ unscaled and scale $x(t)$, we get another interpretation of the wavelet transform. We may see (5) as a "zoom lens" at different time-scales. The concept of scale is related to time-scale, but it also varies inversely with frequency.

The wavelet transform, as the STFT, is also invertible under certain conditions. In fact, for a large set of basic wavelets, $\psi(t)$, $x(t)$ may be recovered from a superposition of wavelets. So, the inverse continuous wavelet transform (ICWT) is defined as

$$x(t) = \frac{1}{C_\psi} \iint CWT\{x(t);a,b\} \frac{1}{\sqrt{a}} \psi\left(\frac{t-b}{a}\right) \frac{dadb}{a^2} \tag{6}$$

if and only if the *admissibility condition* is satisfied, i.e.,

$$C_\psi = \int \frac{|\Psi(\omega)|^2}{|\omega|} d\omega \; < \; \infty. \tag{7}$$

A basis function interpretation is implied in (6); the CWT measures the projection of $x(t)$ onto the basis $\psi_{a,b}(t)$, which means that the CWT is the inner product of $x(t)$ and $\psi_{a,b}(t)$. A further explanation of this interpretation can be found in [Mallat] [Daubechies].

3.2.4 Wavelet Series (WS) and Discrete Wavelet Transform (DWT)

Wavelet series (WS) coefficients are sampled CWT coefficients. Time remains continuous but time-scale parameters (b, a) are sampled in the time-scale plane (b, a) [Rioul]. The WS coefficients (called discrete wavelet transform in [Quatieri] and other publications) are defined as

$$C_{j,k} = \int x(t)\psi^*_{j,k}(t)dt, \tag{8}$$

where

$$\psi_{j,k}(t) = \frac{1}{\sqrt{|a_j|}} \psi\left(\frac{t-b_k}{a_j}\right). \tag{9}$$

The WS coefficients represent the inner product of $x(t)$ with the discretized wavelet basis $\psi_{j,k}(t)$, which are the original wavelets sampled in scale and in shift. A usual (and by this time we could say "traditional" or even "standard") sampling in scale and shift is that of dyadic (or octave) sampling, where for each scale $a_j = 2^j$ for $j = 1, 2, 3, \ldots$ we shift at $b = k2^j$

for $k = 1, 2, 3...$ This may be considered a "natural" sampling because as the scale increases by a factor of two (this means the bandwidth of the wavelet decreases by a factor of two), the sampling rate of shift decreases by a factor of two (half the bandwidth requires half the sampling rate) [Quatieri]. Figure 3.6 provides a pictorial description of the stated above.

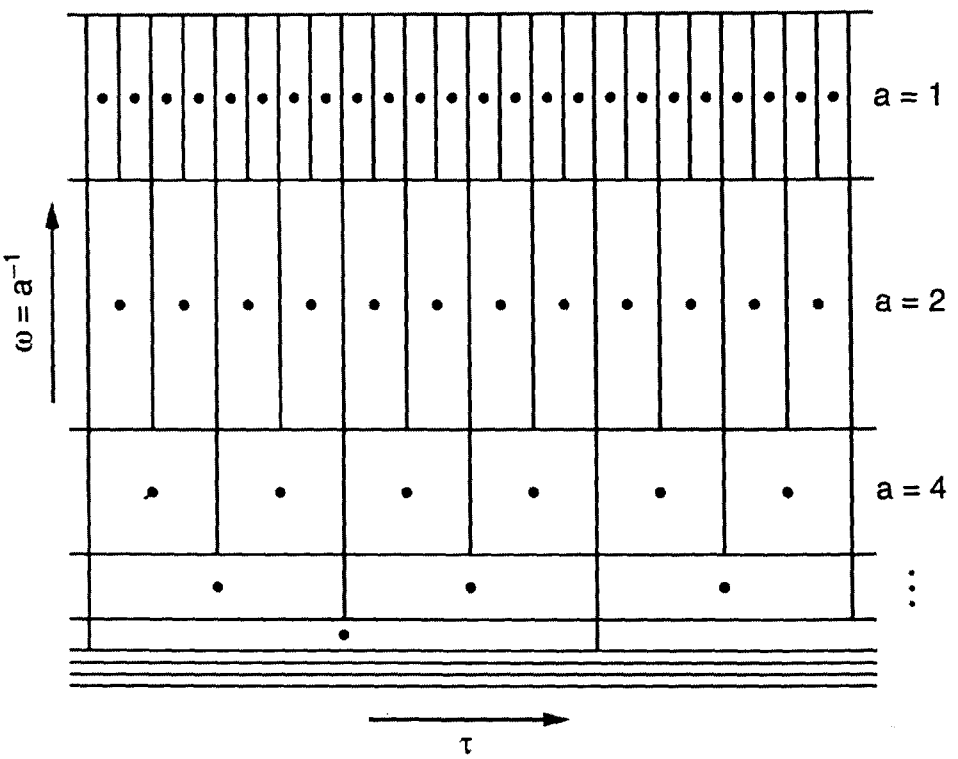


Figure 3.6. Sampling of scale and shift in a dyadic wavelet basis [Quatieri].

Indeed, a common definition of dyadic time-scale sampling WS coefficients is

$$C_{j,k} = CWT\{x(t); a = 2^j, b = k2^j\}, \quad j, k \in \mathbf{Z}, \tag{10}$$

in which case the wavelets are

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k). \tag{11}$$

A wavelet series decomposes a signal $x(t)$ onto a basis of continuous-time wavelets $\tilde{\psi}_{j,k}(t)$ as shown [Rioul].

$$x(t) = \sum_{j\in Z}\sum_{k\in Z} C_{j,k}\tilde{\psi}_{j,k}(t). \tag{12}$$

These reconstruction wavelets, as in (11), usually correspond to discretized parameters $a = 2^j$ (where j is called the "octave") and $b = k2^j$.

By several authors, the discrete wavelet transform (DWT) has been recognized as a natural wavelet transform for discrete-time signals (see references in [Rioul]). Both time and time-scale parameters (scale and shift) are discrete. The DWT is actually an octave-band filter bank as far as the structure of computations is concerned [Rioul]. In fact, the bandpass filters impulse response is $2^{-m/2}\psi(2^{-m}t-n)$ [Quatieri]. In Figure 8.7, we can see a comparison of the increasing filter bandwidths of dyadic wavelets and the uniform filter bandwidths of the STFT, as well as the corresponding filter impulse responses.

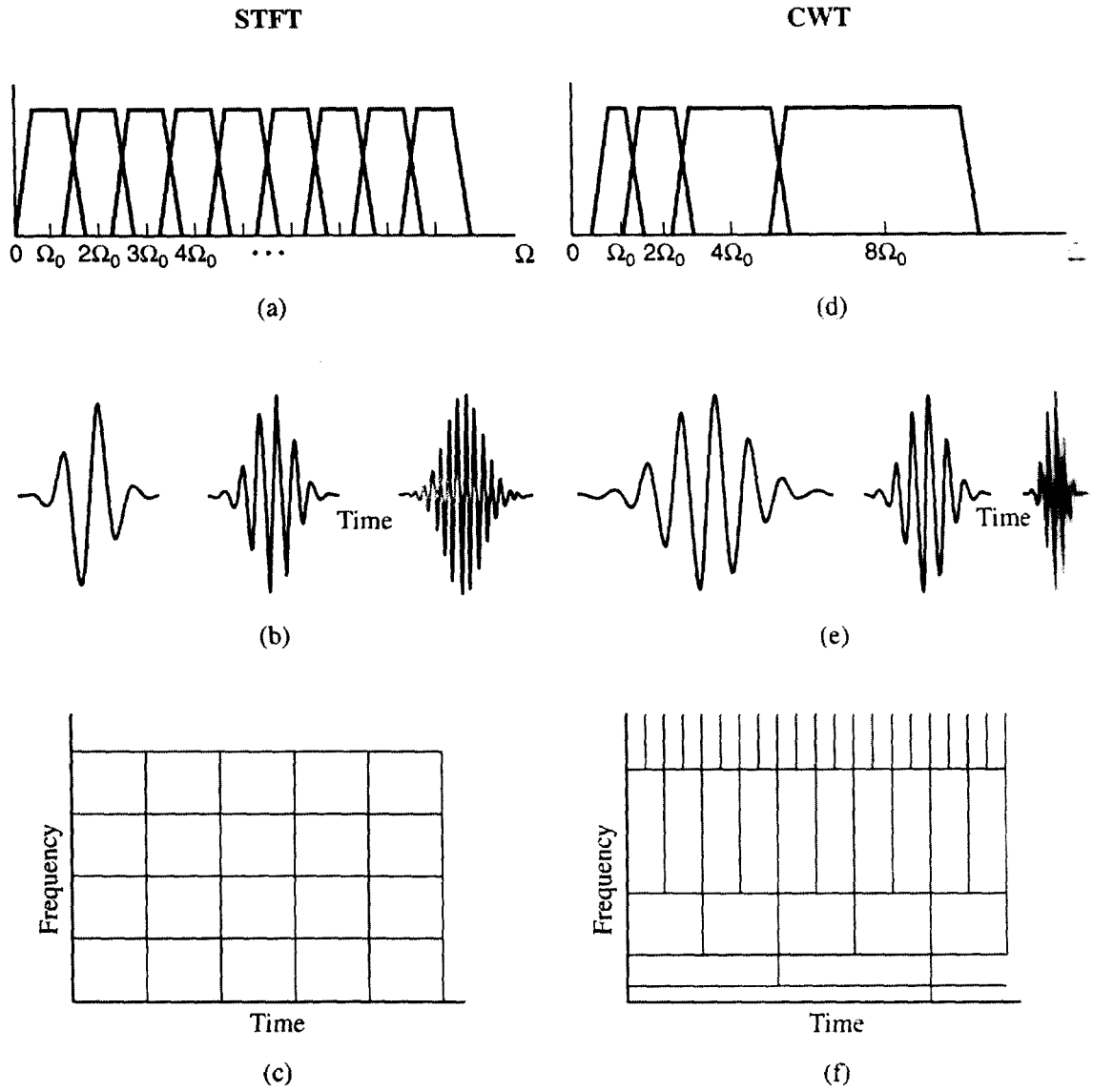


Figure 3.7. Comparison of the sampling requirements for the discrete STFT and the DWT from a filtering perspective. Panels (a) and (d) show the required filters in frequency, while (b) and (e) show their counterparts in time. The discrete STFT filters have constant bandwidth while the discrete dyadic wavelets have constant-Q bandwidth. Panels (c) and (f) give the respective time-frequency "tiles" that represent the essential concentration of the basis in the time-frequency plane [Quatieri].

The (DWT) filter bank can be easily implemented by repeated application of identical stages as shown in Figure 3.8. Each stage consists of a low-pass (by $P(\omega)$) and highpass (by $Q(\omega)$) decomposition of the signal followed by a downsampling by 2 . The sequences $p[n]$ and $q[n]$ are obtained from the basic discrete-time wavelet $\psi[n]$.

Several algorithms for computing the DWT have been known for some time, namely the Mallat algorithm [Mallat], and the "*à trous*" algorithm of Holschneider *et al.* [Holschneider]. For a proof of this algorithms and implementation we refer the reader to [Mallat]. We conform our description to a frugal presentation of the DWT.

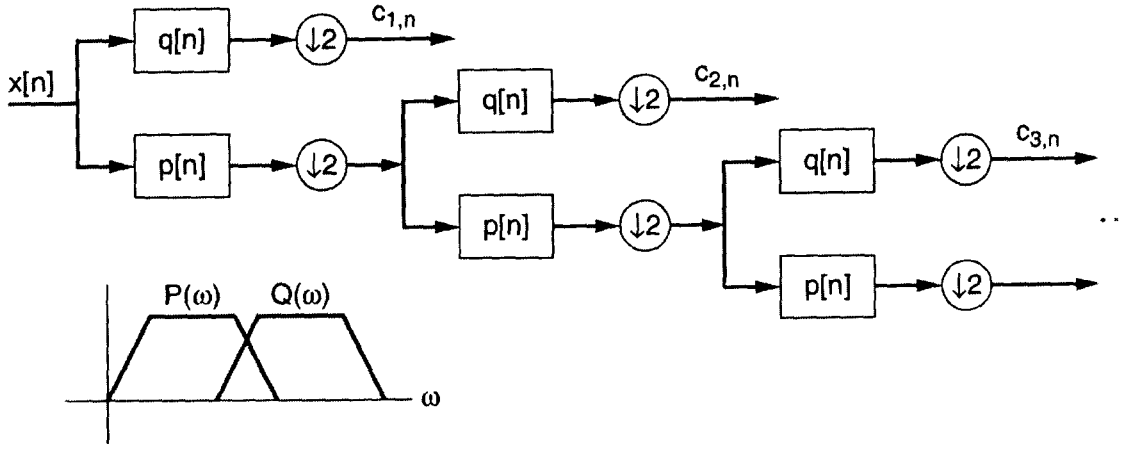


Figure 3.8. Iterative filtering implementation of DWT. A similar iterative structure exists for the inverse DWT [Quatieri].

The "standard" DWT whose coefficients are sampled over the dyadic grid $a = 2^j$, $b = k2^j$ in the time-scale plane yields a multiresolution decomposition of $x[n]$ on J octaves labeled by $j = 1, ..., J$ given by

$$x[n] = \sum_{j=1}^{+\infty} \sum_{k \in Z} c_{j,k} \tilde{h}_j[n - 2^j k] + \sum_{k \in Z} b_{J,k} \tilde{g}_J[n - 2^J k]. \tag{13}$$

The $\tilde{h}_j[n - 2^J k]$ are the *synthesis wavelets*, the discrete equivalents to $2^{-j/2} \tilde{\psi}(2^{-j}(t - 2^j k))$ [Rioul]. An additional low-pass term is used to ensure perfect reconstruction; the corresponding basis functions $\tilde{g}_J[n - 2^J k]$ are called (synthesis) "scaling sequences" ("scaling functions", as stated by Rioul and Duhamel, can be defined in a similar manner for wavelet series).

As explained in [Rioul], the DWT computes "wavelet coefficients" $c_{j,k}$ for $j = 1, ..., J$ and "scaling coefficients" $b_{J,k}$, given by

$$DWT\{x[n]; 2^j, k2^j\} = c_{j,k} = \sum_n x[n] h_j^*[n - 2^j k] \tag{14a}$$

and

$$b_{J,k} = \sum_n x[n] g_J^*[n - 2^J k], \tag{14b}$$

where the $h_j[n-2^J k]$ are the analysis discrete wavelets and the $g_J[n-2^J k]$ are the analysis scaling sequences.

It should be noted that in the filtering implementation shown in Figure 3.8, downsampling the highpass filter outputs at each stage (each stage representing different scales) gives the wavelet coefficients at different scales of the continuous-time function *x(t)* [Daubechies].

It can be proved that the inverse DWT reconstructs the signal from its coefficients by (13), which is conveys a similar structure as that in Figure 3.8. The condition for invertibility on the basic discrete-time wavelet $\psi[n]$, when dyadically sampled in shift and scale, is intimately related to the "perfect reconstruction" constraint on quadrature mirror and conjugate mirror digital filters [Quatieri]. A thorough discussion on this subject can be found in [Vaidyanathan].

3.2.5 Wavelet Packets

In the previous section we presented the DWT as an octave-band, tree-structured filter banks. Now we briefly generalize the concept to arbitrary tree structures beginning with a single block of two-channel filter bank where each filter is followed by a two-to-one downsampler, as in Figure 3.8. With this single blocks or stages a tree can be built up to a depth *J* with different shapes (or branches). In Figure 3.9 all the possible tree structures of depth less or equal to two are presented.

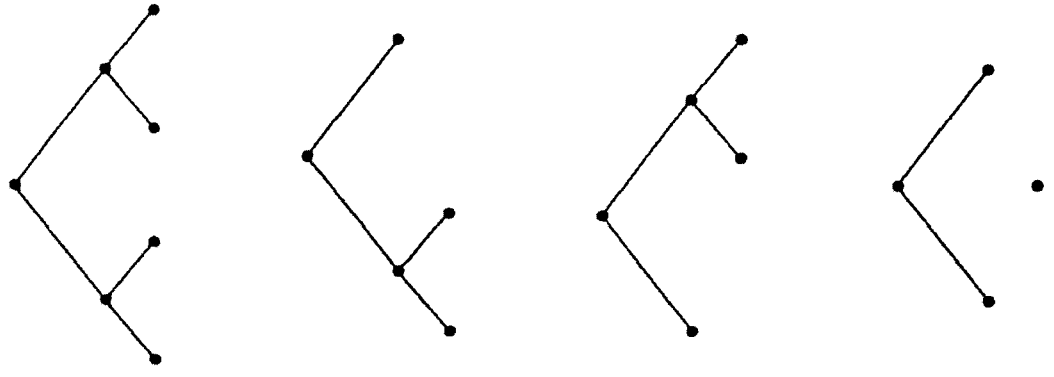


Figure 3.9. All possible combinations of tree-structured filter banks of depth 2. Symbolically, a fork stands for a two-channel filter bank with the lowpass on the bottom. From left to right is the full tree (STFT like), the octave band tree (wavelet), the tree where only the highpass is split further, the two-band tree and finally the nil-tree tree (not split at all) [Vetterli].

It is important to observe that the full tree (first from left to right), which yields a linear division of the spectrum is similar to the STFT, and that the octave band tree (second from left to right) actually performs a DWT as discussed in the previous section. All these arbitrary tree structures are known as wavelet packets and were introduced as a family of

orthonormal bases for discrete-time signals [Vetterli]. As might be intuitively perceived, wavelet packets offer a full menu of decomposition (and reconstruction) structures from which we can choose a particular tree depending on our requirements or based on a particular criterion (whose discussion is out of the scope of this work. See, for example [Mallat]).

To visualize the time-frequency analysis achieved by different wavelet packets filter banks three different (binary) tree structures and its corresponding time-frequency tiling are schematically sketched in Figure 3.10. It should be noted the time localization and frequency resolution trade-offs for each one.

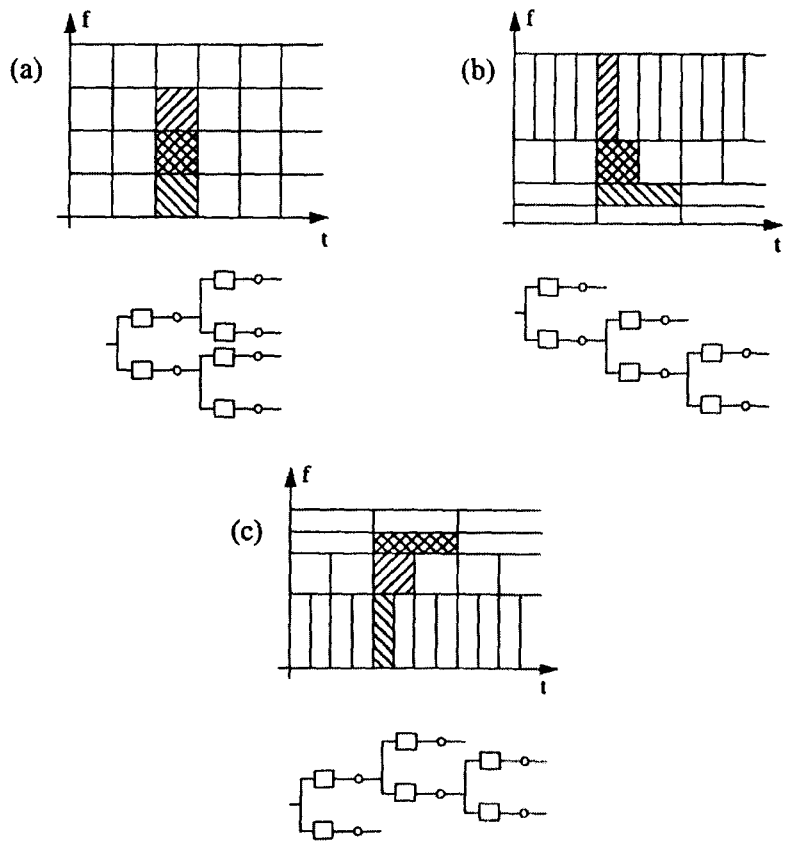


Figure 3.10. Time-frequency analysis achieved by different binary subband trees. The trees are on bottom, the time-frequency tilings on top. (a) Full tree or STFT. (b) Octave-band tree or wavelet series. (c) Arbitrary tree or one possible wavelet packet [Vetterli].

3.5 Noise issues

Finally, in this section we state the three most widely used (and treated in [Deller]) definitions of the *Signal-to-Noise Ratio* (SNR), which includes the classical SNR, segmental SNR, and frequency-weighted segmental SNR.

Let us define three energy signals at time n: a noisy speech signal $y(n)$, with a noise-free equivalent $s(n)$, and an enhanced processed signal $\hat{s}(n)$. Then we may express the error signal relating them as

$$\varepsilon(n) = s(i) - \hat{s}(i) \tag{15}$$

thus, the error energy will be

$$E_\varepsilon = \sum_{n=-\infty}^{\infty} \varepsilon^2(i) = \sum_{n=-\infty}^{\infty} [s(i) - \hat{s}(i)]^2 \tag{16}$$

Furthermore, the energy contained in the speech signal $s(n)$ is

$$E_s = \sum_{n=-\infty}^{\infty} s^2(n) \tag{17}$$

and then, the resulting *classical* SNR measure in *dB* is

$$\text{SNR} = 10\log_{10}\frac{E_s}{E_\varepsilon} = 10\log_{10}\frac{\sum_n s^2(n)}{\sum_n [s(n) - \hat{s}(n)]^2}. \tag{18}$$

As is the case of this work, this quality measure could be used to evaluate the recognition system, provided that the original speech signal $s(n)$ is available, as in fact, it is.

Obviously, we are using this kind of measures in simulation.

Although, this measure represents an average error over time and frequency for a processed signal, it is known [Deller] that classical SNR is a poor estimator of speech quality for many different kinds of speech distortions.

Moreover, [Deller] explains that, the speech energy is time-variant, and if we are dealing with broadband noise distortion with little energy fluctuation, the SNR measure should vary on a frame-by-frame basis. In other words, for example, a deceptively high SNR can be obtained if an utterance contains a high concentration of voiced segments, since noise has a greater perceptual effect in low-energy segments, as unvoiced fricatives.

A much better improvement proposed for this kind of quality measures is an averaged SNR measured over short frames. This frame-based measure is called the *segmental SNR* (SNR_{seg}), and is defined as [Deller]

$$\mathrm{SNR}_{\mathrm{seg}} = \frac{1}{M}\sum_{j=0}^{M-1} 10\log_{10}\left[\frac{\sum_{n=m_j-N+1}^{m_j} s^2(n)}{\sum_{n=m_j-N+1}^{m_j}[s(n)-\hat{s}(n)]^2}\right] \tag{19}$$

where $m_0, m_1, \ldots m_{M-1}$ are the end-times for the M frames, each of length N. This means that a (classical) SNR is computed for each (typically 15 to 25 ms) frame over an utterance, and then the $\mathrm{SNR}_{\mathrm{seg}}$ is obtained by averaging the SNRs of that utterance. This *segmentation* allows an objective measure to assign equal weight to loud and soft portions of the speech [Deller].

It is important to note that in some cases, problems arise with this segmental measure if frames of silence are included (as in speech recognition), because (19) yields large negative $\mathrm{SNR}_{\mathrm{seg}}$. However, there are several approaches to avoid this problem. A convenient approach may be to identify silence frames and discard them regarding $\mathrm{SNR}_{\mathrm{seg}}$ calculations. But also, we can set a lower threshold and replace all frames with $\mathrm{SNR}_{\mathrm{seg}}$ below it with the threshold itself. A common reasonable lower threshold is 0-dB. On the other hand, frames with $\mathrm{SNR}_{\mathrm{seg}}$ greater than 35 dB are not perceived by listeners as being significantly different [Deller *et al*], and an upper threshold is needed to replace any unusually high $\mathrm{SNR}_{\mathrm{seg}}$. Normally, a 35 dB upper threshold is used.

The last SNR variant or definition treated here is the *frequency weighted segmental SNR* ($\mathrm{SNR}_{\mathrm{fw\text{-}seg}}$). This measure allows SNR (perceptual) weighting at different frequency bands, which might be adapted to the human auditory system critical bands, and produce a SNR closely related to a listener's perceived notion of quality [Deller]. The definition is stated as

$$\mathrm{SNR}_{\mathrm{fw-seg}} = \frac{1}{M}\sum_{j=0}^{M-1} 10\log_{10}\left[\frac{\sum_{k=1}^{K} w_{j,k}\, 10\log_{10}\left[E_{s,k}(m_j)/E_{\varepsilon,k}(m_j)\right]}{\sum_{k=1}^{K} w_{j,k}}\right] \tag{20}$$

where M is the number of speech frames (indexed by $m_0, \ldots, m_{M-1}$), K is the number of frequency bands, $w_{j,k}$ are the noise-dependent perceptual weights, $E_{s,k}(m_j)$ the short-term signal energy contained in the kth frequency band for the frame of noise-free speech indexed by m_j, and $E_{\varepsilon,k}$ the similar quantity for the noise sequence $\varepsilon(n)$.

CHAPTER 4

A WAVELET-BASED FRONT END FOR ROBUST ASR

In this chapter we present the main structure, guidelines and basic implementation issues of our experiments. We refer to the theory outlined in previous chapters and present some new practical definitions.

As we do not present or refer to specific algorithmic and programming issues, we consider that it should be noted that this report is and it should be treated as a brief statement of theory and results attained in the most relevant (and only a very few) of our experiments realized by a scheduled deadline.

The overall achievement of this work relies in months of self-learning wavelet and multirate signal processing theory with extensive c-programming, scripting, debugging and testing on the ITESM Speech Group Workstations. As the theory became clearly understood we were able to build simple programs to test the basic concepts presented in [Wickerhauser]. While reading recent literature that applied wavelet concepts to speech recognition tasks we also introduced our selves to the Sphinx 3 codebase. When we were able to decide which features to implement, we developed a set of C functions including a wavelet packet library, and test it with single frame signals. The final step was to adapt the *wave2feat* Sphinx 3 preprocessor to work with wavelet packets using our libraries, carefully devising it as to avoid further modification of the Sphinx 3 codebase.

Even though, all of this has been described in this last single paragraph it was the most time-consuming task of our work; actually in the range of months of work. But, in the sake brevity and to accomplish the intended purpose or this report we present the most relevant theoretic issues in a small number of sections without treating any programming issues.

The first section describes the baseline implementation; this is the Mel-Frequency Cepstral Coefficients (MFCC), which are the front end parameters Sphinx 3 was originally engineered to work with. Then we cover the two types of experiments scheduled to be included for this report, namely, extracting cepstral parameters from a wavelet-based generated spectrum, and extracting wavelet transformed coefficients from a DFT generated spectrum.

Results are presented in the next chapter in executive-summary fashion, but following this chapter's order.

4.1 The MFCC Baseline

4.1.1 Description

Mel-Frequency Cepstral Coefficients (MFCC) are the most common parameterization algorithms used in current recognition tasks [Hansen]. To compute the MFCCs as presented in [Davis] we must first window the speech waveform and STFT it. Then we compute the energy in the STFT weighted by each mel-scale filter frequency response. Defining $V_l(\omega)$ as the frequency response of the lth mel-scale filter, the resulting energies for each speech frame at time n and for the lth mel-scale filter, are yielded by

$$E_{mel}(n,l) = \frac{1}{A_l}\sum_{k=L_l}^{U_l} |V_l(\omega_k)X(n,\omega_k)|^2 \tag{1}$$

where L_l and U_l denote the lower and upper frequency indices over which each filter is nonzero and where

$$A_l = \sum_{k=L_l}^{U_l} |V_l(\omega_k)|^2 \tag{2}$$

which normalizes the filters according to their varying bandwidths so as to give equal energy for a flat input spectrum [Quatieri].

With all this, the real cepstrum associated with $E_{mel}(m,l)$ is called the mel-cepstrum and, as presented in [Davies] is computed for the speech frame at time n as

$$C_{mel}[n,m] = \frac{1}{R}\sum_{l=0}^{R-1} \log\{E_{mel}(n,l)\}\cos\left[\frac{2\pi}{R}\left(l+\frac{1}{2}\right)m\right] \tag{3}$$

where R is the number of filters. It should be noted that the even property of the real cepstrum has been used to write the inverse transform in terms of the cosine basis, referred to as the *discrete cosine transform* [Quatieri].

4.1.2 Experiments Specifications

In our experiments, we used a 40 filter mel-scale filter bank. Even though, other window durations were used, only the 25 ms window results are reported due to the marginal variability attained by the other experiments. Only the first 13 coefficients yielded by (3) are kept as features. This includes the frame energy $C_{mel}[n, 0]$. As kept the same configuration for both databases, which means results are not to be compared between both databases, but each database result against itself in different front end configurations. All these because the different sampling rate of the databases.

4.2 Subband-Based Cepstral Parameters

4.2.1 Description

To compute the SBC parameters we first compute the wavelet packet transform with the (24-bands) tree shown in Figure 1. Then the wavelet coefficients at the end of each final branch (or subband) of the tree are used to calculate the subband signal energies for each frame as

$$E_{wpt}(n,i) = \frac{1}{N_i}\sum_{j=1}^{N_i} p_{i,j}^2 \tag{4}$$

where N_i is the number of coefficients in each subband, and $p_{i,j}$ is the jth wavelet packet coefficient in the ith subband [Goswami].

This method proposed in [Erzin] yields the subband energies of the signal with frequency bands similar to the Mel-frequency division, which exploits the properties of the human auditory system.

At last, we compute the cepstral coefficients using energies yielded by (4). In fact, we follow (3) and may rewrite it here for clarity as

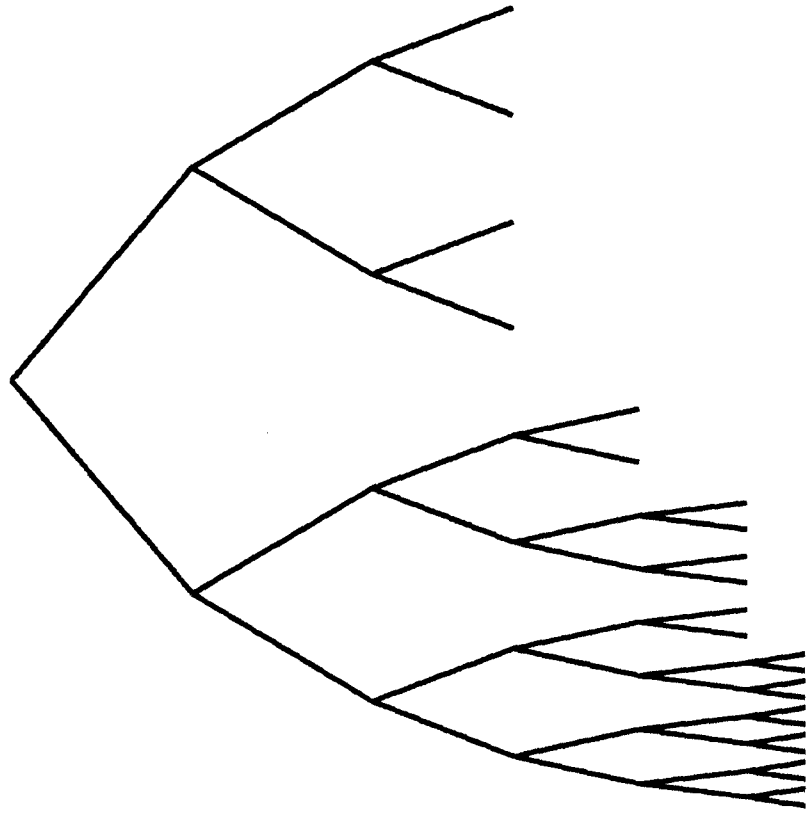


Figure 4.1. Wavelet packet tree for computing mel-like subband energies presented in [Farooq].

$$C_{mel}[n,m] = \frac{1}{R}\sum_{l=0}^{R-1} \log\{E_{wpt}(n,l)\} \cos\left[\frac{2\pi}{R}\left(l+\frac{1}{2}\right)m\right]$$

where R here is the number of bands.

Wavelet Filter				Mel Filter	
Filter Number	Lower cut off frequency (Hz)	Higher cut off frequency (Hz)	Bandwidth (Hz)	Central frequency (Hz)	Bandwidth (Hz)
1	0	125	125	100	100
2	125	250	125	200	100
3	250	375	125	300	100
4	375	500	125	400	100
5	500	625	125	500	100
6	625	750	125	600	100
7	750	875	125	700	100
8	875	1000	125	800	100
9	1000	1125	125	900	100
10	1125	1250	125	1000	100
11	1250	1375	125	1149	160
12	1375	1500	125	1320	184
13	1500	1750	250	1516	211
14	1750	2000	250	1741	242
15	2000	2250	250	2000	278
16	2250	2500	250	2297	320
17	2500	2750	250	2639	367
18	2750	3000	250	3031	422
19	3000	3500	500	3482	484
20	3500	4000	500	4000	556
21	4000	5000	1000	4595	639
22	5000	6000	1000	5278	734
23	6000	7000	1000	6063	843
24	7000	8000	1000	6954	969

Table 3. Frequency bands achieved by the 24-band wavelet packet decomposition and a 24 filter mel-scale filter bank [Farooq].

4.2.2 Experiments Specifications

We experimented extracting SBC parameters using three different filters to compute the wavelet packet coefficients. This are the Daubechies 20, Vaidyanathan 24, and Beylkin 18. Daubechies 20 (tap) filters maximize the smoothness of the associated scaling function by maximizing the rate of decay of its Fourier transform, and are a popular choice among the literature [see Daubechies]. The Beylkin 18 filter coefficients was designed by placing roots for the frequency response polynomial close to the Nyquist frequency on the real axis, thus concentrating power spectrum energy in the desired band. The Vaidyanathan 24

correspond to the filter sequence #24B constructed by [Vaidyanathan and Huong]. This filter has been optimized, for its length, to satisfy the standard requirements for effective speech coding [Wikerhauser].

In summary, the front end computes de wavelet packet coefficients for each band using the tree shown in Figure 4.1, then computes the log-energy of each one and extracts the first 13 cepstral coefficients following equation (3). All this for each filter. Table 3 specifies the frequency bands achieved by the wavelet packet decomposition versus those of a mel-scale filter bank [Farooq]. We trained Sphinx 3 with noise-free data and decode with noise-free data and five different signal-to-noise ratios noisy speech.

4.3 Wavelet Transform Parameters

4.3.1 Description

Wavelet parameters have been used in different approaches in speech recognition tasks [Hansen] [Farooq] [Goswami]. The most frequently used approach is to compute the log-subband energies using a wavelet packet transform as for SBC parameters, and then apply a wavelet packet or dyadic transform to the log-subband energies, and choose 12 or 13 coefficients from this last decomposition as features.

We propose to compute the DWT up to a *4*th level to extract wavelet or approximation coefficients from the STFT spectrum of the speech signal. We use the spectrum rather then subband energies because we use the cepstrum (extracted from the IDFT applied to the log-spectrum) of the speech signal to perceptually have a better intuition as to select which wavelet coefficients carry vocal tract information without contributing to the pitch part of the signal's cepstrum.

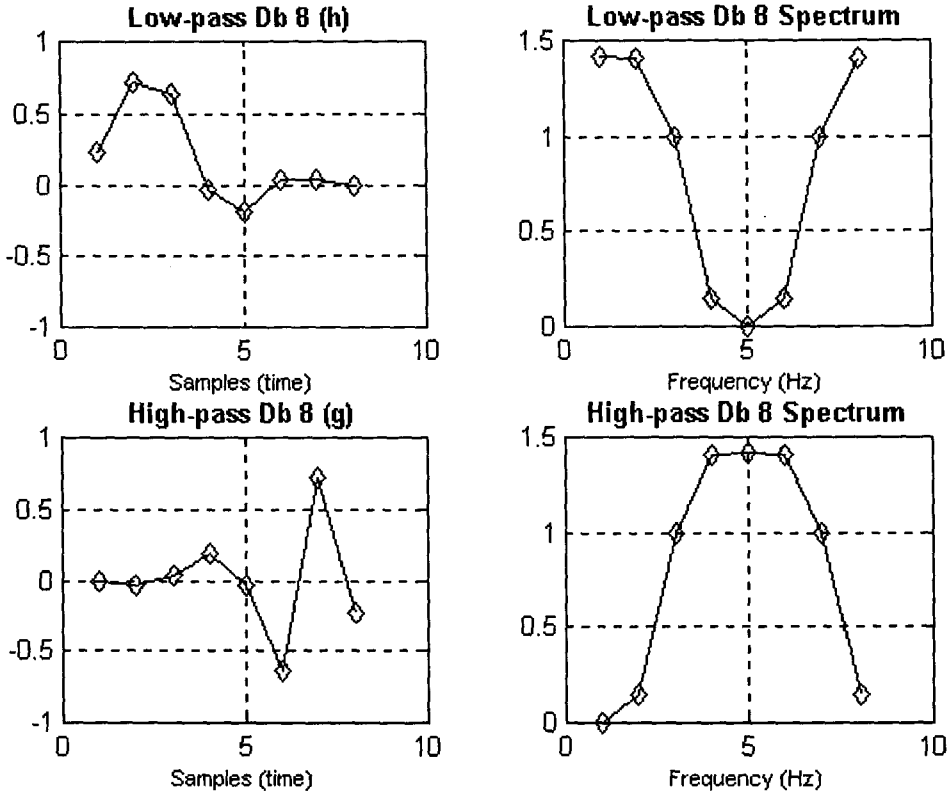


Figure 4.2. Daubechies 8 high-pass and low-pass filters and its respective spectrum.

With this intuitive approach, we decompose the log-spectrum of a speech frame with a 4-level DWT. Then we put to zero, say, the wavelet coefficients of the fourth level, and reconstruct the log-spectrum through the corresponding inverse DWT. With the reconstructed (and, obviously, modified) log-spectrum we compute the cepstrum to perceptually account the changes in its shape. We do this with every wavelet space

(coefficients) and several combinations of zeroed wavelet spaces. When we find that the cepstrum has no pitch effect we define a feature vector from the wavelet coefficients that yielded the reconstructed log-spectrum corresponding to the ("pitchless") cepstrum.

In the proposal presented in this report we use a Daubechies 8 filter (shown in Figure 4.2) for the experiments. We show the case of a 25 ms speech frame of an /e/ sound sampled at 8 kHz (200 samples). Figure 4.3 illustrates the hamming-windowed zero-padded signal, its spectrum, log-spectrum and cepstrum. This is the same sound shown in Figure 3.1, but resampled at 8 kHz. The cepstrum in Figure 3.1 shows the pitch spike around the 50th sample, while in Figure 4.3 is around the 25th sample. Mainly, our goal in this part of the experiment is getting a flat line rather than this (pitch) spike in the cepstrum.

The Figure 4.4 shows the different recovered log-spectrums calculated tagging the wavelet spaces as **w***x* where *x* is the level of the space, **w** stands for wavelet (or detail) space, and **v** refers to the approximation space. It is interesting to note that until we start combining zeroed wavelet spaces the recovered log-spectrum becomes smoother.

Obviously, as we zero lower-level approximation spaces we get a more shapeless noise-like reconstructed signal as illustrated in Figure 4.5.

Finally, Figure 4.7 shows the different cepstrums for each one of the reconstructed log-spectrums in Figure 4.4. It may be clear that we could even use the reconstructed log-spectrums for taking a decision regarding which spaces carry better vocal tract information.

Figure 4.8 shows the wavelet and scaling coefficients of the four different spaces.

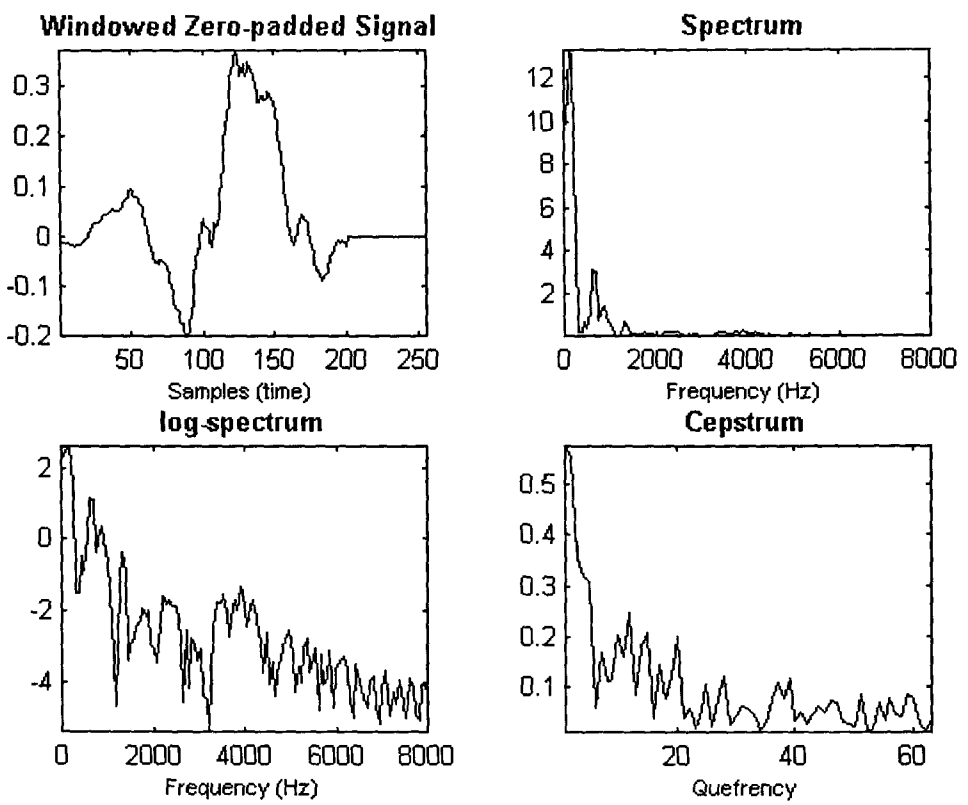


Figure 4.3. Signal analysis of a 400 samples /e/ sound. The waveform was windowed with a hamming window and zero-padded to last 256 samples.

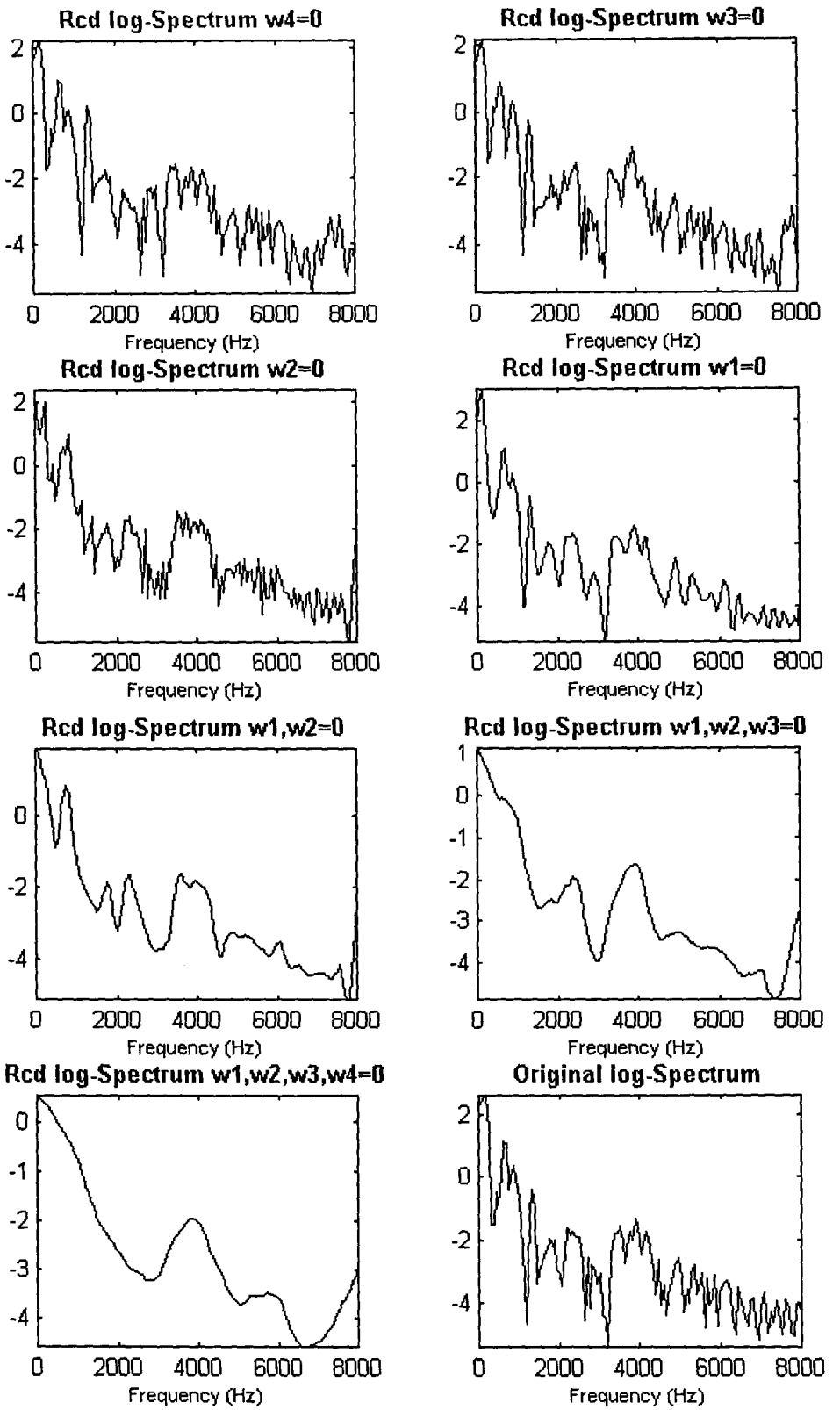


Figure 4.4. Different reconstructed log-spectrums. The iDWT is applied to the DWT of the original log-spectrum (bottom-right), but with different options of zeroed spaces.

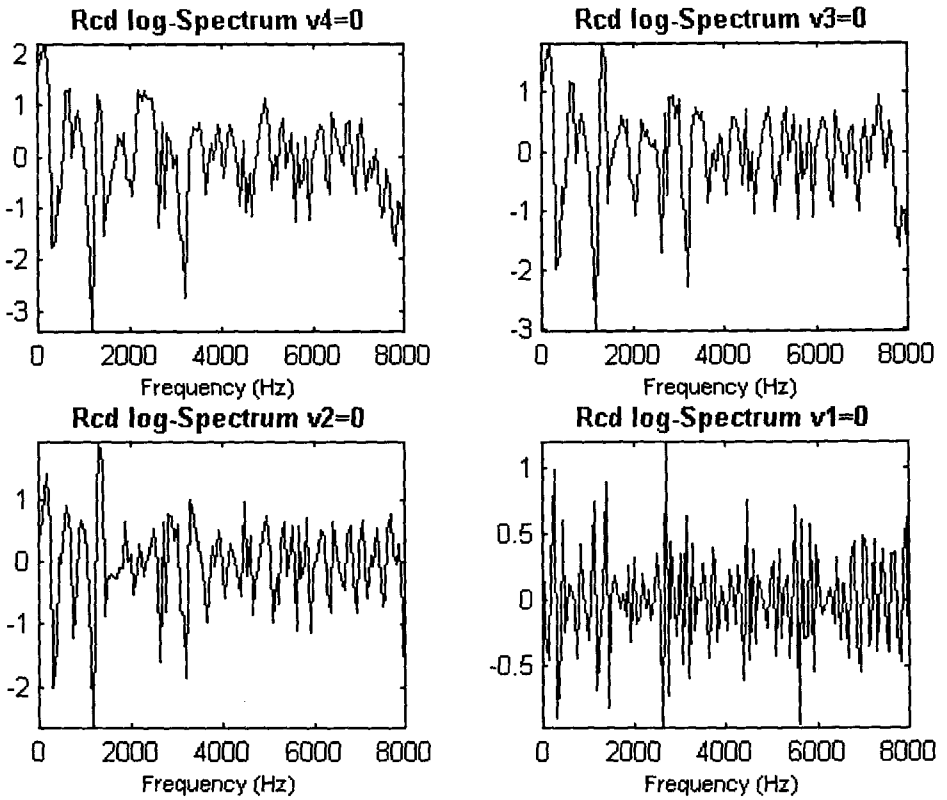


Figure 4.5. Reconstructed log-spectra when approximation spaces have been put to zero.

4.3.2 Experiments Specifications

We realized experiments involving the design of features consisting in the energy of the spaces **V4**, **W4**, and **W3** (see Figure 4.6), and the 2th to 12th coefficients of the approximation space **V4** computed with the Daubechies 8 filters for a total of 13 feature elements per speech frame.

We also experimented with a feature vector consisting in the energy of the space **V4** and the 2th to 14th coefficients of the approximation space **V4**.

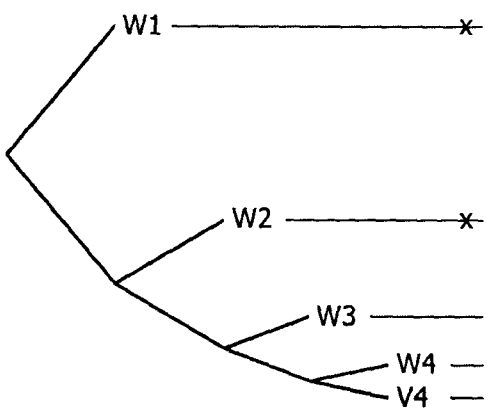


Figure 4.6. DWT tree used for computing wavelet transform parameters.

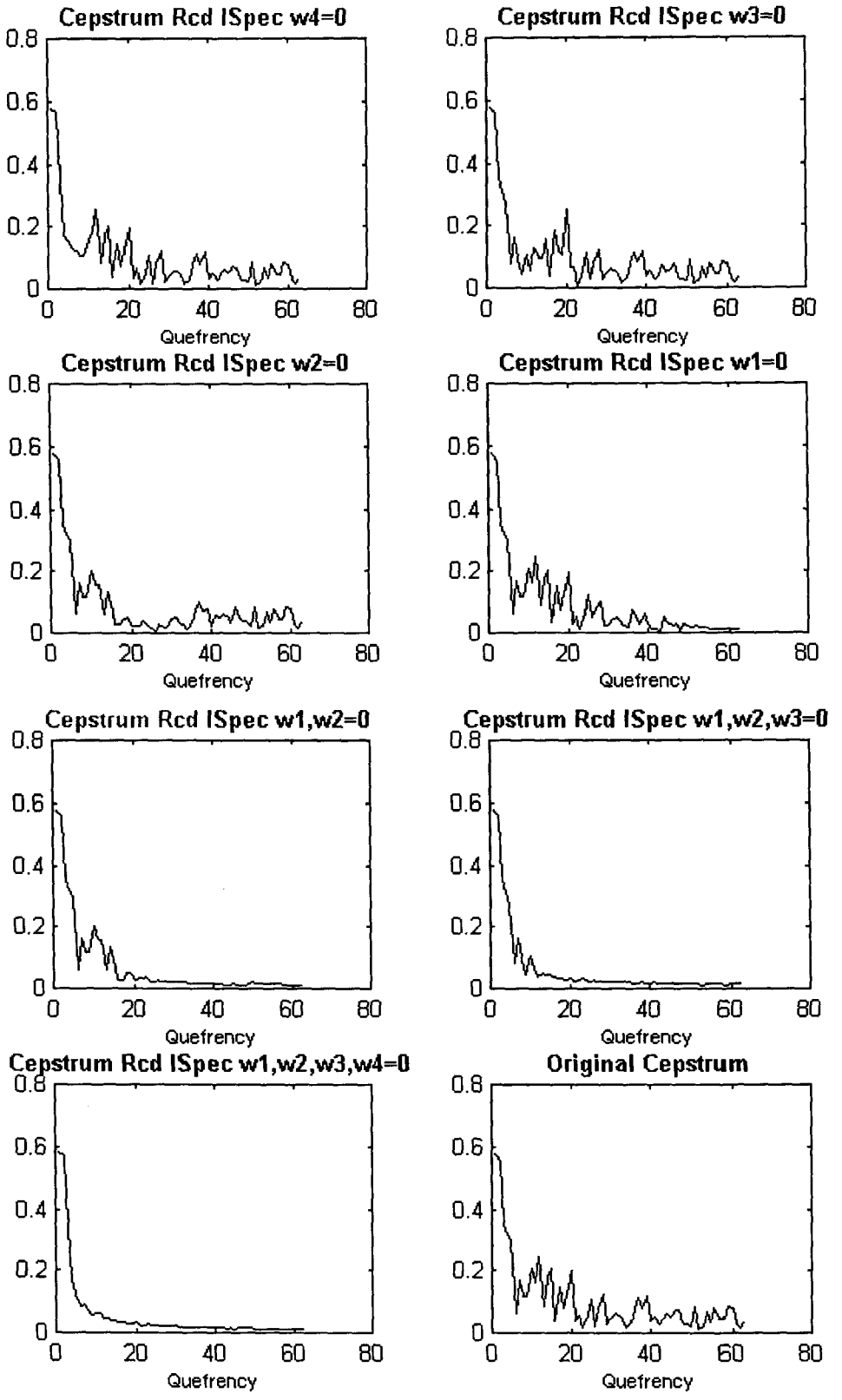


Figure 4.7. Cepstrums attained from the corresponding reconstructed log-spectra shown in Figure 4.4. The original cepstrum is shown in the bottom-right.

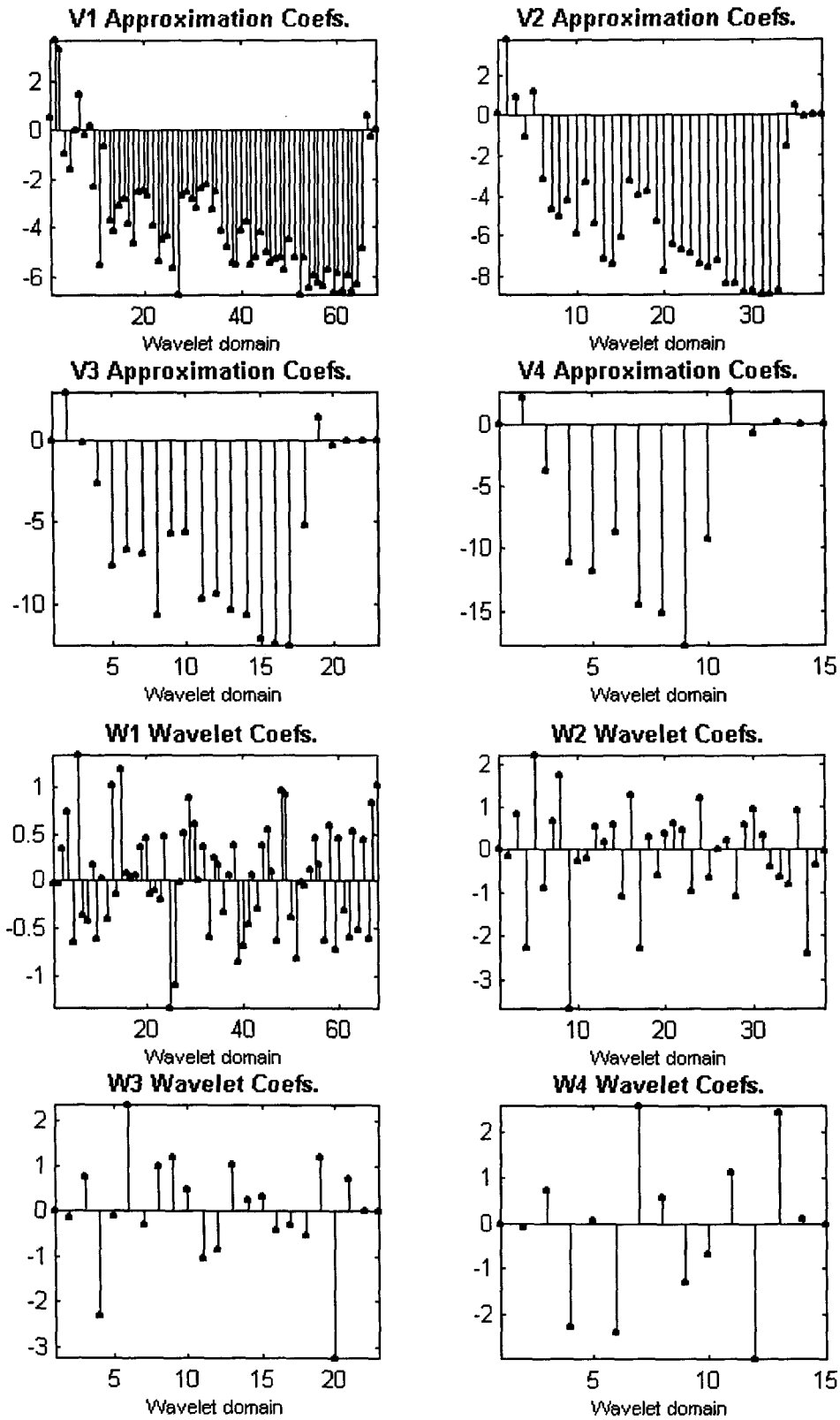


Figure 4.8. Approximation coefficients, Vx, and wavelet coefficients, Wx, for the corresponding x level.

CHAPTER 5

RESULTS

This chapter is an executive summary of the results attained by the experiments explained in the previous chapter. A brief description of each experiment (with references were needed) precedes the tables of results. Not all the experiments described in the last chapter are presented, but only those that provided relevant information by the time this report was required. Other experiments are still in progress and we are engineering new features to be tested on Sphinx 3. An actual description of our work and publications can be found on the internet (http://juang.mty.itesm.mx).

Results could be better (and will be), although development of this experiments didactically enlightened us to insight the knowledge of feature generation and wavelet transform; it prepared us for future work.

Results are presented in the following tables. It should be noted that in all the experiments we trained Sphinx 3 with noise free data.

Table 1 provides the word accuracy percentages (which will be referred simply as accuracy in this chapter) attained with the experiments realized using the CSDigit database with different signal-to-noise ratios in the utterances. Also the corresponding bar graphs are illustrated in the top of each column of the table. The columns show the accuracy of each front end against the different signal-to-noise ratios data utterances of CSDigit. The first column corresponds to the baseline MFCC front end, which is configured to yield a feature vector of 13 coefficients per frame using Equation (3) of chapter 4. The complete specifications are explained in subsection 4.1.2. This MFCC front end attains the highest accuracy over the others front ends when noise-free data utterances are used. Though, the differences are marginal; all four front ends reach a 94% accuracy with CSDigit noise-free data.

The second column displays the results for the subband based coefficients extracted with the front end using a Daubechies 20 wavelet filter as explained in section 4.2.2 with the wavelet packet tree shown in Figure 4.1. With noise-free and 20 dB SNR data Daubechies 20 is not better then the MFCC front end, but as the signal-to-noise ratio decreases in the utterances Daubechies 20 clearly outperforms the baseline front end.

The third column displays the results of the front end with the Vaidyanathan 24 filter as explained also in section 4.2.2. This filter has been optimized, for its length, to satisfy the standard requirements for effective speech coding [Wikerhauser], and attains better results than the baseline for all five different signal-to-noise ratios in the CSDigit utterances.

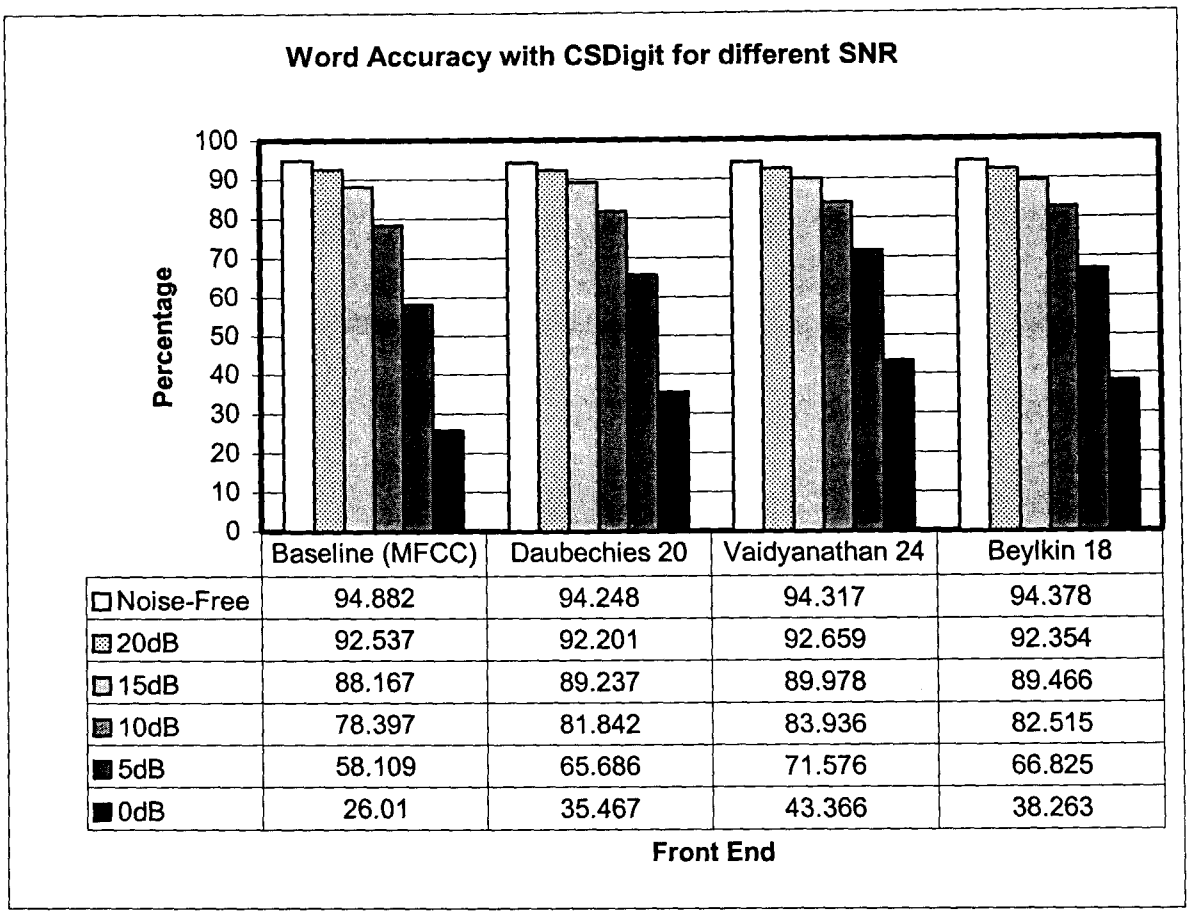
Finally, the fourth column refers to the front end with the Beylkin 18 filter, which attains similar but better results than the Daubechies 20 front end. It also outperforms the baseline front end for the 15 dB, 10 dB, 5 dB, 0 dB SNRs, while being as good as the MFCC front end for the cases of noise-free and 20 dB SNR CSDigit utterances.

It can be observed in a global view of the graphs on Table 1, how the three proposed front ends attain better results than the MFCC baseline front end as noise increases (i.e. SNR decreases).

Another perspective of the same results is illustrated in Figure 5.1. Here we have groups of four bar graphs, each group corresponding to each signal-to-noise ratio (or noise-free data) and each bar corresponding to each front end, as specified in the label on the right of the graph.

With this view it becomes clearer how the front ends attain almost the same (94%) accuracy with noise free CSDigit data, and also in the 20 dB SNR case. The differences are gradually more apparent as the signal-to-noise ratio in the utterances decreases; the baseline MFCC front end is outperformed by all the proposed front ends.

Above 50% of accuracy, the most significant difference is found to be when 5 dB CSDigit data is recognized by the Sphinx 3 platform. The best results in this case are attained by the Vaidyanathan 24 filter front end, as it is in the rest of the experiments presented in Figure 5.1. The actual difference between the Vaidyanathan 24 and the MFCC front end for this case of 5 dB data is of 13.467%. under 50% of accuracy, namely the 0 dB SNR case, this difference increases up to 17.356%.



	Baseline (MFCC)	Daubechies 20	Vaidyanathan 24	Beylkin 18
Noise-Free	94.882	94.248	94.317	94.378
20dB	92.537	92.201	92.659	92.354
15dB	88.167	89.237	89.978	89.466
10dB	78.397	81.842	83.936	82.515
5dB	58.109	65.686	71.576	66.825
0dB	26.01	35.467	43.366	38.263

Table 1. Word accuracy results of experiments with the CSDigit database. Quantities are percentages. Graph on top shows different SNR word accuracy for each front end.

All this could be restated as shown in Figure 5.2, which is also a table, which illustrates the percentage of improvement of accuracy of each proposed front end relative to the baseline MFCC front end. The numbers plotted are calculated using the following formula:

$$\text{Impr} = \frac{FE_{acc} - BFE_{acc}}{100 - BFE_{acc}} \times 100$$

where *Impr* refers to improvement percentage, FE_{acc} is the proposed front end accuracy attained in a given (signal-to-noise ratio) experiment, and BFE_{acc} is the baseline (MFCC) front end accuracy attained for the same experiment conditions.

This last kind of graph we present shows the recognition improvement based in word recognition accuracy for the three different front ends as the SNR goes from 20 dB to 0 dB, i.e. as the signal is getting overwhelmed by the gaussian noise. The improvement percentage presented is calculated for each SNR using the difference of accuracy between the proposed and the baseline front end, over the baseline front end distance to achieving perfect word recognition.

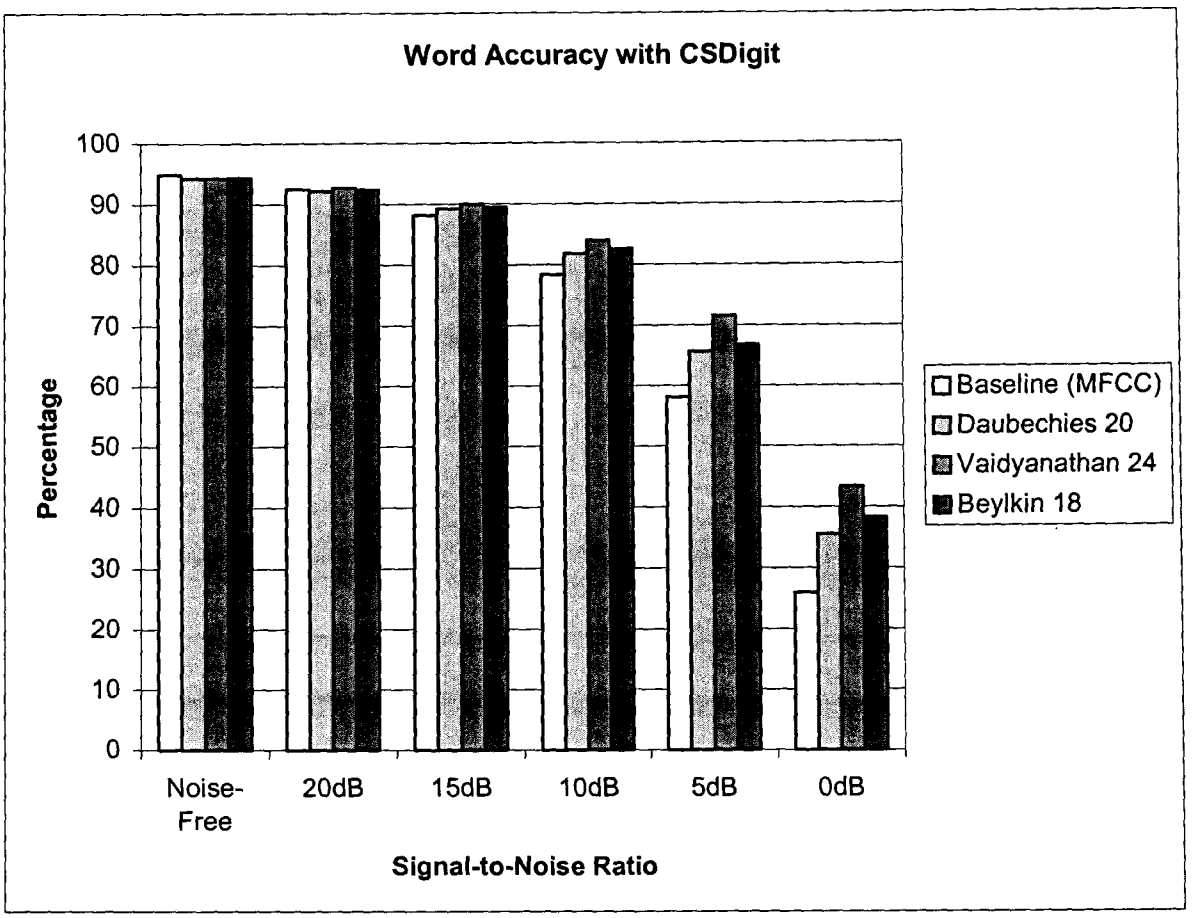


Figure 5.1. Front ends word accuracy grouped for each SNR on CSDigit data.

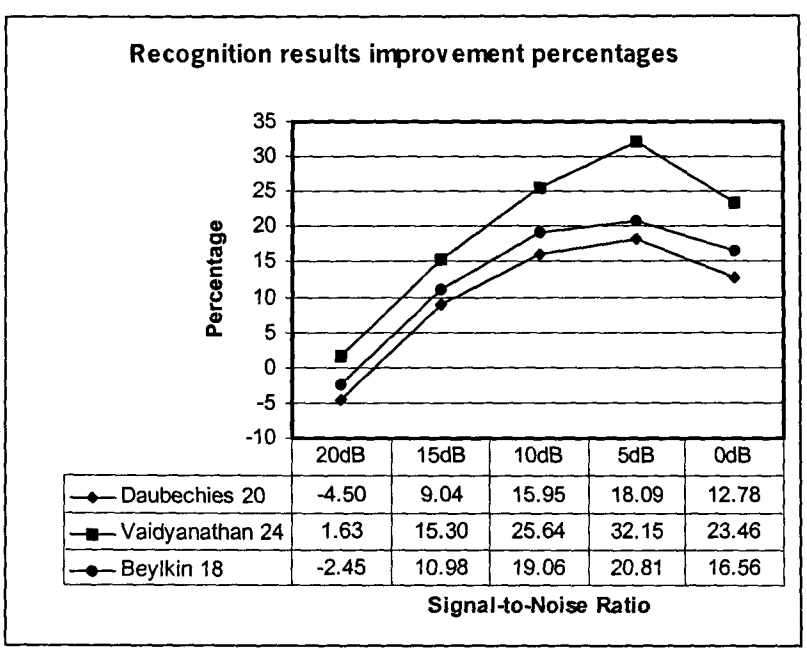


Figure 5.2. Front ends improvement on CSDigit data.

Wavelet transform parameters of section 4.3.2 are not reported here, as by this time we have not been able to reach a significant level of accuracy. Nevertheless, design of a collection of parameters using this scheme is still in progress. An actual state and description of our work is found on the internet (http://juang.mty.itesm.mx).

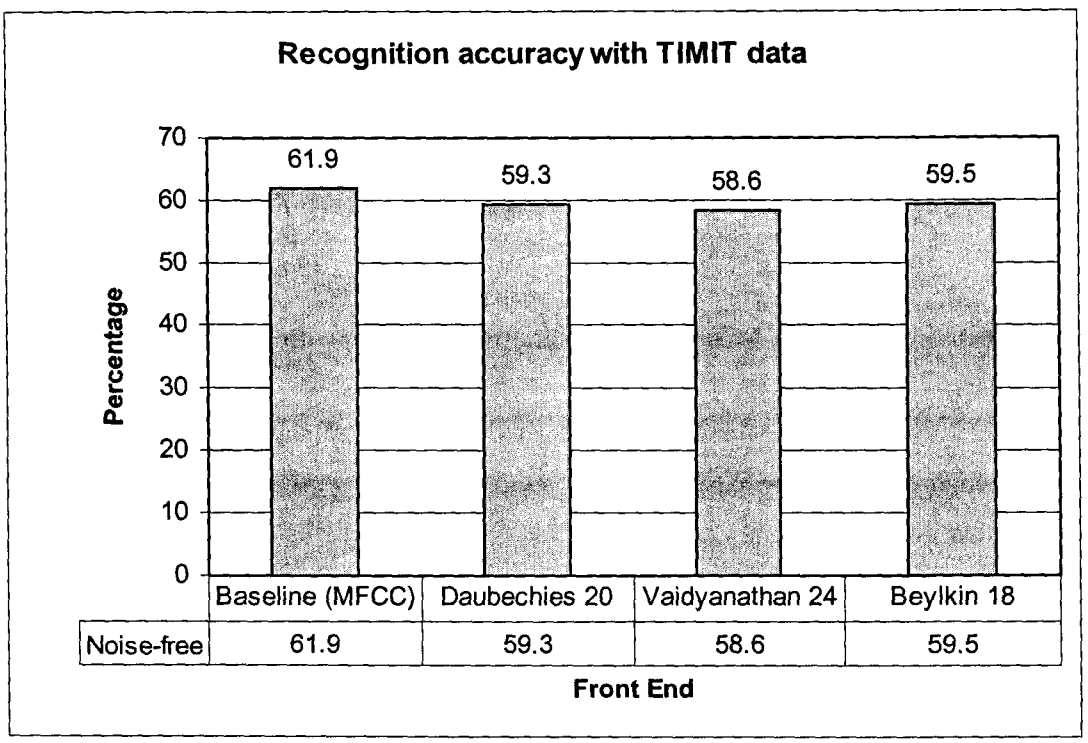


Table 2. Word accuracy results of experiments with the TIMIT database. Quantities are percentages.

Finally, we present in Table 2 the results of the same experiments explained until now, but with noise-free TIMIT data. As with CSDigit data, it can be noted that, although the MFCC baseline front end is not outperformed when experimenting with noise free utterances, the differences are also minimal, indeed.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this work we present results of three different proposed front ends, which features extraction techniques are based on cepstrum analysis of a wavelet generated spectrum. Now we compare this results with the baseline front ends results, namely the Mel-Frequency Cepstral Coefficients (MFCC).

First of all, we have enough information to conclude that all three proposed front ends are as good as the MFCC baseline, regarding noise free speech recognition; the differences of accuracy are minimal.

And, as can be seen for CSDigit data on Table 1 and Figure 5.1 of the previous chapter, as we decrease the signal-to-noise ratio the wavelet-based front ends perform better than MFCC. Figure 5.2 is very enlightening because it shows the prevalence of better

performance of Vaidyanathan 24 over all SNR experiments with a maximum of 32% improvement for 5 dB SNR CSDigit utterances.

Therefore, our main conclusion is that wavelet-based parameters uphold their promise of providing better features for speech recognition. Our certainty increases as we deal with greater SNR speech data.

Future Work

It is clear for us that a further understanding and *fine tuning* of the Sphinx 3 recognition stages, modeling and overall system is needed before continuing.

On the other hand, though we were not able to show results on wavelet transform parameters as explained in subsection 4.3.2, it has been stated [Hansen] that wavelet features can be found to outperform the MFCC baseline for stressed speech. The insight as the one presented in chapter 4 further expands the variety of experiments still to be realized.

Our future work will consist of further analysis of the features we presented and of those not reported for a better consideration of the acoustic model. A start could be to use *principal-component analysis* [Acero] to compute the optimum basis vector of first- and second-order dynamic features Sphinx 3 implements for recognition.

Future work can be easily scheduled if we keep digging in the actual wavelet literature. Different wavelet spaces can be proposed following compression and denoising wavelet techniques. Moreover, a different wavelet mapping can be proposed similar to principal-component-analysis to find efficiently discriminative wavelet-based features.

REFERENCES

[Acero] Huang, X., Acero, A. Han, H.W., Reddy, R. *Spoken language processing: a guide to theory, algorithm and system development.* Prentice Hall PTR, 2001.

[Daubechies] Daubechies I. *Ten lectures on wavelets.* Society of Industrial and Applied Mathematics, 1992.

[Davis] Davis, S.B. and Mermelstein, P. *Comparision of parametric representations for monosyllabic word recognition in continuously spoken sentences.* IEEE Trans. Acoust., Speech, Signal Processing. Vol. 28, pp. 357-366, August 1980.

[Deller] Deller, J.R., Hansen, J., Proakis, J.G. *Discrete-Time processing of speech signals.* IEEE Press; reprint edition, January 2000.

[Erzin] Erzin, E., Cetin, A.E., Yardimci, Y. *Subband analysis for speech recognition in the presence of car noise.* ICASSP-95, vol. 1, pp. 417-420.

[Farooq] Farooq, O., Datta, S. *Mel Filter-Like Admissible Wavelet Packet Structure for Speech Recognition.* IEEE Signal Processing Letters. Vol. 8, No. 7, July 2001.

[Goswami] Goswami, J.C., Chan, A.K. *Fundamentals of wavelets. Theory, algorithms, and applications.* Wiley-Interscience, 1999.

[Hansen] Sarikaya, R., Hansen, J. *High Resolution Speech Feature Parametrization for Monophone-Based Stressed Speech Recognition.* IEEE Signal Processing Letters. Vol. 7, No. 7, July 2000.

[Harrington] Harrington, J., Cassidy, S. *Techniques in speech acoustics.* Kluwer Academic Publishers, 1999.

[Holschneider] Holschneider, M., Kronland-Martinet, R., Morlet, J., Tchamitchian, Ph. *A real-time algorithm form signal analysis with the help of the wavelet transform,* in Wavelets, Time-Frequency Methods and Phase Space. Combes, J.M., Grossmann, A., and Tchamitchian, Ph., Eds. Springer, 1989.

[Jelinek] Jelinek, F. *Statistical methods for speech recognition.* The MIT Press, 1998.

[Mallat] Mallat, S. *A wavelet tour of signal processing.* Academic Press, 1999.

[Meyer] Meyer, Y. *Wavelets and operators.* Cambridge University Press, 1992.

[Quatieri] Quatieri, T.F. *Discrete-time speech signal processing, principles and practice.* Prentice Hall PTR, 2001.

[Rabiner] Rabiner, L., Juang, B.H. *Fundamentals of speech recognition*. Prentice Hall PTR, 1993.

[Rioul] Rioul, O., Duhamel, P. *Fast Algorithms for Discrete and Continuous Wavelet Transforms*. IEEE Transactions on Information Theory. Vol. 38, No. 2, March 1992.

[Sundberg] Sundberg, J. *The Acoustics of the Singing Voice*. Scientific American, March 1977.

[Sphinx FAQ] CMU Sphinx FAQ. Online: http://www.speech.cs.cmu.edu/sphinx/doc/sphinx-FAQ.html

[TIMIT] TIMIT CD-ROM README file. National Institute of Standards and Technology. NIST Speech Disc CD1-1.1, October 1990.

[Vetterli] Vetterli, M., Kovacevic, J. *Wavelets and subband coding*. Prentice Hall PTR, 1995.

[Wickerhauser] Wickerhauser, M.V. *Adapted wavelet analysis from theory to software*. IEEE Press, 1993.

[Vaidyanathan] Vaidyanathan, P.P. *Multirate systems and filter banks*. Prentice Hall PTR, 1993.

[Vaidyanthan and Huong] Vaidyanathan, P.P., Huong, P.Q. *Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks*. IEEE Transactions on Acoustics, Speech, and Signal Processing. Vol. 36, No. 1, pp. 81-94, January 1988.