1-1-1995

Multiresolution wavelet analysis of EEG signals for the detection of Alzheimer's disease

Robi Polikar
Iowa State University

Multiresolution wavelet analysis of EEG signals

for the detection of Alzheimer's disease

by

Robi Polikar

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Electrical and Computer Engineering
Interdepartmental Program: Biomedical Engineering
Co-Majors: Electrical Engineering
Biomedical Engineering

Approved: Members of the Committee:

Signatures have been redacted for privacy

Signatures have been redacted for privacy

Ames, Iowa
1995

DEDICATION

To all Alzheimer's disease patients...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

This study is the product of the endless efforts of many individuals to whom I am deeply indebted.

I consider myself very fortunate to have Dr. Mary Helen Greer and Dr. Lalita Udpa as my major professors. This study would have never been completed without their help, guidance, advice, encouragement, support and friendship. They have been more than just major professors to me.

Special thanks go to my two other committee members, Dr. Fritz Keinert and Dr. William H. Brockman. Dr. Keinert was very kind to work with me on an independent study on wavelet analysis despite his very tight schedule. He has been a great help to me both in understanding the fundamental concepts of the wavelet analysis, and putting these concepts to work. He is also the coolest math professor I have ever met.

Dr. Brockman has always been my resource for all my academic questions in the electrical and computer engineering department. He even helped me choose my classes when I first came to Iowa State and had no clue where to go and what to do.

I would like to acknowledge Dr. Phillip West of the Georgia Institute of Technology Research Institute for providing me the data to use in this study, as well as for his valuable suggestions. This study would not have been possible without his

help.

I would like to extend my special thanks to Dr. Satish S. Udpa without whose efforts signal processing would not have become of such interest to me.

I would like to remember with great appreciation Dr. Ertugrul Yazgan of Istanbul Technical University, my major professor in my undergraduate studies, who first introduced me to the world of signal processing and wavelet analysis.

I am also grateful to many other professors, who made my last two years a great experience here at Iowa State. Among them I would like to name Dr. Jennifer L. Davidson, Dr. John F. Doherty, Dr. Mani Mina, and Dr. David L. Carlson from electrical and computer engineering, Dr. Richard L. Engen and Dr. Malcolm H. Crump from veterinary physiology and pharmacology, and Dr. James Noland from the English department.

I cannot forget the valuable suggestions, encouragement and help from many graduate students, among whom I especially would like to thank to Michael Shiu Chan, Mark E. Schlarmann, Clay G. Owsley, Irfan Furniturewela, Atul S. Athavale, Koray Kulcu, Kai Sun, Sarit Sharma, Ashish Dixit, Wassef M. Masri, Subhadra Gunawardana, Bum-Sup Lee, Kari L. Bjerke, and Andrea Gorrell.

I would also like to thank two hidden heroes, Mrs. Linda B. Clifford and Mrs. Nancy R. Stone, the graduate secretaries in electrical and computer engineering and biomedical engineering, respectively. They took care of all the necessary paperwork, and without their help I would never be able to fill out all those forms and catch all the deadlines.

Most importantly, I would like to thank my parents who have struggled with many difficulties, and devoted their lives to my well being and to providing me a

superior education. They were always there when I needed them. I would not be who I am and where I am today without them.

Oh...! Of course... my car... How would I have ever survived winter nights in lovely Ames, Iowa without it when I had to commute back and forth between Durham Center (home sweet Durham) and the College of Veterinary Medicine???

CHAPTER 1. BIOMEDICAL ENGINEERING, SIGNAL PROCESSING, AND THE FOURIER TRANSFORMS

1.1 Introduction

The processing and analysis of biological signals are of crucial importance in biomedical engineering for the detection of physiological abnormalities in a biological system. The development of signal processing software and hardware has enabled doctors to diagnose a patient's condition without using invasive surgical operations. Today, it is often possible to tell whether a patient has a physiological disorder by analyzing the relevant biological signals. For instance, a cardiovascular disease can often be detected by analyzing the electrocardiogram (ECG), or a neurological disease can often be detected by analyzing the electroencephalogram (EEG), etc. Therefore, the applications of signal processing are of immense value to the medical community.

There are many bioelectric signals that can be detected from the body. However, signal processing of biological signals is not limited to bioelectric signals. Non-electrical signals can be easily converted to electrical signals by transducers and can be analyzed thereafter.

This study presents an investigation of the feasibility of time-frequency analysis of non-stationary biological signals consisting of EEG waveforms from patients with Alzheimer's disease. The processed EEG signals are input to a classification algo-

rithm to distinguish EEGs from patients with the neurological disorders, from EEGs from healthy people. Both neural networks and clustering algorithms were used for classification.

In the last few decades, many signal processing techniques have found applications in biomedical engineering especially after the development of fast computers and fast algorithms.

The ultimate goal of signal processing is to extract information from a signal which is not obvious, or which can not be seen, when the signal is in its raw form. Fourier transform based techniques are most often used for analyzing a signal in terms of its spectral information, i.e., to extract one particular type of information from the signals, namely the *frequency* or *spectral* information.

Late in the 17^{th} century, a French mathematician Joseph Fourier showed that any periodic signal can be written as an infinite sum of sine and cosine terms [1]. This theory has been developed and generalized, first for non-periodic functions, and then for discrete functions (or signals). The mathematical definition of Fourier transform is expressed as

$$X(f) = \int_{-\infty}^{\infty} x(t).e^{-2j\pi ft}\, dt \tag{1.1}$$

$$x(t) = \int_{-\infty}^{\infty} X(f).e^{2j\pi tf}\, df \tag{1.2}$$

where $X(f)$ is the Fourier transform and $x(t)$ is the time domain signal.

Equation (1.1) computes the Fourier transform of the signal, and Equation (1.2) computes the inverse Fourier transform to recover the original signal from the transform. The time domain and the frequency domain representations give different

perspectives of the same signal.

A signal is said to be stationary, if the spectral components are constant at all times, requiring all spectral components to exist at all times. However, if the signal is not stationary, then the spectral content varies in time suggesting that different spectral components exist at different times. The Fourier transform (FT) of such signals only provides what spectral components exist in the signal and does not include *where in time* these spectral components exist. In other words, the Fourier transform gives a global frequency representation of the signal, which is inadequate for fully describing non-stationary signals.

Figures 1.1 and 1.2 demonstrate this concept. The time domain signal in Figure 1.1 consists of three frequency components of 50, 100 and 150 Hz. These spectral components exist in the signal at all times during the entire duration of the signal. The FT of this signal is shown below the time domain signal. Figure 1.2 shows a different signal, again formed with the sinusoids of the same frequencies. However, in this signal the spectral components are *localized* in time, i.e., different frequency components exist at different times. The plot below the time domain signal shows the FT of this signal. Note that the frequency domain representations are very similar to each other. The ripples in the second plot are due to sudden changes in frequencies. Other than these ripples, the two frequency representations are almost identical, whereas the signals are not.

The distinguishing information between these two signals can only be observed by a *time-frequency* representation.

An important point that needs to be made clear at this time is addressed by the following natural question:

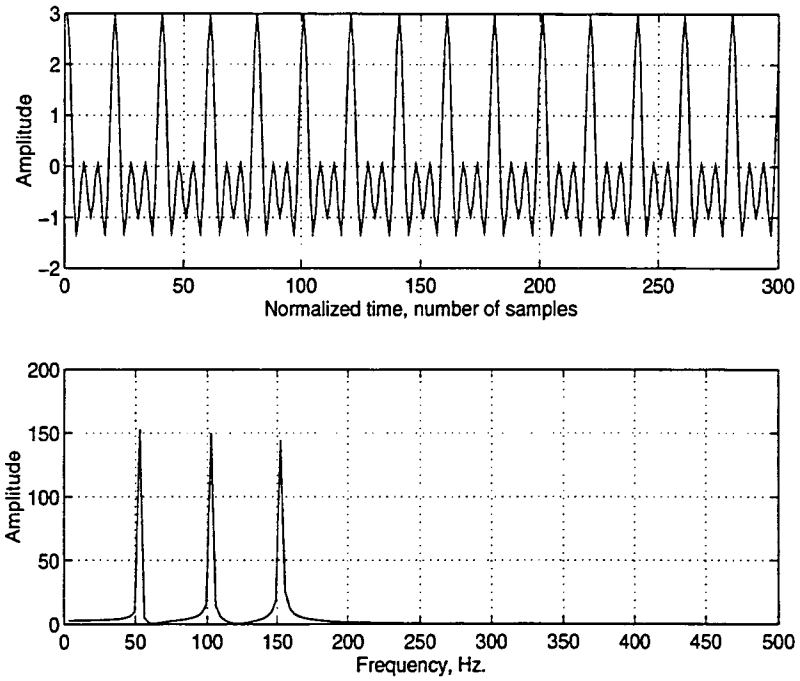

Figure 1.1: A stationary signal with three spectral components at all times and its FT

If FT gives (almost) the same transform for two different signals, how can it recover the two entirely different originals from similar looking transforms?

Fourier transform uses complex exponentials, and therefore, it is a complex valued transform. The FTs shown in Figure 1.1 and 1.2 are the *magnitudes* of these complex-valued transforms. For almost all practical applications, the magnitude of the FT is used since it can be interpreted very easily. The phase of the signal, however, is usually difficult to interpret, and therefore, it is seldomly used, unless the signal needs to be reconstructed. For the above example, the inverse transforms will yield the correct original signals in each case, because the distinguishing information between the two time domain signals lies in the phase of the Fourier transform which is utilized when computing the inverse Fourier transform.

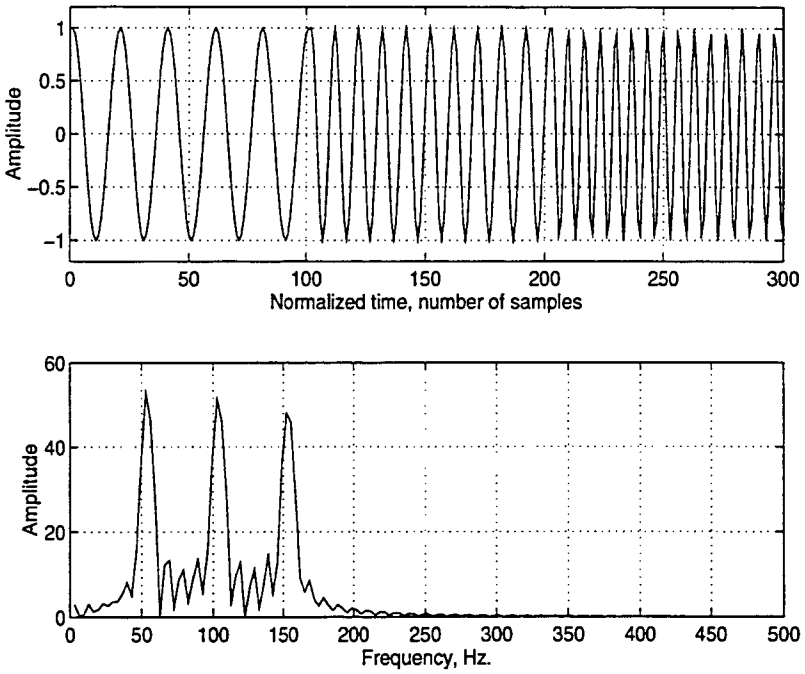

Figure 1.2: A non-stationary signal with three spectral components at different times and its FT

In summary, FT can be used to analyze any signal regardless of stationarity, if *one is only interested in the spectral content* of a signal. All bioelectric signals especially the EEG, are highly non-stationary with time varying spectra.

One of the first techniques that allowed researchers to deal with non-stationary signals is the *short-time Fourier transform (STFT)* which consists of the Fourier transform of segments of a signal that are windowed in time. A constant window function is used to partition the signal into segments, short enough to satisfy conditions of stationarity, and the Fourier transform of every segment is then computed. The end result of this procedure is a time-frequency representation of the signal. Although this technique found extensive applications in speech processing, a drawback of this technique is the use of a constant length window which leads to a limited (and

constant) resolution in time and frequency.

An alternative to the STFT, namely the wavelet transform, was developed and introduced to the signal processing community by Meyer [2] , Mallat [3], and Daubechies [4], [5] in the mid and late eighties. The resolution problem of the STFT was solved by using a variable length window in the analysis. The wavelet transform simply calculates the correlation between the signal and a window function at different scales (or frequencies), providing different time and frequency resolutions at different scales, in contrast to the constant time and frequency resolutions provided by the short time Fourier transform. The name "wavelet analysis (transform)" was thought to be appropriate since the kernel of this transform is a "little wave".

Thakor et al. [6] and Trejo and Shensa [7] have recently reported promising results for the application of the wavelet analysis on evoked potentials.

A brief overview of the short time Fourier transform will be discussed in Chapter 2 to present the time and frequency resolution concepts including the effects of window length on these resolutions.

The continuous wavelet transform will be introduced in Chapter 3, emphasizing its capability to solve the resolution problem of the STFT. A number of examples will be presented to demonstrate the effects of different parameters on time and frequency resolutions. The discrete wavelet transform, for obtaining the time-frequency representation of a discrete signal, will be introduced in Chapter 4.

The classification and clustering algorithms used in this study, namely, the multilayer perceptron with backpropagation algorithm and the k-means algorithm for clustering are discussed in Chapter 5.

Since the EEG signals are used in this study, electroencephalography (the electri-

cal activity of the brain) and its features that are used to detect neurological disorders are presented in Chapter 6. The properties of event-related potentials that are used to detect dementias such as Alzheimer's disease are employed.

The experiment performed to obtain the data, the implementation of the overall algorithm to distinguish the signals from dementic and normal patients are explained in Chapter 7. Finally the results obtained are discussed along with some concluding remarks.

CHAPTER 2. TIME-FREQUENCY REPRESENTATIONS AND THE SHORT TIME FOURIER TRANSFORM

2.1 Time Frequency Representations

In the first chapter the Fourier transform was briefly introduced and the inadequacy of Fourier transforms for the analysis of non-stationary signals was illustrated. If the signal is non-stationary, i.e., it has a *time varying spectrum*, it is essential that a transform be used that provides both time domain and frequency domain information at the same time. A number of time-frequency transforms have been developed over the years. A discussion of these time-frequency representations is presented here.

Time-frequency representations (TFR) are used to analyze signals with time varying spectra. There are many TFRs available, and they can be categorized as *linear TFRs* and *quadratic TFRs* [8]. Two main linear TFRs, namely, the STFT and the wavelet transform (WT) were used in this study. Linear TFRs satisfy the linearity condition

$$x(t) = a_1.x_1(t) + a_2.x_2(t) \Rightarrow T_x(t,f) = a_1.T_{x_1}(t,f) + a_2.T_{x_2}(t,f) \qquad (2.1)$$

In this equation, $T(.)$ represents the transformation, $x(t)$ represents the signal, a_1 and a_2 any two constant numbers, and t and f are the time and frequency variables, respectively. As can be seen from Equation (2.1), the transformation $T(.)$ transforms

a one dimensional signal $x(t)$ into a two dimensional signal of time and frequency $T_x(t, f)$.

The STFT will be explained in some detail in this chapter since it provides the underlying principle to the wavelet transform.

2.2 The Short Time Fourier Transform (STFT)

Consider the following approach for obtaining a two-dimensional time-frequency representation.

If a short segment of the signal is *windowed* out and the FT of this segment is computed, the frequency information obtained would be local, corresponding to the time location of the window. If this window is narrow enough, the signal in the segment can be assumed to be stationary and the computed FT will be an acceptable frequency representation of *that portion* of the signal. If this window is shifted in time to *window* other portions of the signal, the computation of Fourier transforms of each portion results in a time - frequency representation of the signal. This is the underlying principle of the short-time Fourier transform (STFT).

The window used is a function of compact support in time, i.e., it has a finite duration in time. When this signal is multiplied by the window, all information that falls outside the window is suppressed, retaining only the part of the signal overlapping the time duration of the window. The simplest window is the rectangular window defined as

$$w(t) = \begin{cases} 1, & 0 < t < T \\ 0, & otherwise \end{cases} \tag{2.2}$$

where T is the duration of the interval, i.e., window length in time.

However, the rectangular window is very seldom used because of the high frequency components introduced by the sharp transitions, when the window function goes from zero to one and from one to zero. This causes ripples, an unwanted noise like effect. This effect is known as the *Gibb's Phenemonon*, and it can be reduced to a great extent if the window is a smoothly changing function. The Gaussian, Bartlett and Hamming windows are some of the most commonly used window functions.

The short time Fourier transform is defined by the following equation

$$STFT_x^{\omega}(t_0, f) = \int_{-\infty}^{\infty} [x(t).\omega^*(t - t_0)].e^{-2j\pi ft}\, dt \tag{2.3}$$

In the above equation *x(t)* is the signal to be transformed, *ω(t)* is the window function centered at time $t = t_0$, and "$*$" represents the complex conjugate. $STFT_x^{\omega}(t_0, f)$ is read as: Short time Fourier transform of $x(t)$ with respect to the windowing function $\omega(t)$ at time t_0, and frequency f. Note that although the integral is over all values of time, *for every* t_0, the non-zero contribution comes only from the time duration of the window function.

By sliding the window in time and computing the Fourier transform of the windowed segment for different values of t_0, the time-frequency representation of the signal can be obtained.

Figure 2.1 shows an arbitrary discrete-time, time-domain signal $x(t)$. The x-axis indexes the sample numbers of the signal. In other words, the signal in the figure runs from $t = -20$ to $t = 20$ with integer increments resulting in a total of 41 samples. Three windows are superimposed on the signal to demonstrate the shifting of the window. The first window is centered around $t_0 = -12$ and the window function, $w(t)$, can be mathematically denoted as $w(t + 12)$. At this instant the part of the

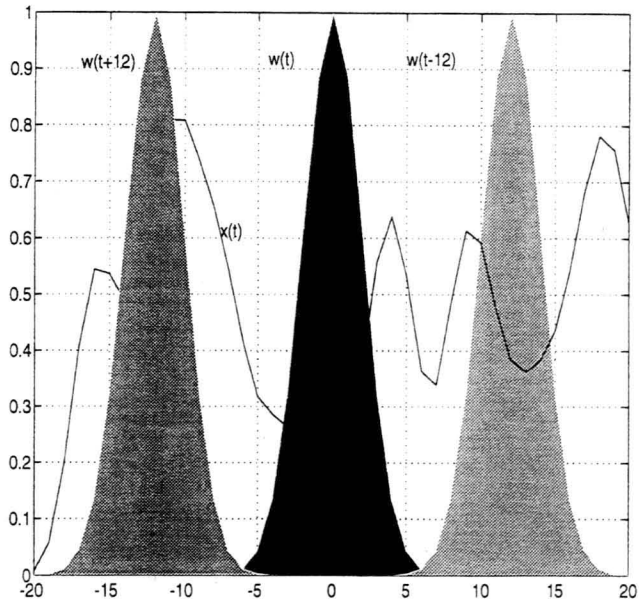

Figure 2.1: Short time Fourier transform

signal that falls outside the window will not contribute to the integral in Equation (2.3). The STFT at this time instant will be the Fourier transform of the function within the compactly supported window.

At time instant $t = -11$ the window function will shift to the right by one sample, and the above procedure will be repeated. Figure 2.1 shows two more windows centered at $t = 0$ where the window function is simply $w(t)$, and $t = 12$ where the window function is $w(t - 12)$. At every time instant different segments of the signal are selected by the window, and the corresponding Fourier transforms are computed.

The STFT is dependent on the choice of the window function. This window is called the *analysis window* and must be chosen wisely according to the signal to be analyzed.

If a wide window function is chosen, the FT will correspond to a long time interval, and therefore, it will yield poor time resolution. In contrast, narrow window functions will result in more accurate time resolution.

Can the window to be chosen be infinitely narrow so that a frequency representation for every time *instant* will be obtained? The answer is no as explained below.

Although choosing an arbitrarily narrow window will give very good time resolution, all frequency information will be lost. To demonstrate this, consider the extreme case, where the window function is the delta function $\delta(t)$. The definition and the important properties of the delta function are given below.

$$\delta(t) = \begin{cases} \infty & \text{for t=0,} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

$$\delta(t - t_0) = \begin{cases} \infty & \text{for t=}t_0\text{ ,} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

$$\int_{-\infty}^{\infty} x(t).\delta(t)\,dt = x(0) \tag{2.6}$$

$$\int_{-\infty}^{\infty} x(t).\delta(t - t_0)\,dt = x(t_0) \tag{2.7}$$

When we use the delta function for windowing:

$$\omega(t) = \delta(t) \Rightarrow STFT_x^{\omega}(t, f) = \int_{-\infty}^{\infty} [x(t).\delta^*(t - t_0)].e^{-2j\pi ft}\,dt \tag{2.8}$$

From the property of the delta function given in Equation (2.6), the integral in Equation (2.7) will only have a non-zero value at t=t_0. The transformed signal at $t = t_0$ will be the signal itself at $t = t_0$, with a phase factor.

$$\omega(t) = \delta(t - t_0) \Rightarrow STFT_x^{\omega}(t, f) = x(t_0).e^{-j2\pi f t_0} \tag{2.9}$$

Since, we are interested in the magnitude of the transforms, and the magnitude of the complex exponential term is "1", *the magnitude of the transformed signal, and the signal itself will be exactly the same*, i.e, $STFT_x^{\omega}(t, f) = x(t_0)$. In other words, the time information is fully retained, but all frequency information is lost. Figure 2.2 illustrates this concept.

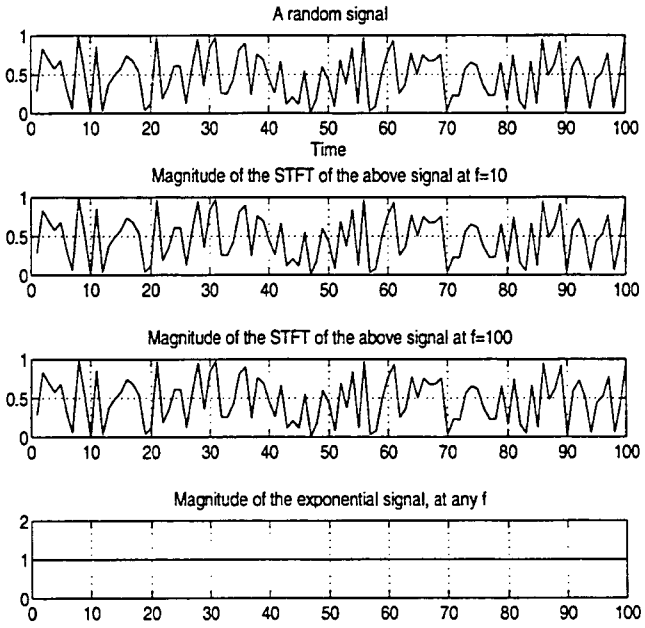

Figure 2.2: Sample transforms for $\omega(t) = \delta(t)$

The first plot in Figure 2.2 is a random signal to be transformed using the delta function for windowing. The second and third plots are calculated using Equation (2.8). In the second plot "f" was taken as ten, and in the third one it was taken as one hundred, to show that the frequency information is really lost. This follows from the following identity

$$|e^{ix}| = 1 \qquad \text{for all x} \tag{2.10}$$

where $i = \sqrt{-1}$.

Now, consider the other extreme condition where the window function is infinitely long, e.g., $w(t) = 1$. In this case, the STFT will be no different from FT, where all the frequency information is kept perfectly, but all the time information is lost. Note that the FT of an infinitely long time window is an infinitely narrow frequency window. $(x(t) = 1 \Rightarrow X(f) = \delta(f))$. Therefore, in order to get a good frequency resolution, the window should be as wide as possible in the time domain which corresponds to a narrow window in the frequency domain.

From the above discussion, it follows that the wider the window function, the better the accuracy in frequency information (frequency resolution), and the narrower the window function, the better the accuracy in time information (time resolution). This paradox is the biggest handicap of the STFT. What kind of window should be chosen? The answer is application dependent.

2.3 Heisenberg Inequality, Uncertainty Principle, and the Limitations of the STFT

The uncertainty principle was first introduced by the German physicist Heisenberg to explain the relationship between the momentum and the location of a moving particle. The original Heisenberg's inequality is as follows

$$2\pi h\Omega.T \geq h/2 \tag{2.11}$$

where, T and $2\pi h\Omega$ are the uncertainties in the position and the momentum of the particle, respectively, and h is the Planck's constant. The above equation states that it is not possible to know exactly the momentum and the location of a moving object and that the product of the uncertainties in the position and the momentum of the particle cannot be less than a constant number $(h/2)$. This argument can be applied to the computation of the STFT.

Two spikes in the time domain can only be distinguished from each other if they are Δt apart, and two spectral components in the frequency domain can only be distinguished from each other if they are Δf apart. The uncertainty principle states that Δt and Δf cannot be arbitrarily small. Their product is lower bounded:

$$\Delta t.\Delta f \geq \frac{1}{4\pi} \tag{2.12}$$

Δt and Δf are measures of time and frequency resolution, respectively. Δt and Δf are defined as follows for windows centered around the origin [9]

$$\Delta f^2 = \frac{\int f^2.|\Omega(f)|^2\, df}{\int |\Omega(f)|^2\, df} \tag{2.13}$$

$$\Delta t^2 = \frac{\int t^2 . |\omega(t)|^2 \, dt}{\int |\omega(t)|^2 \, dt} \tag{2.14}$$

where, $\Omega(f)$ is the Fourier transform of the window function $\omega(t)$ that is used in the STFT. In the case of STFT, $\omega(t)$ is usually a Gaussian function since Gaussian functions satisfy Equation (2.12) with the equality.

The implication of Equation (2.12) is that *good* time resolution can only be obtained at the expense of good frequency resolution and vice versa.

In summary, the STFT provides a time-frequency representation of the signal by using a moving window to modulate the signal and computing the Fourier transform for each position of the window. If the window is narrow then the procedure will result in good time resolution and poor frequency resolution, whereas wider windows will result in poor time resolution and good frequency resolution.

The examples presented in the following section will make the above discussion clear.

2.4 Examples

A computer program for a discrete implementation of the continuous transform was developed. Although there is a discrete STFT, it will not be explained here, since a similar discussion will take place in Chapter 4, where discrete wavelet transform is introduced.

The examples given in this part will correspond to simple test signals, mainly composed of sinusoids of different frequencies, occuring at different times.

The Gaussian window function defined by

$$\omega(t) = e^{-at^2/2} \tag{2.15}$$

was used where the constant "a" determines the width of the window.

2.4.1 Example 1

The first example considers a single frequency sinusoidal signal. Figure 2.3 shows a 100 millisecond long, 250 Hz pure sinusoidal signal, sampled at 1 kHz.

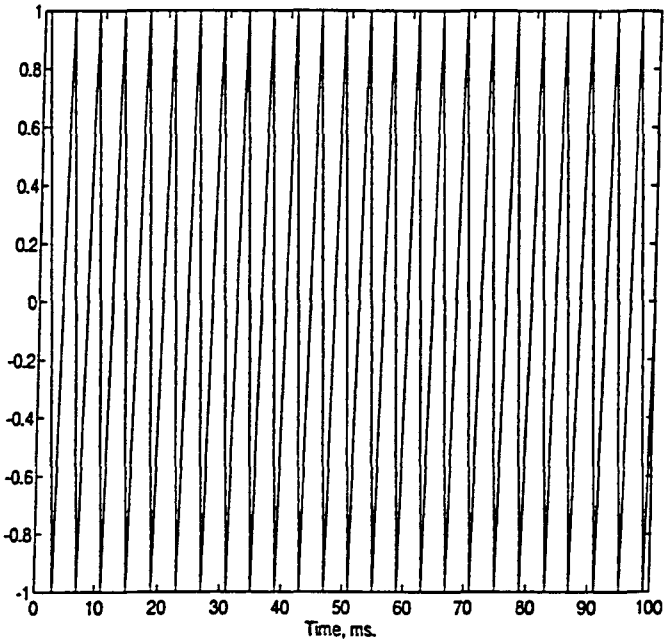

Figure 2.3: Single frequency cosine signal at f=250 Hz

Figure 2.4 shows the window function. The width of this function was chosen as 20 ms. However, for this particular example, it really does not matter since the signal is stationary. The width is changed by using different values of a, in the definition of the window function given above. The effect of "a" will be shown in a later example.

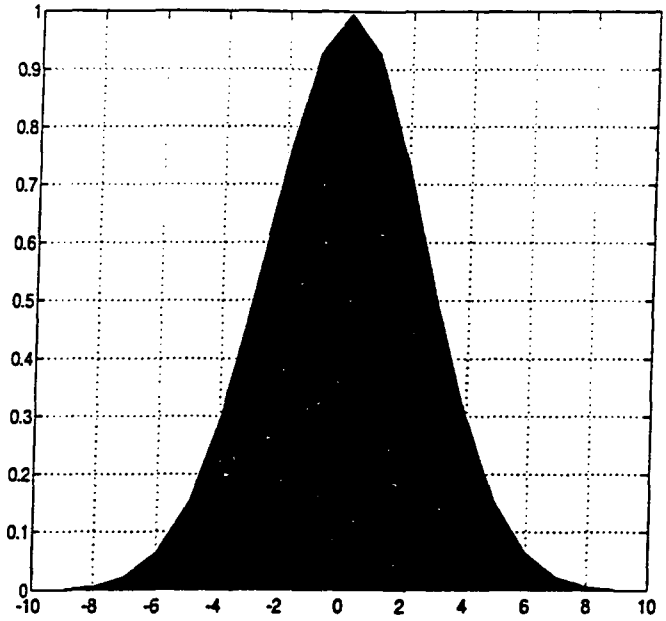

Figure 2.4: The window function

Figure 2.5 shows the STFT of this signal. The two peaks seen on the graph correspond to 250 Hz. The two peaks in Figure 2.5 are a result of the symmetry property of the Fourier transform and are due to the fact that discrete Fourier transforms are always periodic in 2π. It should be noted that the Fourier transforms computed here are actually discrete Fourier transforms. The frequency axis corresponds to 2π radians. For discrete time signals, frequency is measured in terms of radians, rather than Hz. The frequency axes in the plots are given in terms of Hz for easy interpretation only. The sampling frequency for this signal was 1000 samples/sec, and therefore the STFT is periodic around 500 Hz (or π radians). The frequencies corresponding from π to 2π are sometimes referred as negative frequencies.

For single frequency sinusoids as in this case, the Fourier transform integral will

result in two impulses:

$$x(t) = \cos 2\pi 250t \Leftrightarrow X(f) = \frac{1}{\pi}\delta(-250) + \frac{1}{\pi}\delta(250) \tag{2.16}$$

Note, however, that the two peaks corresponding to 250 Hz are not impulses in Figure 2.5. They include other spectral components around 250 Hz. This is a result of using a finite length window in the STFT (as opposed to the infinite length window in the Fourier transform) which smears the impulses as seen in Figure 2.5. The other spectral components around 250 Hz are introduced due to the transition edges of the window.

Figure 2.6 is the same graph as the previous one, except that it shows the plot from a different angle. There are two angular parameters used for this purpose. The first one is azimuth, or horizontal rotation. The plot in Figure 2.5 had a rotation of $-37.5°$ and the plot in Figure 2.6 has a rotation of 90°. Azimuth rotates the plot about the z-axis, with positive values indicating counter-clockwise rotation of the viewpoint. The second parameter is vertical elevation. Positive values of elevation correspond to looking at the plot from above, negative values correspond to looking from below. The elevation was 30° in Figure 2.5 and 20° in Figure 2.6 . These parameters will be denoted by "HR" (for horizontal rotation) and "VE" (for vertical elevation).

Figure 2.7 corresponds to another view of the same plot with the viewing parameters HR=$-120°$, VE=50°.

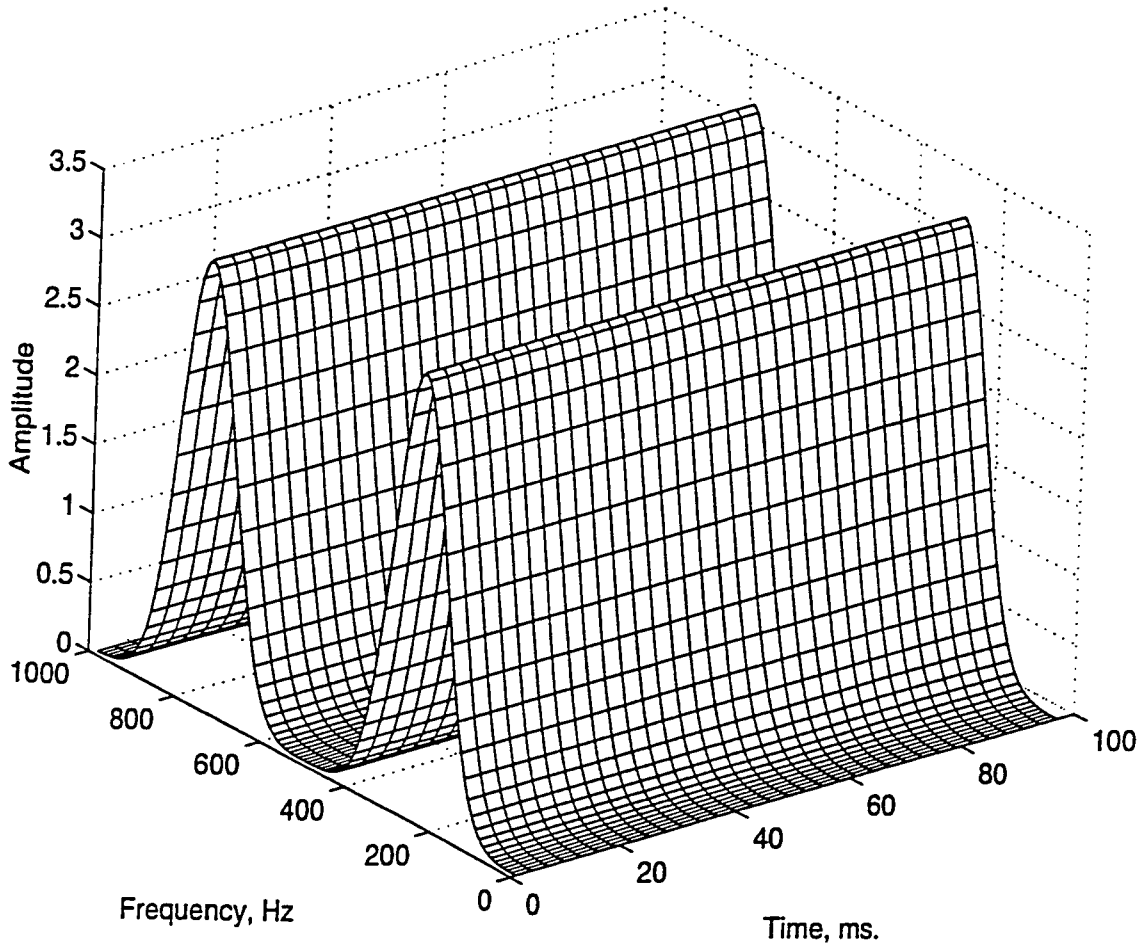

Figure 2.5: STFT of the given signal

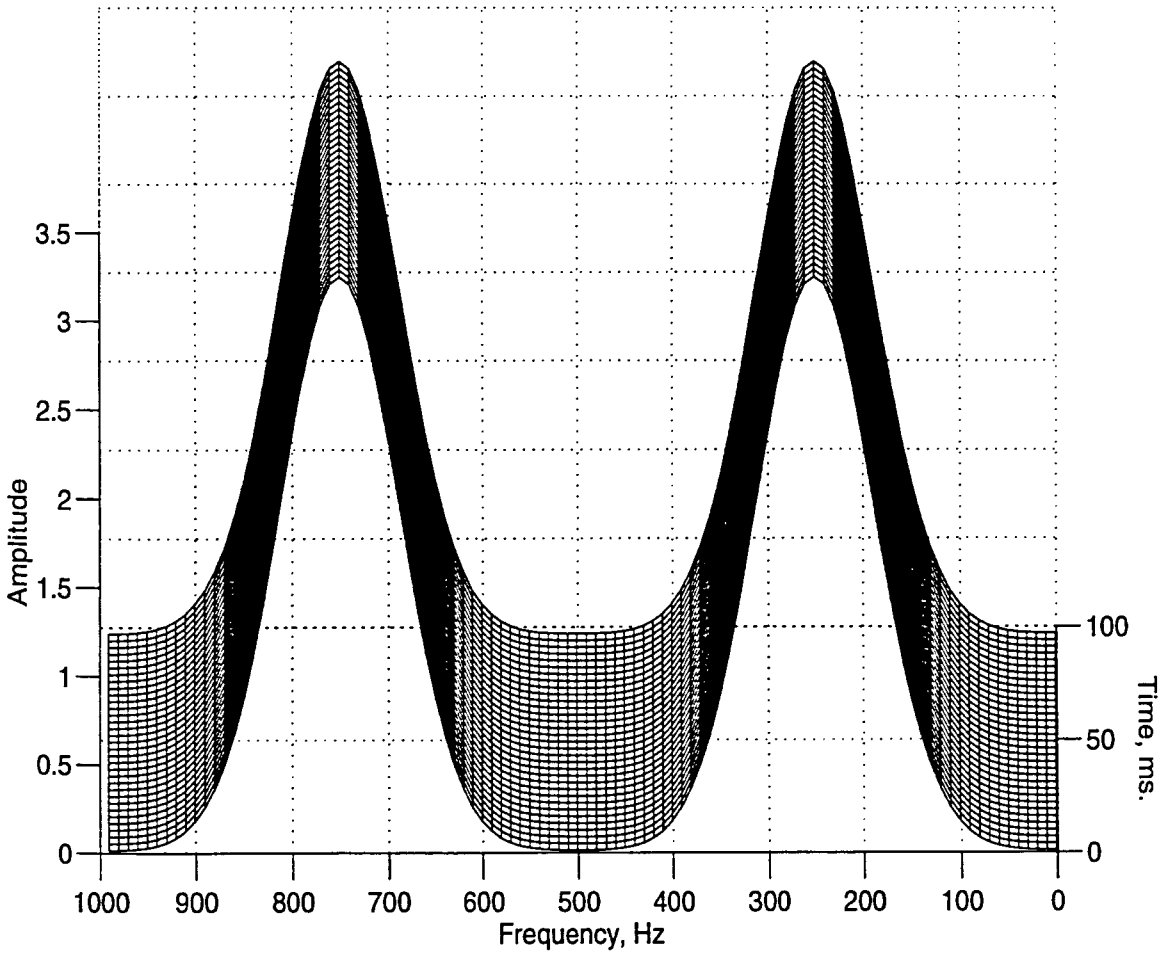

Figure 2.6: STFT of the signal in Figure 2.3, HR=−90°, VE=20°

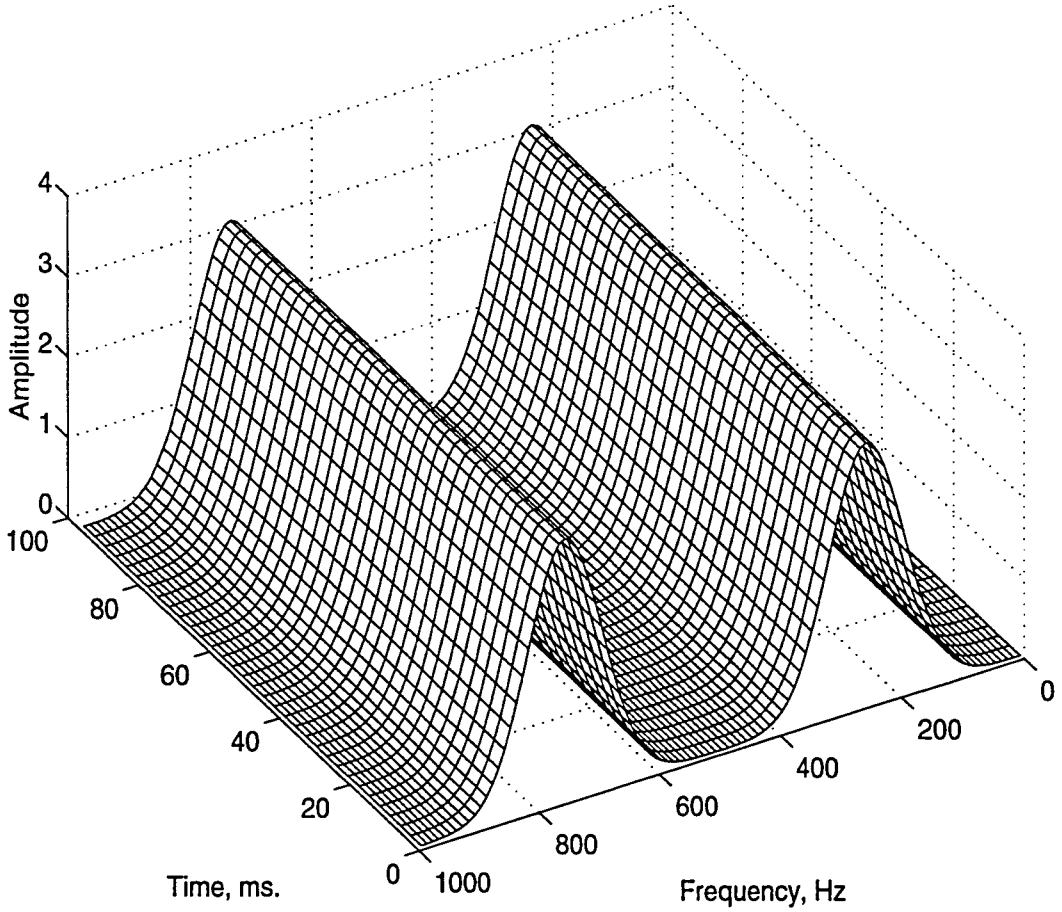

Figure 2.7: STFT of the signal in Figure 2.3 , HR=−120°, VE=50°

2.4.2 Example 2

The following example is intended to show the effects of the window length on the short time Fourier transform of a stationary signal with more than one spectral component. A stationary signal is chosen for this example so that the effect of window length on the frequency resolution only can be seen without worrying about the time resolution.

Figure 2.8 shows the test signal chosen for this example. This signal has two spectral components at 50 Hz and at 250 Hz. These frequencies exist through out the entire duration of the signal, i.e., the signal is stationary.

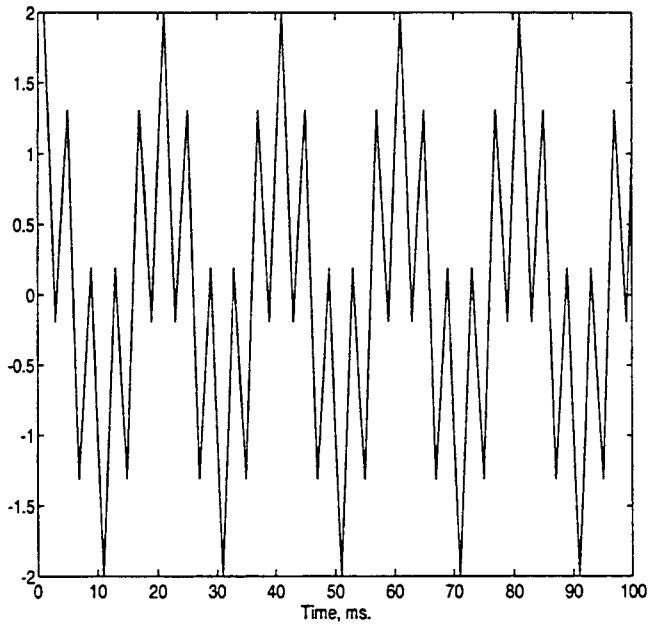

Figure 2.8: A simple, sinusoidal signal with two spectral components at 50 and 250 Hz

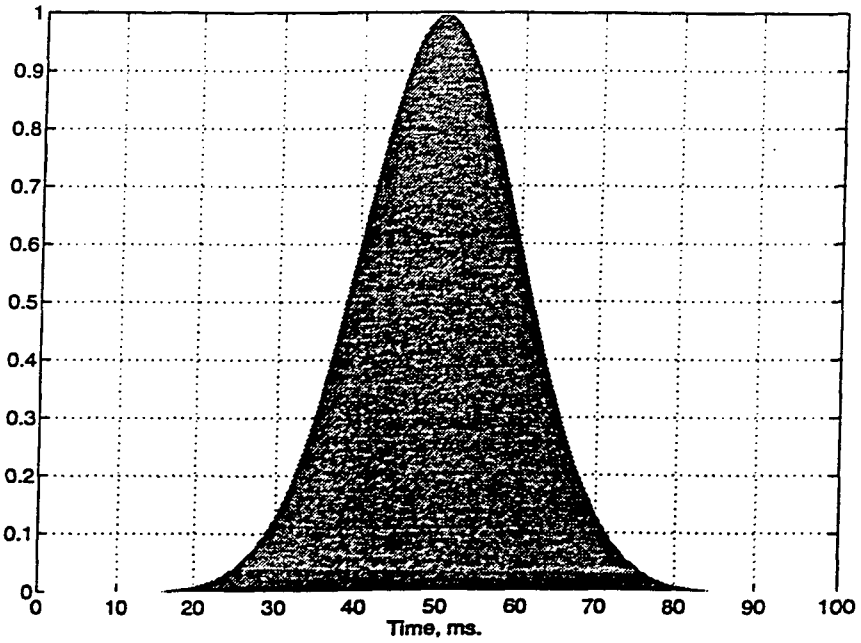

Figure 2.9: The window function, a=0.01

The STFT of this signal was computed for two different window lengths. Figure 2.9 shows the first window which is relatively wide in time. This window function can mathematically be expressed as

$$\omega(t) = e^{-0.01t^2/2} \tag{2.17}$$

Figures 2.10 and 2.11 show two views of the computed STFT for the signal using the window given by Equation (2.17). Note that the peaks correspond to 50 and 250Hz. The two peaks are well separated from each other which shows that the window used was "wide" enough in time domain or narrow enough in frequency domain to give the desired frequency resolution.

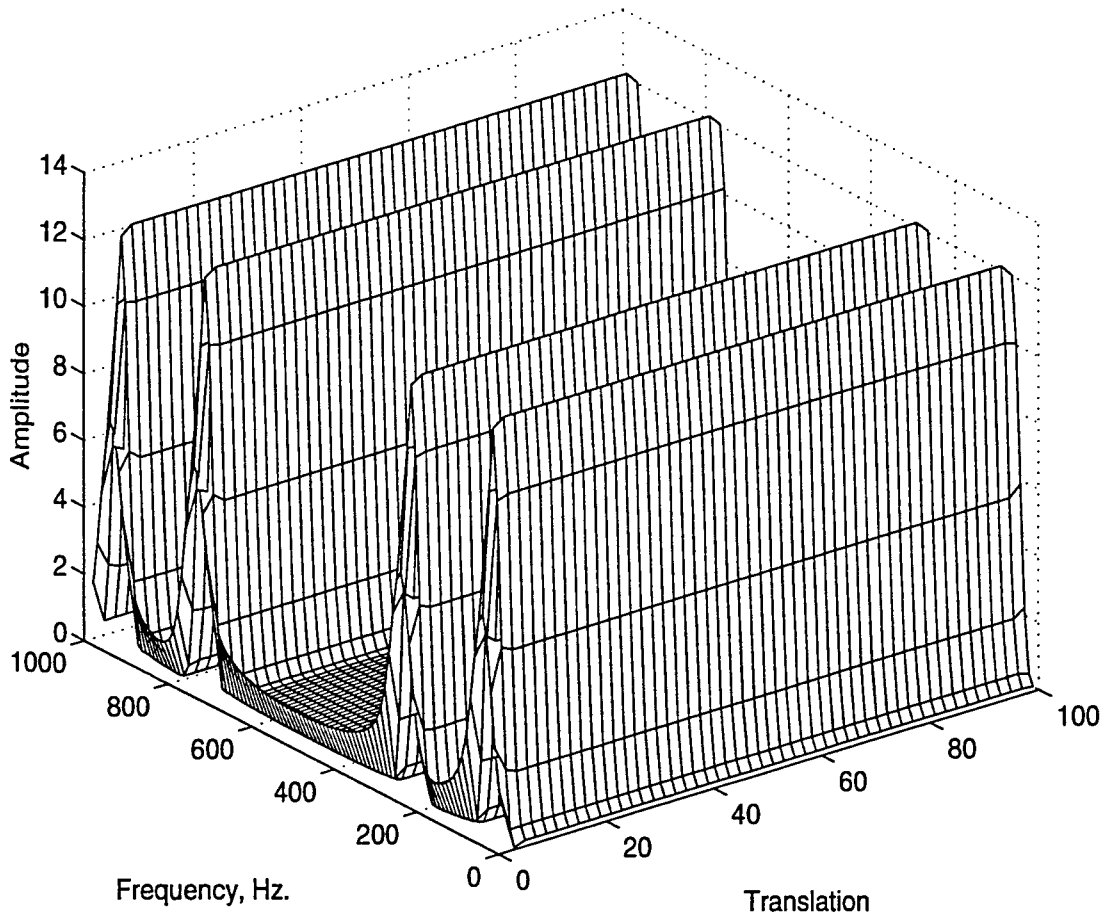

Figure 2.10: STFT of the signal in Figure 2.8, HR=−37.5°, VE=30°

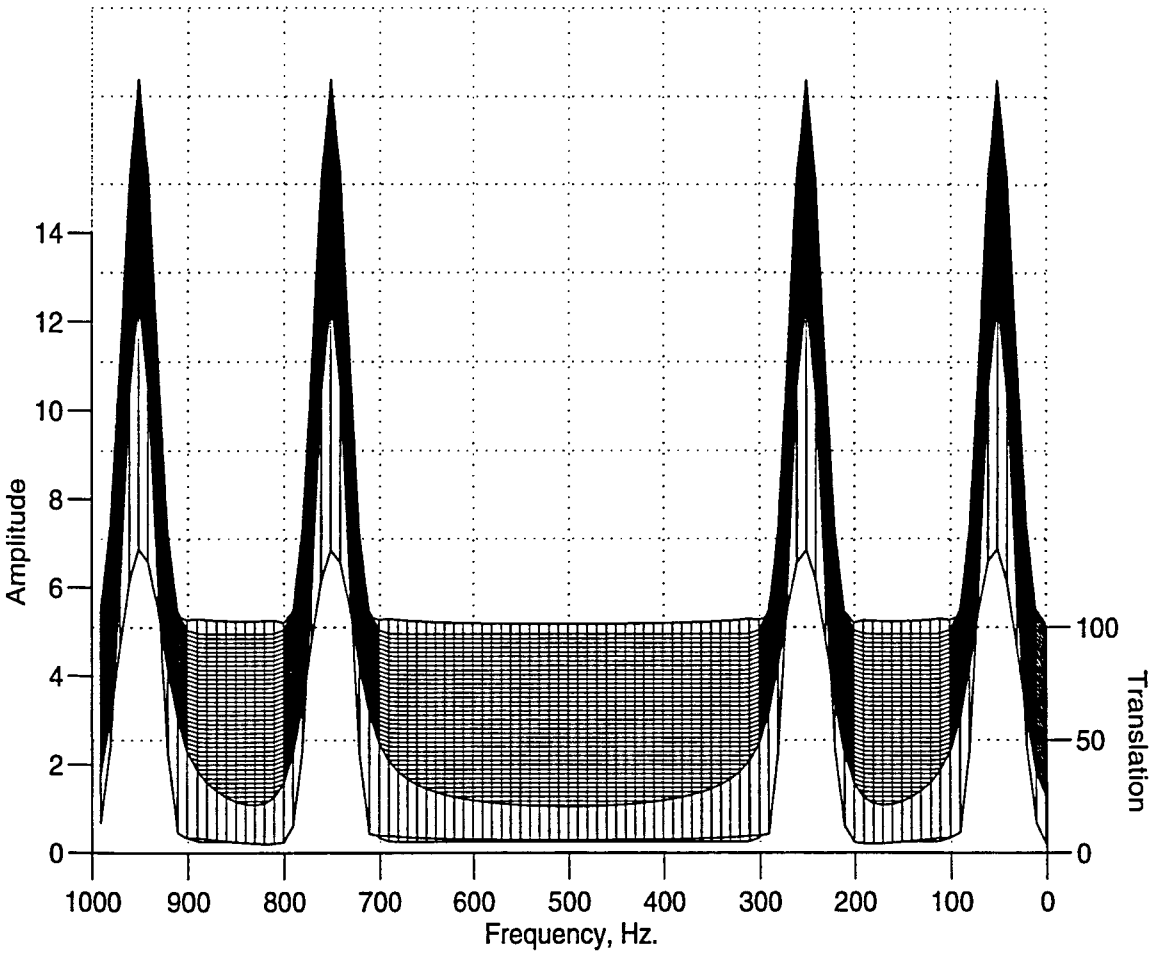

Figure 2.11: STFT of the signal in Figure 2.8 , HR=−90°, VE=20°

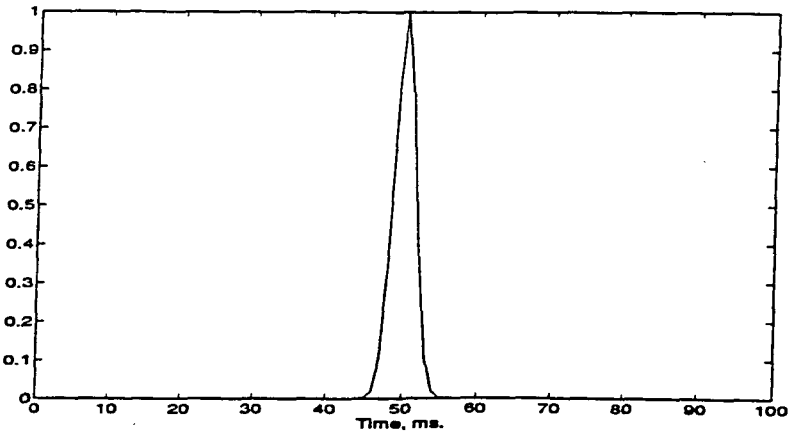

Figure 2.12: The window function, a=0.5

If the signal contains spectral components that are relatively close to each other, a wider window (in time) is necessary to capture them in the frequency domain. However, a large value for a (wide window) may violate the stationarity condition.

Consider next the window function with a=0.5 which is defined as in Equation (2.18) and plotted in Figure 2.12. This makes the window narrower in the time domain but wider in the frequency domain relative to the window in Figure 2.9. The transform results are presented in Figures 2.13 and 2.14. Note that the peaks corresponding to two frequencies are not distinguishable.

$$\omega(t) = e^{-0.5t^2/2} \tag{2.18}$$

Figures 2.13 and 2.14 suggest that the new value of "a" was inadequate to give the frequency resolution needed to separate two peaks. From the given examples, it follows that it may be necessary to sacrifice time resolution in order to obtain good frequency resolution or vice versa.

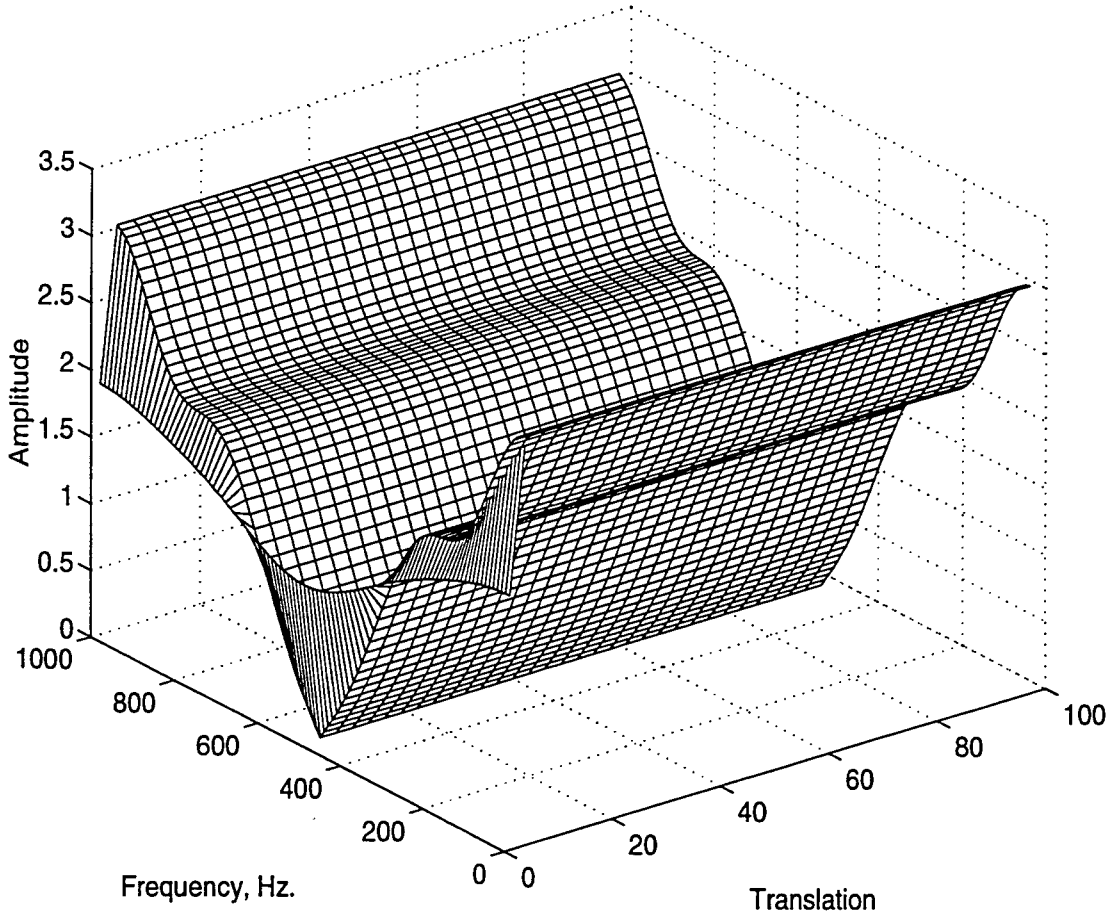

Figure 2.13: Same STFT with Figure 2.10 with a new window, HR=−90°, VE=20°

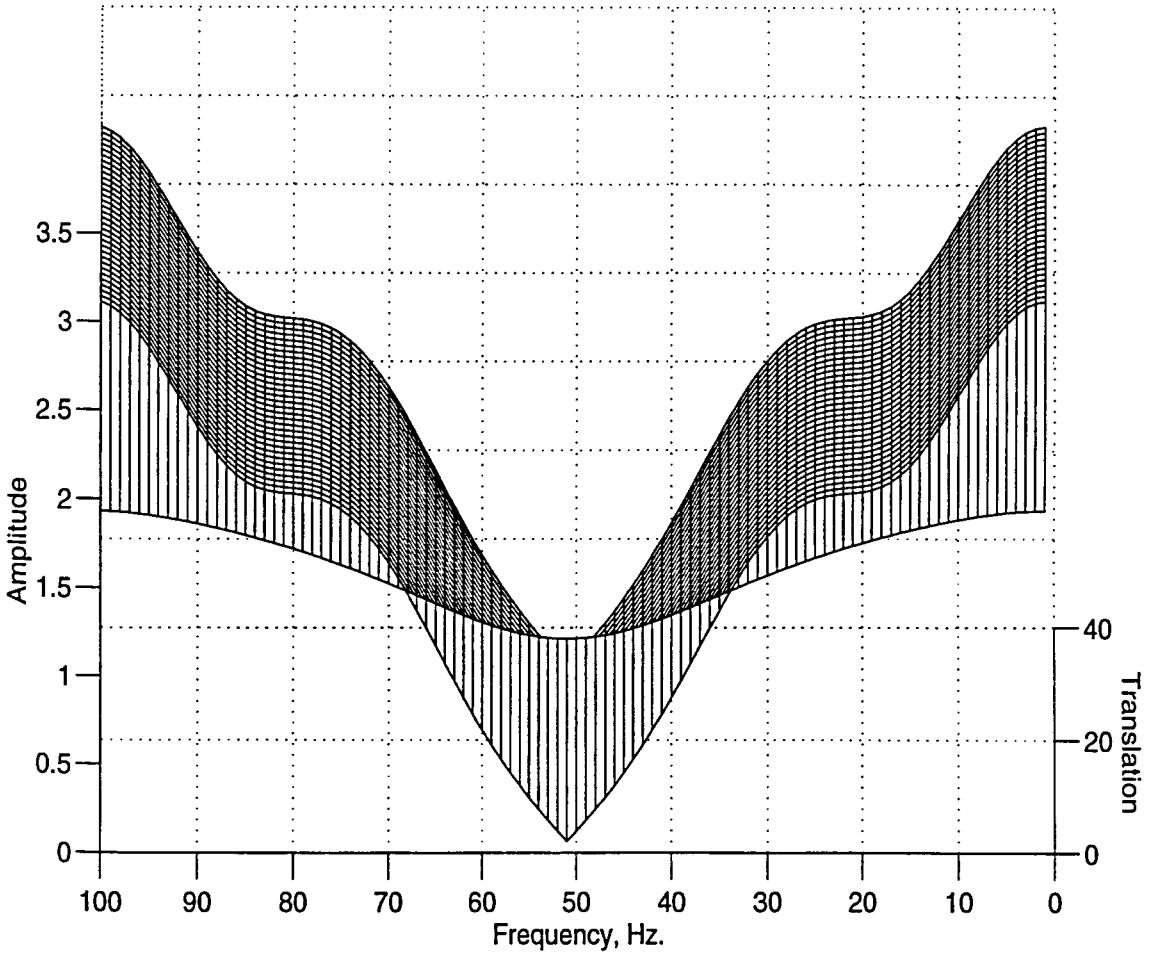

Figure 2.14: Same STFT with Figure 2.11 with a new window, HR=−90°, VE=20°

2.4.3 Example 3

This example demonstrates the performance of the STFT on non-stationary signals. The previous example was given to demonstrate the basic principles of STFT using the stationary signals. However, STFT is hardly ever used for stationary signals since the conventional FT gives a perfect frequency resolution, and time resolution is not necessary since the spectral components do not change in time for stationary signals.

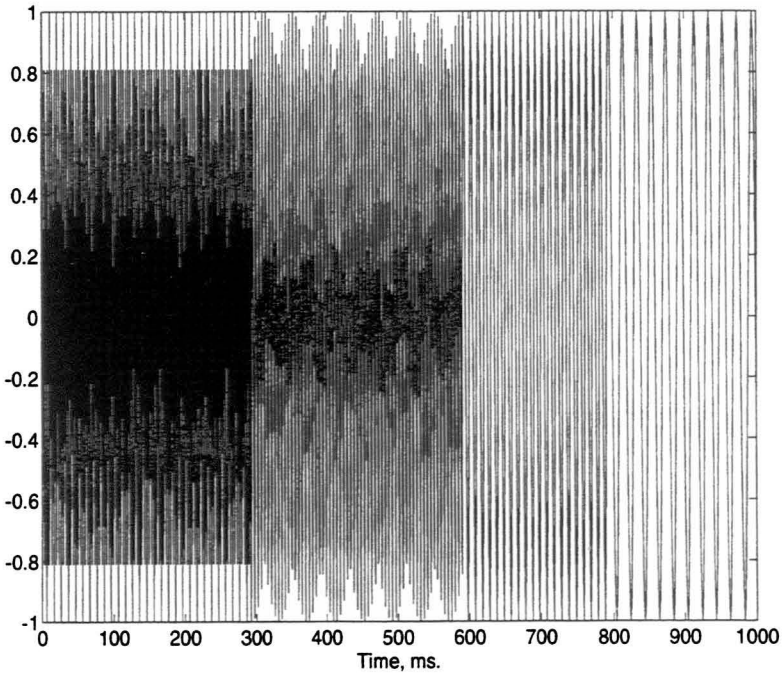

Figure 2.15: A non-stationary signal with four spectral components at different times

From the previous discussions, it is known that the window function should be chosen according to the signal. For the particular signal shown in Figure 2.15 three STFTs will be shown corresponding to three different windows. These windows are shown in Figure 2.16.

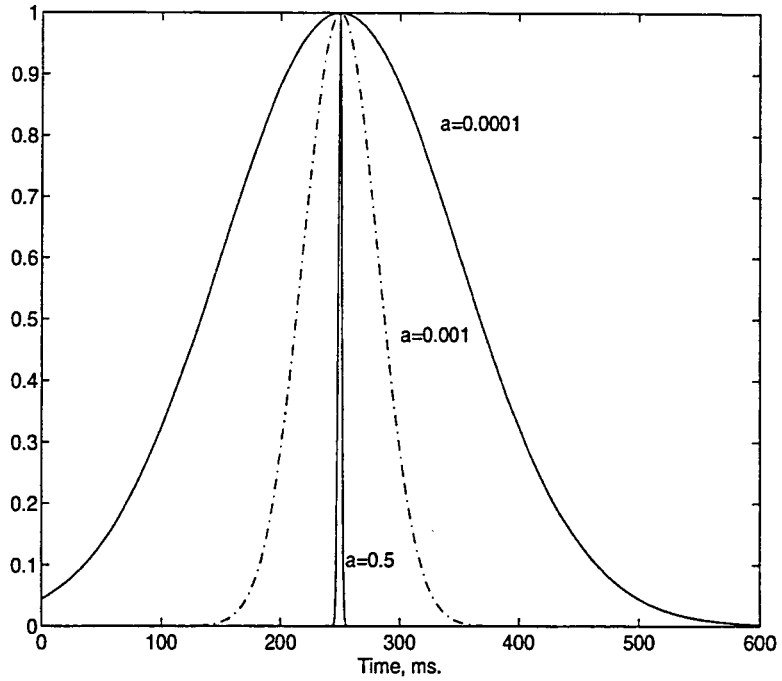

Figure 2.16: The window functions used to transform the non-stationary signal

The narrowest window (with a=0.5) will show a good time resolution, but a poor frequency resolution. The widest window (with a=0.0001) will produce the opposite effect, where the different spectral components can be distinguished very well in frequency, but not so well in time. The spectral components of this signal were chosen to be quite far from each other so that a suitable window could be easily found. For many practical purposes, however, choosing a suitable window can be quite challenging.

Figures 2.17 and 2.18 are the STFTs computed by using the window with a=0.001. The two viewing parameters were chosen to make the time and frequency resolutions as observable as possible. The four frequency components can be easily seen. Figure 2.17 shows that the spectral components are well separated in time.

However, some superposition can also be seen. This means that the spectral components are not *perfectly* separated in time.

Figure 2.18 is the same plot with different viewing parameters. The spectral components are well separated from each other in frequency. No spectral component corresponding to a center frequency spreads into a spectral component of another center frequency, even though each component has a variance of itself around their center frequencies (as a result of the uncertainty principle). When the variance of a spectral component does not smear the other components, that small variance can be neglected, and that resolution is considered to be good enough for most practical purposes. Therefore, *this particular window* provides a good frequency resolution *for this particular signal.*

Figures 2.19 and 2.20 correspond to the STFTs of the same signal with the widest window. Figure 2.19 shows the view from the time axis of the transformed signal. The overlapping of the spectral components in time is clearly seen in this figure. This, of course, is the result of the wide window covering a large time interval which results in poor time resolution.

Figure 2.20 shows the transformed signal from the frequency axis. A very good frequency resolution, even better than the previous one, is seen here, due to the wider window. This choice of the window function demonstrated an extreme situation in which there was a very good frequency resolution, but very poor time resolution.

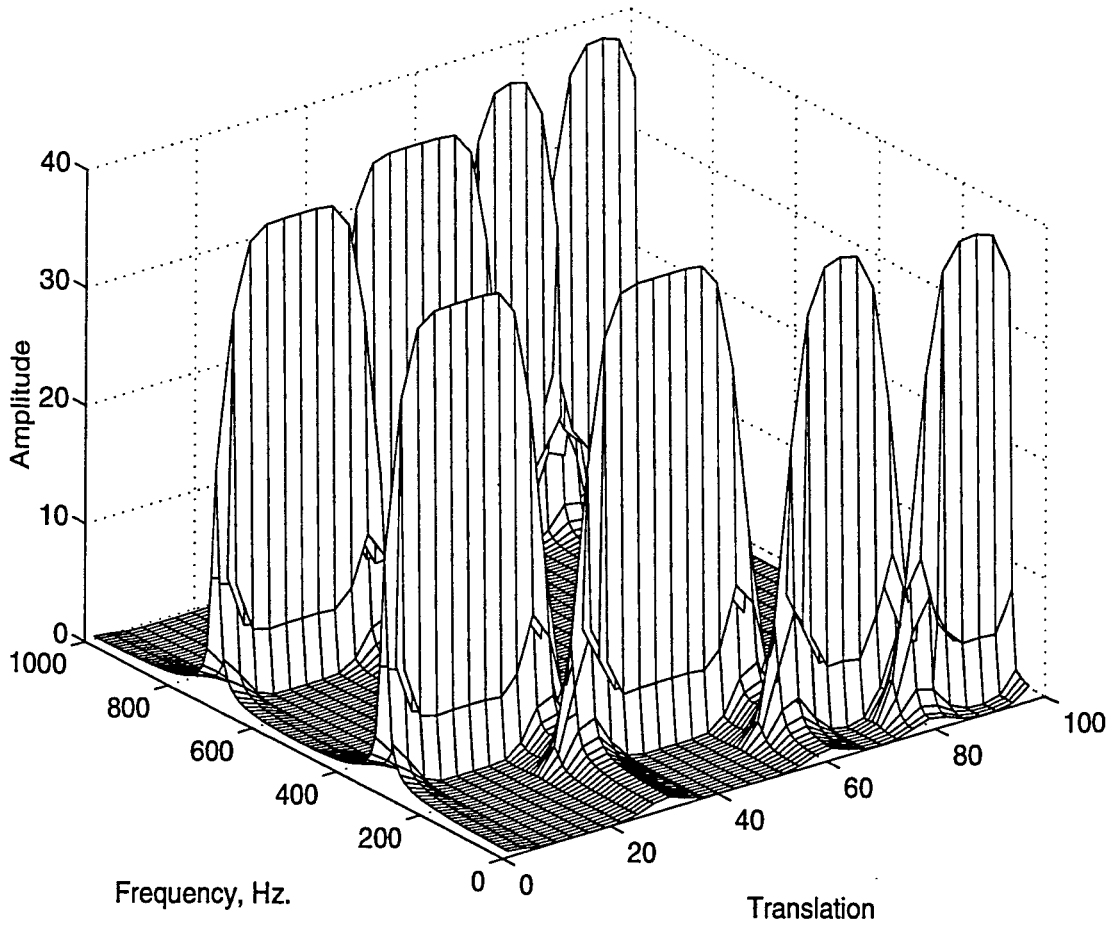

Figure 2.17: STFT of the signal in Figure 2.15, HR=−37.5°, VE=30°

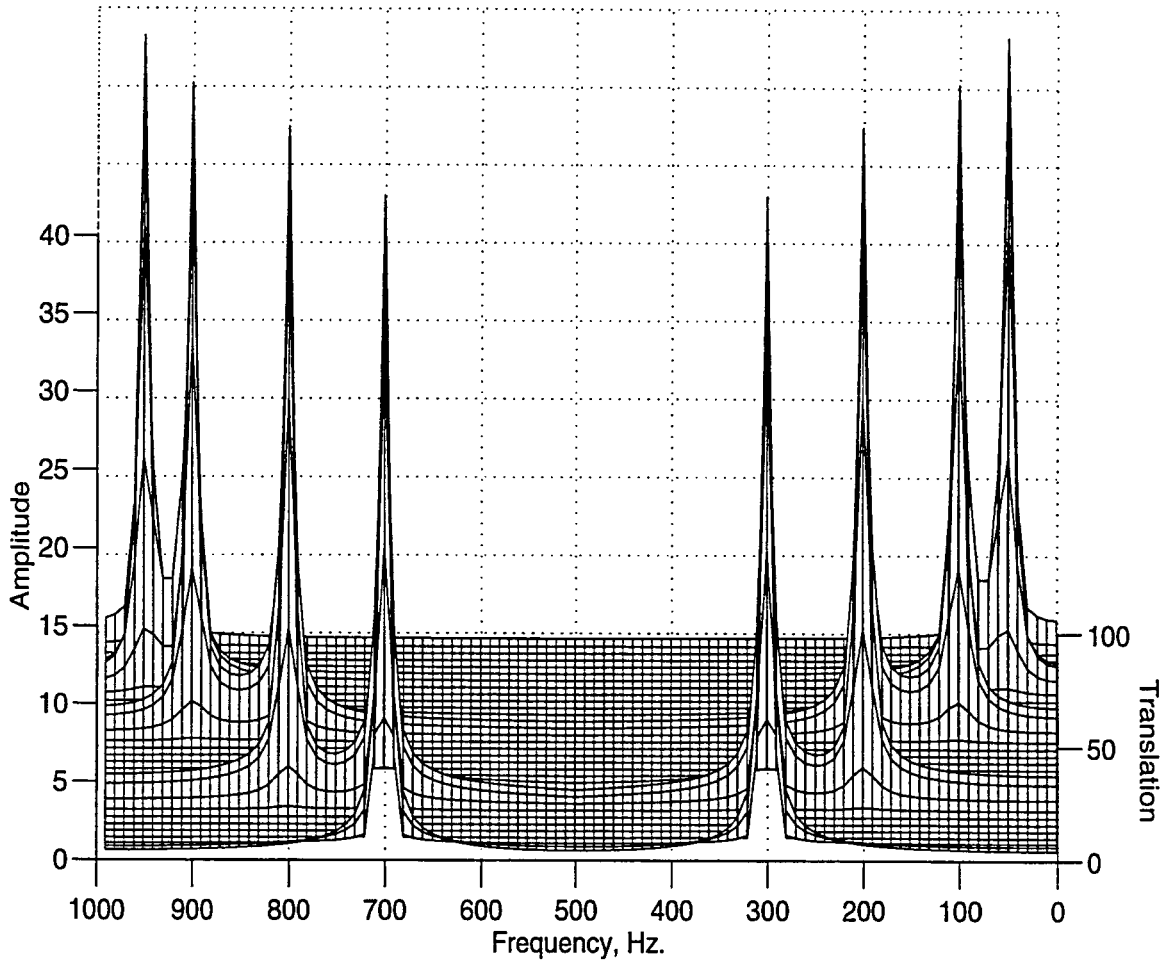

Figure 2.18: STFT of the signal in Figure 2.15 , HR=−90°, VE=20°

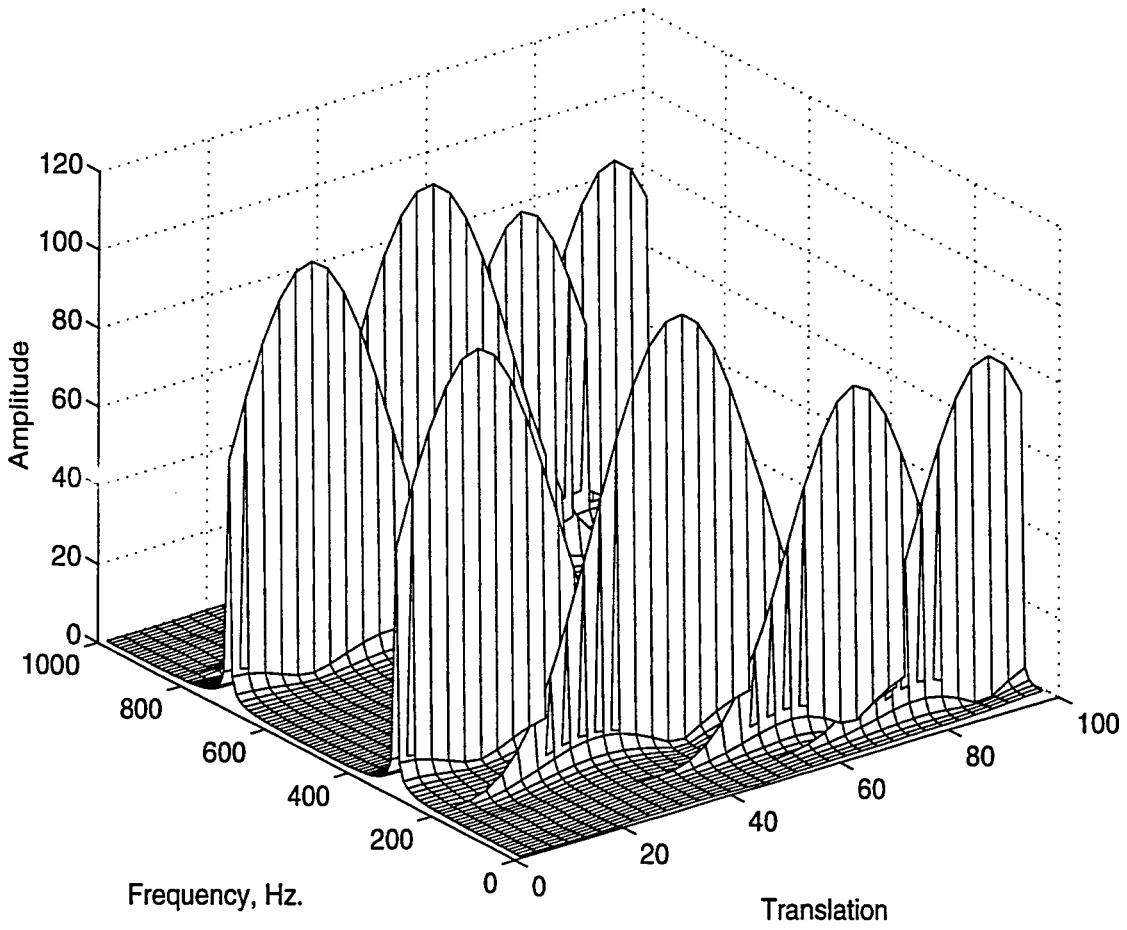

Figure 2.19: STFT of the signal in Figure 2.15 ,with window with a=0.0001

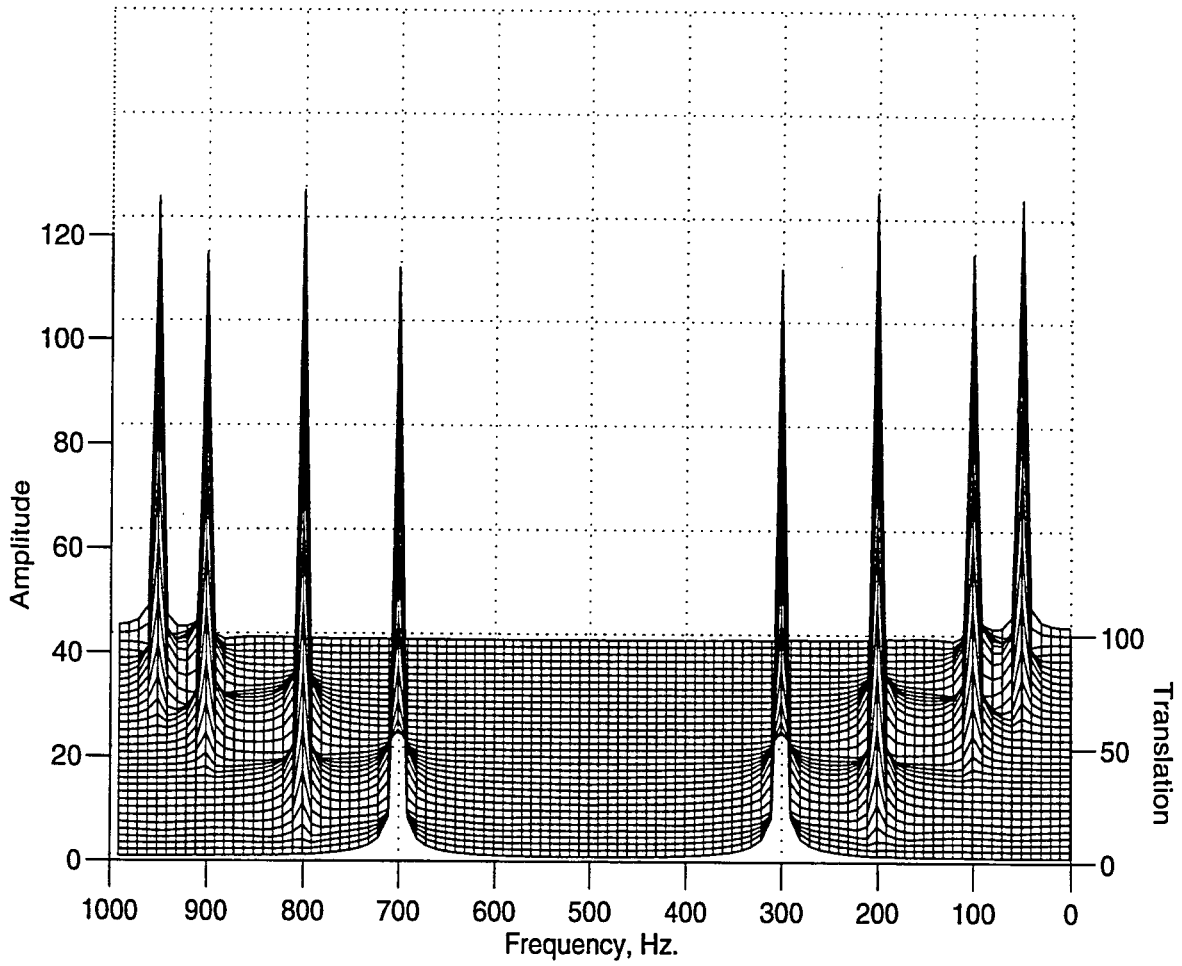

Figure 2.20: STFT of the signal in Figure 2.15, with window with a=0.0001, HR=−90°, VE=20°

Finally, the narrowest window was used to compute the STFTs of the same non-stationary signal. Figures 2.21 and 2.22 illustrate the time-frequency representation of the non-stationary signal with respect to this narrow window.

Following the same order, the transformed signal as viewed from the time axis is shown first in Figure 2.21. The spectral components corresponding to four different time intervals can be easily seen in this figure. For this case, every FT computed corresponds to a very short interval of time and yields good time resolution. Figure 2.22, showing the frequency axis side of this plot, dramatically illustrates the price paid for the good time resolution: The spectral components are hardly distinguishable.

For the practical signal, the choice of the best window can be quite challenging, particularly if the signal contains a variety of different spectral components. This is the main handicap of the STFT, and this made researchers look for alternative methods.

2.5 Signal Reconstruction and Inverse Transform

The short time Fourier transform is a reversible transform, and therefore, the transformed signal can be reconstructed under certain conditions. For continuous time STFT, the only major condition is on the choice of the window function. This condition is not very restrictive and can be formulated as [18]

$$\int_{-\infty}^{\infty} \xi(t).\omega^*(t)\, dt = 1 \tag{2.19}$$

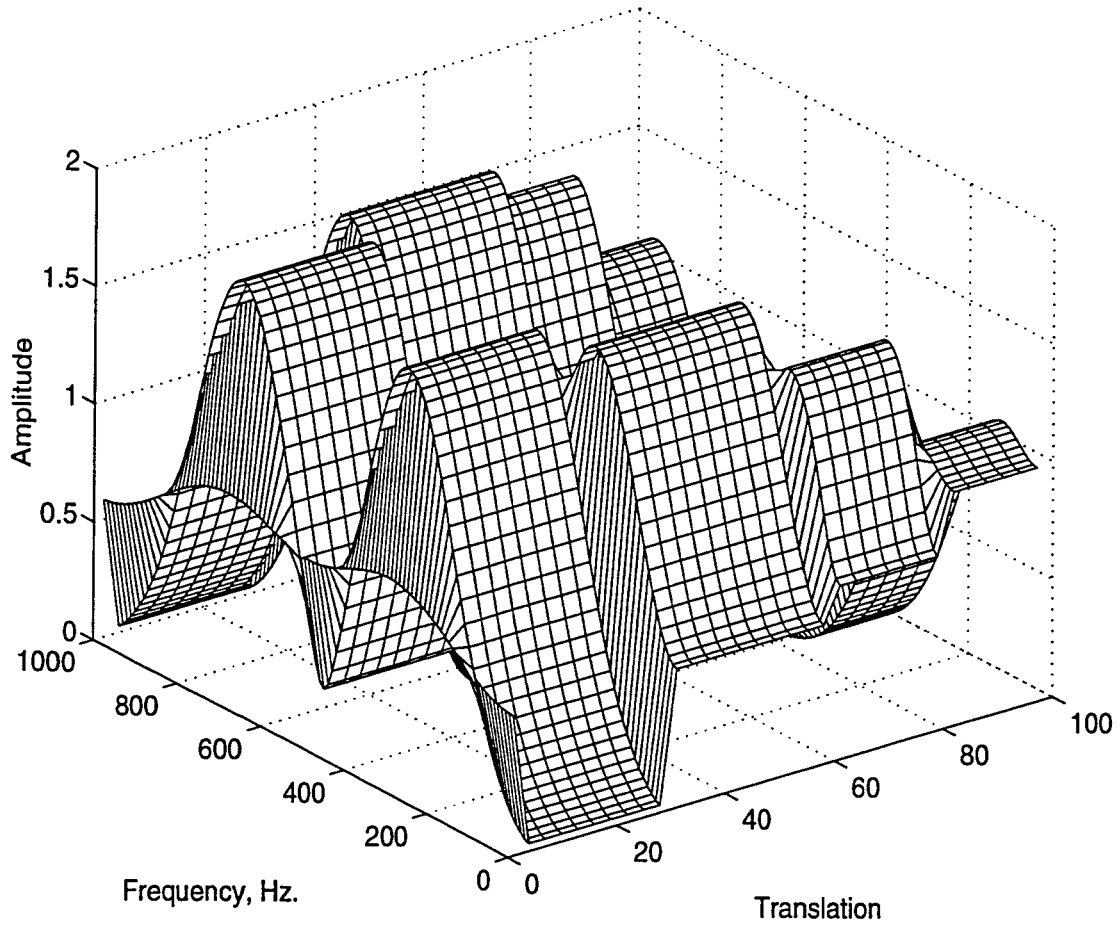

Figure 2.21: STFT of the signal in Figure 2.15, with window with a=0.5, HR=−37.5°, VE=30°

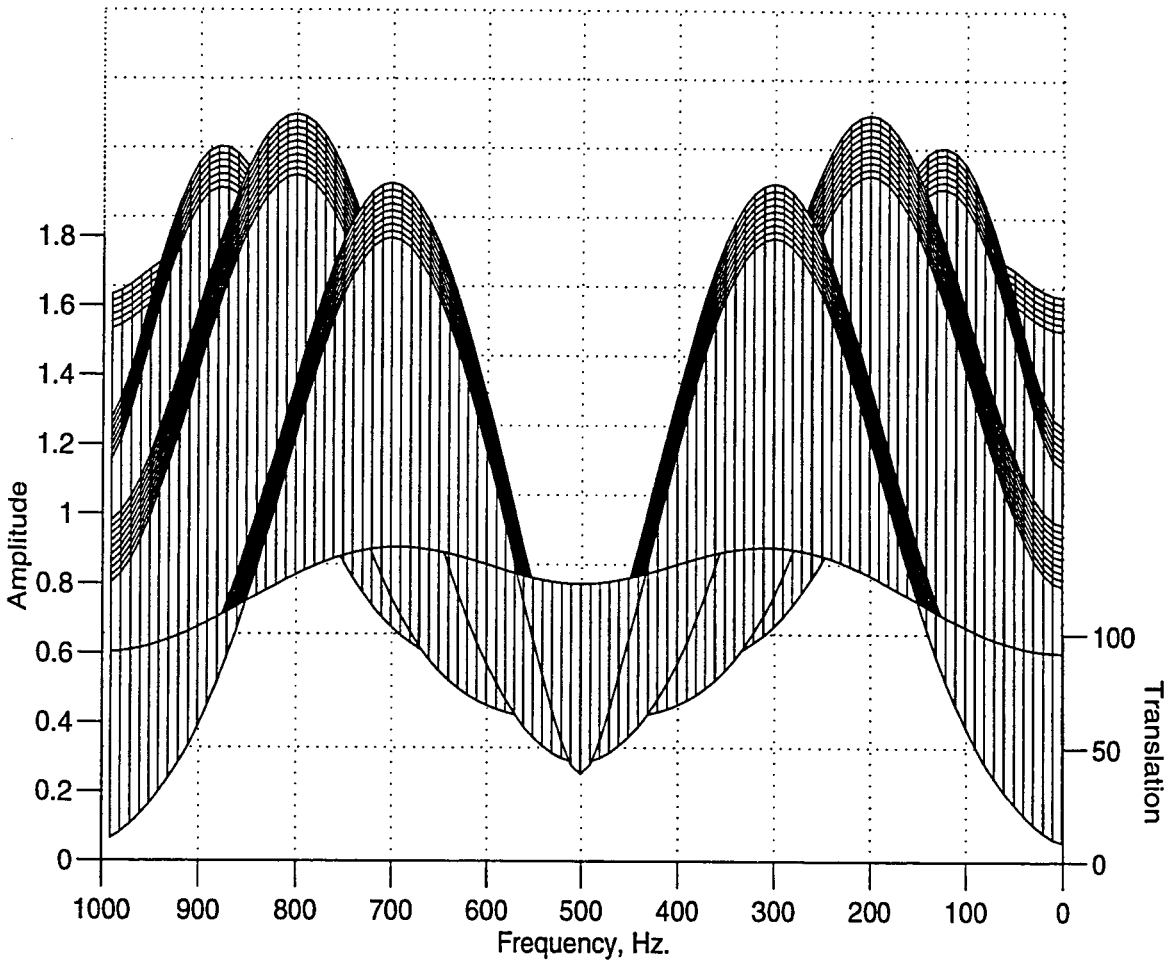

Figure 2.22: STFT of the signal in Figure 2.15, with window with a=0.5, HR=−90°, VE=20°

The function $\xi(t)$ is known as the *synthesis (reconstruction) window* as opposed to the *analysis window* $\omega(t)$. Recall that the inverse FT formula (Equation (1.2)) is very similar to the forward transform formula (Equation (1.1)), with the only difference in the sign of the exponent of the kernel. This similarity holds for the STFT with one major difference. Note that STFT maps a univariate function onto a bivariate function. This requires the use of a double integral in the inverse transform (reconstruction) formula. The inverse transform is given by the following equation.

$$x(t) = \int_{t'} \int_{f'} STFT_x^{\omega}(t', f').\xi(t - t').e^{j2\pi f' t}\, dt'\, df' \tag{2.20}$$

2.6 Discrete STFT

For computer applications, the STFT algorithm can be discretized by sampling the time (translation) and frequency parameters on the time-frequency plane. Let the time interval between each sample (time sampling period) be T, and the frequency interval (frequency sampling period) be F. Two indexes are needed to keep track of the discrete variables. Let m be the index to time, and n be the index to frequency. The discrete time-frequency plane can be represented by a grid of points that are T apart in time, and F apart in frequency. Note that the intervals in time and frequency are constant for the entire plane.

According to the above explanation, discrete time STFT analysis can be written as follows [8].

$$STFT_x^{\omega}(mT, nF) = \int_t x(t).\omega^*(t - mT).e^{-j2\pi(nF)t}\, dt \tag{2.21}$$

Similarly, the reconstruction formula can be discretized in time and frequency. The synthesis formula will then be

$$x(t) = \sum_m \sum_n STFT_x^{\omega}(mT, nF).\xi(t - mT).e^{j2\pi(nF)t} \tag{2.22}$$

where $\omega(t)$ is the analysis window, and $\xi(t)$ is the synthesis window.

For perfect reconstruction, a similar condition to the continuous time case is required. However, this condition is more restrictive than that for the continuous case.

$$\frac{1}{F}\sum_m \xi\left(t + n\frac{1}{F} - mT\right)\omega^*(t - mT) = \delta(n) \tag{2.23}$$

where $\delta(n)$ is the delta function, as defined in Equation(2.3) . Two points must be noted in the application of STFT to non-stationary signals:

1. If only signal analysis but not synthesis is of importance, then it is not necessary to select window functions that satisfy the above equation. In many applications, signal reconstruction may not be required, and any window that is compactly supported in time (finite in time) can be used for signal analysis.

2. The above definition of discrete STFT is not a discrete transform, but a discrete implementation of the continuous transform. However, for all practical purposes, the integral can be changed to summation, and the sum can be taken over a finite interval. This will result in a good approximation of the continuous time STFT. The program that was written to compute the given examples was coded in this manner.

2.7 An Alternative Interpretation for STFT

The following is a slightly different approach to STFT computations. It will be relevant in explaining the wavelet theory.

The STFT defined by Equation (2.2) is thought of as a correlation between the window function and the signal to be analyzed. The window function has a constant width, and it is modulated by the complex exponential term $e^{-j2\pi ft}$. For every value of f the frequency of the kernel function is varied and the window function is modulated at that frequency and multiplied by the signal. If the signal has a component at that particular f, the product and the integration result in a large (relatively) value. If the signal does not have a major spectral component at f, the product of the signal and the window modulated at frequency f followed by the integration will give (relatively) a small value. This means that the signal and the modulated window are not correlated.

This interpretation is illustrated next. Figure 2.23 shows four different plots. The first one corresponds to a window function, similar to the ones that have been used so far. The other three plots show the product of this window function with a complex exponential kernel, $e^{-j2\pi ft}$. Note that the plots illustrate the real parts of the products. Recall that the magnitudes of the products of the window function with all exponential kernels will look exactly like each other, since the magnitude of the complex exponential is constant and 1. The second, third, and fourth plots correspond to products of the window function, with the kernel of frequencies f=10 Hz, f=50 Hz, and f=100 Hz, respectively, i.e., these plots illustrate the *modulated window function* at three different frequencies.

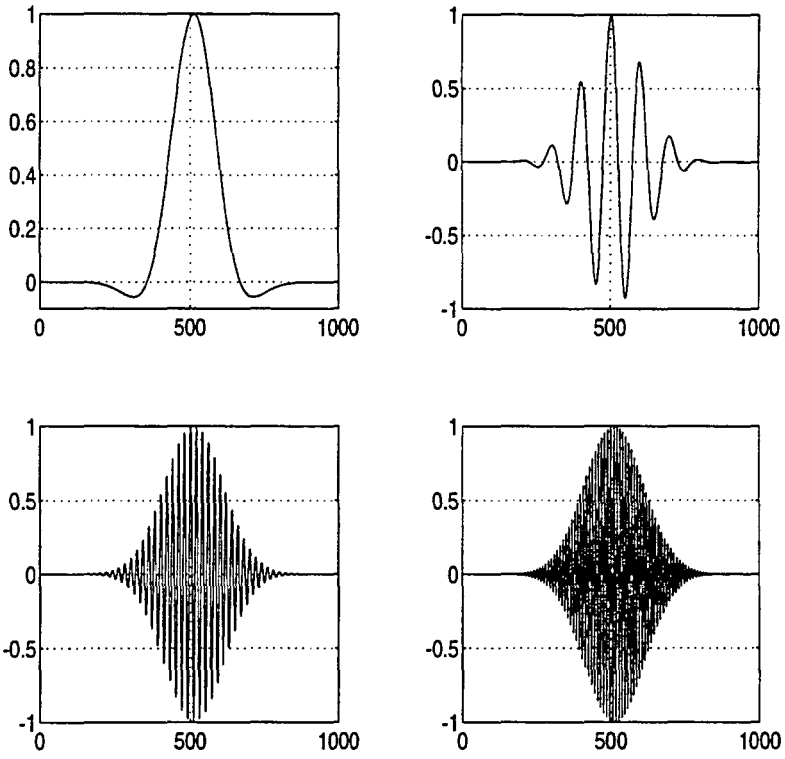

Figure 2.23: The modulated windows at different modulation frequencies

For instance, if the signal has a major 10 Hz component, the STFT computed for 10 Hz will be relatively large, since the harmonics corresponding to 10 Hz in the signal, and the modulated window will have a large correlation. In contrast, if the signal does not have a major 10 Hz component, the STFT computed for 10 Hz will be relatively small. This interpretation is also true for the Fourier transform.

In the next chapter the wavelet transform will be discussed from a similar perspective, namely, the correlation between a window function (which will be called the

"wavelet") and the signal at different times and frequencies.

2.8 Applications of the STFT

STFT has been used for many applications in signal processing, mainly for the analysis of time-varying signals. System identification, spectral estimation, signal detection, parameter estimation, determination of group velocity, speech analysis, speaker identification, speech coding, compression of acoustic signals are some of these applications (as cited in[8]).

In summary, the major drawback of the short time Fourier transform is related to its resolution in the time-frequency plane. The resolution is determined by the width of the window, which is kept constant throughout the process once it is chosen. Consequently, the resolution is also constant in time and frequency over the entire time-frequency plane.

A plausible solution to this problem is to consider the use of a different window at different frequencies. A wide window (in time) is needed for good frequency resolution and a narrow window is needed for good time resolution. It turns out that, for most practical applications, high frequency components (spikes) occur from time to time for short durations in a signal which has low frequency components of long durations. In this case, very good time resolution for low frequencies is not needed since it is already known that they exist over almost the entire duration of the signal. However, good time resolution for the spikes (high frequency components) is needed.

Good time resolution at high frequencies and good frequency resolution at low frequencies, sacrificing the good time resolution at low frequencies and good frequency resolution at high frequencies, will be satisfactory. Such a feature is offered by the

wavelet transform, and it is achieved by changing the window width at different frequencies.

CHAPTER 3. CONTINUOUS WAVELET TRANSFORM

3.1 Multiresolution Analysis

Although the time and frequency resolution problems are results of a physical phenomenon (the uncertainty principle) and exist regardless of the transform used, it is possible to analyze any signal by using an alternative approach called the *multiresolution analysis (MRA)*. Multiresolution analysis, as implied by its name, analyzes the signal at different frequencies with different resolutions. Every spectral component is not resolved equally as was the case in the STFT.

Multiresolution analysis is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies. This approach makes sense especially when the signal at hand has high frequency components for short durations and low frequency components for long durations. Fortunately, the signals that are encountered in practical applications are often of this type. For example, Figure 3.1 shows a signal of this type. It has a relatively low frequency components throughout the entire signal and relatively high frequency components for a short duration somewhere around the middle.

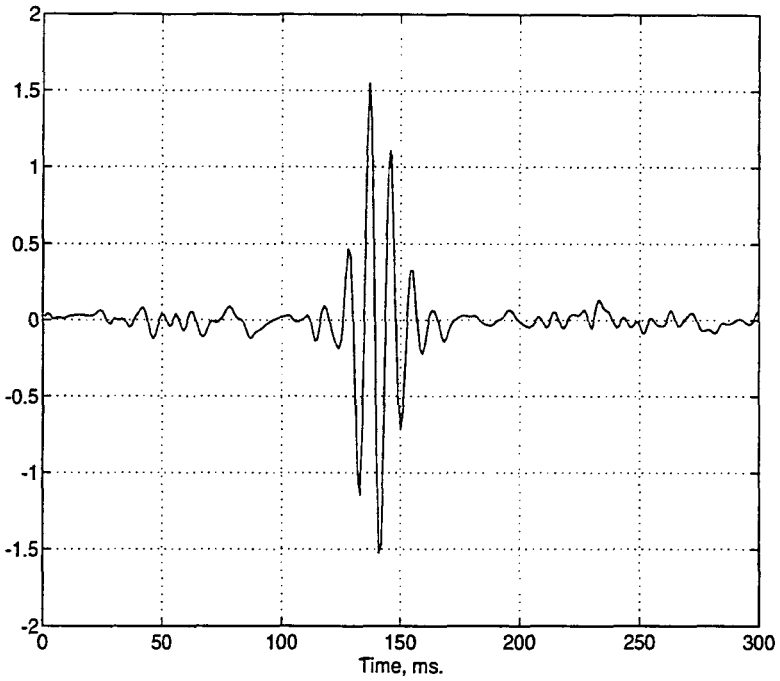

Figure 3.1: A non-stationary signal with high frequency components of short duration

3.2 The Continuous Wavelet Transform

The continuous wavelet transform (CWT) was developed as an alternative approach to the short time Fourier transform to overcome the resolution problem. The wavelet analysis is done in a similar way to the STFT analysis, in the sense that the signal is multiplied with a function, *the wavelet*, similar to the window function in the STFT, and the transform is computed separately for different segments of the time-domain signal. However, there are two main differences between the STFT and the CWT:

1. The Fourier transforms of the windowed signals are not taken, and therefore single peak will be seen corresponding to a sinusoid, i.e., negative frequencies are not

computed.

2. The width of the window is changed as the transform is computed for every single spectral component, which is probably the most significant characteristic of the wavelet transform.

The continuous wavelet transform is defined as follows

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t)\psi^* \left(\frac{t-\tau}{s}\right) dt \tag{3.1}$$

As seen in Equation (3.1), the transformed signal is a function of two variables, τ and s, the *translation* and *scale* parameters, respectively. $\psi(t)$ is the transforming function, and it is called *the mother wavelet.* The term *mother wavelet* gets its name due to two important properties of the wavelet analysis as explained below:

The term *wavelet* means a *small wave* . The smallness refers to the condition that this (window) function is of finite length (*compactly supported*). The wave refers to the condition that this function is oscillatory. The term *mother* implies that the functions with different regions of support that are used in the transformation process are derived from one main function, or the mother wavelet. In other words, the mother wavelet is a *prototype* for generating the other wavelets.

The term *translation* is used in the same sense as it was used in the STFT; it is related to the location of the window, as the window is shifted through the signal. This term, obviously, corresponds to time information in the transform domain. However, we do not have a *frequency* parameter, as we had before for the STFT. Instead, we have a scale parameter which is defined as $1/frequency$. The term *frequency* is reserved for the STFT. Scale is described in more detail in the next section.

3.2.1 The Scale

The parameter *scale* in the wavelet analysis is similar to the scale used in maps. As in the case of maps, high scales correspond to a non-detailed global view (of the signal), and low scales correspond to a detailed view. Similarly, in terms of frequency, low frequencies (high scales) correspond to a global information of a signal (that usually spans the entire signal), whereas high frequencies (low scales) correspond to a detailed information of a hidden pattern in the signal (that usually lasts a relatively short time). Cosine signals corresponding to various scales are given as examples in Figure 3.2 .

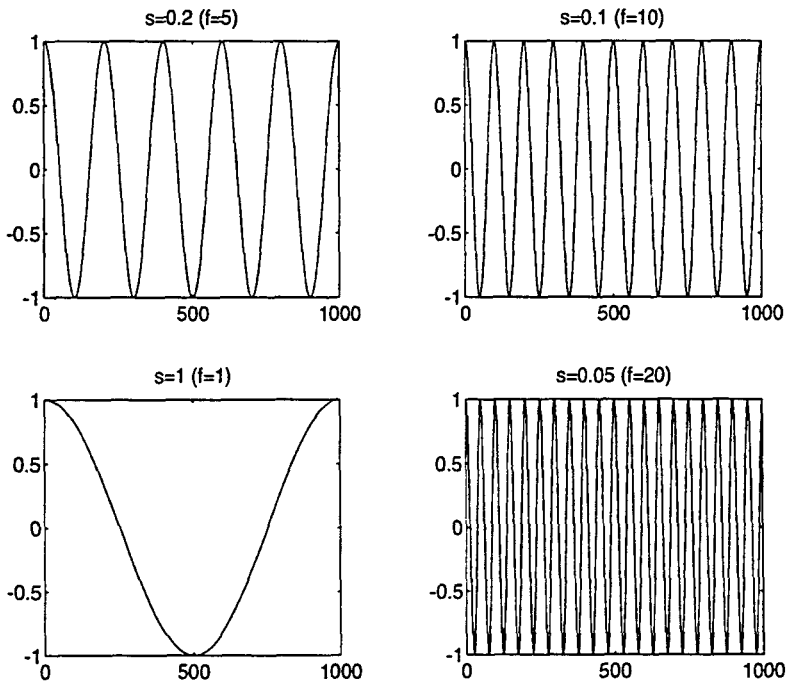

Figure 3.2: Sample cosine signals at different scales

Fortunately in practical applications, low scales (high frequencies) do not last for the entire duration of the signal, unlike those shown in Figure 3.2, but they usually appear from time to time as short bursts, or spikes. High scales (low frequencies) usually last for the entire duration of the signal.

Scaling, as a mathematical operation, either dilates or compresses a signal. Larger scales correspond to dilated (or stretched out) signals and small scales correspond to compressed signals. All of the signals given in Figure 3.2 are derived from the same cosine signal, i.e., they are dilated or compressed versions of the same function. In Figure 3.2, "$s = 0.05$" is the smallest scale, and "$s = 1$" is the largest scale.

In terms of mathematical functions, if $f(t)$ is a given function $f(ft)$ corresponds to a contracted (compressed) version of $f(t)$ if $f > 1$ and to an expanded (dilated) version of $f(t)$ if $f < 1$.

However, in the definition of the wavelet transform, the scaling term is used in the denominator (in Equation (3.1), $f(\frac{t-\tau}{s})$), and therefore, the opposite of the above statements holds, i.e., scales $s > 1$ dilates the signals whereas scales $s < 1$, compresses the signal. This interpretation of scale will be used throughout this text.

3.2.2 Computation of the CWT

Interpretation of Equation (3.1) will be explained in this section. Let $x(t)$ be the signal to be analyzed. The mother wavelet is chosen to serve as a prototype for all windows in the process. All the windows that are used are the dilated (or compressed) and shifted versions of the mother wavelet. There are a number of functions that are used for this purpose. The Morlet wavelet and the Mexican hat function are

two candidates, and they are used for the wavelet analysis of the examples which are presented later in this chapter.

Once the mother wavelet is chosen the computation starts with $s = 1$ and the continuous wavelet transform is computed for all values of s, smaller and larger than "1". However, depending on the signal, a complete transform is usually not necessary. For all practical purposes, the signals are bandlimited, and therefore, computation of the transform for a limited interval of scales is usually adequate. In this study, some finite interval of values for s were used, as will be described later in this chapter.

For convenience, the procedure will be started from scale $s = 1$ and will continue for the increasing values of s, i.e., the analysis will start from high frequencies and proceed towards low frequencies. This first value of s will correspond to the most compressed wavelet. As the value of s is increased, the wavelet will dilate.

The wavelet is placed at the beginning of the signal at the point which corresponds to time=0. The wavelet function at scale "1" is multiplied by the signal and then integrated *over all times.* The result of the integration is then multiplied by the constant number $\frac{1}{\sqrt{s}}$. This multiplication is for energy normalization purposes so that the transformed signal will have the same energy at every scale. The final result is the value of the transformation, i.e., the value of the continuous wavelet transform *at time zero* and *scale s=1.* In other words, it is the value that corresponds to the point ($\tau = 0$, $s = 1$) in the time-scale plane.

The wavelet at scale $s = 1$ is then shifted towards right by τ amount to the location $t = \tau$, and Equation(3.1) is computed to get the transform value at ($t = \tau$, $s = 1$) in the time-frequency plane.

This procedure is repeated until the wavelet reaches the end of the signal. *One*

row of points on the time-scale plane for the scale $s = 1$ is now completed.

Then, s is increased by a small value. Note that, this is a continuous transform, and therefore, both τ and s must be incremented *continuously* . However, if this transform needs to be computed by a computer, then both parameters are increased by *a sufficiently small step size.* This corresponds to sampling the time-scale plane.

The above procedure is repeated for every value of s. Every computation for a given value of s fills the corresponding single row of the time-scale plane. When the process is completed for all desired values of s, the CWT of the signal has been calculated.

Figure 3.3 illustrates the entire process step by step.

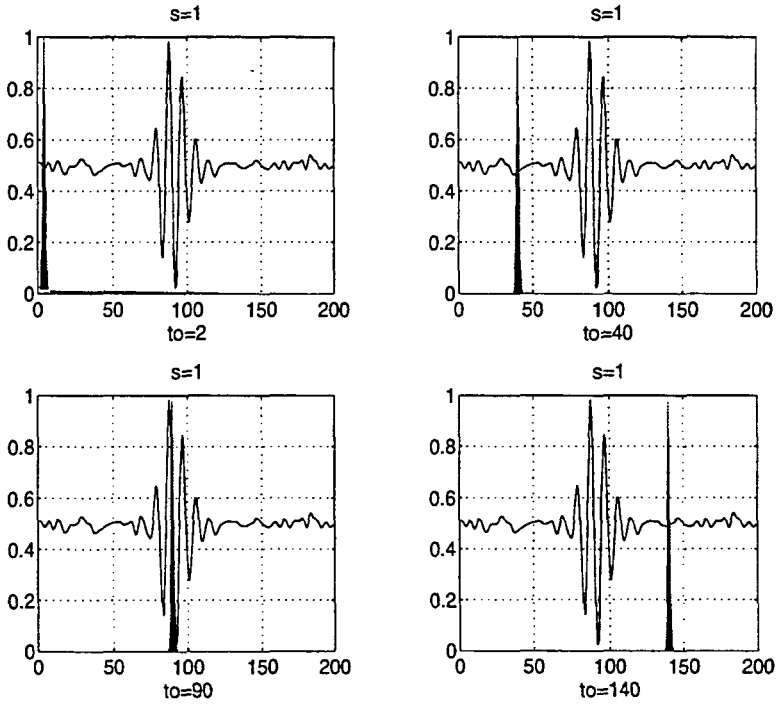

Figure 3.3: The CWT computation at s=1

In Figure 3.3, the signal and the wavelet function are shown for four different values of τ. The signal is a truncated version of the signal shown in Figure 3.1. The scale value is 1, corresponding to the lowest scale, or highest frequency. Note how compact it is. It should be as narrow as the highest frequency component that exists in the signal. Four distinct locations of the wavelet function are shown in the figure at $t_0 = 2$, $t_0 = 40$, $t_0 = 90$, and $t_0 = 140$. At every location, it is multiplied by the signal. Obviously, the product is nonzero only where the signal falls in the region of support of the wavelet, and it is zero elsewhere. By shifting the wavelet in time, the signal is localized in time, and by changing the value of s, the signal is localized in scale (frequency).

If the signal has a spectral component that corresponds to the current value of s (which is 1 in this case), the product of the wavelet with the signal **at the location where this spectral component exists** gives a relatively large value. If the spectral component that corresponds to the current value of s is not present in the signal, the product value will be relatively small, or zero. The signal in Figure 3.3 has spectral components comparable to the window's width at $s = 1$ around t=100 ms.

The continuous wavelet transform of the signal in Figure 3.3 will yield large values for low scales around time 100 ms, and small values elsewhere. For high scales, on the other hand, the continuous wavelet transform will give large values for almost the entire duration of the signal, since low frequencies exist at all times.

Figures 3.4 and 3.5 illustrate the same process for the scales s=5 and s=20, respectively. Note how the window width changes with increasing scale (decreasing frequency). As the window width increases, the transform starts picking up the lower

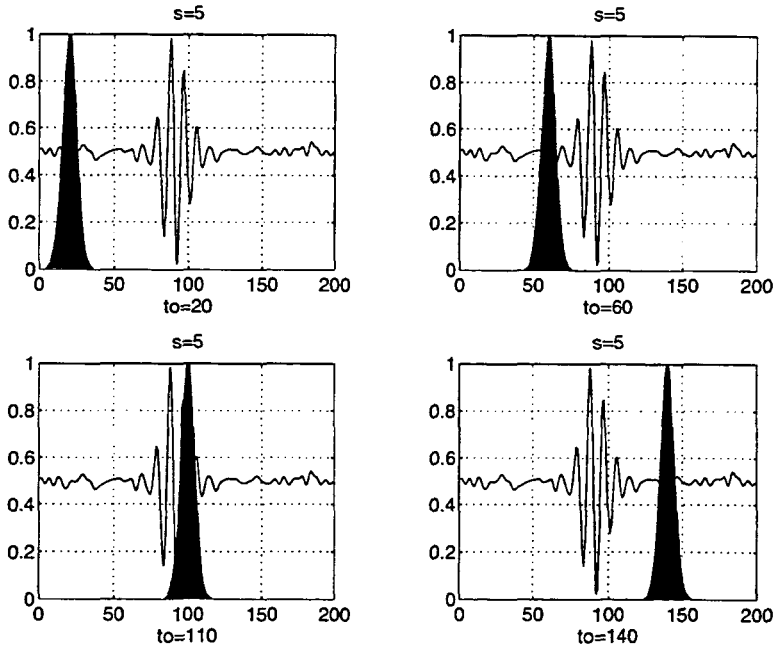

Figure 3.4: The CWT computation at s=5

frequency components.

As a result, for every scale and for every time (interval), one point of the time-scale plane is computed. The computations at one scale construct the rows of the time-scale plane, and the computations at different scales construct the columns of the time-scale plane.

Figure 3.6 is the computed two dimensional CWT of this signal. Note that the axes are translation (time) and frequency, not scale. Time-frequency representation is relatively easy to interpret. In contrast, time-scale plots are usually more difficult to interpret as shown later in this chapter.

The axes in Figure 3.6 are normalized and should be evaluated accordingly. Roughly speaking, the 50 points in the translation axis correspond to 200 ms, and

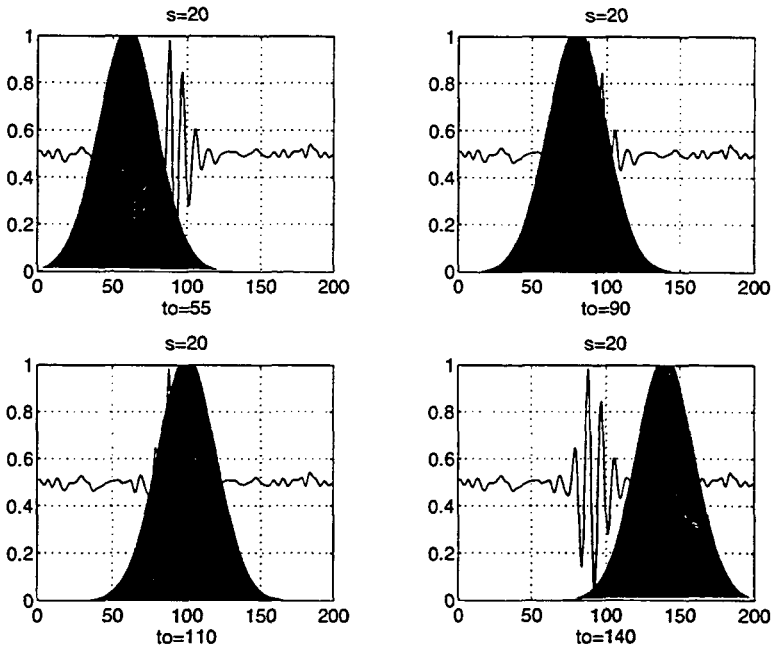

Figure 3.5: The CWT computation at s=20

small values on the frequency axis correspond to lower frequencies. Higher values of the frequency axis correspond to higher frequencies. Note that the zero on the frequency axis is not necessarily the DC value, but simply the lowest frequency value used in the analysis. From this perspective, notice the relatively small peak that occurs at a translation of 20-25 units (which corresponds to time around 100 ms.) indicates a high frequency component. Also note that low frequency components are located at all times (translations).

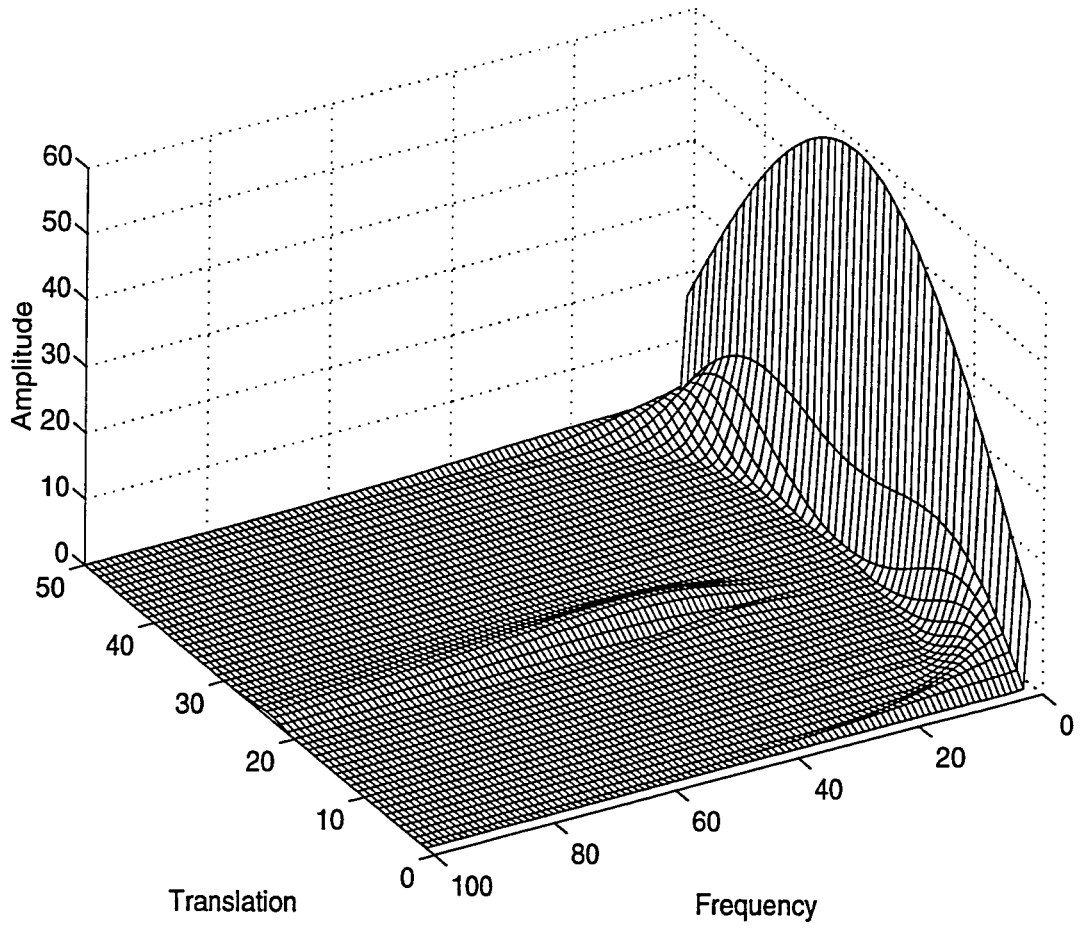

Figure 3.6: The computed CWT for the test signal given in Figure 3.1

3.2.3 Time and Frequency Resolutions

The most important characteristic of the wavelet transform is the time and frequency resolutions. The illustrations in Figures 3.7 and 3.8 are commonly used to explain how time and frequency resolutions should be interpreted. Figure 3.7 shows the time-frequency discretization of the STFT. The time and frequency planes are divided into equal size rectangles which indicates that time and frequency resolutions are constant. Every box corresponds to one computed value of the STFT. For the time-frequency plane given in Figure 3.7, there is one value computed for the region covered by each box. That value represents all the points that fall into that box. The value that is actually computed usually corresponds to the center of the rectangular box.

In contrast, Figure 3.8 shows how wavelet transform discretizes time and frequency variables (as covered in more detail in Chapter 4). In this case, the width of the rectangles decreases with increasing frequency, which shows the increasing time resolution at high frequencies. However, heights of the rectangles increase with increasing frequency which indicates decreasing frequency resolution. The opposite holds for low frequencies. Heights of the rectangular boxes decrease, indicating increase in frequency resolution. Widths of the boxes increase, which indicates decrease in time resolution. Similar to the case of STFT, each box corresponds to one computed value of the continuous wavelet transform integral.

However, regardless of the dimensions of the boxes, the areas of all boxes both in STFT and WT, are the same and determined by Heisenberg's inequality. The area of a box is fixed for each window function (STFT) or mother wavelet (CWT), whereas different windows or mother wavelets can result in different areas. However,

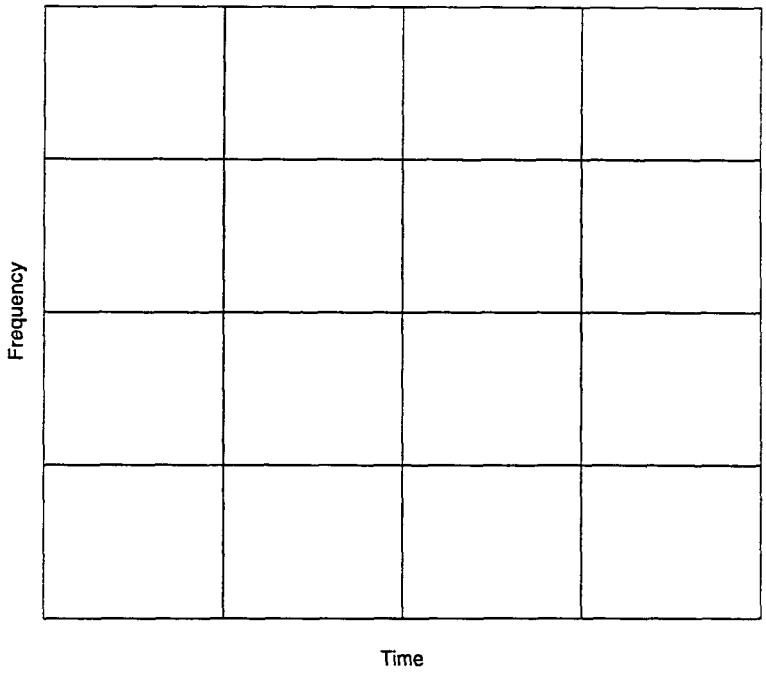

Figure 3.7: The partition of the time-frequency plane for the STFT

all areas are lower bounded by $1/4\pi$. On the other hand, for a given mother wavelet the dimensions of the boxes can be changed, while keeping the area the same. This is exactly what wavelet transform does.

3.3 Introduction to The Wavelet Theory: A Mathematical Approach

This section describes the main idea of wavelet analysis theory, which can also be considered to be the underlying concept of most of the signal analysis techniques. The FT defined by Fourier [1] use *basis functions* to analyze and reconstruct a function. *Every vector in a vector space can be written as a linear combination of the basis vectors in that vector space*, i.e., by multiplying the vectors by some constant

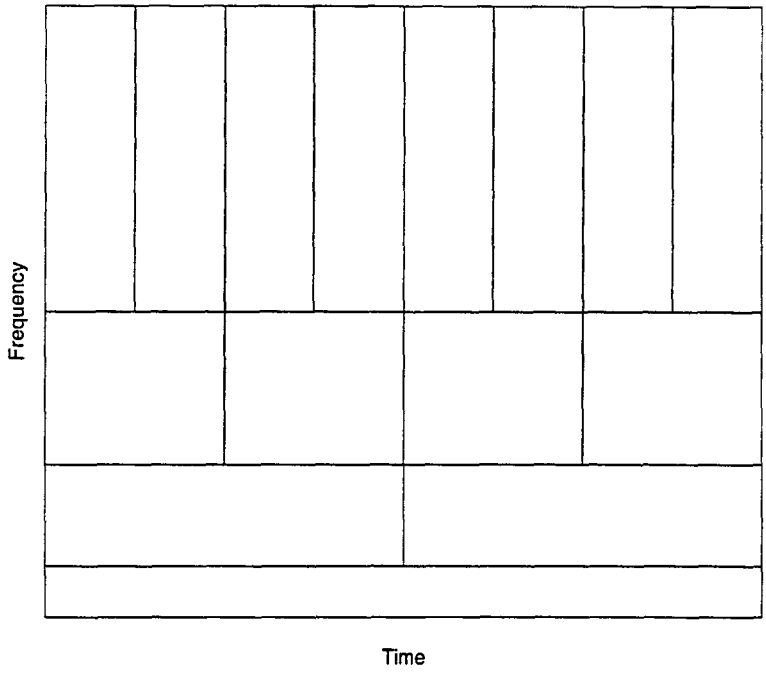

Figure 3.8: The partition of the time-frequency plane for the WT

numbers, and then by taking the summation of the products. The analysis of the signal involves the estimation of these constant numbers (transform coefficients, or Fourier coefficients, wavelet coefficients, etc). The synthesis, or the reconstruction, corresponds to computing the linear combination equation.

All the definitions and theorems related to this subject can be found in [10], but an introductory level knowledge of how basis functions work is necessary to understand the underlying principles of the wavelet theory. Therefore, this information will be presented in this section.

3.3.1 Basis Vectors

A *basis* of a vector space V is a set of linearly independent vectors, such that any vector $\mathbf{v}\epsilon V$ can be written as a linear combination of these basis vectors. (Note that vectors are typed in bold face.) There may be more than one basis for a vector space. However, all of them have the same number of vectors, and this number is known as the *dimension* of the vector space. For example in two-dimensional space, the basis will have two vectors.

$$\mathbf{v} = \sum_k \nu^k \mathbf{b}_k \tag{3.2}$$

Equation (3.2) shows how any vector $\mathbf{v}$ can be written as a linear combination of the basis vectors $\mathbf{b}_k$ and the corresponding coefficients ν^k .

This concept, given in terms of vectors, can easily be generalized to functions, by replacing the basis vectors $\mathbf{b}_k$ with basis functions $\phi_k(t)$, and the vector v with a function $f(t)$. Equation (3.2) then becomes

$$f(t) = \sum_k \mu_k \phi_k(t) \tag{3.3}$$

The complex exponential (sines and cosines) functions are the basis functions for the FT. Furthermore, they are orthogonal functions, which provide some desirable properties for reconstruction.

Let f(t) and g(t) be two functions in $L^2[a,b]$. ($L^2[a,b]$ denotes the set of square integrable functions in the interval [a,b]). The inner product of two functions is defined as follows:

$$< f(t), g(t) >= \int_a^b f(t).g^*(t)\, dt \tag{3.4}$$

According to the above definition of the inner product, the CWT can be thought of as the inner product of the test signal with the basis functions $\psi_{\tau,s}(t)$:

$$CWT_x^{\psi}(\tau, s) = \Psi_x^{\psi}(\tau, s) = \int x(t).\psi_{\tau,s}^*(t)\, dt \tag{3.5}$$

where,

$$\psi_{\tau,s} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{3.6}$$

This definition of the CWT shows that the wavelet analysis is a measure of similarity between the basis functions (wavelets) and the signal itself. Here the similarity is in the sense of similar frequency content. The calculated CWT coefficients refer to the closeness of the signal to the wavelet *at the current scale.*

This further clarifies the previous discussion on the correlation of the signal with the wavelet at a certain scale. If the signal has a major component of the frequency corresponding to the current scale, then the wavelet (the basis function) at the current scale will be *similar* or *close* to the signal at the particular location where this frequency component occurs. Therefore, the CWT coefficient computed at this point in the time-scale plane will be a relatively large number.

3.3.2 Inner Products, Orthogonality, and Orthonormality

Two vectors $\mathbf{v}, \mathbf{w}$ are said to be *orthogonal* if their inner product equals zero:

$$< \mathbf{v}, \mathbf{w} >= \sum_n v_n w_n^* = 0 \tag{3.7}$$

Similarly, two functions f and g are said to be orthogonal to each other if their inner product is zero:

$$< f(t), g(t) >= \int_a^b f(t).g^*(t)\, dt = 0 \tag{3.8}$$

A set of vectors $\{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}\}$ is said to be *orthonormal*, if they are pairwise orthogonal to each other, and all have length "1". This can be expressed as:

$$< \mathbf{v_m}, \mathbf{v_n} >= \delta_{mn} \tag{3.9}$$

Similarly, a set of functions $\{\phi_k(t)\}$, k=1,2,3,..., is said to be orthonormal if

$$\int_a^b \phi_k(t)\phi_l^*(t)\, dt = 0 \;\; k \neq l \text{ (orthogonality cond.)} \tag{3.10}$$

and

$$\int_a^b \{|\phi_k(t)|\}^2\, dx = 1 \tag{3.11}$$

or equivalently

$$\int_a^b \phi_k(t)\phi_l^*(t)\, dt = \delta_{kl} \tag{3.12}$$

where, δ_{kl} is the *Kronecker delta* function, defined as follows.

$$\delta_{kl} = \begin{cases} 1 & \text{if } k = l \\ 0 & \text{if } k \neq l \end{cases} \tag{3.13}$$

As stated above, there may be more than one set of basis functions (or vectors). Among them, the orthonormal basis functions (or vectors) are of particular importance because of the nice properties they provide in finding these analysis coefficients. The orthonormal bases allow computation of these coefficients in a very simple and straightforward way using the orthonormality property.

For orthonormal bases, the coefficients, μ_k, can be calculated as

$$\mu_k =< f, \phi_k >= \int f(t).\phi_k^*(t)\, dt \tag{3.14}$$

and the function f can then be reconstructed by Equation (3.3) by substituting the μ_k coefficients. This yields

$$f(t) = \sum_k \mu_k \phi_k(t) \tag{3.15}$$

$$= \sum_k < f, \phi_k > \phi_k(t) \tag{3.16}$$

Orthonormal bases may not be available for every type of application where a generalized version, *biorthogonal* bases can be used. The term "biorthogonal" refers to two different bases which are orthogonal to each other, but each do not form an orthogonal set.

In some applications, however, biorthogonal bases also may not be available in which case *frames* can be used. Frames constitute an important part of wavelet theory, and interested readers are referred to [10].

Following the same order as in chapter 2 for the STFT, some examples of continuous wavelet transform are presented next. The figures given in the examples were generated by a program written to compute the CWT.

3.4 Examples

All of the examples that are given below correspond to non-stationary signals. For easy understanding, the test signals are chosen so that their CWT can be predicted and thus compared with the actual results that are computed by the program.

3.4.1 Example 1

The window function used for this analysis was the *Mexican hat* wavelet which is defined as the second derivative of the Gaussian function.

$$w(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-t^2}{2\sigma^2}} \tag{3.17}$$

$$\psi(t) = \frac{1}{\sqrt{2\pi}\sigma^3} \left(e^{\frac{-t^2}{2\sigma^2}} . \left(\frac{t^2}{\sigma^2} - 1 \right) \right) \tag{3.18}$$

The function resembles a Mexican hat. Two of its important properties are that it is real and that it has a zero integral. The implication of these properties, as opposed to the Morlet wavelet which is complex and does not has a zero integral, will be explained below.

Figure 3.9 shows the test signal used in this example. It contains two spectral components at different times. The high frequency component is four times the low frequency one. The signal is 512 ms long.

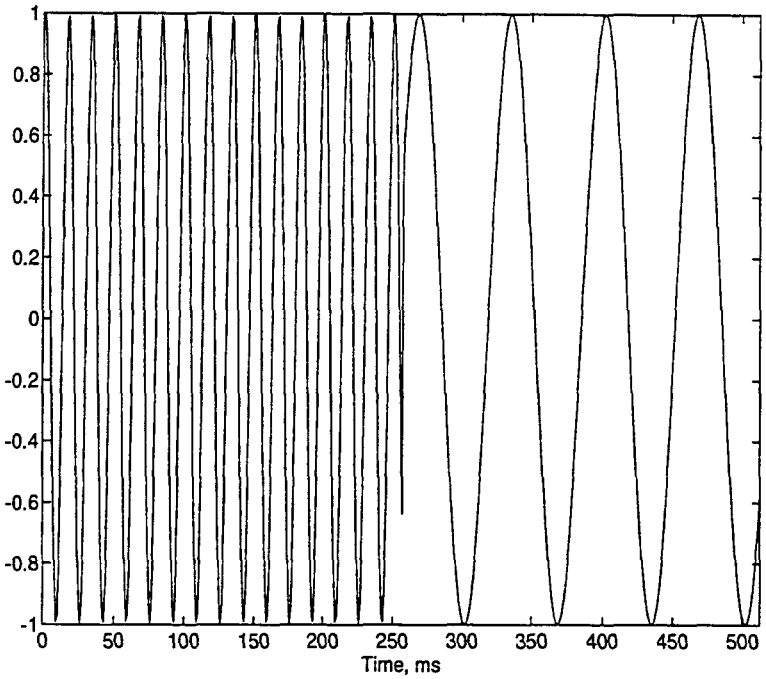

Figure 3.9: Non-stationary signal with two frequency components

Figure 3.10 shows the continuous wavelet transform of the sample signal given above. This figure actually plots the absolute value (magnitude) of the CWT coefficients. The wavelet used for this transform is real, and therefore, there is actually no reason for plotting the magnitude. However, the time-frequency (scale) representation can be interpreted more easily from the magnitude plot. (The original CWT is given in Figure 3.11 for comparison.)

The two different frequencies can be easily seen in Figure 3.10 (or in Figure 3.11). In the time interval between zero and 256 ms, a peak occurs where frequency is marked as 10, and in the time interval between 256 ms to 512 ms, a peak occurs on the frequency axis at 2-3. This shows exactly what frequency components exist

at what times.

Two important features can be seen immediately in this figure. The first one is the difference in amplitudes. The reason why the amplitude at the lower frequency is higher follows from the fact that, when the frequency is low, the window is wider in time and takes more terms into integration. This can be changed by using an amplitude scaling factor.

The second feature is the difference in the widths of the peaks. The wavelet transform corresponding to high frequencies is more spread out in frequency compared to the low frequencies. This is exactly what was expected. At higher frequencies there is good time resolution, but poor frequency resolution. At lower frequencies, there is good frequency resolution, but poor time resolution.

Until now, the time-frequency representations of the CWT were given, although, the CWT is a time-scale analysis. Figure 3.12 shows the continuous wavelet transform of the same signal in the time-scale plane.

Now, low scales correspond to high frequencies, and high scales correspond to low frequencies. Therefore, scale 1 corresponds to the highest frequency in the analysis, whereas scale 50 corresponds to a lower frequency in the analysis. The lower bound of the range of the scales covered by the analysis depends on the parameters of the initial window (the mother wavelet), which is given by σ in the above equation of the Mexican hat function. Every window function can be started from an initial scale by introducing a scaling term in the equation of the window function.

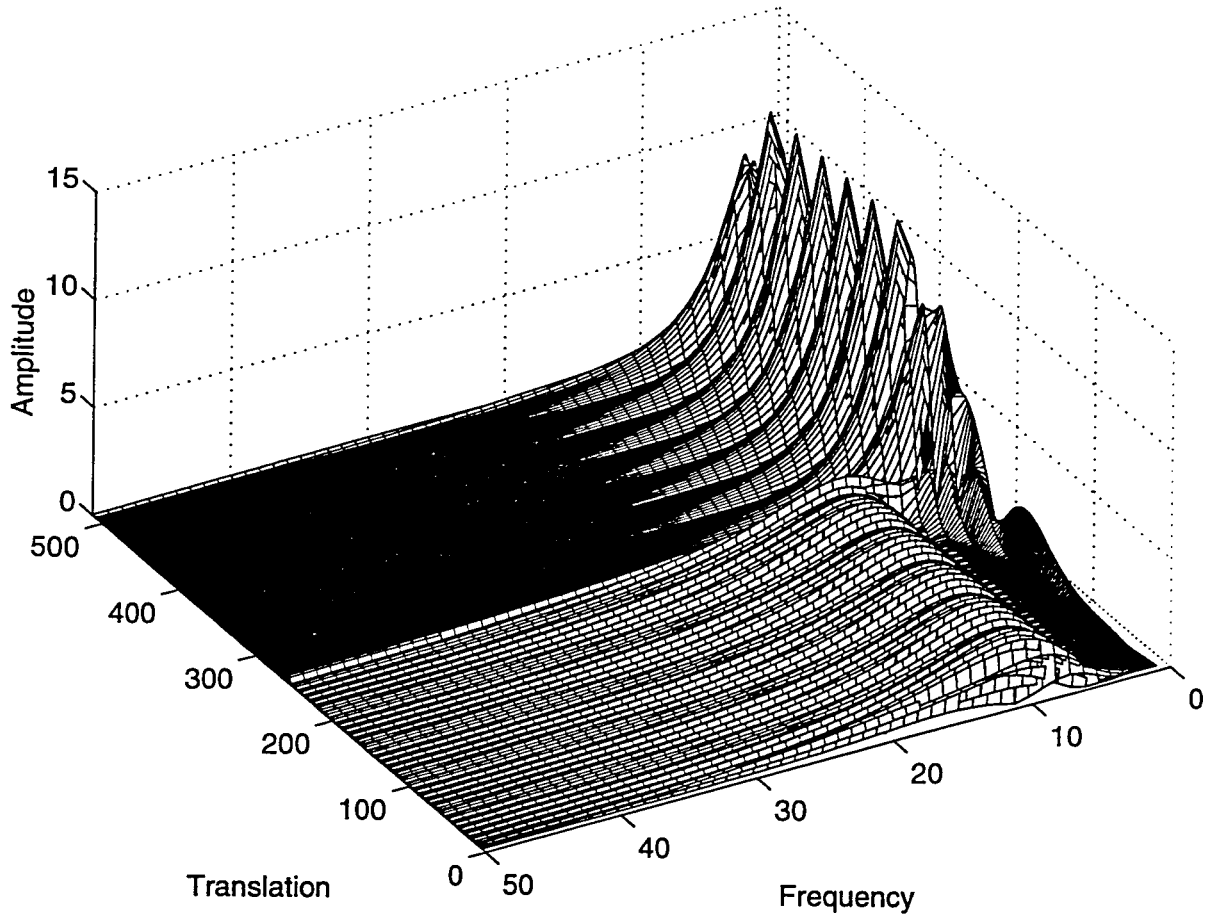

Figure 3.10: Magnitude of the CWT for the signal in Figure (3.9)

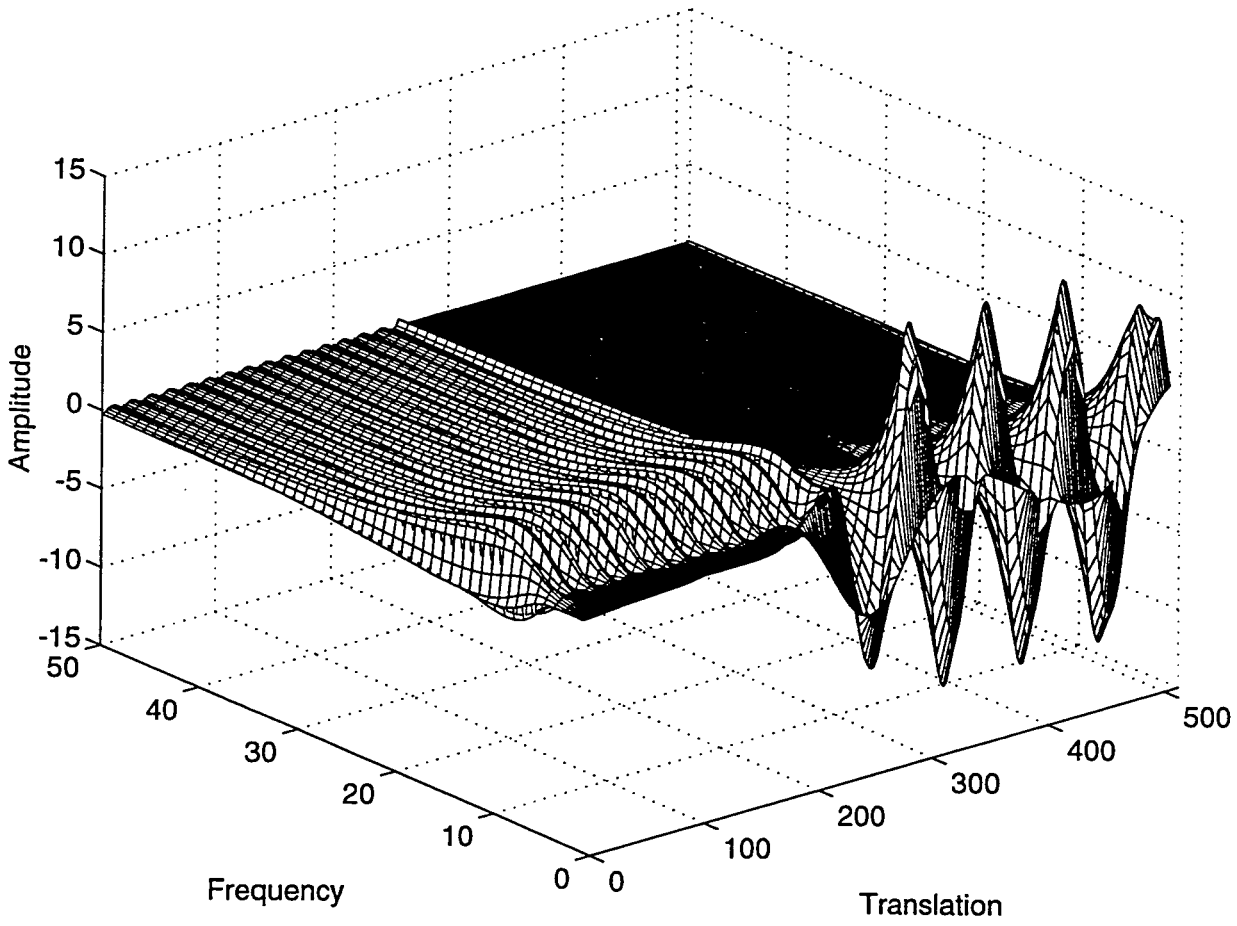

Figure 3.11: CWT for the signal in Figure (3.9)

The resolution properties of the transform in Figure 3.12 should be interpreted with care. As opposed to what is explained in the previous sections, the transform seems to produce poor frequency resolution at low frequencies and good frequency resolution at higher frequencies. Note that the portion of the transform corresponding to the lower frequencies (high scales) is spread out in **frequency**, whereas that portion of the transform corresponding to the high frequencies (low scales) is compact in **frequency**.

Although the high frequencies look more compact, it should be noted that they are actually compact in *scale*, not in *frequency* . The low scale (high frequency) portion is compact in scale which means that it is not compact in frequency, since scale and frequency are reciprocals of each other.

The time resolution, on the other hand, is the same as that in Figure 3.10, since time is the same in both cases. The time resolution properties looks even more obvious in Figure 3.12. Note how the low scale (high frequency) portions are well bounded in time, whereas at higher scales (lower frequencies) the transform gets smeared in time. The spread corresponds to poor time resolution.

Finally, Figure 3.13 shows the same signal's CWT using a Morlet wavelet. The Morlet wavelet is defined as

$$w(t) = e^{iat}.e^{-\frac{t^2}{2\sigma}} \tag{3.19}$$

where a is a modulation parameter, and σ is the scaling parameter that affects the width of the window.

It is given here to show that some differences may occur from one wavelet transform to another. Some wavelets may resolve some frequencies better than others.

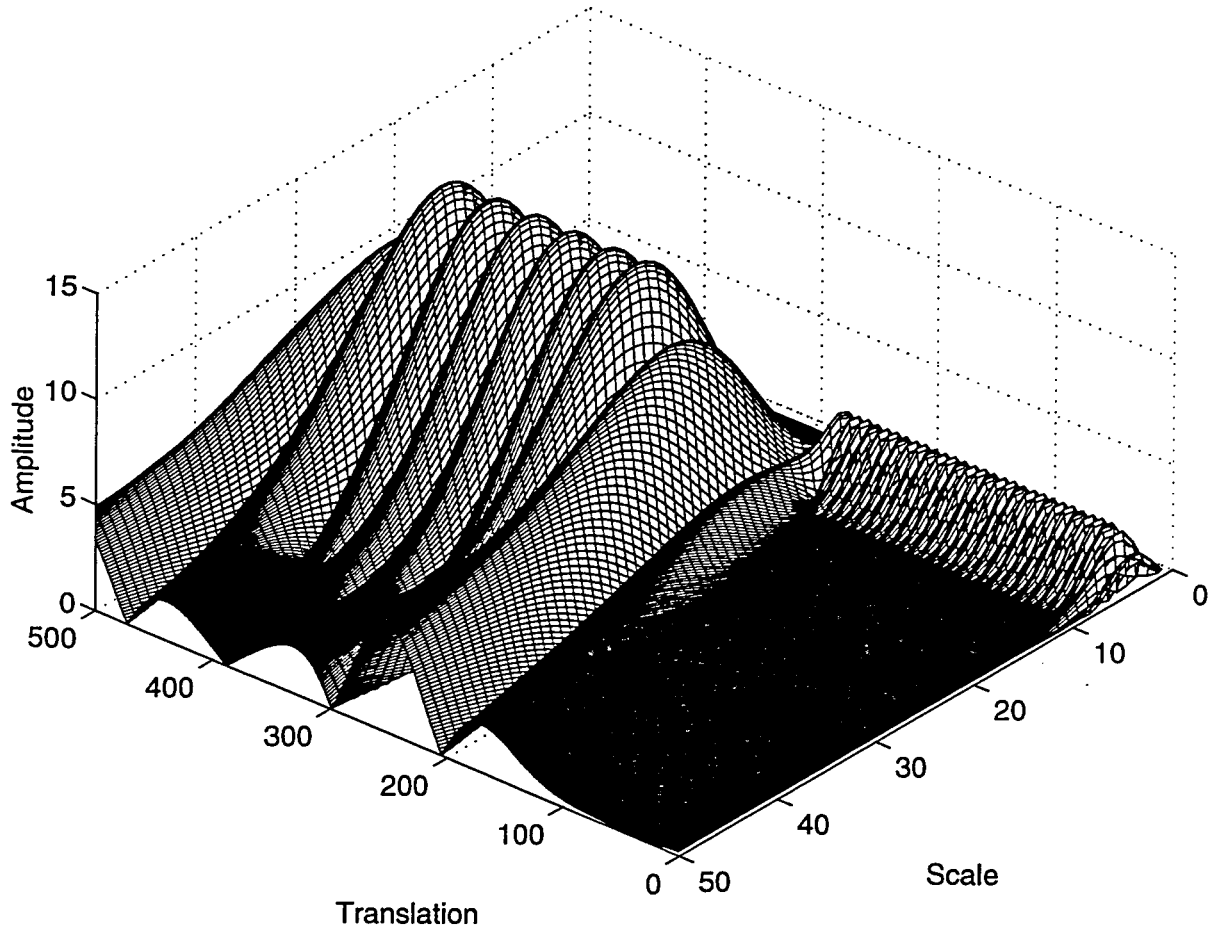

Figure 3.12: CWT for the signal in Figure (3.9)

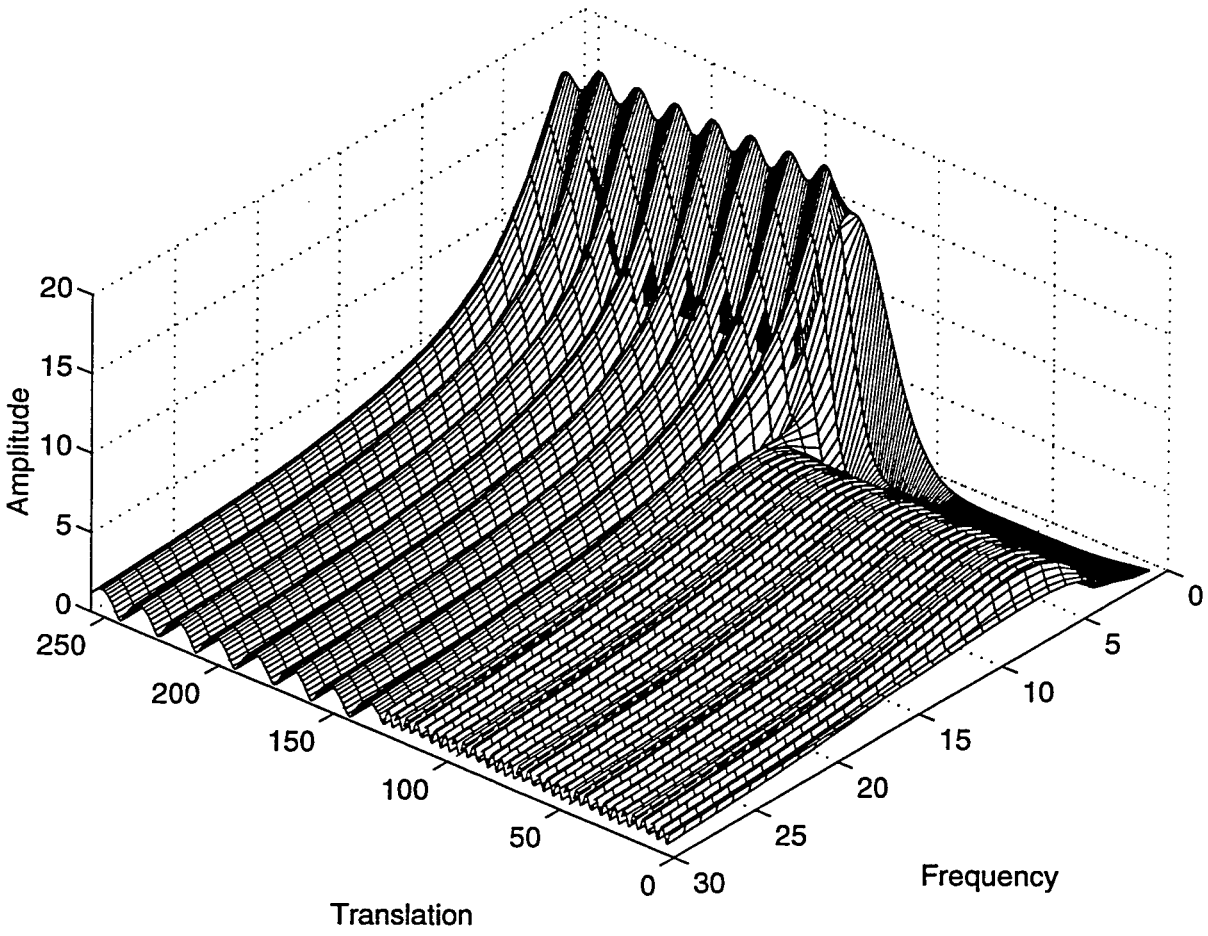

Figure 3.13: CWT for the signal in Figure 3.9 with the Morlet wavelet

However, it should be noted that there are no criteria for selecting the optimum wavelet for an application. It is usually chosen by trial and error. In some cases, using a different wavelet may result in significant changes.

3.4.2 Example 2

In the second example, a rectangular pulse is introduced between two cosine waves of different frequencies. The rectangular region represents a major discontinuity.

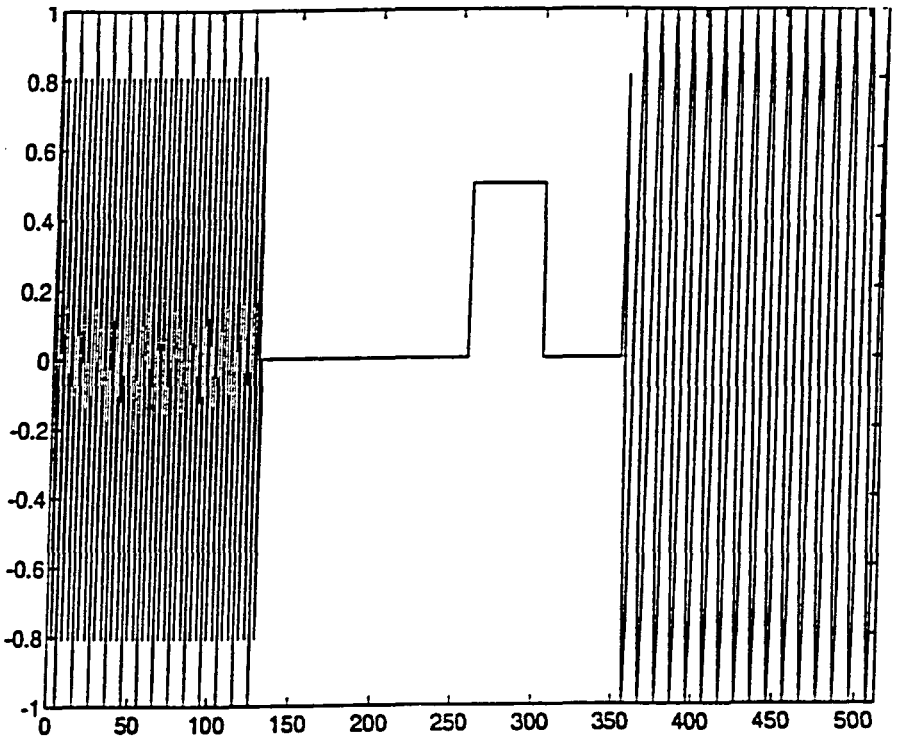

Figure 3.14: The signal used for Example 2

The signal has a 300 Hz component for the first 128 ms, then it is zero for the next 128 ms with a steep discontinuity, a rectangular pulse of width about 45 ms, followed by another 50 ms of zero, and 160 ms of a 100 Hz sinusoid.

The continuous wavelet transform of this signal is computed with Morlet and Mexican hat wavelets. Different Morlet wavelets, with different starting scales were used in the computation. The following figures illustrate the results.

Figure 3.15 is the CWT of the signal with respect to the Morlet wavelet. The high frequency components are seen first in time, then there is a zero region, then comes the rectangular region, followed by another zero region, and the low frequency components, as expected.

Figure 3.16 is the CWT of the same signal with a different Morlet wavelet. The initial scale value has been changed to obtain the spectral components that might have been missed by the first wavelet. However, it looks like the first Morlet wavelet localized the signal better in time.

The signals chosen for these examples are relatively non-complicated, composed of sinusoids and/or simple discontinuities, and therefore, a significant difference between the transforms with different wavelet functions cannot be seen. However, for practical signals which are considerably more complex than the ones used in these examples, a major improvement can be made in the signal analysis if *a better* wavelet is chosen. Unfortunately, as stated before, there is no deterministic way of choosing the right wavelet, or even the right initial parameters of a specific wavelet.

Figure 3.17 shows the real part of this CWT (Morlet wavelet is complex). Note the outline of the rectangular region at the lowest frequency, which draws the real part of the Morlet window.

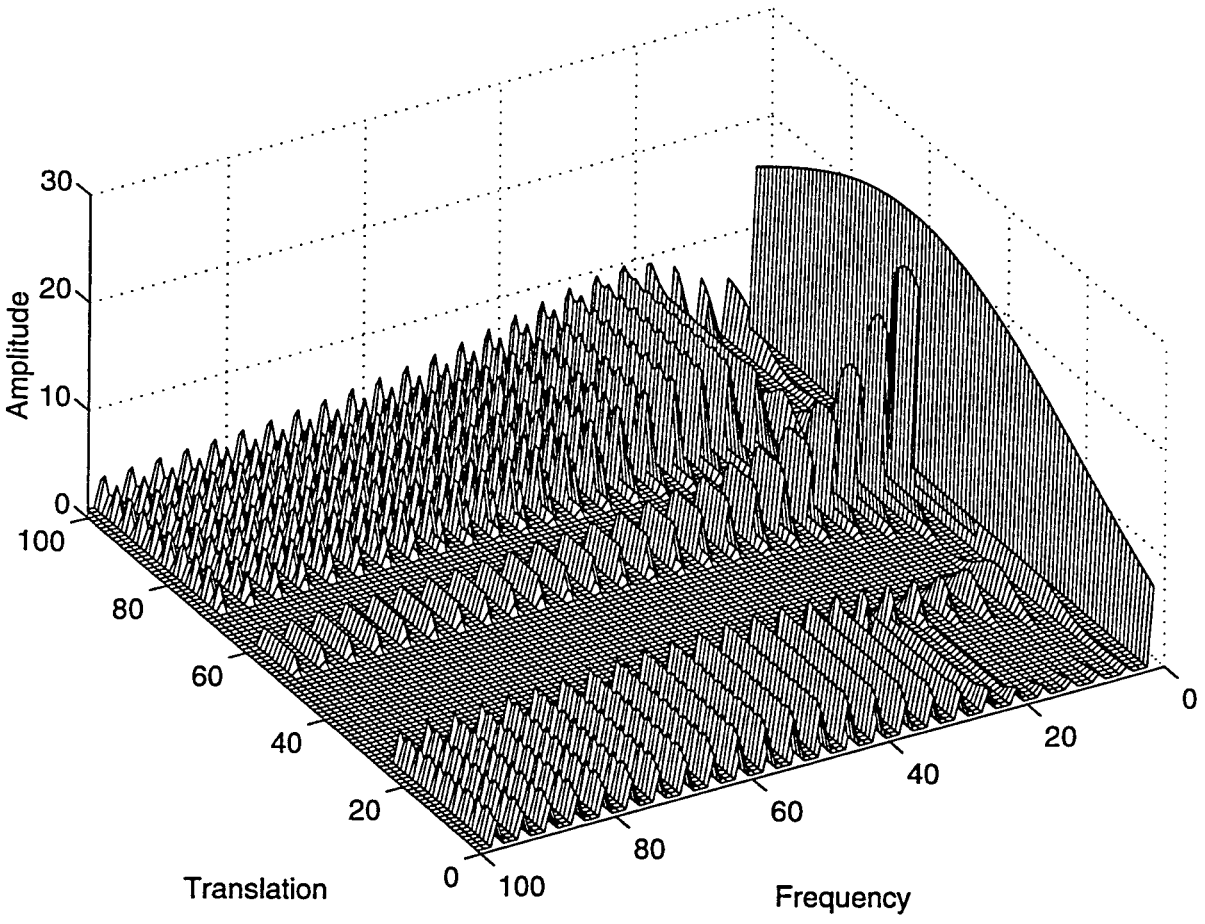

Figure 3.15: The CWT of the signal for example 2, with respect to Morlet wavelet

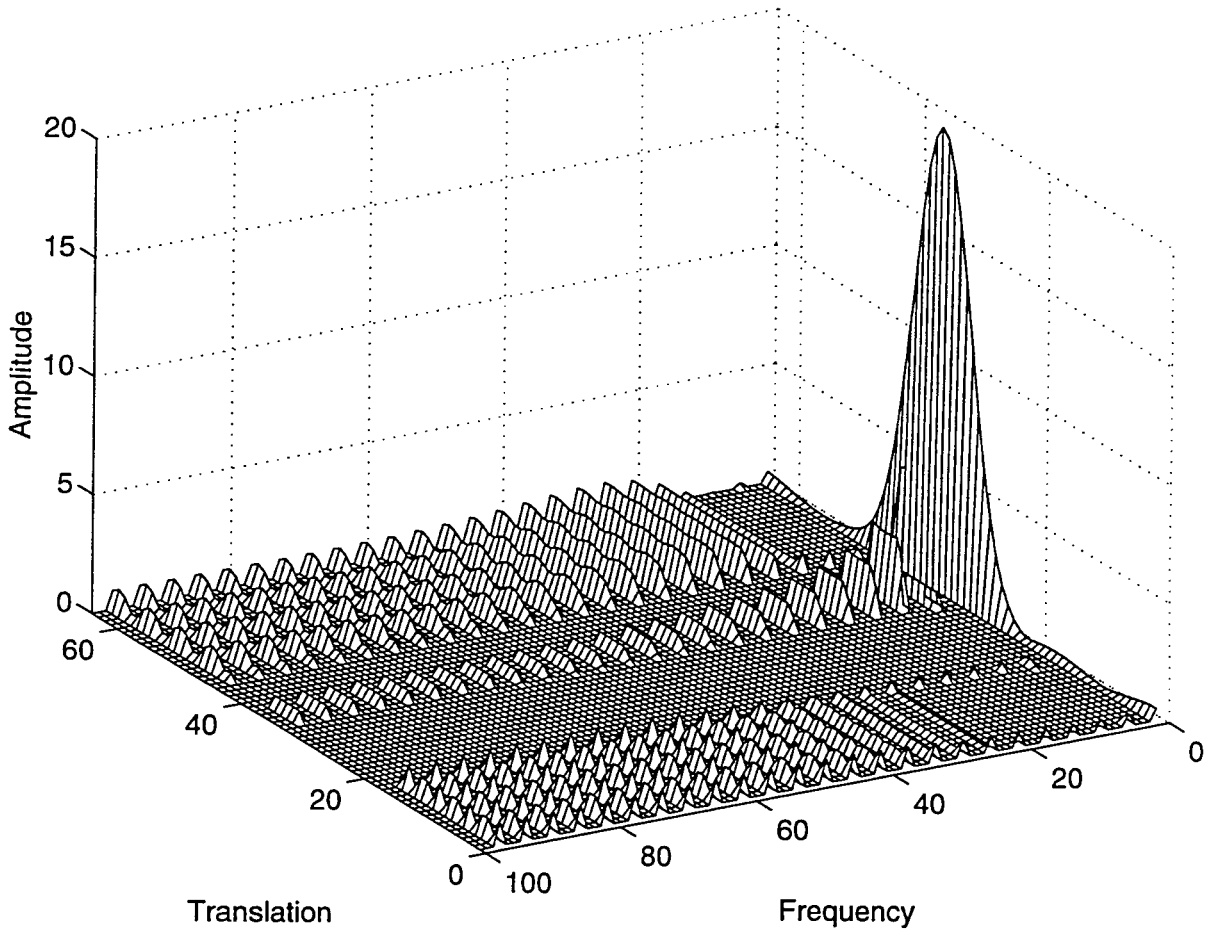

Figure 3.16: Same CWT with respect to a different Morlet wavelet

The continuous wavelet transform using the Mexican hat wavelet is shown in Figure 3.18. Note the differences from the one computed by the Morlet wavelet. Significant differences can be seen at the lowest frequency, and in the times corresponding to the rectangular discontinuity. There is the outline of the rectangle, corresponding to the transition edges, and the in-between area is zero.

This is actually more accurate than the previous CWT with the Morlet wavelet because the region between the two sides of the rectangle is constant in time, which corresponds to the zero frequency. The Mexican hat wavelet localized the spectral components corresponding to that region better than the Morlet wavelet.

3.5 The Wavelet Synthesis

The continuous wavelet transform is a reversible transform, provided that Equation (3.21) is satisfied. Fortunately, this is a very *non-restrictive* requirement. The continuous wavelet transform is reversible if Equation (3.21) is satisfied, even though the basis functions are in general not orthonormal. The reconstruction is possible by using the following reconstruction formula:

$$x(t) = \frac{1}{c_\psi^2} \int_s \int_\tau \Psi_x^\psi(\tau, s) \frac{1}{s^2} \psi\left(\frac{t-\tau}{s}\right) d\tau \, ds \tag{3.20}$$

where c_ψ is a constant that depends on the wavelet used. The success of the reconstruction depends on this constant called, *the admissibility constant*, to satisfy the following *admissibility condition*:

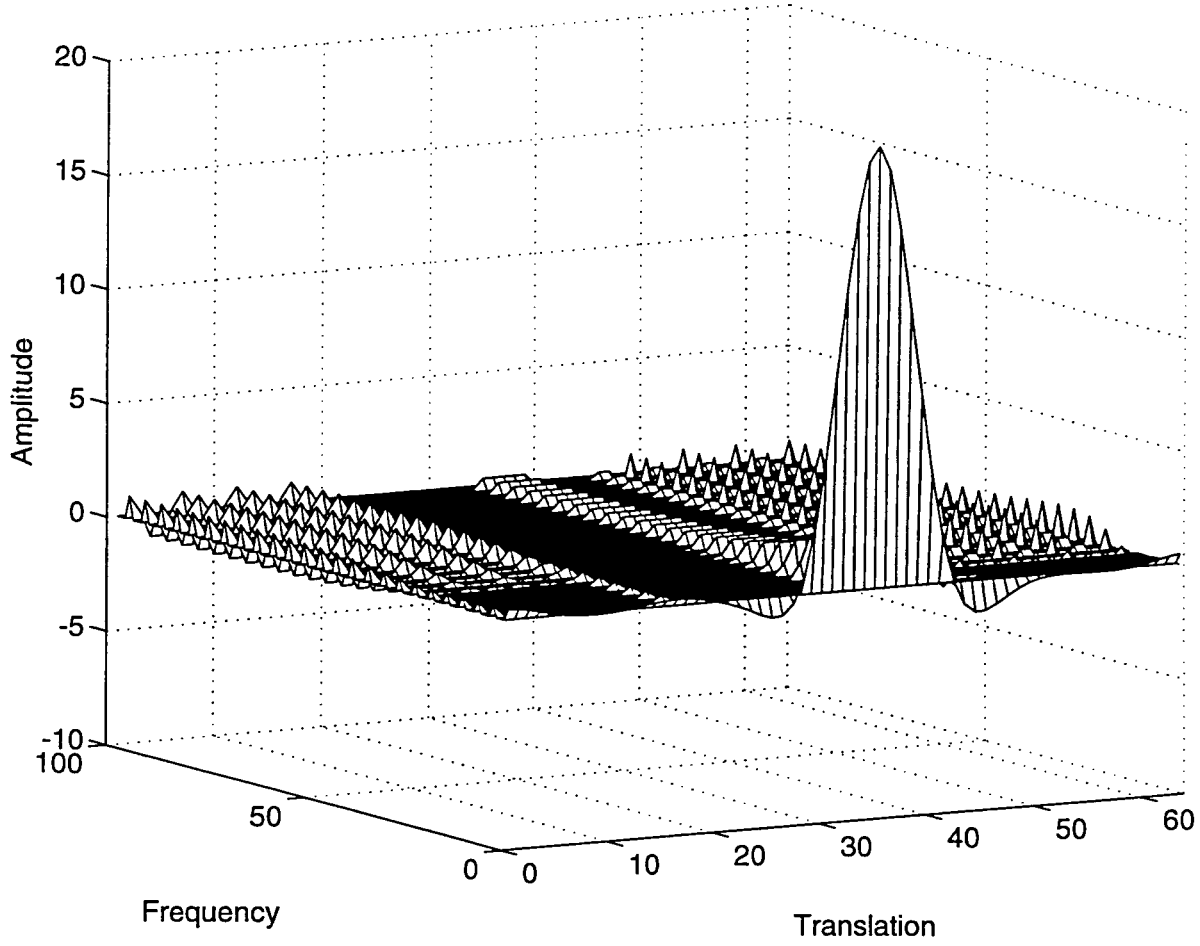

Figure 3.17: The real part of the complex CWT

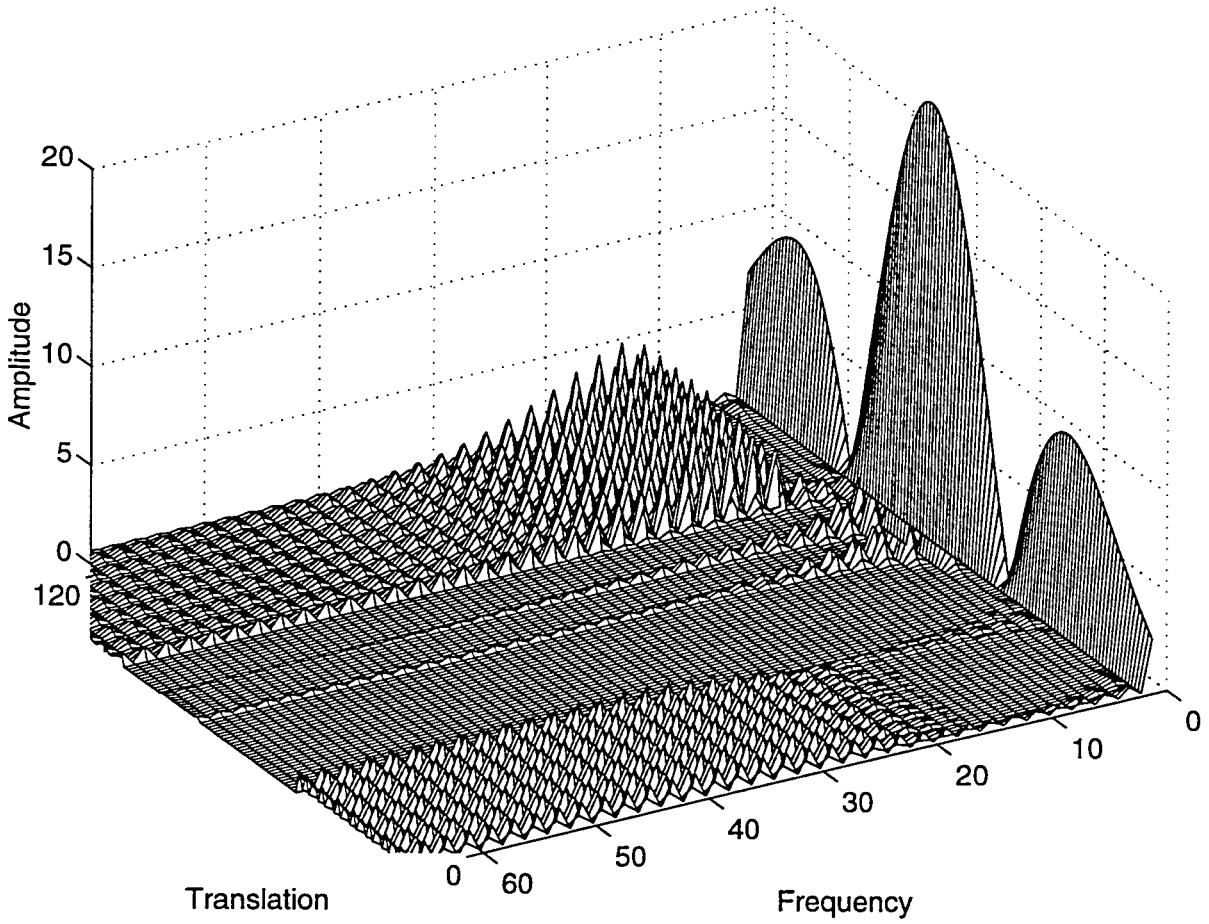

Figure 3.18: The CWT of the signal for Example 2, with respect to the Mexican Hat wavelet

$$c_\psi = \left\{ 2\pi \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\xi)|^2}{|\xi|} d\xi \right\}^{1/2} < \infty \tag{3.21}$$

where $\hat{\psi}(\xi)$ is the FT of $\psi(t)$. Equation (3.21) implies that $\hat{\psi}(0) = 0$, which is

$$\int \psi(t)\, dt = 0 \tag{3.22}$$

As stated above, Equation (3.22) is not a very restrictive requirement since many wavelet functions can be found whose integral is zero. For Equation (3.22) to be satisfied, the wavelet must be oscillatory.

3.6 Discretization of the Continuous Wavelet Transform: The Wavelet Series

In today's world, computers are used to do most computations. It is apparent that neither FT, nor STFT, nor CWT can be practically computed by using analytical equations, integrals, etc. It is therefore necessary to discretize the transforms. As in FT and STFT, the most intuitive way of doing this is simply sampling the time-frequency (scale) plane. And, again intuitively, sampling the plane with a uniform sampling rate sounds like the most natural choice. However, in the case of CWT, the scale change can be used to reduce the sampling rate.

At higher scales (lower frequencies), the sampling rate can be decreased, according to Nyquist's rule. In other words, if the time-scale plane needs to be sampled with a sampling rate of N_1 at scale s_1, the same plane can be sampled with a sampling rate of N_2, at scale s_2, where, $s_1 < s_2$ (corresponding frequencies $f1 > f2$) and $N_2 < N_1$. The actual relationship between N_1 and N_2 is

$$N_2 = \frac{s_1}{s_2} N_1 \tag{3.23}$$

or

$$N_2 = \frac{f_2}{f_1} N_1 \tag{3.24}$$

In other words, at lower frequencies the sampling rate can be decreased which will save a considerable amount of computation time.

It should be noted at this time, however, that the discretization can be done in any way without any restriction, as far as the analysis of the signal is concerned. If synthesis is not required, even the Nyquist criteria is not necessary to be satisfied. The restrictions on the discretization and the sampling rate become important if the signal reconstruction is desired. Nyquist's sampling rate is the minimum sampling rate that allows the original **continuous time** signal to be reconstructed from its **discrete** samples. The basis vectors that are mentioned in section 3.3 are of particular importance for this reason.

As mentioned earlier, the wavelet $\psi(\tau, s)$ satisfying Equation (3.21), allows reconstruction of the signal by Equation (3.20). However, this is true for the continuous transform. The question is: can we still reconstruct the signal if we discretize the time and scale parameters? The answer is "yes", under certain conditions.

The scale parameter s is discretized first on a logarithmic grid. The time parameter is then discretized with respect to the scale parameter, i.e., a different sampling rate is used for every scale. In other words, the sampling is done on the *dyadic* sampling grid shown in Figure 3.19.

Think of the area covered by the axes as the entire time-scale plane. The CWT

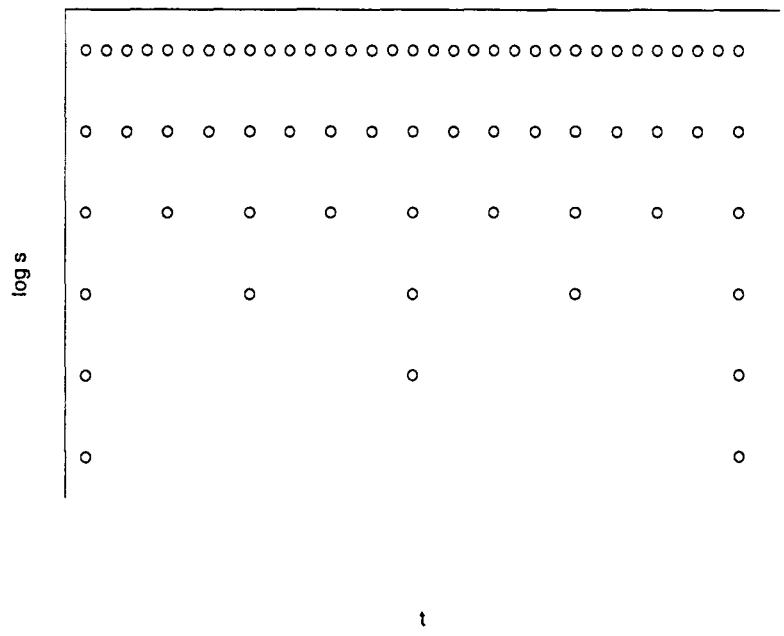

Figure 3.19: The dyadic sampling grid

assigns a value to the continuum of points on this plane. Therefore, there are an infinite number of CWT coefficients. First consider the discretization of the scale axis. Among that infinite number of points, only a finite number are taken, using a logarithmic rule. The base of the logarithm depends on the user. The most common value is 2 because of its convenience. If 2 is chosen, only the scales 2, 4, 8, 16, 32, 64,...etc. are computed. If the value was 3, the scales 3, 9, 27, 81, 243,...etc. would be computed. The time axis is then discretized according to the discretization of the scale axis. Since the discrete scale changes by factors of 2, the sampling rate is reduced for the time axis by a factor of 2 at every scale.

Note that at the lowest scale (s=2), only 32 points of the time axis are sampled. At the next scale value, s=4, since the scale is increased by a factor of 2, the sampling

rate of time axis is reduced by a factor of 2, and therefore, only 16 samples are taken. At the next step, s=8 and 8 samples are taken in time, and so on.

Although it is called the time-scale plane, it is more accurate to call it the *translation-scale* plane because "time" in the transform domain actually corresponds to the shifting of the window in time. For the wavelet series, the actual time is still continuous.

Similar to the relationship between continuous Fourier transform, Fourier series and the discrete Fourier transform, there is a continuous wavelet transform, a semi-discrete wavelet transform (also known as wavelet series) and a discrete wavelet transform.

Expressing the above discretization procedure in mathematical terms, the scale discretization is $s = s_0^j$, and translation discretization is $\tau = k.s_0^j.\tau_0$ where $s_0 > 1$ and $\tau_0 > 0$. Note, how translation discretization is dependent on scale discretization with s_0.

The continuous window function

$$\psi_{\tau,s} = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{3.25}$$

becomes

$$\psi_{j,k}(t) = s_0^{-j/2}\psi(s_0^{-j}t - k\tau_0) \tag{3.26}$$

by inserting $s = s_0^j$, and $\tau = k.s_0^j.\tau_0$.

If $\{\psi_{j,k}\}$ constitutes an orthonormal basis, the wavelet series transform becomes

$$\Psi_x^{\psi_{j,k}} = \int x(t)\psi_{j,k}^*(t)\,dt \tag{3.27}$$

$$x(t) = c_\psi \sum_j \sum_k \Psi_x^{\psi_{j,k}} \psi_{j,k}(t) \tag{3.28}$$

if $\{\psi_{j,k}\}$ are orthonormal.

A wavelet series requires that $\{\psi_{j,k}\}$ are either orthonormal, or biorthogonal, or frame. If $\{\psi_{j,k}\}$ are not orthonormal, Equation (3.27) becomes

$$\Psi_x^{\psi_{j,k}} = \int x(t) \hat{\psi_{j,k}^*}(t)\, dt \tag{3.29}$$

where $\hat{\psi_{j,k}^*}(t)$ is either the *dual biorthogonal basis* or *dual frame.*

If $\{\psi_{j,k}\}$ are orthonormal or biorthogonal, the transform will be non-redundant, where as if they form a frame, the transform will be redundant. On the other hand, it is much easier to find frames than it is to find orthonormal or biorthogonal bases.

The following analogy may make this concept more clear. Consider the whole process as looking at a particular object. The human eyes first determine the coarse view which depends on the distance to the object. That corresponds to adjusting the scale parameter s_0^{-j}. When looking at a very close object, with great detail, j is negative and large (low scale, high frequency, analyses the detail in the signal). Moving the head very slowly, and with very small increments (of angle, of distance, depending on the object that is being viewed), corresponds to small values of $\tau = k.s_0^j.\tau_0$. Note that when j is negative and large, it corresponds to small changes in time, τ, (high sampling rate) and large changes in $s_0^- j$ (low scale, high frequencies, where the sampling rate is high). The scale parameter can be thought of as magnification too.

How low can the sampling rate be and still allow reconstruction of the signal? This is the main question to be answered to optimize the procedure. The most convenient value (in terms of programming) is found to be "2" for s_0 and 1 for τ.

Obviously, when the sampling rate is forced to be as low as possible, the number of available orthonormal wavelets is also reduced.

The continuous wavelet transform examples that were given in this chapter were actually the wavelet series of the given test signals. The parameters were chosen depending on the signal. Since the reconstruction was not needed, the sampling rates were sometimes below the critical value where s_0 varied from 2 to 4, and τ_0 varied from 2 to 8, for different examples.

The discrete wavelet transform is discussed in the next chapter in detail.

CHAPTER 4. THE DISCRETE WAVELET TRANSFORM AND MULTIRESOLUTION ANALYSIS

4.1 Why is the Discrete Wavelet Transform Needed ?

Although the discretized continuous wavelet transform enables the computation of the continuous wavelet transform by computers, it is not a true discrete transform. As a matter of fact, the wavelet series is simply a sampled version of the CWT, and the information it provides is highly redundant, as far as the reconstruction of the signal is concerned. This redundancy, on the other hand, requires a significant amount of computation time and resources.

The discrete wavelet transform (DWT) is sufficient for most practical applications as well as the reconstruction of the signal. The discrete wavelet transform provides enough information, and offers an enormous reduction in the computation time.

The DWT is considerably easier to implement when compared to the continuous wavelet transform. In this chapter the mathematical foundation of the discrete wavelet transform will be introduced, along with its properties and the algorithms used to compute it. As in the previous chapters, examples are provided to aid in the interpretation of the discrete wavelet transform.

4.2 The Discrete Wavelet Transform (DWT)

The foundations of the DWT go back to 1976 when Croiser, Esteban, and Galand devised a technique to decompose discrete time signals [11]. A similar work was done by Crochiere, Weber, and Flanagan on coding of speech signals in the same year [12]. They named the analysis *subband coding.* In 1983, Burt defined a technique very similar to subband coding [13] and named it *pyramidal coding* which is also known as *multiresolution analysis.* Later in 1989, Vetterli and Le Gall made some improvements to the subband coding scheme, removing the existing redundancy in the pyramidal coding scheme [14]. Both of these techniques are explained below. A detailed coverage of the discrete wavelet transform and theory of multiresolution analysis can be found in [4], [15], [16], [9] and [17].

4.2.1 The Pyramidal Coding and The Multiresolution Analysis

The main idea is the same as it is in the CWT. A time-scale (frequency) representation of a digital signal is obtained, using digital filtering techniques. Recall that the CWT is a correlation between a wavelet at different scales and the signal, with the scale (or the frequency) being used as a measure of similarity. The continuous wavelet transform is computed by changing the scale of the analysis window, shifting the window in time, multiplying by the signal, and integrating over all times. In the discrete case, filters of different cutoff frequencies are used to analyze the signal at different scales. The signal is passed through a series of high pass filters to analyze the high frequencies, and it is passed through a series of low pass filters to analyze the low frequencies.

The resolution of the signal which is a measure of the amount of detailed infor-

mation in the signal is changed by the filtering operations, and the scale is changed by *upsampling and subsampling or downsampling* operations.

Subsampling a signal corresponds to reducing the sampling rate, or removing some of the samples of the signal. For example, subsampling by two refers to dropping every other sample of the signal. Subsampling by a factor n reduces the number of samples in the signal n times.

Upsampling a signal corresponds to increasing the sampling rate of a signal by adding new samples to the signal. For example, upsampling by two refers to adding a new sample, usually a zero, between every two samples of the signal. Upsampling a signal by a factor n increases the number of samples in the signal by a factor of n.

Although it is not the only possible choice, the *standard discrete wavelet transform*, where the DWT coefficients are sampled from the CWT on a dyadic grid, i.e., $s_0 = 2$ and $\tau_0 = 1$, yielding $s = 2^j$ and $\tau = k2^j$, as described in Chapter 3 is used in this study.

Since the signal is a discrete time function, the term *function* and *sequence* will be used interchangeably in the following discussion. This sequence will be denoted by $x[n]$, where n is an integer.

The procedure starts with passing this signal (sequence) through a half band digital lowpass filter with impulse response $h[n]$. Filtering a signal mathematically corresponds to convolving the signal with the impulse response of the filter. The convolution operation in discrete time is defined as follows

$$y[n] = x[n] * h[n] \tag{4.1}$$

$$= \sum_k x[k].h[n-k] \tag{4.2}$$

$$= \sum_k h[k].x[n-k] \tag{4.3}$$

A half band lowpass filter is a filter which removes all the frequencies that are above half of the highest frequency in the signal. For example, if a signal has a maximum of 1000 Hz component, then half band lowpass filtering removes all the frequencies above 500 Hz.

The unit of frequency is of particular importance at this time. In discrete signals, frequency is expressed in terms of radians. Accordingly, the sampling frequency of the signal is equal to 2π radians in terms of radial frequency. Therefore, the highest frequency component that exists in a signal is π radians, if the signal is sampled at Nyquist's rate (which is twice the maximum frequency that exists in the signal). Therefore using *Hz* is not appropriate for discrete signals. However, *Hz* is used whenever it is needed to clarify a discussion, since it is very common to think of frequency in terms of *Hz*. It should always be remembered that the unit of frequency for discrete time signals is *radians*.

After passing the signal through a half band lowpass filter, half of the samples can be eliminated according to the Nyquist's rule, since the signal now has a highest frequency of $\pi/2$ radians instead of π radians. Simply discarding every other sample will subsample the signal by two, and the signal will then have half the number of points. The scale of the signal is now doubled. Note that the lowpass filtering removes the high frequency information, but leaves the scale unchanged. The scale

is changed only by the subsampling process.

Resolution, on the other hand, is related to the amount of information in the signal, and therefore, it is affected by the filtering operations. Half band lowpass filtering removes half of the frequencies, which can be interpreted as losing half of the information. Therefore, the resolution is halved after the filtering operation. Note, however, the subsampling operation **after** filtering does not affect the resolution, since removing half of the spectral components from the signal makes half the number of samples redundant anyway. Half the samples can be discarded without any loss of information.

In summary, the lowpass filtering halves the resolution, but leaves the scale unchanged. The signal is then subsampled by 2 since half of the number of samples is redundant. The scale will be doubled after the subsampling.

This procedure can mathematically be expressed as

$$y[n] = \sum_{k=-\infty}^{\infty} h[k].x[2n-k] \tag{4.4}$$

Equation (4.4) is simply the convolution of the signal $x[n]$ with the filter $h[n]$ followed by subsampling.

The original signal is then separated into two parts, one corresponding to the detail information in the signal, and the other corresponding to a coarse approximation of the signal. For coarse approximation, the opposite procedure is followed, that is, the filtered and subsampled signal is upsampled by a factor of two and passed through another half band lowpass filter, with impulse response $h'[n]$.

The upsampling is done by simply inserting zeros between every sample of the signal $y[n]$. This can be expressed as $y'[2n] = y[n]$, $y'[2n+1] = 0$. which creates

an array of numbers whose even indexed terms are equal to terms of the previously obtained signal y[n], and the odd indexed terms are equal to zero. Filtering the upsampled signal with a new filter $h'[n]$ will provide an approximation to the original signal. Call this approximation of $x[n]$ as $\hat{x}[n]$:

$$\hat{x}[n] = \sum_{k=-\infty}^{\infty} h'[k]y'[n-k] \tag{4.5}$$

Note that the filter $h'[n]$ is not a reconstruction or synthesis filter. It just interpolates the zero samples to the best approximation. Therefore the obtained signal $\hat{x}$[n] is not equal to x[n]. The difference $d[n] = \hat{x}[n] - x[n]$ is the detail information that is lost by lowpass filtering the signal. This shows that the original signal can be obtained by adding the detail and the coarse approximation signals. The overall operation is shown in Figure 4.1 .

This process separates the signal into two parts, namely a lowpass coarse approximation of the signal, and a detail (high pass) signal. The approximation of the signal contains frequencies from zero to $\pi/2$, and the detail information contains the frequencies from $\pi/2$ to π. Both signals have only half of the information, and therefore, they are at half the resolution.

The same procedure is then repeated for both half resolution signals. That is, both of them are passed through a halfband lowpass filter, subsampled by two, then upsampled by two, followed by a highpass filter. The signals obtained will have one fourth of the original information. The second level approximation will have frequencies from zero to $\pi/4$, and the detail of the (previous) approximation will have the frequencies from $\pi/4$ to $\pi/2$. The approximation of the (previous) detail signal will have the frequencies from $\pi/2$ to $3\pi/4$, and finally, the detail of the

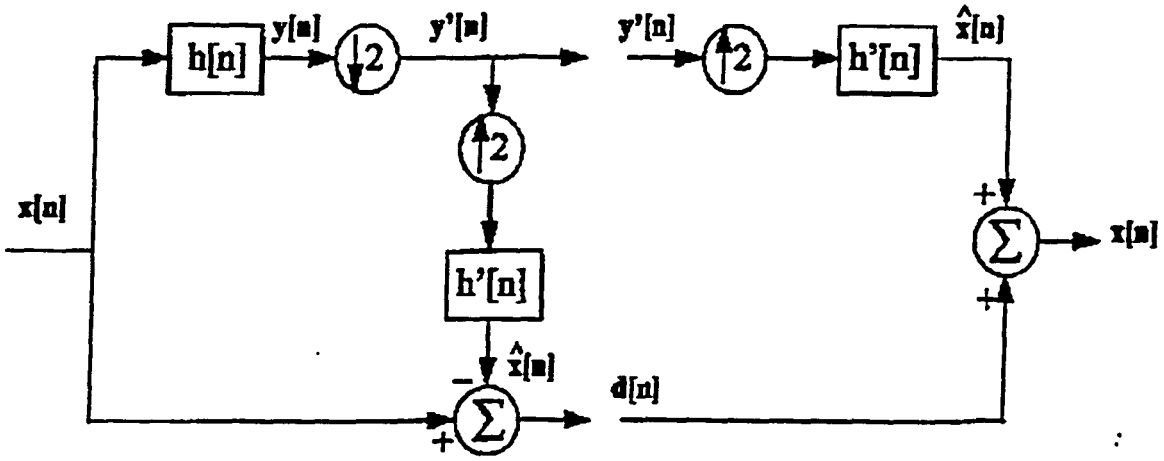

Figure 4.1: Pyramidal Coding (modified from [9])

(previous) detail signal will have the frequencies from $3\pi/4$ to π.

By continuing this process, it is possible to *zoom* into a signal and analyze every frequency band separately. This process, however, is usually iterated only with the approximation parts, and it produces lower resolution signals at higher scales.

4.2.2 Subband Coding

The above algorithm is a fast way of obtaining the time-scale representation of a signal, but it still has redundant information as far as the total number of samples is concerned. Every level of pyramid decomposition generates two signals. The first part contains half the number of points, and it is a low resolution signal (that was labeled as $y[n]$). The second is a detail of the signal, but it contains the same number

of samples as the original signal. The total number of samples is 50% more than the number of samples in the original signal.

The following technique, however, does not cause any redundancy and, therefore, is used very often for the computation of the DWT.

As in pyramidal coding, the low resolution, subsampled signal $y[n]$ is first obtained. But instead of computing the detail signal as a difference from the upsampled and interpolated signal, the other half of the signal is computed using a half band highpass filter $g[n]$. In other words, the signal $x[n]$ is passed through a lowpass filter and then it is subsampled. The output signal has the information from zero to $\pi/2$ as explained above. Then $x[n]$ is also passed through a half-band highpass filter, followed by subsampling. The output is a signal that contains the frequencies from $\pi/2$ to π.

Note in this case that both of the signals are *subsampled* by 2 and that both of them have half the total information. Therefore, two signals at half the resolution and twice the scale are obtained. More importantly, the total number of samples is unchanged, i.e., there is no redundant information. The above procedure is one level of DWT computation. The output of the highpass portion is called *level 1, DWT coefficients.*

The procedure is repeated by using the level 1 lowpass output. That is, the output of the lowpass filter at level 1 is again divided into two by passing it through halfband lowpass and highpass filters, and subsampling by two. The end result is two more signals, at one fourth of the resolution and four times the scale. The output of the highpass portion of this level is called *level 2 DWT coefficients.* The lowpass portion is then taken and further decomposed. The procedure continues until only

one sample is left.

An alternative interpretation of subband coding is explained next with an example. As stated above, filtering a signal (lowpass and highpass), and then subsampling the outputs, yields two signals with half the number of samples each. Note that, although the number of the samples is halved, the signals still correspond to the entire duration of the original signal, which means that half the number of points are available to represent the signal. Therefore, time resolution is halved by two.

Suppose the procedure was started with a 1024 points long signal which has frequencies from 0 to π. At the end of the first level, there would be two 512 samples long signals, both corresponding to a $\pi/2$ long frequency interval; one of them from zero to $\pi/2$, and the other from $\pi/2$ to π. Then the low frequency portion would be decomposed again. Therefore, in the second layer, there would be two 256 samples signals, both of them corresponding to a $\pi/4$ long frequency interval, one of them from zero to $\pi/4$ and the other from $\pi/4$ to $\pi/2$.

At the beginning there was a 1024 points signal corresponding to a frequency interval of π radians. In the first level 512 points represent the original 1024 points signal, and in the second level 256 points represent the original signal. Therefore, the *time resolution* is halved at level 1 and it is further halved (to the one fourth of the original) at level 2. In other words, the time resolution at level 0 (the original signal), is twice as good as the one at level 1, and it is four times as good as the one at level 2 and so forth.

The frequency resolution, on the other hand, gets better as the algorithm progresses. In the first level, the two signals correspond to a $\pi/2$ radians long interval, and in level 2, the signals correspond to a $\pi/4$ radians long interval. Therefore, the

frequency resolution at level 0 is the poorest since it is the original time-domain signal, and there is no discriminating frequency information. The signal corresponds to a π radians long interval, i.e., all the frequencies that exist in the signal. The frequency resolution at level 1 is twice the frequency resolution at level 0, and the frequency resolution at level 2 is twice the frequency resolution at level 1 and so forth.

In summary, at higher levels (higher scales, lower frequencies) there is better frequency resolution since the frequency range gets narrower and narrower; however, the time resolution gets worse since only half of the number of samples are used to characterize the signal at each level.

The change in the time and the frequency resolutions of the discrete wavelet transform are exactly the same as those of the continuous wavelet transform. (see Figure 3.8). Figure 4.2 illustrates the subband coding procedure.

One important property of the discrete wavelet transform is the relation between the impulse responses of the highpass and lowpass filters. The highpass and lowpass filters are not independent of each other, and they are related by

$$g[L-1-n] = (-1)^n h[n] \tag{4.6}$$

where $g[n]$ is the highpass, $h[n]$ is the lowpass filter, and L is the filter length (in number of points). Note that the two filters are odd index alternated reversed versions of each other. Lowpass to highpass conversion is provided by the $(-1)^n$ term.

The two filtering and subsampling operations can be expressed by

$$y_{high}[k] = \sum_n x[n].g[-n+2k] \tag{4.7}$$

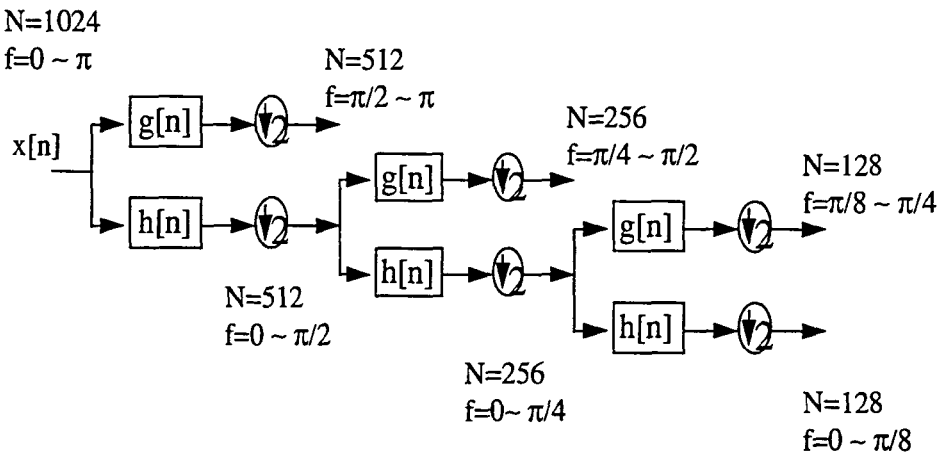

Figure 4.2: Subband coding block diagram

$$y_{low}[k] = \sum_n x[n].h[-n + 2k] \tag{4.8}$$

The reconstruction in this case is very easy since halfband filters form orthonormal bases. The above procedure is followed in reverse order for the reconstruction. The signals at every level are upsampled by two, passed through the *synthesis filters* $g'[n]$, and $h'[n]$ (highpass and lowpass, respectively), and then added. The interesting point here is that the analysis and synthesis filters are identical to each other, except

for a time reversal. Therefore, the reconstruction formula becomes (for each layer)

$$x[n] = \sum_{k=-\infty}^{\infty} [y_{high}[k]g[-n+2k] + y_{low}[k]h[-n+2k]] \tag{4.9}$$

However, if the filters are not ideal halfband, then perfect reconstruction cannot be achieved. Although it is not possible to realize ideal filters, under certain conditions [18], it is possible to find filters that provide perfect reconstruction. These filters will be named *wavelet function* and *scaling function* for the highpass and lowpass filters, respectively. However, the relation between the wavelet and scaling functions will be similar to the relation between the highpass and lowpass filters. The most famous ones are known as *Daubechies wavelets* [4]. These are used in the examples given below.

4.3 Examples

The plots for the examples in this chapter will be somewhat different from the ones in the previous chapters, and they should be interpreted as explained below.

The program written for the computation of DWT accepts signals whose lengths are powers of two. If the signals' length is not a power of two, then the signal is extended periodically to a length which is a power of two, or the decomposition stops whenever the number of samples can not be divided by 2. The number of levels that the signal is decomposed is determined by the length of the signal. For example, if the signal length is 1024, ten levels of decomposition are possible. However, it is quite unnecessary to go that far since the number of samples in the signal decreases by a factor of "2" at every level. For example for a 1024 sample signal, there are only 32 points left at the fifth level to represent the 1024 sample signal, and the time

resolution gets very poor. The typical number of levels depends on the signal length.

Recall that the DWT coefficients are the outputs of the highpass filters followed by subsampling. However, both outputs of the last level are included in the transformed signal.

The DWT is displayed in two different ways. The first is used most often. In this display technique, all the computed coefficients are simply concatenated one after the other. For example, suppose we have a 200 points long signal. This will be decomposed into 3 levels. In the first level it will be decomposed into two 100 points signals. The highpass 100 points will be the level 1 DWT coefficients. The lowpass 100 points are then decomposed into two 50 points signals. The highpass 50 points become the level 2 DWT coefficients, and the lowpass 50 points are further decomposed into two 25 points signals. The decomposition will stop here since the total number of points is not divisible by 2. The level 3 coefficients will be the 25 highpass *and 25 lowpass* transform coefficients placed one after the other. Therefore, the transformed signal will be 25 lowpass points at level 3, 25 highpass points at level 3, 50 highpass points at level 2 and 100 highpass points at level 1, concatenated in this order. Note that the total number of samples adds up to 200, which is the number of samples in the original signal.

This display method is usually difficult to interpret since it gives a one-dimensional plot in which both time and frequency information are embedded. However, the signal in this format is more suitable for use by a neural network, and therefore, it is commonly followed.

The other display method is placing these coefficients in a matrix where the rows represent the frequency and the columns represent the translation. The matrix

is then color coded on a gray scale according to the amplitude of the values. This display method creates a two dimensional time-frequency representation.

4.3.1 Example 1

The first example is a *chirp* signal, whose frequency changes continuously. The chirp signal is shown in Figure 4.3.

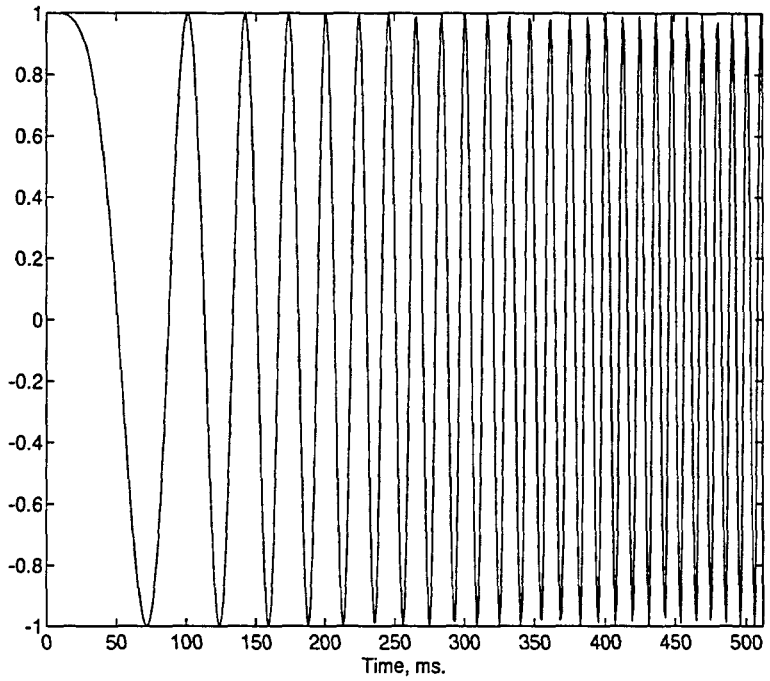

Figure 4.3: The chirp signal

For this signal, there is a constant increase of frequency along the time axis. The computed DWT coefficients are displayed in Figures 4.4 and 4.5 in the two models described above.

Figure 4.4 is the DWT of the chirp signal in *augmented time format* created with

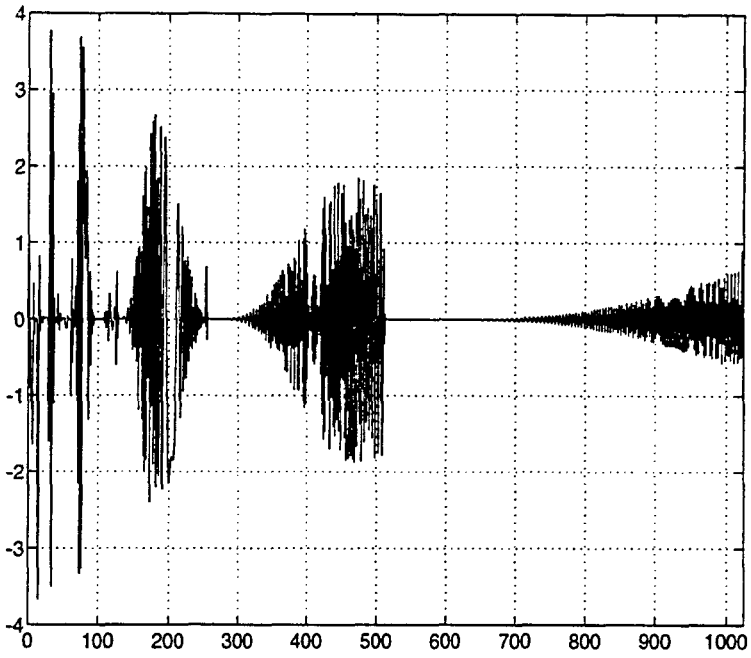

Figure 4.4: DWT of the chirp signal

the first method. The horizontal axis corresponds to time. Note that the frequency information can not be seen in this figure easily; although, it is embedded in the plot. Recall that the last 512 points correspond to level 1 ($\pi/2$ to π). The 256 points preceding level 1 correspond to level 2 ($\pi/4$ to $\pi/2$), the 128 points preceding level 2 correspond to level 3 ($\pi/8$ to $\pi/4$), and so forth.

Figure 4.5 is the DWT of the chirp signal in *matrix display format*. The formation of this display matrix in Figure 4.5 can be of special interest. The matrix has N columns and rows where N is the signal length, provided that N is a power of 2. In this example, the signal is of length 1024. (Half of the signal is shown in Figure 4.3.)

The first row is filled with the single value of the last level's lowpass output. That is, the entire first row is filled with one number. The second row is filled with

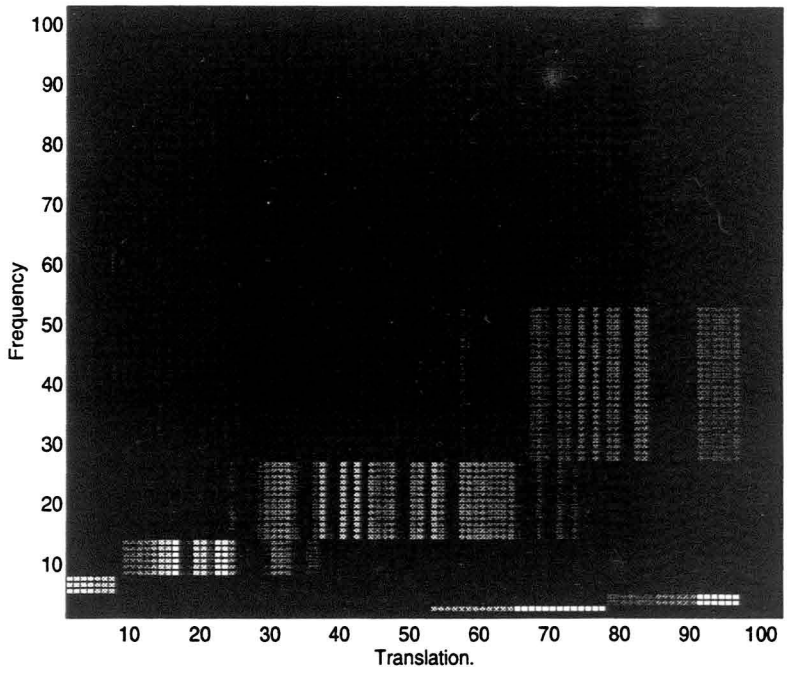

Figure 4.5: DWT of the chirp signal, matrix display format

highpass output of the last level, again the same value for the entire row, since there is only one coefficient at the last level. The time axis is 1024 points long (as the original signal), and the rows are only 1 point wide.

The third and fourth rows are filled with the highpass output of the *number of layers-2* (the second before the last) layer. There are two coefficients at this layer; the first 512 points of the third and the fourth rows are filled with the first of the two coefficients, and the second 512 points are filled with the second coefficient. Note that, the frequency resolution is decreased, (frequency interval increased to 2 rows) and the time resolution is increased (the time interval decreased from 1024 to 512) by a factor of 2.

The next "4" rows (5,6,7,8) are filled with the "4" coefficients of the highpass

output of the *number of layers-3* (third from the end) layer. The four coefficients at this level are placed on the four consecutive regions of the time axis similar to the previous level. The first coefficient is placed through columns 1 to 256, the second coefficient is placed through columns 257 to 512, the third coefficient is placed through columns 513 to 768 and the fourth coefficient is placed through columns 767 to 1024.

Finally, the last 512 rows(very poor frequency resolution at high frequency) are filled with the 1024 coefficients (very good time resolution) one value for every column. The result is displayed in Figure 4.5 (for the first 512 rows).

One can see the increasing height of the columns at higher frequencies, indicating decreasing frequency resolution. The time resolution properties cannot be seen from this figure.

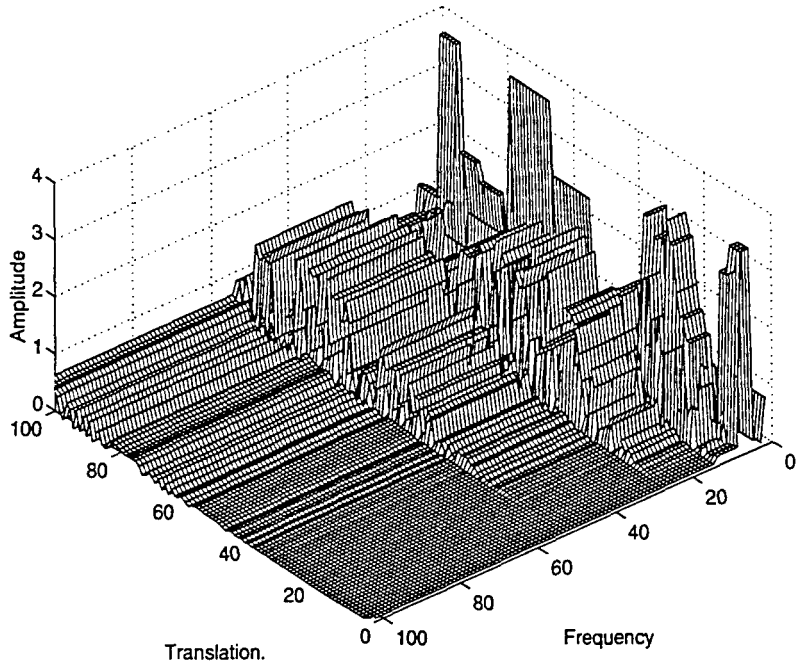

Figure 4.6: DWT of the chirp signal, matrix display format

The 3-D plot corresponding to the display matrix format (i.e., the amplitudes are not color coded) is given in Figure 4.6. It is not possible to tell much from this figure, except that the sinusoids are seen corresponding to different frequencies at different times. Among the above given three displays, the matrix format (the second) gives the best interpretation when viewed in color.

4.3.2 Example 2

The second example is the same as Example 2 used in the continuous wavelet transform given in chapter 3 (Figure 3.14). Again the augmented time and matrix format displays are given. The signal is 512 points long, and therefore, nine levels are used.

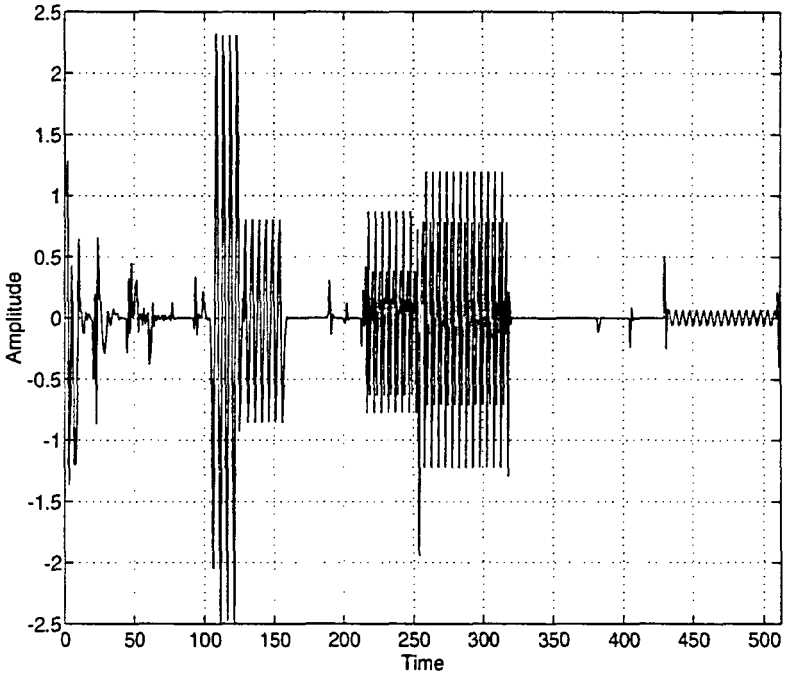

Figure 4.7: DWT of the signal in Figure 3.14, augmented time format

Figure 4.7 shows the augmented time format. Similar to the previous example, the last 256 points corresponds to the first level. The preceding 128 points correspond to level 2 and so forth. The maximum energy can be seen at level 2 and level 4 signals give information about the existence of the corresponding frequencies.

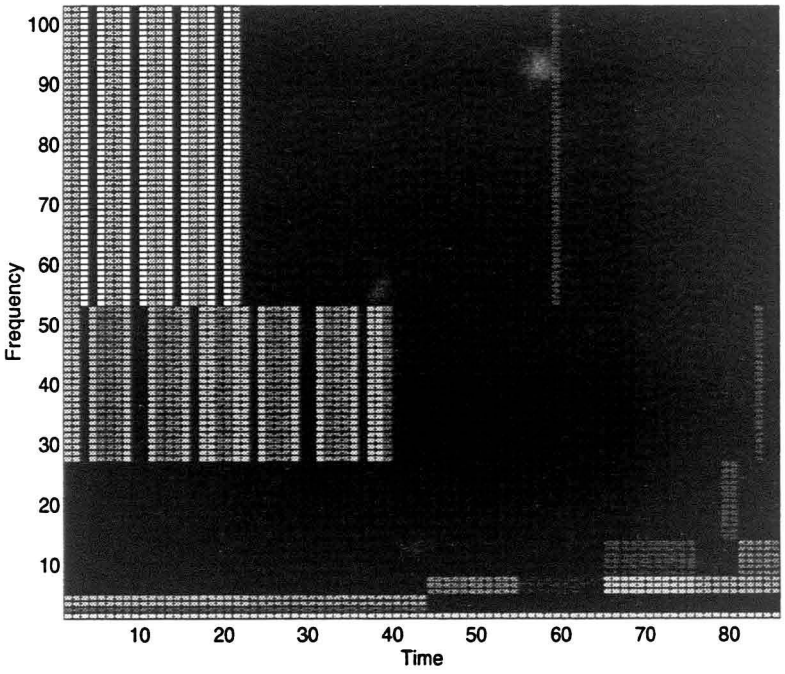

Figure 4.8: DWT of the signal in Figure 3.14, matrix format

Figure 4.8 shows the matrix format. The high frequency components at early times and low frequency components at later times can easily be seen from the figure. Also notice the height of lines increasing and the width decreasing at high frequencies. This once again indicates good time resolution and poor frequency resolution. At low frequencies, lines are shorter but wider, which is due to good frequency resolution but poor time resolution.

CHAPTER 5. ARTIFICIAL NEURAL NETWORKS

Signal classification and clustering have always been of special interest in signal processing. Signal classification is the second step after processing the signal, and this step is usually accomplished by conventional *pattern recognition* techniques. More recently *artificial neural networks*, or simply *neural nets*, have found considerable application for signal classification.

Neural networks try to simulate the pattern recognition capabilities of the human brain. Neural networks, physically realized by silicon chips, are five to six (a factor of $10^5 - 10^6$) orders of magnitude faster than the neurons in a human brain. However, a neural network cannot get close to the performance of the brain because the brain typically contains nearly 10 billion neurons with 60 trillion synapses or connections [19]. This, of course, is not possible for a physically realizable network, where the number of neurons and connections are around six orders of magnitude less than that of the brain.

One of the main objectives of this study was to investigate the performance of wavelet transforms on biological signals, specifically EEG signals. The wavelet transform processed signals were presented to two different neural network architectures to be classified or clustered. (The difference between classification and clustering will be described in the following sections.)

There has been an enormous amount of work on neural networks in the last 50 years, and there have been many books, articles and tutorials [21], [22] published. Therefore, only a brief overview of neural networks will be given in this chapter, emphasizing the *multilayer perceptron* with backpropagation learning rule and the *k-means clustering algorithm.*

5.1 Components of Basic Neural Networks

Neural networks consist of layers of simple processing units called *neurons* or *nodes.* These nodes perform simple mathematical operations, similar to the operations performed by real neurons. A finite set of inputs, also called *input nodes*, constitute the first layer of the neural network. However, the input nodes basically feed the data into the neural network; they perform no mathematical operations. Therefore, the input layer is usually not counted when the number of layers are specified for a particular network.

Every neural network also has *output nodes* which perform mathematical operations. The nodes which perform computations are also called *processing elements (PEs).* Every processing element has a *state* associated with it. This state is a scalar number which changes during the *learning procedure.*

The input and output nodes exist in every neural network. However, most neural networks also have one or more *hidden layers* which also include a set of nodes. The hidden layer nodes are processing elements like the output nodes. All the nodes in all the layers are connected to each other. The connection scheme determines the flow of the information between the nodes. For example, if all the nodes of one layer are connected to all the nodes of the next layer, the neural network is said to be *fully*

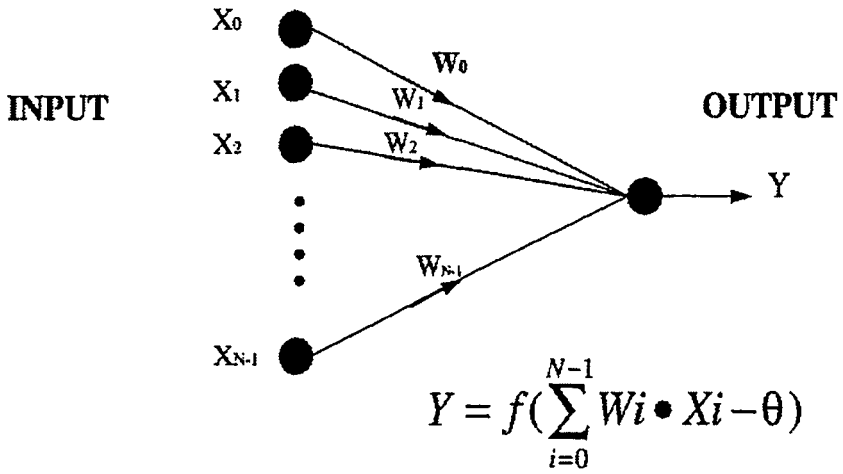

Figure 5.1: Single layer neural network

interconnected. Every connection is usually directed from one node to another, and every connection has a *weight* associated with it. A single layer neural network is illustrated in Figure 5.1.

The N input nodes are named X_i, $i = 0, 1, ..., N-1$. All the input nodes are connected to the output node Y, and every connection has a *weight value*, W_i, $i = 0, 1, ..., N-1$. The only node in Figure 5.1 that performs a computation is the output node. This node basically adds the *weighted values* of the input nodes. If applicable, a threshold value, θ, is subtracted from the weighted input value. The resulting value is passed through a non-linear *activation function*, $f(x)$. This final value becomes the new state of the output node. The activation function is usually selected to be one of three popular functions, namely, the threshold function (hard-

limiter), piecewise linear function, or the sigmoid function.

5.2 Activation Functions

5.2.1 The Hard Limiter (Threshold Function)

The input-output characteristic of the hard-limiter function is illustrated in Figure 5.2 . The hard-limiter function is defined as

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{5.1}$$

The hard limiter function is often used in classification problems in many neural networks, such as Hopfield networks, where binary outputs are needed.

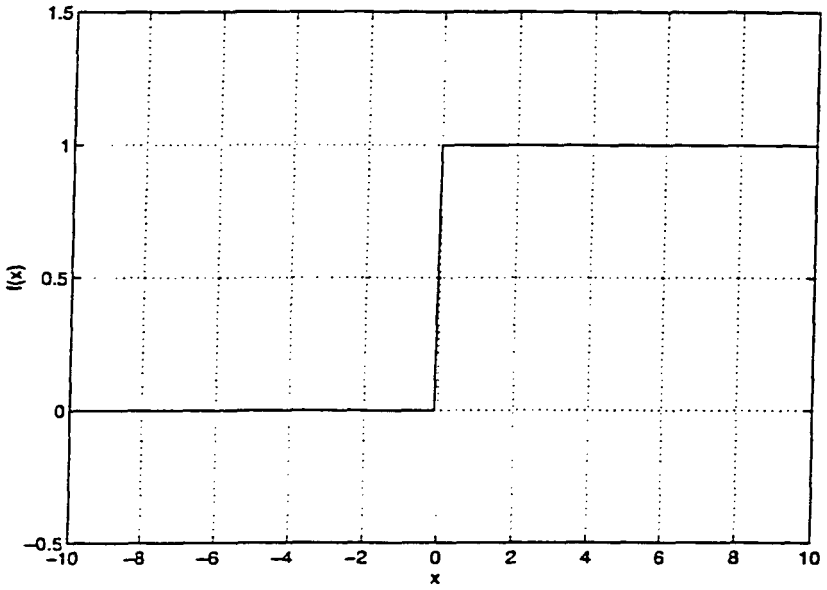

Figure 5.2: The hard limiter function

5.2.2 The Piecewise Linear Function

The piecewise linear function has an output value equal to the input value in a given interval. The piecewise linear function is plotted in Figure 5.3 and defined as follows:

$$f(x) = \begin{cases} 1 & \text{if } x \geq 1/2 \\ x & \text{if } -1/2 < x < 1/2 \\ 0 & \text{if } x \leq -1/2 \end{cases} \tag{5.2}$$

The piecewise linear function is usually used for function approximation.

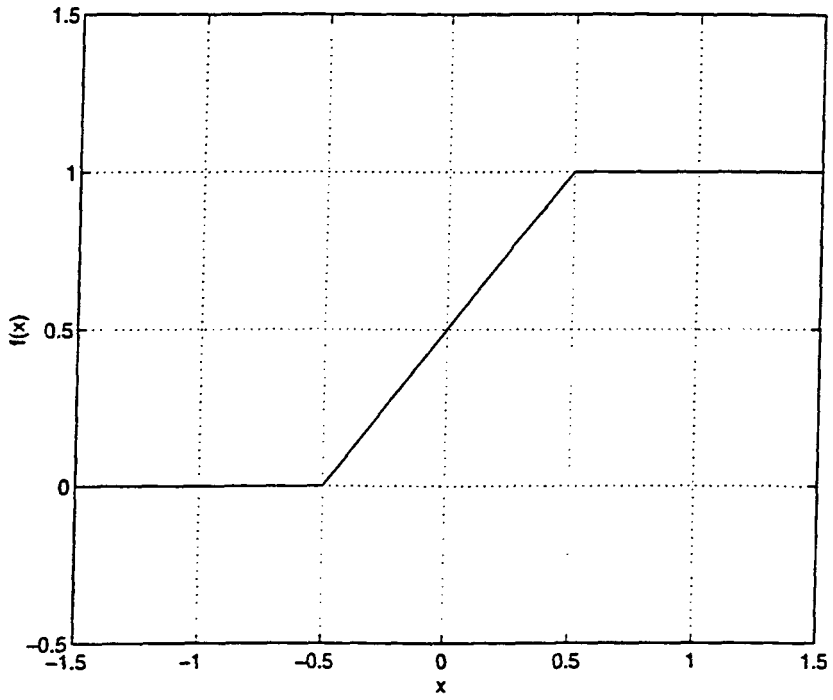

Figure 5.3: The piecewise linear function

5.2.3 The Sigmoid Function

The sigmoid function, the most commonly used function, forces any value between minus and plus infinity into the [0 1] range. The multilayer perceptron with backpropagation usually uses the sigmoid function because it is differentiable. This is useful for implementing the backpropagation algorithm.

The input-output characteristics of the sigmoid function are given in Figure 5.4, and it is defined as follows:

$$f(x) = \frac{1}{1 + e^{-kx}} \tag{5.3}$$

The parameter k determines the slope of the sigmoid function; as k approaches infinity, the sigmoid function approaches the hard limiter.

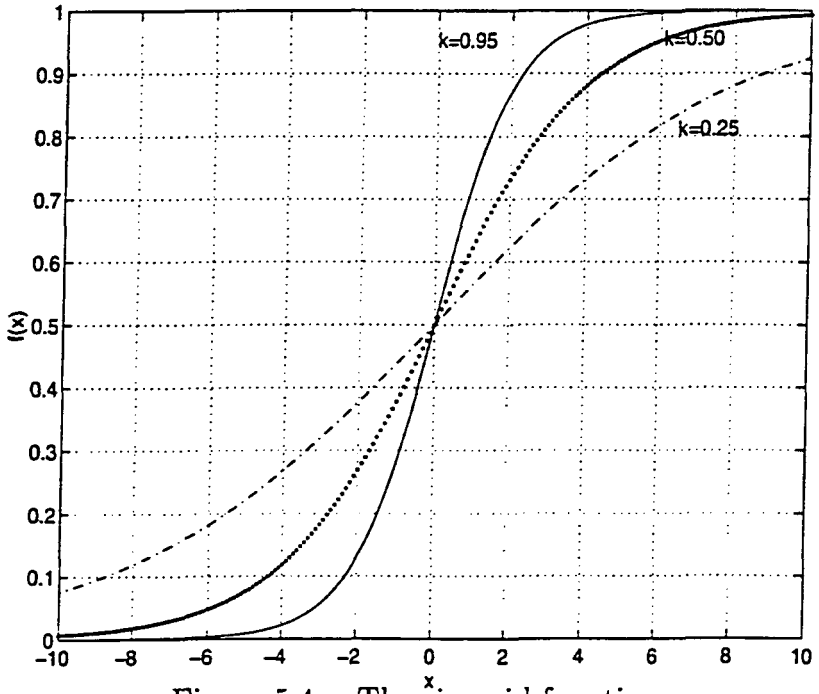

Figure 5.4: The sigmoid function

5.3 The Learning Process

Neural networks can be classified into two classes according to their learning rule. The majority of neural networks are trained with a set of data of known classes. This data is passed through the network many times through an iterative procedure. At every iteration the network learns more about the data and changes the weight and/or node values, accordingly. This data, with known classes, is called *training data*, and this type of learning is called *supervised learning.* The Hopfield network, the Hamming network, and single and multilayer perceptrons (with backpropagation) use supervised learning algorithms.

The iterative learning procedure continues until the training is complete. The training ends when the neural network can classify all the training data correctly and/or when the weights do not change from one iteration to the next. This is called the *convergence of the learning algorithm.* After training is over, the network is presented a set of test data for validation and classification. The weights calculated during the training are used for this classification.

The learning rules for neural networks are somewhat random procedures. The initial selection of the weights, the number of hidden layers, and the number of nodes in the hidden layer all affect the performance of the neural network. However, there are no criteria for the selection of the number of nodes, number of hidden layers, or initial selection of the weights. Therefore, neural networks with various parameters are implemented a number of times to get the optimum performance.

There are many other learning algorithms and network architectures available, and many books and articles have been published in this area. Simon Haykin's recent book, *Neural Networks, A Comprehensive Foundation* is an excellent reference guide

[19].

However, training data of known classes may not always be available. Even the number of classes may not be known. In these cases the learning procedure is said to be *unsupervised.* The networks using unsupervised learning rules usually cluster the data into groups. Self organizing feature maps (Kohonen network), the k-means clustering algorithm, and the minimax clustering algorithm are examples of unsupervised learning rules.

5.4 The Perceptron Model

The perceptron model, developed by Rosenblatt in the early sixties [23] is a single layer network similar to the one shown in Figure 5.1. It uses the hard limiter activation function. It accepts continuous valued inputs, and outputs binary values. The perceptron is a *feedforward* network in which the information flows only from input towards the output.

The perceptron model is a supervised learning rule which is guaranteed to converge, if and only if, the data are *linearly separable.* This is a very restrictive requirement since this is not the case in many applications. An example of linearly separable and not linearly separable data sets is given in Figure 5.5 .

For two sets of N dimensional data to be linearly separable, a decision boundary of dimension $N-1$ should be found. In the upper plot of Figure 5.5 the two dimensional data sets can be separated with a one dimensional decision boundary, a line. However, the data sets in the bottom plot cannot be separated with a one dimensional decision boundary; therefore they are not linearly separable. Multilayer perceptrons, however, are *theoretically* capable of classifying sets having arbitrary

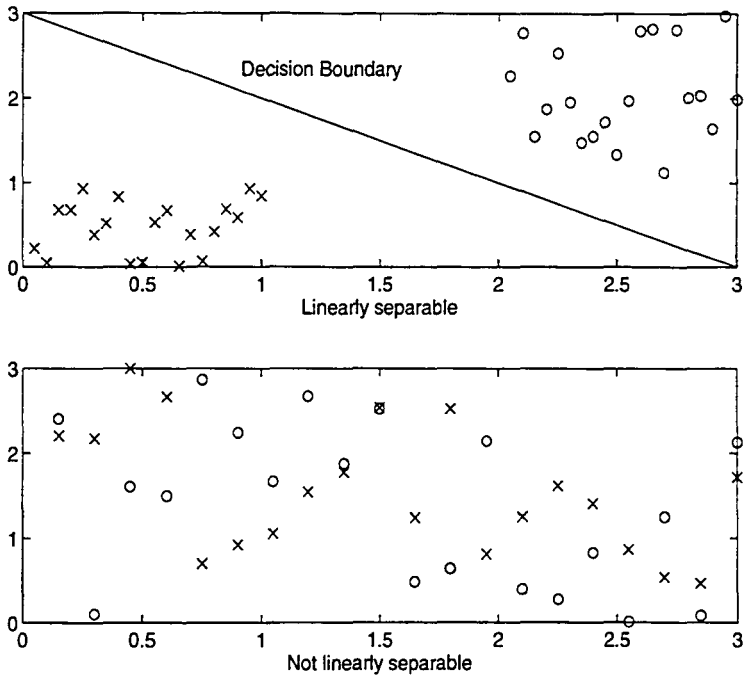

Figure 5.5: Linearly separable, not linearly separable sets of data

(not necessarily linear) decision boundaries.

5.4.1 Perceptron Convergence Algorithm

The single layer perceptron algorithm is given below. The network architecture is identical to the one given in Figure 5.1 if there is one output node. If there is more than one output, all the input nodes are connected to all the output nodes. The network is, therefore, fully interconnected.

1. Randomly initialize all the weights to some small numbers.

2. Present input to the input nodes. There must be as many input nodes as the number of samples in the data. This is called the dimension of the data.

3. Calculate the output value for each output node:

$$y(k) = f\left(\sum_{i=1}^{N} W_i(k).a_i - \theta\right) \tag{5.4}$$

where f is the activation function, k is the iteration number, N is the number of input nodes, a_i is the current value of the i^{th} input node, W_i is the weight that connects the i^{th} input node the output node, and θ is a threshold value. If there is more than one output node, the above equation is repeated for every output node.

4. Adapt the weights according to Equation (5.5), where the weight values at the $(k+1)^{th}$ iteration are written in terms of the learning rate η, current values of the input and output nodes, the correct output value d, and the weight values at the k^{th} iteration.

$$W_i(k+1) = W_i(k) + \eta.[d - y(k)].a_i \qquad \text{i=1,2,....,N} \tag{5.5}$$

Increasing the learning rate, η, may reduce the convergence time, but increasing it too much may yield incorrect results. The pair (**a**,d) is called a *training pair.*

5. Continue iteration by returning to step 2 (increase k by one) until convergence. The network is said to be converged when the weight values do not change from one iteration to the next. In other words, continue iteration until $W_i(k+1) = W_i(k)$.

As mentioned above, the single layer perceptron will not converge if the data are not linearly separable. Therefore, it is seldom used. However, the perceptron algorithm becomes extremely powerful if one or more hidden layers are added to the network architecture. The resulting network architecture is called the *multilayer perceptron.*

5.5 Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) is a feed-forward network with one or more hidden layers between the input and the output nodes. At each level, the node values are computed exactly as they are computed in the single layer perceptron (with Equation 5.4). The MLP network architecture is given in Figure 5.6.

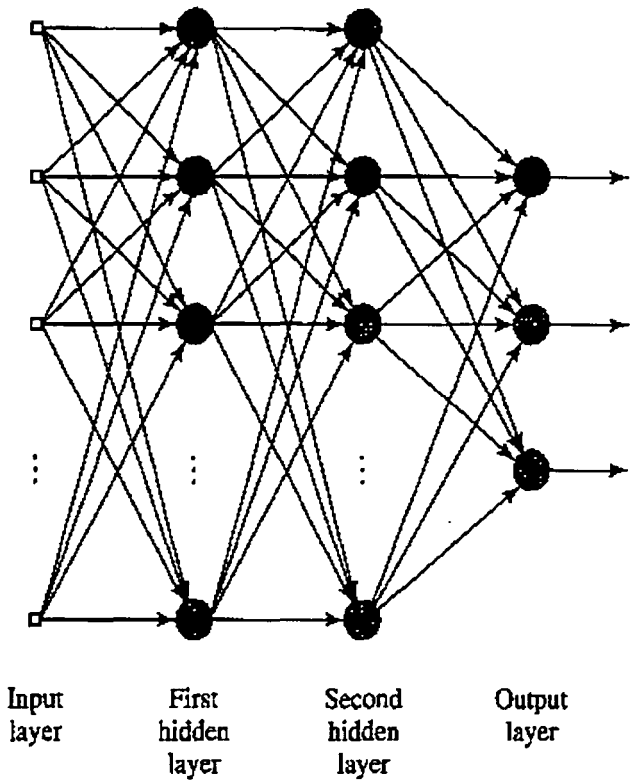

Figure 5.6: The MLP with two hidden layers (from [19])

The MLP uses *the backpropagation learning algorithm* for updating the weights. This is a supervised learning rule where there is a set of training data with known classes available. Multilayer perceptron/ backpropagation (MLP/BP) uses the sigmoid activation function because it can produce continuous output values, and it can

be differentiated.

The weight update rule for the backpropagation algorithm will be explained below, followed by the step by step algorithm. The derivation of this algorithm will not be given here since it can be found in many books and articles such as [19] and [23].

5.5.1 Backpropagation Learning Rule for the MLP

For most practical applications only one hidden layer (two layers in total with the output layer) are used in MLPs. Three hidden layer MLPs are considerably more difficult to code, and their performance is not necessarily better. Three hidden layer MLPs are usually used to realize very complex decision boundaries.

Figure 5.7 shows a two layer MLP with nodes and weights. This is a generic MLP architecture used in many MLP implementations. The algorithm presented below uses the notation for nodes and for weights as illustrated in Figure 5.8.

x_i, $i = 1, 2, ...l$, are the input nodes, h_j, $j = 1, 2, ..., m$ are the hidden layer nodes, and the o_k, $k = 1, 2, ..., n$ are the output nodes. Also note that one more node is added to both input and hidden layers with a constant value of 1. This node serves as the threshold θ, in Equation (5.4). This node always has the same state, 1, and does not change throughout the training process. It is usually faster to include the threshold term as an extra node instead of subtracting it every time as seen in Equation (5.4). The state of this extra node is chosen as 1 for convenience.

The weights connecting the input nodes to the hidden layer nodes are named w_{ij} and interpreted as *weights connecting the input node x_i to the hidden layer node h_j*. The weights connecting the hidden layer nodes to the output nodes are named

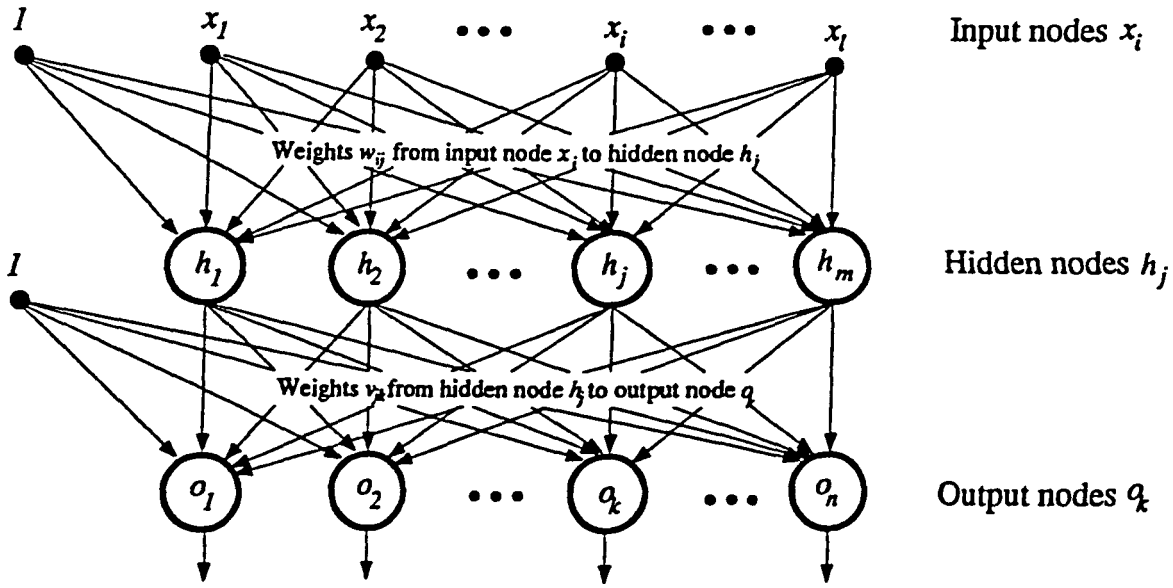

Figure 5.7: One hidden layer MLP architecture, from [24]

v_{jk} and interpreted as *weights connecting hidden layer node h_j to the output node o_k.*

Again, for convenience, Equation (5.4) is re-evaluated in terms of x_i, h_j, o_k, w_{ij}, and v_{jk} in Equation (5.6) and Equation (5.7). Since the network architecture given in Figure 5.7 is a two layer network (one hidden layer and one output layer network), Equation (5.4) needs to be evaluated for both hidden layer nodes, h_j, and output nodes, o_k.

$$h_j = f\left(w_{0j} + \sum_{i=1}^{l} w_{ij}.x_i\right) \qquad j = 1, 2, ..., m \tag{5.6}$$

$$o_k = f\left(v_{0k} + \sum_{j=1}^{m} v_{jk}.h_j\right) \qquad k = 1, 2, ..., n \tag{5.7}$$

The activation function, f, used in the above equations is the sigmoid function as defined by Equation (5.3). The difference between Equation (5.4) and the above equations should be noted: the threshold value, θ, is replaced by w_{0j} and v_{0k}. Also note that the index value of the weights, w_{0j} and v_{0k} is zero. Recall that $i = 1, 2, ..., l$ and $j = 1, 2, ..., m$. In other words, there is no input or hidden layer node with index number 0. The index 0 is used to characterize the extra node added, to replace the threshold value, θ.

The correct classes will be denoted with a n-dimensional binary valued vector. A binary valued vector is a vector whose elements are 0 or 1. For example, if there are 5 classes there will be 5 output nodes, and if the correct class for a signal is, for example 3, then the correct class vector for that input will be (0 0 1 0 0).

Since the activation function is a sigmoid, rather than a hard limiter, the actual outputs computed by the net will converge towards 0 for incorrect classes, and towards 1 for the correct classes, but they will not be exactly 0 or 1 (they will be close to these numbers). Therefore, a thresholding is used at the output. Usually, if the value of the output node is above 0.95, it is accepted as 1, and if it is below 0.05 it is accepted as 0. This thresholding should not be confused with the thresholding used in Equation (5.4), θ.

A correct classes vector, $\mathbf{c}$, is required for every input vector, $\mathbf{p}$, since backpropagation is a supervised learning rule. Assume there are s input patterns in which case $\{(\mathbf{p}^t, \mathbf{c}^t)\}_{t=1}^{s}$, s input-output pairs are needed. The training process starts with feeding the input nodes with the first input signal (vector), $\mathbf{p^1}$. Note that subscripts

are used to show the index number of an element of a vector, whereas superscripts are used to show index to the vector. In other words, $\mathbf{p^1}$ is the first input signal, and p_1^1 is the first element of the first input vector. The node values of the hidden layers are then computed by Equation (5.6). These values are used to compute the node values (states) of the output nodes by Equation (5.7). The output vector is then subtracted from the correct classes vector (corresponding to the current input vector) to compute the error *at the output level* with Equation (5.8) given in the algorithm below. This error *at the output layer* is *backpropagated* to the hidden layer and used in the computation of the error *at the hidden layer* with Equation (5.9).

Once the errors at both levels are computed, the weights are updated according to the errors at each level. The weights at the second layer, v_{jk}, are updated by using the current weight values, hidden node values and the error at the second layer. Then the first layer weights, w_{ij}, are updated by using the current weight values, input node values (input signal), and the error at the first layer.

Once the weights are updated, the next input is fed to the net, and the same procedure is repeated until all the inputs are passed through the net. It takes many passes of the input data for the net to converge. The algorithm is summarized in the steps below.

5.5.2 Backpropagation Algorithm

1. The weights are initialized randomly to some values. Typically initial values are randomly chosen from the interval [-0.1,0.1].

2. The input vector $\mathbf{p^t}=(p_1^t, p_2^t, ..., p_l^t)$ from the training pair $\{(\mathbf{p}^t, \mathbf{c}^t)\}_{t=1}^s$ is fed to the net and the node values (node states) are computed first for the hidden layer

nodes and then for the output nodes from Equations (5.6) and (5.7), respectively. This is the forward pass of the input signal from the network.

3. The error at the output layer, δ_{o_k}, for $k = 1, 2, ..., n$, is computed from Equation (5.8):

$$\delta_{o_k} = o_k(1 - o_k)(c_k^t - o_k) \tag{5.8}$$

Recall that $\mathbf{c^t} = (c_1^t, c_2^t, ..., c_n^t)$ is the correct class vector for the input vector $\mathbf{p^t}$=$(p_1^t, p_2^t, ..., p_l^t)$, and o_k is the current value of the k^{th} output node.

4. The errors computed at the second layer (output) are *backpropagated* to the first level to compute the error, δ_{h_j}, $j = 1, 2, ..., m$, at the hidden layer nodes:

$$\delta_{h_j} = h_j(1 - h_j) \sum_{k=1}^{n} \delta_{o_k}.v_{jk} \tag{5.9}$$

where h_j, for $j = 1, 2, ..., n$ are the current values of the hidden layer nodes, and δ_{o_k}, for $k = 1, 2, ..., n$, are the errors computed at the output layer by using Equation (5.8).

5. After computing the errors at both layers, the weights are updated. First the weights between the hidden and output nodes, v_{jk}, are computed by

$$v_{jk}(t) = v_{jk}(t - 1) + \eta.\delta_{o_k}.h_j \tag{5.10}$$

where t shows the pattern index (t^{th} input), and $0 < \eta \leq 1$ is the *learning rate*. Note that to update weights at iteration t, the weight values at iteration $(t - 1)$ are used. This is called a *recursion*.

6. Finally the weights between the input and hidden layer nodes are updated:

$$w_{ij}(t) = w_{ij}(t-1) + \eta.\delta_{h_j}.p_i^t \tag{5.11}$$

7. Steps **2** through **6** are repeated for every training pair, $\{(\mathbf{p}^t, \mathbf{c}^t)\}_{t=1}^{s}$. However, the iteration does not end here, and the above steps are repeated starting from the first input vector, $\mathbf{p^1}$, but the weights are not initialized again, and the previous weight values are used to compute the new ones. The learning continues until the net converges, i.e., all the training data vectors are classified correctly. Usually, training is terminated when the errors computed by Equations (5.9) and (5.8) (especially error using (5.8)), are less than a predefined value, or the weight values do not change from one iteration to the other. The convergence proof of the above algorithm can be found in [19].

The backpropagation algorithm searches for the minimum of the *error surface* and, therefore, it is a *gradient descent* method. In this error surface there may be more than one minimum, called *local minima*, and the backpropagation algorithm is expected to find the global minimum of the error surface rather than the local minima. However, the backpropagation algorithm often gets stuck in one of the local minima. The error in the local minimum may or may not be satisfactory. If it is not satisfactory, the algorithm can be repeated with different initial weight values or a different number of layers and/or different number of nodes. This will start the search for the minimum of the error surface from a different point, and the local minimum which caught the algorithm could be skipped.

Although the backpropagation algorithm given above is the most commonly used neural network supervised learning rule, it is very seldomly used in the way it is explained above (plain backpropagation) since it is slow, and the algorithm is often

trapped into local minima. Fortunately, there are ways to improve the performance of the backpropagation, as explained in the next section.

5.5.3 Improving Backpropagation

There are three methods for improving the performance of the backpropagation algorithm [25]. These are including a *momentum term*, choosing initial values more wisely (rather than randomly), and using a *variable learning rate*.

Learning With A Momentum Term: A momentum term is used to reduce the sensitivity of the backpropagation to the small details in the error surface, and thus prevent the network from getting trapped in small local minima. Using a momentum term in the training can be thought of as using a lowpass filter, ignoring the minor changes in the error surface.

The momentum term can be included in the algorithm by replacing Equation (5.11) with the Equation (5.12):

$$w_{ij}(t) = w_{ij}(t-1) + \eta.\delta_{h_j}.p_i^t + \alpha(w_{ij}(t-1) - w_{ij}(t-2)) \qquad (5.12)$$

The momentum term, α, can be chosen from the interval [0,1]. If $\alpha = 0$ the weight change is based only on the gradient, and it is the same as the plain backpropagation. When $\alpha = 1$, the weight change is basically mainly set to the last change in the weights, and the gradient is mostly ignored. The typical value for the momentum term is 0.95.

Learning With A Smart Choice of Initial Weights: As cited in [25], Nguyen and Widrow showed that the backpropagation algorithm will converge faster

if the initial weights are not chosen randomly but according to the input vector length, number of layers and number of nodes in each layer. Random choice of initial weights may start the learning far from the local/global minimum. By choosing the initial weights wisely, the learning can be started from a point closer to the error surface minimum and, therefore, the learning time may be reduced [25].

Learning With an Adaptive Learning Rate: As mentioned earlier, there are no criteria for choosing the learning rate, η. Choosing η too small will increase learning time considerably, whereas choosing it too large may cause the net to *jump over*, or miss, the error surface minimum.

An alternative is using an adaptive learning rate. In this method, the learning rate is constantly changed according to the change in the error from one iteration to the next. If the new error is more than a predefined ratio (typically 1.04) times the previous error, the new weights, output values, and errors are discarded, and the learning rate is decreased by a predefined factor (typically 0.7). The learning then continues with the previous weights, error values, and node values. If the new error is less than the predefined ratio times the previous error, the learning rate is increased, again by a predefined factor (usually 1.2) [25].

As a summary, backpropagation with multilayer perceptron can be used for pattern recognition, function approximation, and pattern classification. However, optimum network structure is not predefined for a specific problem. Usually a two layer network (one hidden layer) can create any decision boundary provided that it has enough nodes. Different results can be obtained from one learning session to the other, but generally the performance of the backpropagation algorithm can be

improved by using the momentum term, adaptive learning rate, and carefully chosen initial weight values.

Frequently, this powerful algorithm cannot be used because the correct classes of patterns may not be available. In this case a supervised learning can not be performed. However, unsupervised learning rules can be used for these types of patterns. One major difference between supervised and unsupervised learning is that supervised learning can be used for classification, whereas unsupervised learning can be used for clustering. In the present study on evoked potentials, the correct classes were not available until the final stages of the study, and therefore, an unsupervised clustering algorithm was first used to cluster the signals. The next section will describe one of the most commonly used clustering algorithms which was also used in this study.

5.6 Unsupervised Learning and K-means Algorithm

Unsupervised learning algorithms are used in pattern recognition applications when training data of known classes are not available. A conventional clustering algorithm, namely the k-means clustering algorithm, was used in this study.

One of the most popular network architectures for unsupervised learning is the Kohonen net and Kohonen learning rule [26], [27]. The Kohonen net forms a *self organizing feature map* where each input vector is mapped into a cluster center. The main idea in self organizing feature maps and clustering algorithms is to transform the data from a high dimensional pattern into a lower dimensional feature space. The Kohonen network architecture and the learning rule are discussed in detail in [19] and [27]. The k-means algorithm is a special case of the Kohonen network [19].

Every N samples long signal is considered as a point in N-dimensional space or, equivalently, as an N dimensional vector. In unsupervised learning, output data is arranged in clusters based on a similarity measure. The cluster is characterized by its *cluster center* which is also an N dimensional vector. Originally, the clusters that represent the data are not known, but the number of clusters is usually known.

There are many similarity measures, such as Euclidean distance, Mahalanobis distance, and angular measure of similarity [28]. Euclidean distance is used most often. The Euclidean distance between two vectors $\mathbf{A} = \{a_1, a_2, ..., a_N\}$ and $\mathbf{B} = \{b_1, b_2, ..., b_N\}$ is defined as:

$$\begin{aligned} D(\mathbf{A}, \mathbf{B}) &= \| \mathbf{A} - \mathbf{B} \| \\ &= \sqrt{\sum_{k=1}^{N} (a_k - b_k)^2} \end{aligned} \tag{5.13}$$

and Mahalanobis distance between vectors $\mathbf{A} = \{a_1, a_2, ..., a_N\}$ and $\mathbf{m} = \{m_1, m_2, ..., m_N\}$ is defined as:

$$D = (\mathbf{A} - \mathbf{m})' \mathbf{C}^{-1} (\mathbf{A} - \mathbf{m}) \tag{5.14}$$

where $\mathbf{A}$ is a pattern vector, $\mathbf{m}$ is the mean of the pattern vector $\mathbf{A}$, and $\mathbf{C}$ is the covariance matrix of a pattern population.

A measure of similarity itself is not sufficient to describe a clustering algorithm. A clustering criterion is also required. This can be a heuristic scheme or based on the minimization (or maximization) of a performance index. The performance index (as used by the k-means algorithm) is the *sum of squared errors*, defined as [28]

$$J = \sum_{j=1}^{N_c} \sum_{\mathbf{x} \in S_j} \| \mathbf{x} - \mathbf{m}_j \|^2 \tag{5.15}$$

where J is the performance index, N_c is the number of clusters, $\mathbf{x}$ is the input pattern (a vector), $\mathbf{m}_j$ is the sample mean (a vector) of S_j (the set of input patterns clustered into the cluster j, $j = 1, 2, ..., N_c$ at the current iteration),

$$\mathbf{m}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j} \mathbf{x} \tag{5.16}$$

where N_j is the number of input patterns clustered into the cluster j at the current iteration. The index J is the sum of squared errors between the patterns in a cluster and their corresponding mean. The following is the step by step k-means algorithm (adopted from [28]).

5.6.1 K-Means Clustering Algorithm

The k-means algorithm clusters signals according to the distance between the signals and corresponding cluster centers to minimize the performance index (sum of squared error between the patterns in a class and the cluster centers). Initially, N_c number of cluster centers are randomly chosen. Usually, the first N_c signals are taken as the initial cluster centers. Then, the Euclidean distance between each signal and each cluster center is calculated. Every signal is clustered into the nearest cluster whose cluster center was chosen as the first N_c signals. This separates the data into N_c initial classes, $S_j(t)$, for $j = 1, 2, ..., N_c$, where $S_j(t)$ represents the set of signals clustered into the j^{th} class, and t is the iteration index.

Then new cluster centers are computed at the $(t+1)^{th}$ iteration such that the performance index is minimized. The cluster center that minimizes this error is the

mean of $S_j(t)$, $\mathbf{m}_j$. The number of patterns clustered into a specific cluster changes throughout the process but stays constant when the cluster centers remain unchanged from one iteration to the next. The iteration terminates when the cluster centers remain unchanged. The k-means algorithm is guaranteed to converge as shown in [29], [30].

1. N_c initial cluster centers, $\mathbf{z}_1(1)$,$\mathbf{z}_2(1)$,...,$\mathbf{z}_3(1)$, are chosen as the first N_c patterns from the input set.

2. Distribute the signals into N_c number of classes according to the Euclidean distance between signals and cluster centers.

$$\mathbf{x} \in S_j(t) \quad \text{if} \quad \| \mathbf{x} - \mathbf{z}_j(t) \| < \| \mathbf{x} - \mathbf{z}_i(t) \| \tag{5.17}$$

for all $i, j = 1, 2, ..., N_c$, $i \neq j$. $\mathbf{x}$ is the input pattern and $\mathbf{z}_j(t)$ is the j^{th} cluster center at the t^{th} iteration.

3. The new cluster centers, $\mathbf{z}_j(t+1)$, for all the $j = 1, 2, ..., N_c$ classes are computed such that the squared distances from all of the points in signals in $S_j(t)$ to the new cluster center are minimized. This corresponds to the new cluster center which minimizes the performance index given in Equation (5.15). It can be shown that the new cluster center, $\mathbf{z}_j(t+1)$, that will minimize J_j is the mean of $S_j(t)$. Therefore, the new cluster center is

$$\mathbf{z}_j(t+1) = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j} \mathbf{x} \qquad j = 1, 2, ..., N_c \tag{5.18}$$

where N_j is, as in Equation (5.16), the number of patterns in cluster j at the current iteration t.

4. The iteration is continued from step **2** until the cluster center at $t+1$ is equal to the cluster center at t, i.e., $\mathbf{z}_j(t+1) = \mathbf{z}_j(t)$.

The performance of the k-means clustering algorithm depends on the initial choice of the cluster centers, number of cluster centers, and the order in which the input patterns are presented. The k-means algorithm has been used for over 30 years because of its simplicity and generally good performance.

CHAPTER 6. ELECTROENCEPHALOGRAPHY, EVOKED POTENTIALS AND ALZHEIMER'S DISEASE

...feeble currents of varying direction pass through the multiplier when the electrodes are placed on two points of the external surface, or one electrode on the grey matter, and one on the surface of the skull.

Caton, 1877

6.1 Electroencephalography

The above sentence originally appeared in the *British Medical Journal* in 1877, after Caton's experiments on more than 40 rabbits, cats, and monkeys, and it is regarded as an indication of the birth of the electroencephalography, the study of the electrical activity of the brain. The human electroencephalogram was discovered by Hans Berger in early the 1920's. However, the major increase in interest in the electroencephalogram (EEG) signals have occured in the last 30 years. It parallels the increase in interest in signal processing [31].

EEG signals have been used in many clinical applications to diagnose neurological disorders. In this study, EEG signals were used to detect a well-known dementia, *Alzheimer's disease*. This chapter will give basic background information on how EEG signals have been used in diagnosing dementias. Readers interested in other

aspects of electroencephalography are referred to [31].

In recording the EEG signals, *the international 10-20 system* is used for electrode placement. Ten pair of electrodes are placed on the cranium and named according to their positions with a letter and a number. The letter, usually P, F, C, T, or O represents an area of the brain, namely, *parietal*, *frontal*, *central*, *temporal*, and *occipital*, respectively. The number represents the relative position of the electrode in that area. Odd numbered electrodes are placed on the left, and even numbered electrodes are placed on the right. The midline electrodes are numbered zero, and the numbers increase from left to right for even numbered electrodes, and from right to left for odd numbered electrodes. The standard international 10-20 system is illustrated in Figure 6.1 .

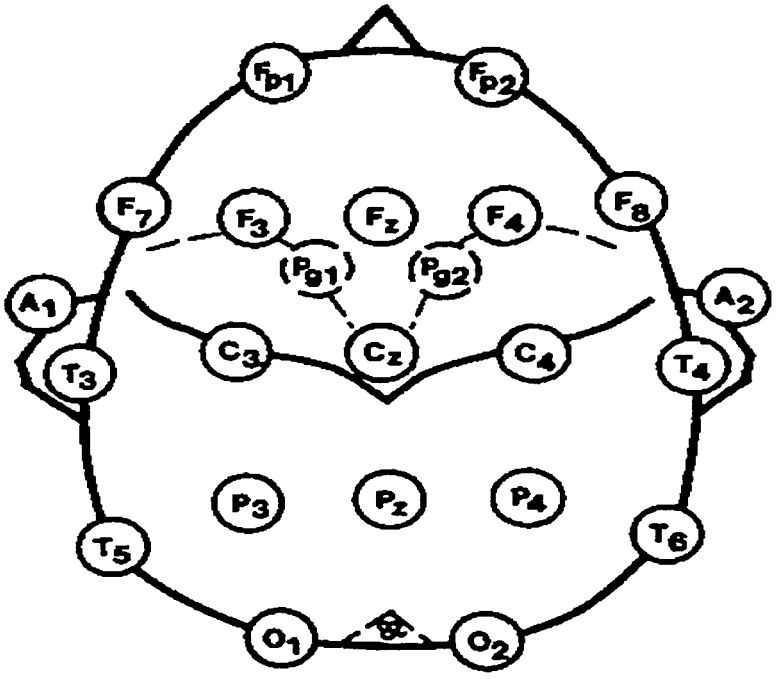

Figure 6.1: International 10-20 electrode placement system

6.1.1 The Evoked Potentials

Evoked potentials is a generic term given to that part of the EEG which occurs in response to a certain stimulus. The stimulus can be auditory, visual, or electrical. For example, if one hears a short beep, an evoked potential will be generated in the EEG in response to that beep.

Evoked potentials are usually separated into two categories as *stimulus-related* and *event-related (ERP)* potentials. The stimulus related evoked potentials are generated when an external stimulus (e.g., auditory or visual) is perceived by the individual, and event-related evoked potentials are generated when the individual deliberately responds to a stimulus. For example, pressing the fire button on a video game when an enemy plane appears on the screen generates an event-related potential.

Event-related potentials have been used for several years in diagnosing dementia, a condition of deteriorated mentality. *Dementia* is a syndrome that consists of a decline in cognitive and intellectual abilities occuring in an awake and alert patient. The decline is severe enough to interfere significantly with work, usual social activities, or relationships with others (American Psychiatric Association, 1987, as cited in [31]). Dementia is a general name for a group of diseases which affect the patient's ability to remember, think, and make judgments. The dementias include Alzheimer's disease, Pick's disease, Parkinson's disease, Huntington's disease, and number of others. Dementias are seen frequently in elderly people.

The effect of dementia on the EEG was first studied by Goodin et al. [32] in 1978. They demonstrated that the dementias have a distinct effect on the *P300* component of the event-related potential.

6.2 The P300 Component of the Event-Related Potentials

One of the most widely and popularly known components of event-related potentials is a positive peak occuring at a latency of approximately 300 ms to 500 ms. The *P*ositive peak at a latency of *300 ms* is named the *P300* or the *P3* component of the ERP. The P300 component is generated when an individual is asked to discriminate a less frequently occuring stimulus from a more frequently occuring stimulus in a series of stimuli involving both. The P300 is, therefore, considered a *cognitive* ERP. The *oddball paradigm* is a simple experiment that is used to record the P300 component.

In the oddball paradigm (as explained in detail later in this chapter), the subjects are asked to perform a simple task, such as tapping their right index finger, pushing a button, or counting every time they hear a high tone (2000 Hz) in a series of beeps with high and low tones (1000 Hz).

Many studies have showed that this experiment can be used to distinguish patients with dementias from patients with other neurological diseases [32], [34]. Many other studies considered the effects of different aspects of this experiment on the P300 component [33], [35], [36], [37], [38], [39]. These aspects included the intensity of the stimuli, the interval between the stimuli, and the probability of occurrence of the two sets of stimuli.

Studies have shown that the latency and the amplitude of the P300 component are subject to change due to age and the existence of dementia. As cited in [31], Fabiani et al. [40], Noldy et al. [41], Paller et al. and [42] showed that the P300 amplitude increases with better memory performance. It has also been shown that there is a close relationship between the information provided by the stimulus and the

amplitude of the P300. This is related to how often the target stimulus occurs. As discussed later in this chapter, if the frequency of occurrence of the target stimulus decreases, the amplitude of the P300 increases. This is related to a well known rule of the probability theory which states that a less likely event has more information.

Another parameter that affects the amplitude is the *interstimulus interval.* Generally speaking, the amplitude of the P300 component increases when the interval between the stimuli increases [39]. These findings suggest that whenever attentional resources are required to recognize a stimulus that is different from other stimuli, an ERP (*P300*) occurs at a latency of 300 ms to 500 ms.

Another important parameter of the P300 is the exact latency of the peak, i.e., how long after the stimulus is given does the response occurs. Generally speaking, shorter P300 latencies indicate better mental performance. Therefore, the P300 latency is related both to age and existence of a dementia. This relationship is shown in Figure 6.2, where the age versus P300 latency is given for normal subjects, patients with neurological diseases, both dementias and non-dementias, and psychiatric patients.

It should be noted at this time that although a strong relationship has been found between mental performance and the P300 component, large variations in latency and amplitude have been reported for different experiments. Therefore, a generalization regarding the relationship between the amplitude and latency of the P300 and the mental performance of an individual can not be made. This variation in the amplitude and latency is interpreted [31] as: "Variation in the P300 amplitude can be considered as a consequence of *attentional resource allocation*, and variations in latency can be considered as a consequence of the speed with which those resources

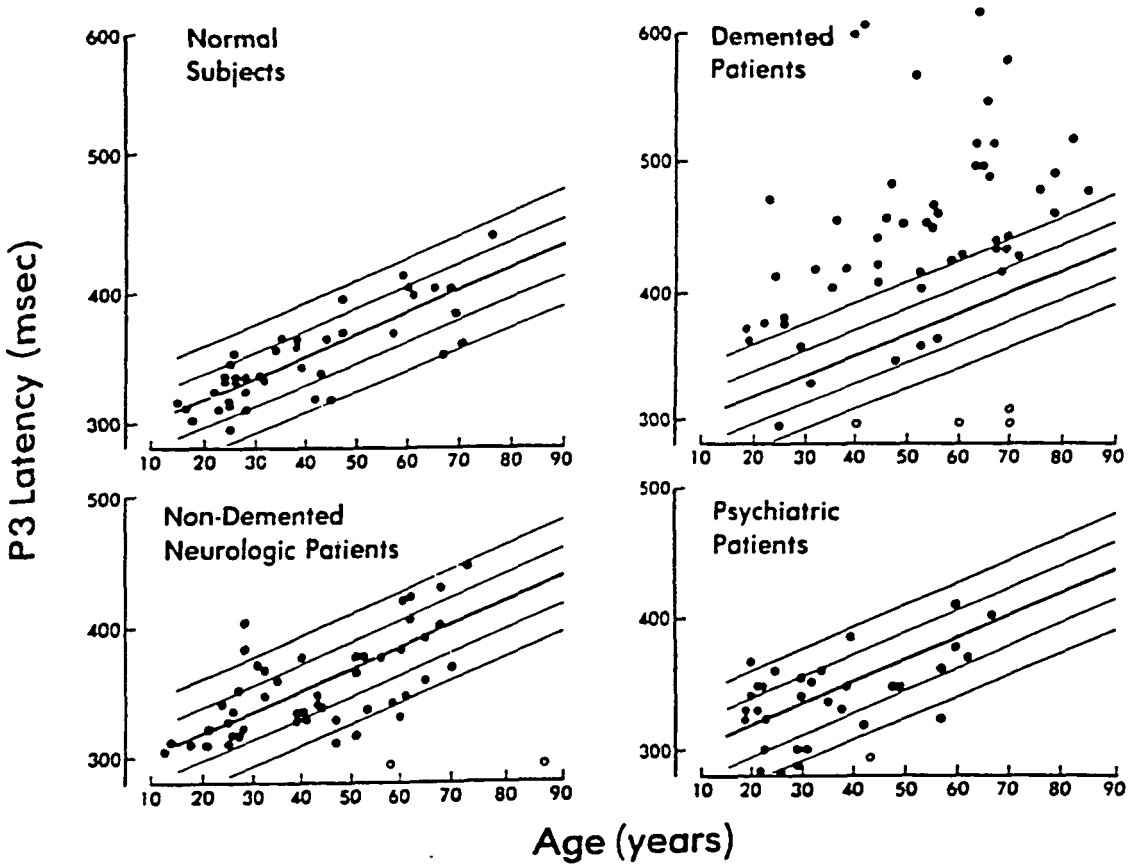

Figure 6.2: Relationship between the age and the P300 latency for different groups of patients (from [43] as cited in [31]).

can be allocated when immediate memory is updated."

It should also be noted that the variations in the P300 components of normal people are large enough to prevent generalizing the relationship between the P300 and mental abilities. Furthermore, there are a number of variables that affect an individual's P300 measurements. These include body temperature, age, recency of food intake, season of the year, personality of the individual, etc. Therefore, the subjects chosen for the experiment should have similar internal (age, food intake, body temperature) and external (season of the year, time of the day) conditions.

The specific effects of these variables can be found in [31].

6.3 The Oddball Paradigm

The most prominent experiment for detection of the P300 component of the ERP is the oddball paradigm experiment. The subject is asked to respond to a target tone of 2000 Hz, *the oddball tone*, which occurs relatively infrequently compared to the non-target tone of 1000 Hz, *the regular tone*. Although auditory stimuli are usually used to generate the P300, other stimuli can be used. Auditory stimuli have been found to be advantageous since they are easy to produce and do not cause an electrooculogram (EOG) artifact.

Four major peaks are observed in the ERP: N1, P2, N2 and P3. N1 is a negative peak that occurs approximately 100 ms after the stimulus. P2 is a positive peak that occurs at a latency of approximately 200 ms. N2 is a negative peak that occurs at a latency of approximately 200 ms, and finally, P3 is the positive peak that occurs at a latency of approximately 300 ms. Typical ERPs generated by the oddball tone and by the regular tone are plotted in Figure 6.3. Note that N1, P1, and N2 are present in both ERPs, but P300 is only present in response to the oddball tone. The axis below the ERPs shows the stimulus type; S being the standard (regular) tone and T being the target (oddball) tone.

As mentioned earlier, there are a number of parameters that affect the ERPs formed by the oddball paradigm experiment. The frequency of the tones, frequency of occurrence of the tones, duration and intensity of the tones, and interstimulus interval are the parameters related to the stimulus. There are also other factors such as the electroencephalograph used for recordings, the task that the subject was told

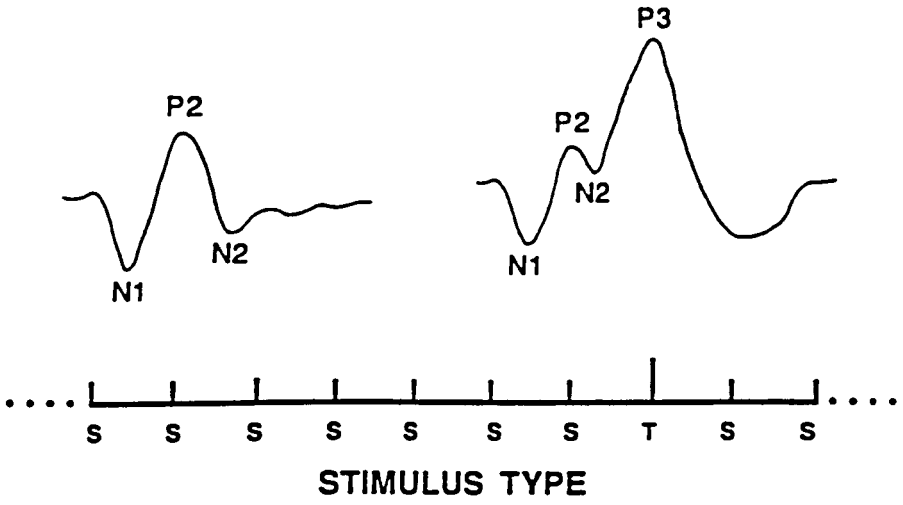

Figure 6.3: ERPs in respond to the standard(regular) and target (oddball) tones from [31]

to perform, etc. There is not a set of parameters that are internationally agreed on, but the parameters given in table 6.1 are commonly used [31].

The specifications given in Table 6.1 have been chosen to give the most robust P300 components. The tone frequencies are chosen as 1 and 2 kHz since these are frequencies generated by human speech and they are easy to perceive. The probabilities of 20% target tone and 80% non-target tone are chosen to generate a high amplitude P300. The intensity, duration, and rise/fall times were also determined to be the optimum values after many experiments.

It is recommended that the subjects sit in a comfortable position so they can concentrate on the task. The eyes should be closed to prevent the artifacts of the EOG. The task can be any of number of tasks, but finger tap/button press have

Table 6.1: Typical parameters for the oddball experiment

Parameter	Typical Value
Stimulus factors	
Tone frequency	2000 Hz. at 20% (target) 1000 Hz. at 80% (standard)
Rise/fall	10 ms.
Duration	50 ms.
Intensity	60 dB
Interstimulus interval	2 s.
Subject and task	
Position	Seated
Eyes	Closed
Task	Finger tap/button press
Electrophysiological recording	
Electrodes	Fz, Cz, Pz, EOG
Reference	A1/A2
Ground	Forehead
Bandpass	0.01-0.5 to 30 Hz.
Epoch Length	750 ms.
Artificial rejection	$\pm 100\ \mu$ V
Target trials in average	20 (or more)
Replications	2 Blocks of 20 trials

been found to be most efficient. In the original experiment in 1978, Goodin asked his patients to count the number of the target tones [32], but today this is not recommended because patients with dementia may get confused and cannot perform the task.

The Fz, Cz, and Pz electrodes are usually used to capture the P300, and all of them should be used for correct identification. This is especially important for elderly people for whom the amplitude of the peak may not be adequate to be noticed from one electrode only. Monopolar active electrodes with reference to A1/A2 (earlobes or mastoids) are recommended.

One of the key parameters in capturing the P300 components is the bandpass filter used. The P300 has relatively low frequency components, around 3 Hz. Normal EEG signals can have frequency components from 0.01 Hz to 100 Hz, and therefore, the bandpass filter should pass at least the 0.01 Hz-30 Hz band.

The recommended recording time (epoch) per session (per beep) is 750 ms since this should be long enough to capture any P300 latency.

Artifacts constitute a major problem in EEG recordings. They should be removed prior to processing the data. For all practical purposes, any signal with an amplitude over ± 100 microvolts is assumed to be an artifact, and therefore, it should be automatically rejected.

The P300 component of the ERPs can not be seen without averaging. Therefore, at least 20 artifact-free trials should be recorded and averaged for a stable P300 component.

Under the above conditions, the P300 should not be very difficult to identify since it is the largest component of the ERPs. However, since the amplitude of the

peaks decreases considerably for elderly people, it can be difficult to identify. For the three main electrode combinations given, the amplitude of the P300 tends to increase from the front to the back of the scalp; therefore, the P300 with the largest amplitude is usually the one captured by the Pz electrode (Figure 6.4).

In summary, the largest positive going peak occuring after the N1, P1, and N2 from the target stimulus, and increasing in amplitude from the frontal to the parietal part of the brain is considered to be the P300 component of the event-related potential [31].

Figure 6.4 shows typical P300 recordings from normal patients and Alzheimer's disease patients for each of the Pz, Cz, and Fz electrode combinations. Note that the plots are grand averages of many patients' P300 components.

The left column of plots shows the responses to the oddball tones, and the right column of plots shows the responses to the regular tone. The solid lines represent the normal patients' responses, and the dashed lines represent the Alzheimer's disease patients' responses. Note that the P300 peak has a considerably lower amplitude and longer latency for Alzheimer's disease patients compared to the normal patients.

Figures 6.5 - 6.8 show some of the signals used in this study. The detailed discussion of the experiment and the results obtained are presented in Chapter 7. The time axis is normalized, and the entire axis corresponds to approximately 1 second. The original signals contained 750 samples, sampled at 600 Hz, and they are truncated to remove the prestimulus artifacts. All the figures correspond to the grand average of a session for that particular patient. Usually 40-50 ERPs are averaged for each patient for every channel. In this study signals acquired from the Pz and Fz electrodes are used. Figures 6.5 to 6.8 are obtained by averaging the signals from the

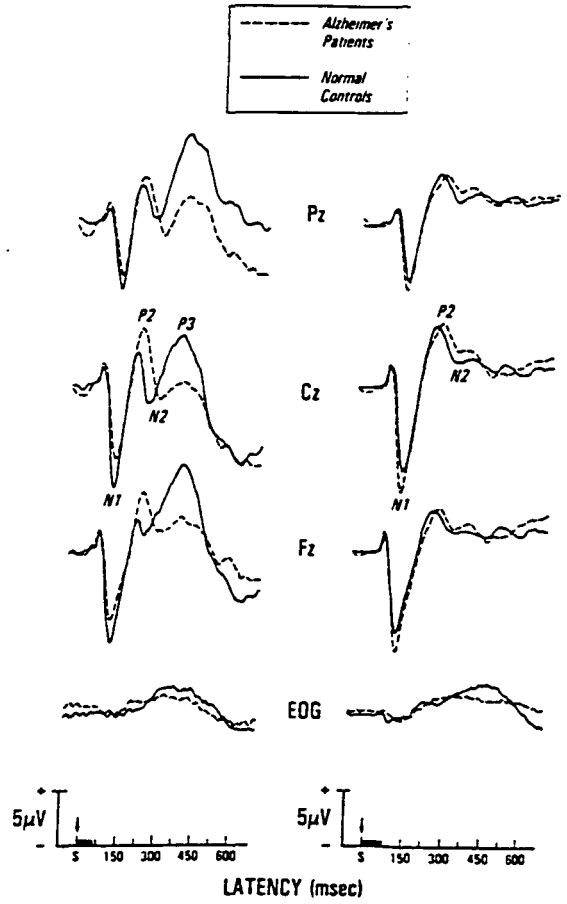

Figure 6.4: Typical P300 responses of normal and Alzheimer's patients to the oddball paradigm experiment (from [44] as cited in [31]).

Pz electrode (since amplitudes are higher in the signals acquired from this electrode) for two patients (one with Alzheimer's disease and one normal).

Figure 6.5 shows the event-related potential recorded from an elderly normal individual. The ERP in this figure is in response to an oddball tone, and the N1, P2, N2, and P3 components can be clearly observed.

Figure 6.6 is an example of an ERP recorded from an Alzheimer's disease patient in response to an oddball tone. All the components, including the P300 (marked as P3), are clearly visible. Also note that the latency of the P300 component is larger

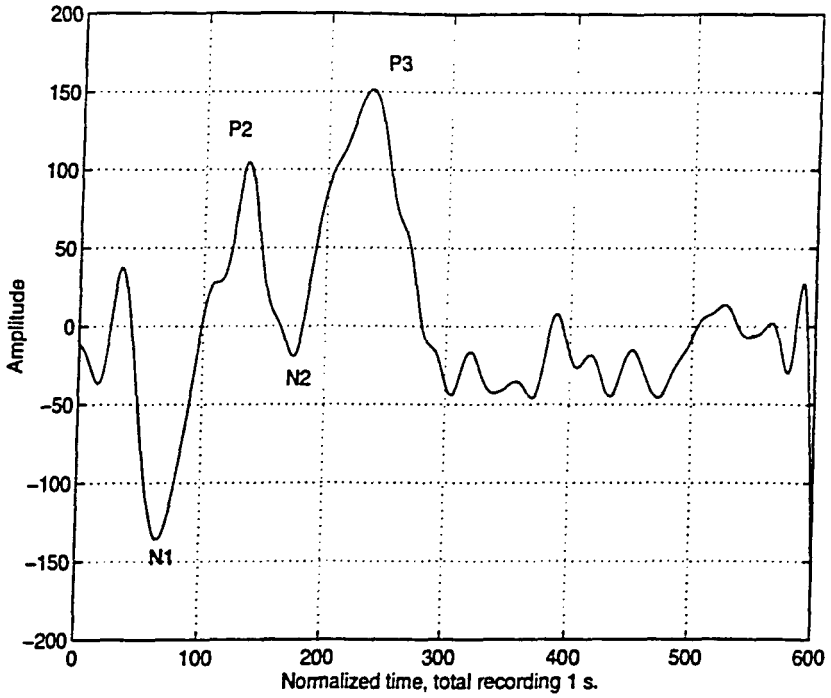

Figure 6.5: An ERP recorded from an elderly normal patient in response to an oddball tone.

for the patient with Alzheimer's disease, compared to the normal patient. However, it should be noted that ERPs are not always as obvious as they are in Figures 6.5 and 6.6. (Compare the complete set of ERPs given in Appendix A.)

Figure 6.7 is an example of an ERP recorded from an elderly normal person in response to a regular tone. As mentioned earlier in this chapter, one of the characteristic properties of the P300 component of the event-related potentials is that it can only be seen in the ERPs recorded in response to a less frequently occuring target stimulus. In the experiment from which these signals were acquired, 80 % of the tones were regular tones; only 20 % of them were target (oddball) tones. As

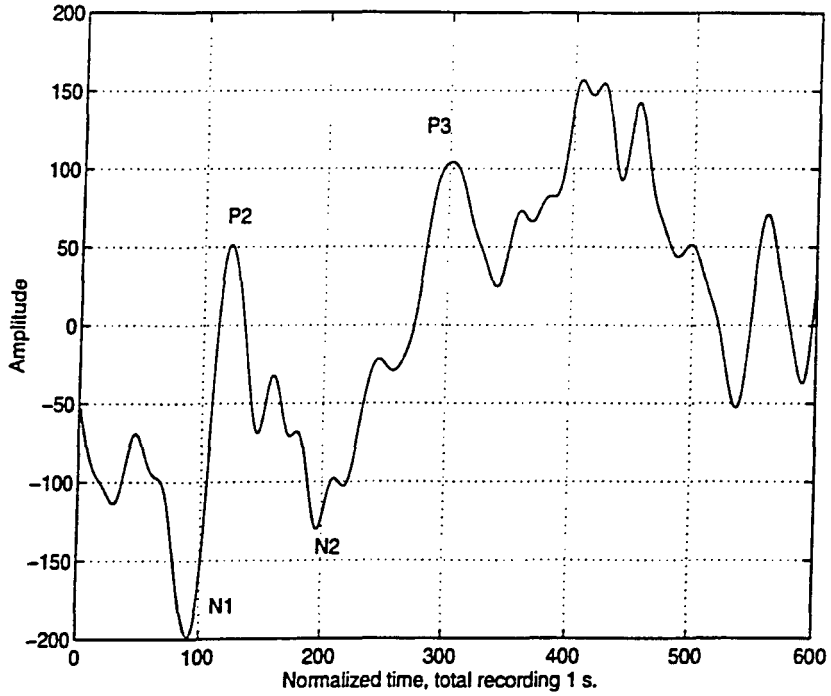

Figure 6.6: An ERP recorded from an Alzheimer's disease patient in response to an oddball tone.

expected, N1, P2, and N2 components are clear, but the P300 component can not be seen.

Finally, Figure 6.8 is the ERP from an Alzheimer's disease patient in response to a regular tone. As in the previous figure, the P300 component can not be seen in the figure, but all the other, N1, P2, and N2, components can be clearly observed.

6.4 Electroencephalography and Dementia

As defined earlier in this chapter, dementias are a group of diseases in which the patients lose their cognitive and intellectual abilities. EEG signals are used

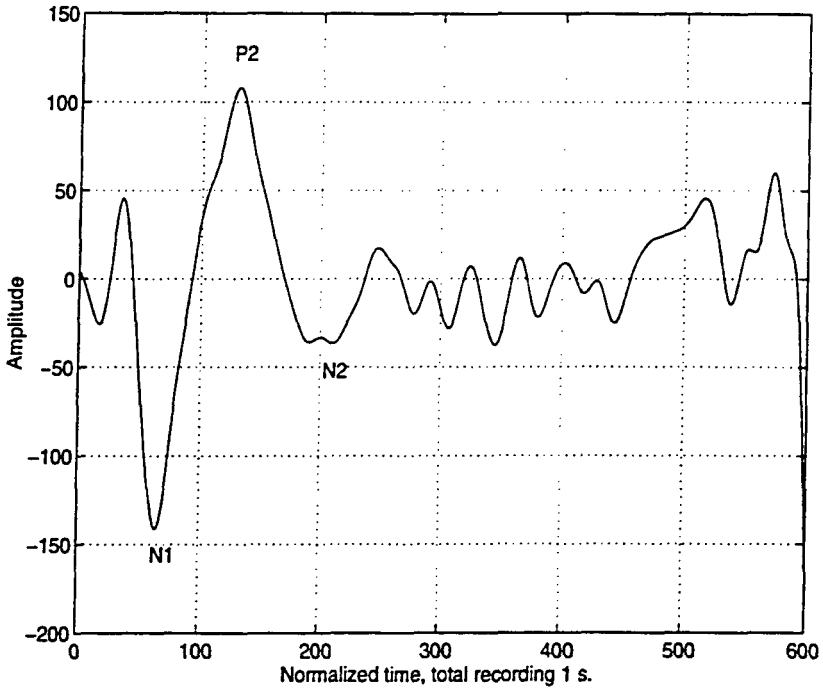

Figure 6.7: An ERP recorded from an elderly normal patient in response to a regular tone.

to distinguish the dementias from other neurological disorders, such as psychiatric abnormalities. Harner, as cited in [31], noticed that the EEG patterns show significant abnormalities for the treatable neurological disorders, whereas they show only slight abnormalities for the non-treatable dementias. Therefore, EEG signals are usually suggested to give a clue about the individual's mental state, rather than diagnosing any abnormalities.

Dementias are generally classified into two main groups, cortical and subcortical dementia. Alzheimer's disease, Pick's disease, Creutzfeldt-Jakob disease are examples of cortical dementia, where as Parkinson's disease and Huntington's disease are

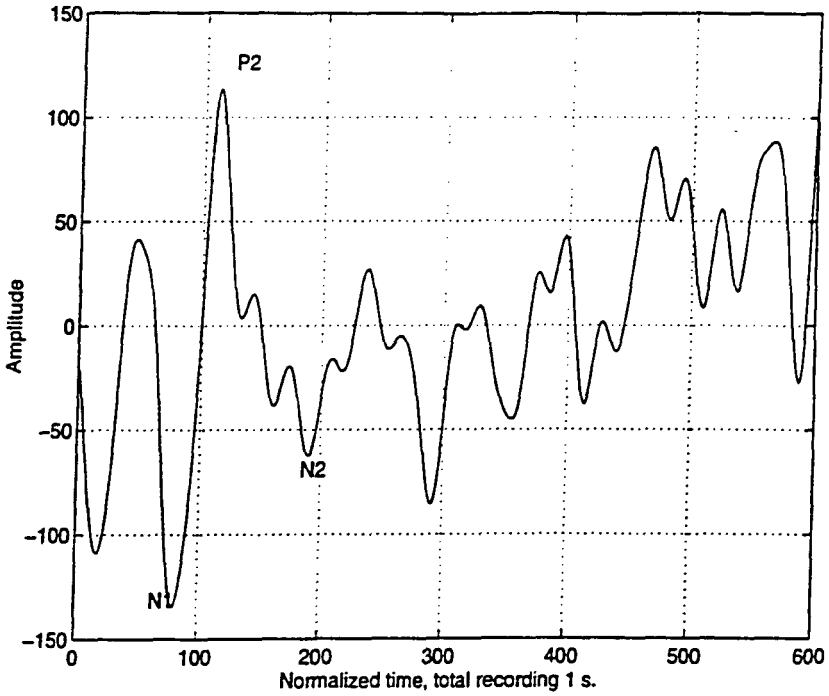

Figure 6.8: An ERP recorded from an Alzheimer's disease patient in response to a regular tone.

examples of subcortical dementia. The following section is devoted to general characteristics of Alzheimer's disease.

6.4.1 Alzheimer's Disease

Alzheimer's disease is the most common cortical dementia. The first symptoms usually appear at 50-60 years of age. Today, there are approximately 2 million Alzheimer's disease patients, and 100,000 Alzheimer's disease patients die every year in the United States [45].

Alzheimer's disease causes a gradual deterioration of the mental abilities, includ-

ing losing memory and losing verbal and reading skills. In its early stages, it causes moodiness, depression, and lack of energy, and, unfortunately, these symptoms are usually ignored. However, as the disease progresses, the patient starts losing first the short term memory, then the mid-term memory, and, eventually, the long-term memory. This makes him/her totally dependent on others. Patients in late stages of the disease forget how to perform basic tasks, forget their home addresses, and eventually forget even their own names.

Patients with Alzheimer's disease lose a significant number of neurons especially from the frontal and temporal lobes, and this loss is suspected to be related to the inadequate production of acetylcholine, a neurotransmitter in the *nucleus basalis* (a cerebral nucleus which plays an uncertain role in memory storage and retrieval) of the *cerebrum.*

Worst of all, Alzheimer's disease can not be 100% diagnosed unless the patient has died because the disease is characterized by the unusually large concentrations of plaques and neurofibrillary tangles in the nucleus basalis and other related portions of the brain. This can only be detected with an autopsy. In addition to plaques and tangles, an unusual protein, called *Alzheimer's disease associated protein (ADAP)*, also appears in the memory related portions of the brain. Fortunately, this protein also appears, in small amounts, in the cerebrospinal fluid of many Alzheimer's disease patients, and therefore, a blood screening test may be available to detect this condition in the near future [45]. Unfortunately, there is currently no cure for Alzheimer's disease.

There has been number of books written on different aspects of Alzheimer's disease, including the recent developments in the research and the social consequences

of the disease. Barry Reisenberg [46] and Ezio Giacoboni with Robert Becker [47] edited a number of papers on the current research in Alzheimer's Disease. Bick et al. [48] edited other papers discussing the early history of Alzheimer's disease. This includes a short preface by Alois Alzheimer. Powell [49] and Light and Lebowitz [50] have recently authored books discussing the social consequences of the Alzheimer's disease.

6.5 Summary

In this chapter some background information on the use of electroencephalography in diagnosing dementias is given. It was shown that the P300 component of the event-related potential from an oddball paradigm experiment can be useful in the diagnosis of cognitive disorders, especially Alzheimer's disease.

The potential to use the EEG for diagnosis have mostly been based on visual analysis of the signals by the neurologists. Recently, researchers have studied the spectral properties of the ERPs of demented patients, and their findings have been consistent with the visual analysis of the neurologists. Many references using spectral analysis are cited [31].

A number of studies showed that the amplitude and the latency of the P300 component are related to the cognitive abilities of the individuals. Therefore, a spectral analysis technique may not be adequate since the information that is being searched for might be hidden in the frequency, amplitude, or latency. Therefore, the feasibility of using a time-frequency analysis technique for diagnosing Alzheimer's disease was proposed. The next chapter will explain the procedure of the experiment, the analysis techniques used for feature extraction, and the classification results.

CHAPTER 7. RESULTS AND DISCUSSION

In the previous chapters, the major issues related to this study were discussed. The ultimate objective of this study was the development of a neural network for the classification of the event-related potentials, preprocessed using a wavelet transform based technique for feature extraction.

This chapter describes the overall implementation procedure including the data acquisition, the preprocessing techniques, the neural network and clustering algorithms used, the problems encountered, and finally, the results of the implementation.

7.1 Obtaining the Data

In response to a request on Internet for normal and abnormal EEGs in an electronic format, the data used in this study were released to us by scientists at the Georgia Institute of Technology Research Institute and Emory University School of Medicine. These data were collected from patients with and patients without Alzheimer's disease for use in the spectral analysis of event-related potentials.

7.2 Data Acquisition

7.2.1 Background and Summary

The goal of the original study was to develop a method for distinguishing subjects with Alzheimer's Disease from those without the disease through the analysis of EEG evoked potential data, neuropsychological measures, or through reaction time data. The primary emphasis was directed toward the analysis of EEG data. The analyses in the study were performed on EEG data obtained from 30 subjects, 15 of whom were known to have Alzheimer's Disease (AD), and the remaining (control) subjects were known to be free from the disease. Four EEG analysis methods, all frequency-based, have been used at Georgia Institute of Technology Research Institute, namely, power spectral density estimation, cross-spectral density estimation, coherence estimation and bispectral analysis. Subject testing and data acquisition and the statistical analysis of the results were performed by researchers at Emory University. Of these 30 records only 28 sets of signals, 14 from Alzheimer's patients and 14 from normal control patients, were used and the other two subjects' data were discarded.

7.2.2 Experimental Setup and Methodology

The EEG data were obtained from auditory oddball tests in which subjects listened to a sequence of tones. The sequence consisted of 200 millisecond beeps repeating at a 1.5 second interval. The *non-target* tones were 1 kHz, and the *target* (oddball) tones were 2 kHz. The subjects reacted by pressing a button when they heard the target tone. Eighty percent of the beeps were regular tones, and the remaining 20% were oddball tones.

EEG recording began 0.25 seconds before each tone occurred and ended one second after the tone started. The sampling rate for the data collection was 600 Hz, or one sample for every 1.667 milliseconds.

For all experiments, data were recorded from the frontal and parietal locations, and, in a few sessions, central data were also captured. After sampling, the data were hand-scored by a neuropsychologist, and the artifactual trials were removed. [51]

7.3 Preprocessing

7.3.1 Removing the Prestimulus

All of the programs used for this study were written for MATLAB, and all the programs were run using MATLAB. The reason for choosing MATLAB was its high efficiency in manipulating large amounts of data in matrix operations. Also many built-in functions were utilized in the programs, especially for the backpropagation algorithm. All source codes are included in appendices.

The data consisted of 56 files, two files for each patient, corresponding to the responses to the oddball tones and responses to the regular tones. Each data file was in matrix format with 8 columns and 32000-45000 rows. (All total row numbers were divisible by 750, suggesting that individual responses to every beep, each consisting of 750 samples, 1250 ms at 600 Hz, were concatenated.) Therefore, each file contained approximately 40 to 64 responses. Each column corresponded to a different lead combination. As suggested in the Chapter 6, two of these electrodes were of particular interest, namely, the Pz and Fz electrodes.

First, these columns are separated from the rest of the matrix for further preprocessing. The first 150 of the 750 samples were removed since this part corresponds

to the prestimulus. The mean of the prestimulus was removed from the rest of the data to remove the *presitimulus artifacts.*

7.3.2 Averaging

Recall the original proposal was to analyze the signal using the continuous wavelet transform and the discrete wavelet transform. The codes for all of these transformations were written in advance and were ready before the data were received. However, the correct classification of the signals was not received with the data. The approach used consisted of obtaining the continuous wavelet transform of the waveform followed by attempting visual pattern recognition. The continuous wavelet transform provides a two dimensional time-frequency (or time-scale) representation of the signals. It was expected that a pattern would be found in 14 of the transforms that was significantly different from that in the other 14.

The continuous wavelet transform is very dense (redundant), and therefore, takes a relatively long time to compute. The removal of the prestimulus reduced the data size by 20%, but it was still too large for processing. The continuous wavelet transform program has the capability of computing the transform at as many frequency (or scale) values as desired. Using trial and error procedures it was decided to compute the transform at around 80 frequency points. This requires the transform to be computed at 80x35000 to 80x45000 points.

The discrete wavelet transform is much faster and can easily handle this amount of data, since it is a one to one transform. However, for an efficient discrete wavelet transform, the signal length needs to be a power of two, or at least a multiple of a power of two. None of the data files had a length equal to a power of two (or a

multiple of it).

It was then decided to average the signals before transforming them. The averaging was performed on all 56 files, reducing every file length to 600 samples, a manageable size for any transformation technique. After the averaging, the P300 components in most of the responses to the oddball tones were apparent. Averaged evoked potentials for every patient are given in Appendix A, where four plots are given for each patient. Two of these plots are the Pz and Fz electrode recordings in response to the oddball tone, and the other two are the same recordings in response to the regular tone.

7.4 Transforming The Data: Feature Extraction

7.4.1 Continuous Wavelet Transform

The continuous wavelet transforms of each data file was then computed at 80 frequency points. This number was decided by trial and error. The transforms for two patients are shown in Figures 7.1 - 7.4. The complete set of continuous wavelet transforms on signals in the database is given in Appendix B.

The time and frequency axes of the plots are normalized. The beginning of the frequency axis marked zero does not necessarily represent the DC value, but it simply represents the *lowest* frequency analyzed in that particular computation. Furthermore, the first five or so frequencies were eliminated from the plots since they did not contain any relevant information. The time axis corresponds to the translation of the wavelet function along the signal. Recall that the preprocessed data had 600 samples. The continuous wavelet transform was computed by shifting the wavelet function by steps of 5 samples ($\tau = 5$) resulting in 120 samples along the

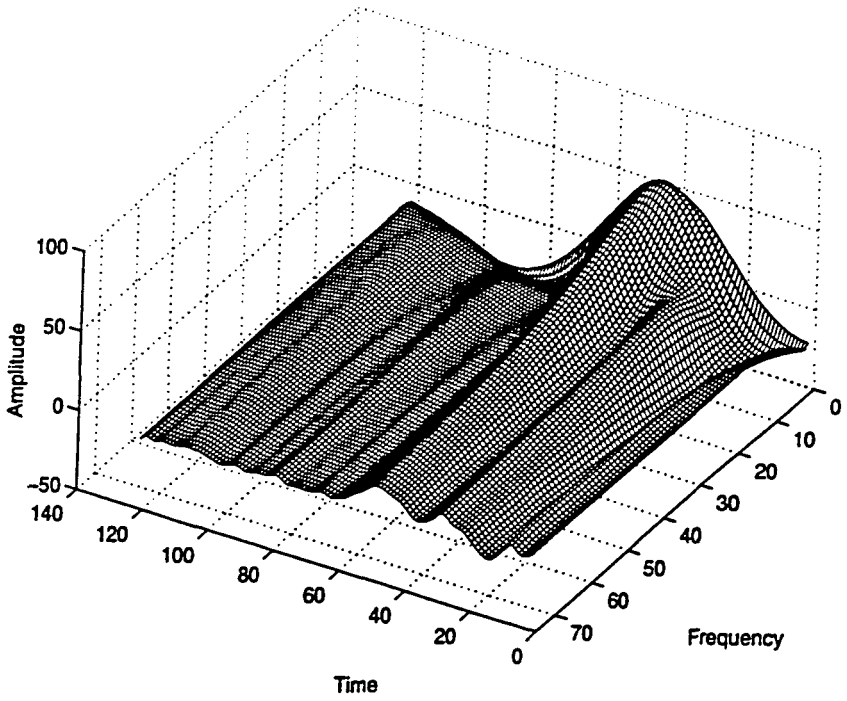

Figure 7.1: CWT of an ERP of an elderly normal person, oddball tone

time axis. A more dense computation was also performed, but it did not provide any additional information. Hence to reduce the computational effort, a value of $\tau = 5$ was used.

Figure 7.1 is the continuous wavelet transform of an ERP of an elderly normal person's response to an oddball tone. The first thing that is observed in the plot is the relatively high peak at low frequencies and at time $t \approx 40$. Also note that no significant information is present at high frequencies.

Figure 7.2 is the continuous wavelet transform of the same patient's response to a regular tone. Note that the peak that occured at low frequencies and time $t \approx 40$ in the previous case did not appear in the response to regular tone. It is also observed from this plot that the major information lies in the lower frequencies.

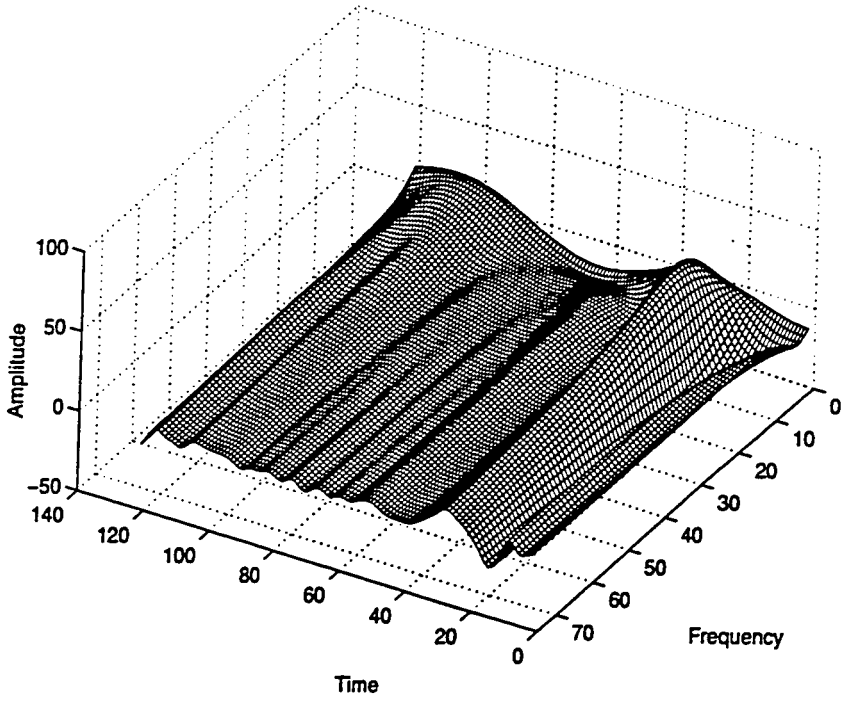

Figure 7.2: CWT of an elderly normal person, regular tone

Figure 7.3 is the continuous wavelet transform of the ERP recorded from an Alzheimer's disease patient in response to an oddball tone. The large peak at lower frequencies can be observed. Note, however, that this peak occurs at a later time, $t \approx 100$, relative to the normal patient. This may be due to the higher latency of the P300 component of the ERP from the Alzheimer's disease patient compared to the P300 component of the ERP from the normal patient.

Finally, Figure 7.4 shows the continuous wavelet transform of the ERP recorded from the same patient with Alzheimer's disease in response to a regular tone. Note that the large peak did not appear in this plot. The relatively large peak that is seen at lower frequencies and at time $t \approx 120$ is not distinguishing information since a similar peak can also be seen in Figure 7.2 which corresponds to the transform of the

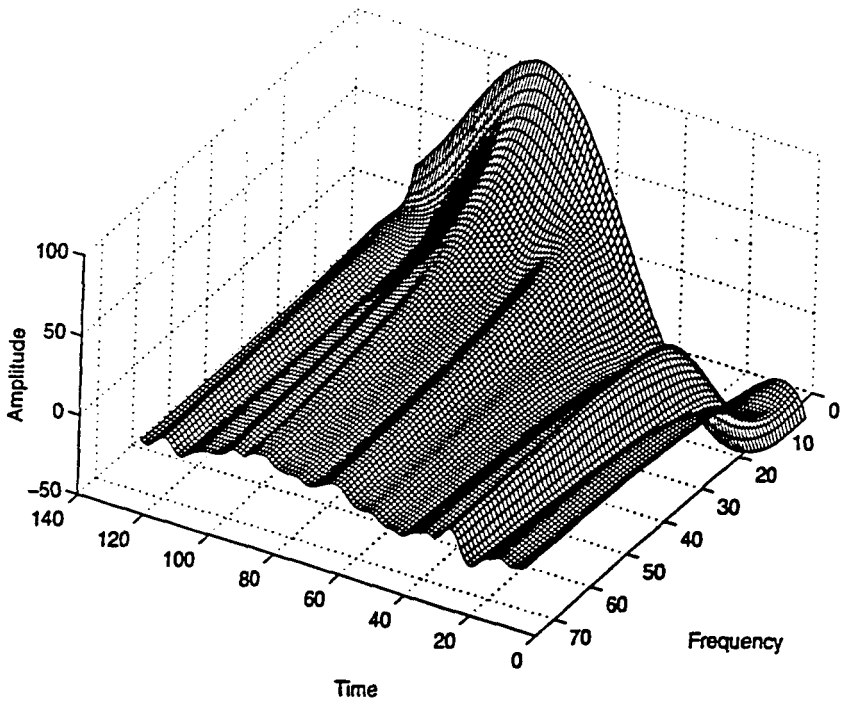

Figure 7.3: CWT of an ERP of an Alzheimer's disease patient, oddball tone

ERP of a normal patient.

From the above observations it might be concluded that the continuous wavelet transform may show a pattern that is significantly different for the ERPs recorded from normal patients and Alzheimer's disease patients. However, not all of the transformed signals showed a similar distinct difference between normal and diseased patients. For complete comparison of the continuous wavelet transformed signals, please refer to Appendix B. Correct classification of the transformed signals by visually analyzing these plots was not possible, particularly when the classification information of the signals was not known.

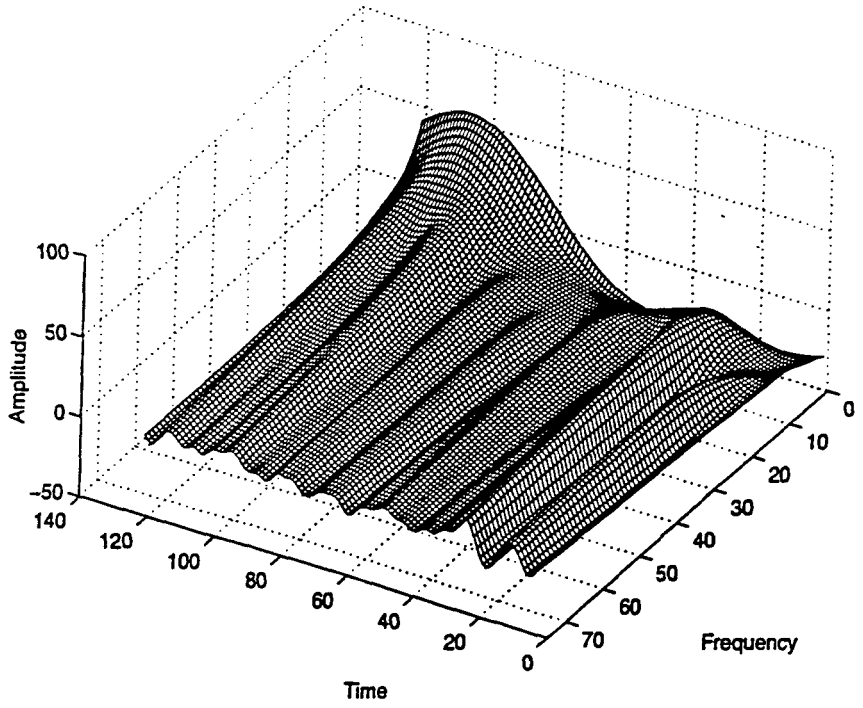

Figure 7.4: CWT of an ERP of an Alzheimer's disease patient, regular tone

7.4.2 Discrete Wavelet Transform

For a computerized classification of the signals by a neural network, training data, for which the correct classes are known, is needed. However, as described in Chapter 5, there are some clustering algorithms which separate the signals into different classes according to a similarity measure. The continuous wavelet transform is not suitable for a neural network classification or clustering since it has redundant information. For neural network classification (or clustering), the data should be as concise as possible. This can be accomplished using a suitable transformation technique to extract the relevant features from the raw data.

Supervised and unsupervised classifications were performed on the discrete wavelet transform data. The discrete wavelet transform was taken at 640 points, after the

signal was augmented by the first 40 samples to increase the signal length to 640, a multiple of a power of two.

The discrete wavelet transform was used in the augmented signal format (see Chapter 4), in which the entire time domain signal at different frequency bands was concatenated. Since the interpretation of the discrete wavelet transform in augmented format is especially difficult a visual pattern recognition cannot be performed. However, this format of the discrete wavelet transformed signal is particularly suitable for use with clustering algorithms and neural networks. Therefore, this signal was input to the k-means clustering algorithm and into a multilayer perceptron (see Chapter 5). Two examples are given in Figures 4.5 and 4.6. Figure 4.5 is the discrete wavelet transforms of the four signals from an Alzheimer's disease patient, and Figure 4.6 shows the transforms of the signals from a normal patient. Note that discriminating information is not readily obvious. The neural network however is capable of finding the discriminating information between these signals. The complete set of discrete wavelet transform signals is given in Appendix C.

7.5 Unsupervised Clustering with K-Means Clustering Algorithm

Since the signal was oversampled at 600 Hz, the data length was reduced to 50 samples by taking the first 50 samples of the discrete wavelet transform. The 50 sample long signals were then clustered by the k-means algorithm into two classes. The clustering procedure was implemented on all four sets of signals per patient. Only the signals from electrode Pz were clustered into two even classes of 14 signals each. At that time the correct classification was still unknown, and therefore, the performance of the clustering algorithm was also unknown.

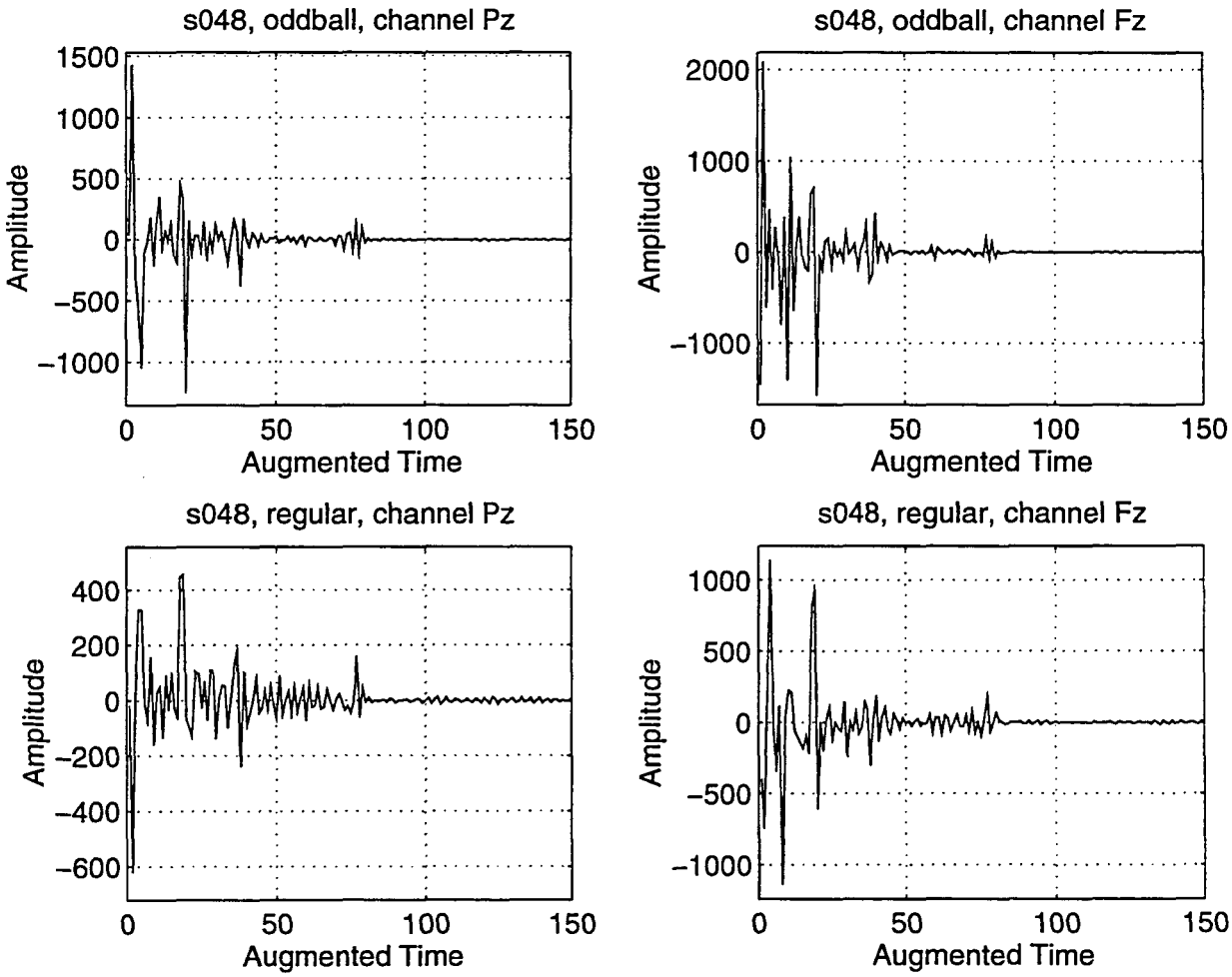

Figure 7.5: DWT of ERPs of an Alzheimer's disease patient, oddball and regular tones

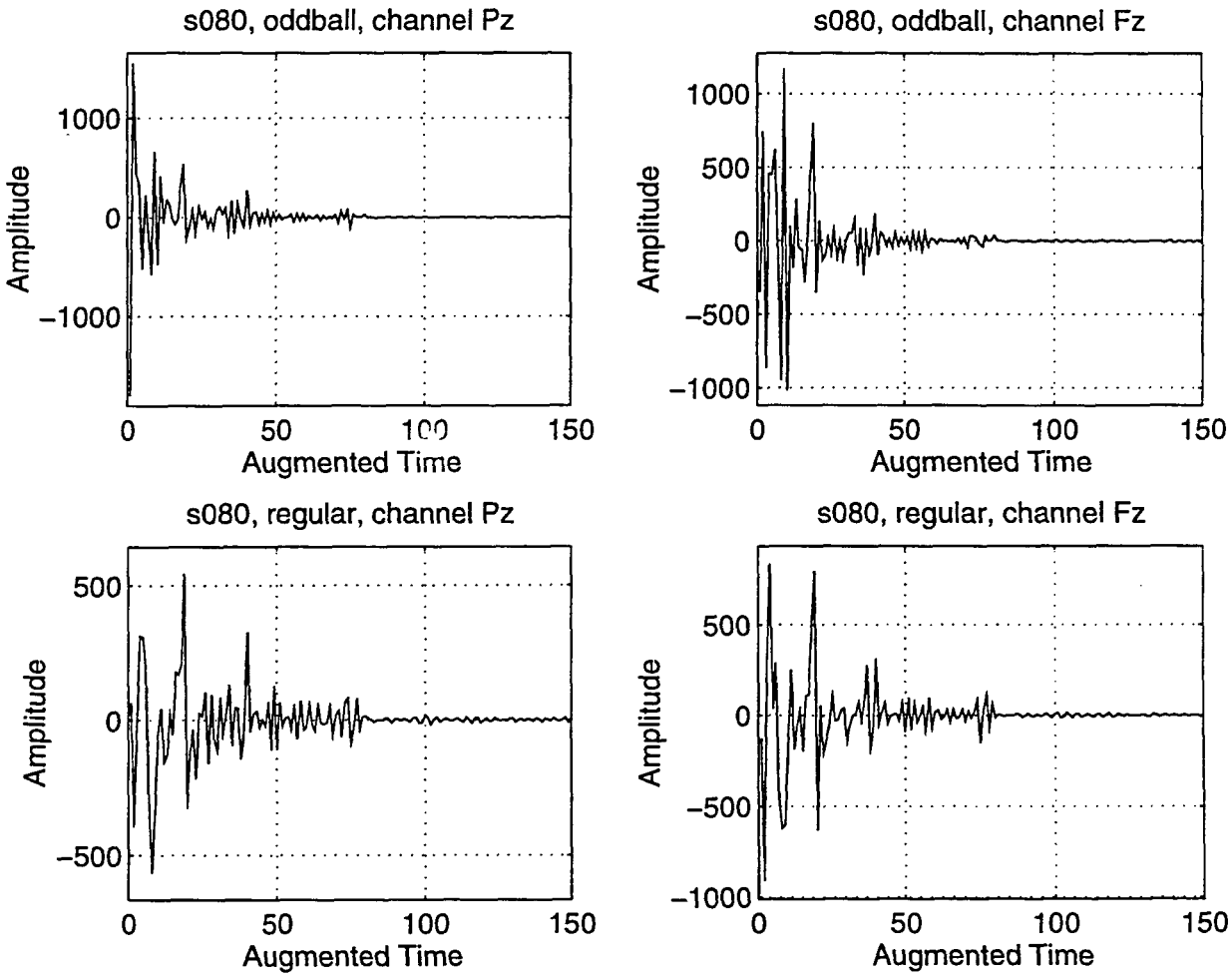

Figure 7.6: DWT of ERPs of an Alzheimer's disease patient, oddball and regular tones

The original, non-transformed, averaged signals were also used with this clustering algorithm. A similar data reduction was performed for the original (time-domain) signals by decimating and taking every twelfth sample. Thus, 50 samples of the time domain signal were input to the k-means algorithm. Interestingly, transformed and original signals were clustered exactly the same for every set of signals. This suggested that the discrete wavelet transform did not provide any additional information. It should be noted, however, that the intercluster distance between the two clusters was larger for the transformed signals. This can mean that transformed signals are somewhat *better* separated than the non-transformed signals, which would be of significant value for larger data bases with larger intercluster variance.

The correct classes of the signals were obtained after the completion of this unsupervised clustering. As expected, the responses to the oddball tones recorded from the Pz electrode gave the best classification performance. Table 7.1 summarizes the results of the k-means clustering algorithm. As seen in Table 7.1, the k-means clustering algorithm performed at 71.42 % by classifying 20 of the 28 signals correctly. The clustering algorithm classified the signals as belonging to class1 or class2 which were mapped to Alzheimer's disease and normal. Considering that it was a completely blind experiment, the 71.42% performance was considered to be very promising.

7.6 Supervised Learning with MLP / Backpropagation

A multilayer perceptron neural network was trained with the supervised backpropagation learning rule, in order to improve the classification. As discussed in Chapter 5, multilayer perceptrons are theoretically able to generate complex decision

Table 7.1: Results of the K-means clustering algorithm

Patient ID	Class	Classified As	Correctly Classified?
s048a1	ALZHEIMER'S	ALZHEIMER'S	YES
s049a1	ALZHEIMER'S	NORMAL	NO
s070a1	NORMAL	NORMAL	YES
s072a1	ALZHEIMER'S	ALZHEIMER'S	YES
s074a1	NORMAL	ALZHEIMER'S	NO
s077a1	NORMAL	NORMAL	YES
s080a1	NORMAL	NORMAL	YES
s081a1	NORMAL	ALZHEIMER'S	NO
s111a1	ALZHEIMER'S	ALZHEIMER'S	YES
s115a1	ALZHEIMER'S	ALZHEIMER'S	YES
s120a1	ALZHEIMER'S	NORMAL	NO
s121a1	NORMAL	ALZHEIMER'S	NO
s124a1	NORMAL	NORMAL	YES
s127a1	ALZHEIMER'S	ALZHEIMER'S	YES
s129a1	ALZHEIMER'S	ALZHEIMER'S	YES
s132a1	NORMAL	NORMAL	YES
s133a1	NORMAL	NORMAL	YES
s134a1	NORMAL	NORMAL	YES
s144a1	NORMAL	NORMAL	YES
s148a1	NORMAL	NORMAL	YES
s149a1	ALZHEIMER'S	NORMAL	NO
s224a1	NORMAL	ALZHEIMER'S	NO
s236a1	NORMAL	NORMAL	YES
s270a1	ALZHEIMER'S	ALZHEIMER'S	YES
s271a1	ALZHEIMER'S	ALZHEIMER'S	YES
s272a1	ALZHEIMER'S	NORMAL	NO
s273a1	ALZHEIMER'S	ALZHEIMER'S	YES
s276a1	ALZHEIMER'S	ALZHEIMER'S	YES

boundaries. However, their performance depends on many parameters, such as the number of nodes in the hidden layer(s), initial selection of the weights, selection of the learning rate, momentum term, etc. The performance of the MLP/BP is also dependent on the choice of the training data. Recall that a training data set is required with known correct classes for the MLP to learn.

Half of the signals were chosen randomly (seven with Alzheimer's disease and seven normal) to train the multilayer perceptron. The remaining 14 signals were kept for testing and were not shown to the neural network. Many different multilayer perceptron architectures were tried with different parameters and different initial weight selections to get as close as possible to the perfect classification.

7.6.1 Selection of the Training Data

Fourteen training data pairs were randomly chosen from the 28 sets of discrete wavelet transform signals. As explained below, this initial selection of the training data classified 11 of the 14 signals correctly. Then the three misclassified signals were added to the training data, and three of the signals used earlier in the training data were moved to the test data base. After this replacement, all the weight values were initialized randomly to reset the network. Note that the net was not trained with an *additional* three signals, but three pairs of signals of the previous data set were *replaced* by another three pairs of signals. In effect, the net was trained with a *better random* set of *fourteen* pairs of signals. The second set of the training data enabled the network to generalize better than the first one. The final training data consisted of the signals shown in Table 7.2.

Table 7.2: Training data

Patient ID	Class
s048a1	ALZHEIMER'S
s049a1	ALZHEIMER'S
s070a1	NORMAL
s081a1	NORMAL
s120a1	ALZHEIMER'S
s124a1	NORMAL
s129a1	ALZHEIMER'S
s132a1	NORMAL
s133a1	NORMAL
s144a1	NORMAL
s224a1	NORMAL
s149a1	ALZHEIMER'S
s270a1	ALZHEIMER'S
s276a1	ALZHEIMER'S

7.6.2 MLP/BP Parameters

All MLP architectures used had one hidden layer. The length of the input vector was varied from 50 samples to 150 samples. The number of hidden nodes was varied from 10 to 50. For every different set of parameters, the network was trained at least twenty times (with different initial weight selections).

Other parameters of the MLP/BP are given in Table 7.3. The meanings of the terms in Table 7.3 are explained in Chapter 5. The parameters given in Table 7.3 are the ones which gave the best performance. The program was run many times with different parameters to get the best performance.

Table 7.3: MLP/BP Parameters

Parameter	Value
Error goal	0.01
Initial learning rate	0.1
Increase rate in learning rate	1.25
Decrease rate in learning rate	0.6
Momentum term	0.95
Error ratio for variable learning rate	1.04
Number of input nodes	150
Number of hidden layers	1
Number of hidden layer nodes	30
Number of output nodes	2
Activation function	logarithmic sigmoid
Number of signals in the training data	14
Number of signals in the testing data	14
Correct classification % of the training data	100%(14/14)
Correct classification % of the testing data	92.87% (13/14)
Number of iterations for convergence	71

7.6.3 Performance of the MLP/BP

In all cases the multilayer perceptron generally converged very quickly to a (possibly local) minimum. When only 50 samples of the signal were used with 10 hidden layer nodes, the net had difficulty in converging. But even under these conditions, the net was able to classify 11 of the previously unseen 14 signals, a classification performance of 78.57%.

The best performance was obtain when 150 samples of the transformed signal were used with 30 hidden layer nodes. However, the training data was also slightly changed. As explained above, the three incorrectly classified signals in the previous experiment were replaced by three other signals. The ratio of 7 signals from Alzheimer's patients to 7 signals from normal people was kept. In this case the net classified 13 out of 14 of the previously unseen signals, a correct classification performance of 92.87%. The only signal that was misclassified was s270a1 which belonged to an Alzheimer's disease patient.

As in the case of unsupervised learning, the multilayer perceptron was also trained with the non-transformed time domain data for comparison purposes. Unlike the k-means algorithm, the MLP/BP did not give similar results for both transformed and non-transformed signals.

The non-transformed data were more likely to trap the network into a local minimum, since the net frequently failed to converge during training. When it did converge, the classification performance was usually poor compared to that obtained using the transformed signal. The typical average correct classification percentage for the non-transformed data was around 64.28% (usually 9/14), whereas for transformed data this average was 78.57% (usually 11/14).

7.7 Conclusion and Recommendation For Future Study

In this study the wavelet transform based time-frequency representation methods were used to analyze the ERPs from patients with and without Alzheimer's disease. ERPs, or generally speaking EEGs, are highly non-stationary signals, and wavelet analysis is a suitable technique for processing such signals.

Although the original time domain signal can be classified with reasonable accuracy, the initial results show that a wavelet transform based technique can further improve the classification of a supervised trained neural network.

However, the number of patient responses available for this experiment is too small to make statistically valid generalizations. Future work on this topic must extend the analysis to a bigger database. Also a selected portion of the continuous wavelet transform could be interfaced with a neural network to allow it to be used for neural network classification purposes.

It appears that a well trained neural network with an adequate number of signals might be useful for diagnosing Alzheimer's disease. As a matter fact, since discrete wavelet transform is particularly fast, this diagnosis could be done on-line. The result of the analysis could be made available while the patient is still connected to the EEG machine. Moreover, a well trained neural network could be implemented in hardware, and a small *Alzheimer's Detector* could be realized. Initial results of this study are found to be very promising, making further work in this area worth pursuing.

BIBLIOGRAPHY

[1] Fourier J.B.J., *Theorie Analytique de la Chaleur, tome premier*, Darboux G. (editor), Gauthiers-Villars, 1888.

[2] Meyer Y., *Wavelets, Algorithms and Applications*, Dyan R.D. (translated), Siam, Philadelphia, PA.,1993.

[3] Mallat S., *A Theory for Multiresolution Signal Decomposition: the Wavelet Representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol:11, no:7, p:674-693, July 1989.

[4] Daubechies, I. *Orthonormal Bases of Compactly Supported Wavelets*, Communications in Pure and Applied Math., vol:41, no:7, p:909-996, 1988.

[5] Daubechies, I. *The Wavelet Transform, Time-Frequency Localization and Signal Analysis*, IEEE Transactions on Information Theory, vol:36, no:5, p:961-1005 , Sept 1990.

[6] Thakor N.V., Xin-rong G., Yi-Chun S., Hanley D.F., *Multiresolution Wavelet Analysis of Evoked Potentials*, IEEE Transactions on Biomedical Engineering, vol:40, no:11, p:1085-1093, 1993.

[7] Trejo L.J., Shensa M.J., *Linear and Neural Network Models for Predicting Human Signal Detection Performance From Event-Related Potentials: A Comparison of the Wavelet Transform With Other Feature Extraction Methods*, downloaded from the ftp server of University of South Carolina Math. department, cite address: maxwell.math.scarolina.edu, directory: pub/wavelet

[8] Hlawatsch F., and Boudreaux-Bartels G.F., *Linear and Quadratic Time-Frequency Representations*, IEEE Signal Processing Magazine, April 1992, p:21-67.

[9] Rioul O., and Vetterli M., *Wavelets and Signal Processing*, IEEE Signal Processing Magazine, p:14-38, October 1991.

[10] Kaiser G., *A Friendly Guide to Wavelets*, Birkhauser, Boston MA., 1994.

[11] Croiser A., Esteban D., and Galand C., *Perfect Channel Splitting by Use of Interpolation, Decimation, Tree Decomposition Techniques*, Int. Conf. on Information Sciences/Systems, Patras , p443-446, Aug. 1976.

[12] Crochiere R.E., Weber S.A., and Flanagan J.L., *Digital Coding of Speech in Subbands*, Bell Syst. Tech. Journal, vol:55, p:1069-1085, Oct. 1976.

[13] Burt P.J. *Multiresolution Techniques for Image Representation, Analysis and Smart Transmission*, Proceedings of SPIE Conf. on Visual Communication and Image Processing, p:2-15, Philadelphia, PA., Nov. 1989.

[14] Vetterli M., and Le Gall D., *Perfect Reconstruction FIR Filter Banks: Some Properties and Factorizations*, IEEE Transactions on Acoust. Speech Signal Proc., vol:37, no:7, p:1057-1071, July 1989.

[15] Rioul O., *Discrete Time Multiresolution Theory* , IEEE Transactions on Signal Processing, vol:41, p:2591-2606, August 1993.

[16] Akansu A.N., and Haddad R.A., *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, Academic Press, Inc. San Diego CA., 1992.

[17] Vetterli M., Herley C., *Wavelets and Filter Banks: Theory and Design* IEEE Transactions on Signal Processing, 1992.

[18] Keinert F., Introduction to Wavelet Theory and Applications, Lecture Notes for Math 485X, Iowa State University, Ames, IA., 1992.

[19] Haykin S., *Neural Networks, A Comprehensive Foundation*, Macmillan College Publishing Company, New York, 1994.

[20] Shepperd G.M., Koch C., "Introduction to the Synaptic Circuits.",In *The Synaptic Organization of the Brain* (Shepherd editor), p:3-31, Oxford University Press, New York, NY., 1990.

[21] Lippmann R.P., *An Introduction to Computing with Neural Nets*, IEEE ASSP Magazine, vol:4, no:2, p:4-22, April, 1987.

[22] Hush D.R., Horne B.G., *Progress in Supervised Neural Networks*, IEEE Signal Processing Magazine, vol:10, no:1, p:8-39, January, 1993.

[23] Rosenblatt F., *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, Psychological Review, vol:65, p:386-408, 1958.

[24] Wilson, *Image Algebra Handbook*, downloaded from the ftp server of University of Florida, cite address: ftp.cis.ufl.edu, directory: pub/ia/documents

[25] Demunt H., Beale M., *Neural Network Toolbox for Use with MATLAB*, The Math Works Inc., Natick MA., 1992.

[26] Kohonen T., *Self-Organized Formation of Topologically Correct Feature Maps*, Biological Cybernetics, vol:43, p:59-69.

[27] Kohonen T., *The Self Organizing Map*, Proceedings of the IEEE, vol:78. p:1464-1480.

[28] Gonzales R.C., and Tou J.T., *Pattern Recognition Principles* Addison-Wesley Publishing Company, Reading, MA., 1974

[29] MacQuinn J., *Some Methods for Classification and Analysis of Multivariate Observation*, In "Proceedings of the 5^{th} Berkeley Symposium on Mathematical Statistics and Probability (LeCun L.M., Neyman J. editors) , Berkeley: University of California Press, vol:1, p:281-297.

[30] Udpa L., Pattern Recognition, Lecture Notes for EE529-G, Iowa State University, Ames, IA.,Fall 1994.

[31] Niedermeyer E., and Lopes Da Silva, F, *Electroencephalography*, Williams and Wilkins, Baltimore, MD., 1993.

[32] Goodin D.S., and Squires K.C., Starr A., *Long Latency Event-Related Components of the Auditory Evoked Potential in Dementia*, Brain, vol:101, p:635-648, 1978.

[33] Putnam L.E., and Roth W.T., *Effects of Stimulus Repetition, Duration, and Rise Time on Startle Blink and Automatically Elicited P300.* Psych-physiology, vol:27, No:3, p:275-297, 1990.

[34] Polich J., Ehlers C.L., Otis S., Mandell A.J., and Bloom F.E., *P300 Latency Reflects the Degree of Cognitive Decline in Dementing Illness*, Electroencephalograpy and Clinical Neurophysiology, vol:63, p:138-144, 1985.

[35] Goodin D.S., *Clinical Utility of Long Latency Cognitive Event-Related Potentials (P3): The Pros*,Electroencephalograpy and Clinical Neurophysiology, vol:76, p:2-5, 1990.

[36] Pfefferbaum A., Ford J.M., and Kraemer H.C., *Clinical Utility of Long Latency Cognitive Event-Related Potentials (P3): The Cons*,Electroencephalograpy and Clinical Neurophysiology, vol:76, p:6-12 , 1990.

[37] Goodin D.S., Squires K.C., Henderson B.H., and Starr A., *Age-related Variations in Evoked Potentials to Auditory Stimuli in Normal Human Subjects*, Electroencephalograpy and Clinical Neurophysiology, vol:44, p:447-458, 1978.

[38] Papanicolaou, A.C, Loring D.C., Raz N., and Eisenberg H.M., *Relationship Between Stimulus Intersity and the P300*, Psychophysiology, vol:22, p:326-329, 1985.

[39] Polich J., *P300, Probability, and Interstimulus Interval*, Psychophysiology, vol:27, p:396-403, 1990.

[40] Fabiani M., Karis D., and Donchin D., *Effects of Mnemonic Strategy Manipulation in a Von Restorff Paradigm*, Electroencephalograpy and Clinical Neurophysiology, vol:75, p:22-35, 1990.

[41] Noldy N., Stelmack R., and Campbell K., *Event-Related Potentials and Recognition Memory for Pictures and Words: The effects of Intentional and Incidental Learning*, Psychophysiology, vol:27, p:417-428, 1990.

[42] Paller K.A., McCarthy G., and Wood C.C.,*ERPs Predictive of Subsequent Recall and Recognition Performance*, Biol. Psychol., vol:26, p:269-276, 1988.

[43] Squires K.C., Chippendale T.J., Wrege K.S., Goodin D.S., and Starr A., *Electrophysiological Assessment of Mental Function in Aging and Dementia* in Aging in the 1980's: Selected Contemporary Issues in the Psychology of Aging, Poon L.W. (editor), Washington D.C.: American Psychological Association, p125-134, 1980.

[44] Polich J., Ladish C., and Bloom F.E., *P300 Assessment of early Alzheimer's Disease*, Electroencephalograpy and Clinical Neurophysiology, vol:77, p:179-189, 1990.

[45] Martini F., *Fundamentals of Anatomy and Physiology*, Prentice Hall, Englewood Cliffs, NJ., 1992.

[46] Reisberg B. (editor) *Alzheimer's Disease, The Standard Reference*, The Free Press, New York, NY., 1983.

[47] Becker R., and Giacobini E., *Current Resaeych in Alzheimer's Therapy*, Taylao and Francis, New York, NY., 1988.

[48] Bick K., Amaducci L., and Giancarlo P. (editors) *The Early History of Alzheimer's Disease*, Liviana Press, Padova, Italy, 1987.

[49] Powell L.S., *Alzheimer's Disease, A Guide For Families*, Addison-Wesley, Reading MA., 1993.

[50] Light E., and Lebowitz B.D., *Alzheimer's Disease Treatment and Family Stress: Directions for Research*, U.S. Department of Health and Human Services, Rockville, 1989.

[51] West P., Personal communications, 1994-1995.

APPENDIX A. AVERAGED SIGNALS

The event-related potentials recorded in response to oddball tones and regular tones from channels Pz and Fz are given in this appendix for each patient. Each signal is averaged and the prestimulus is removed from all of them. The mean of the prestimulus, and the mean of the entire signal is subtracted from each signal. The time axes shows the index to the sample number. Every signal has 600 samples (after removing the first 150 samples corresponding to the prestimulus). The entire signal was 1250 ms with 750 samples. Removing the first 150 samples corresponds to removing the first 250 ms from the data. Therefore 600 samples on the time axis corresponds to 1 second of data, sampled at 600 Hz.

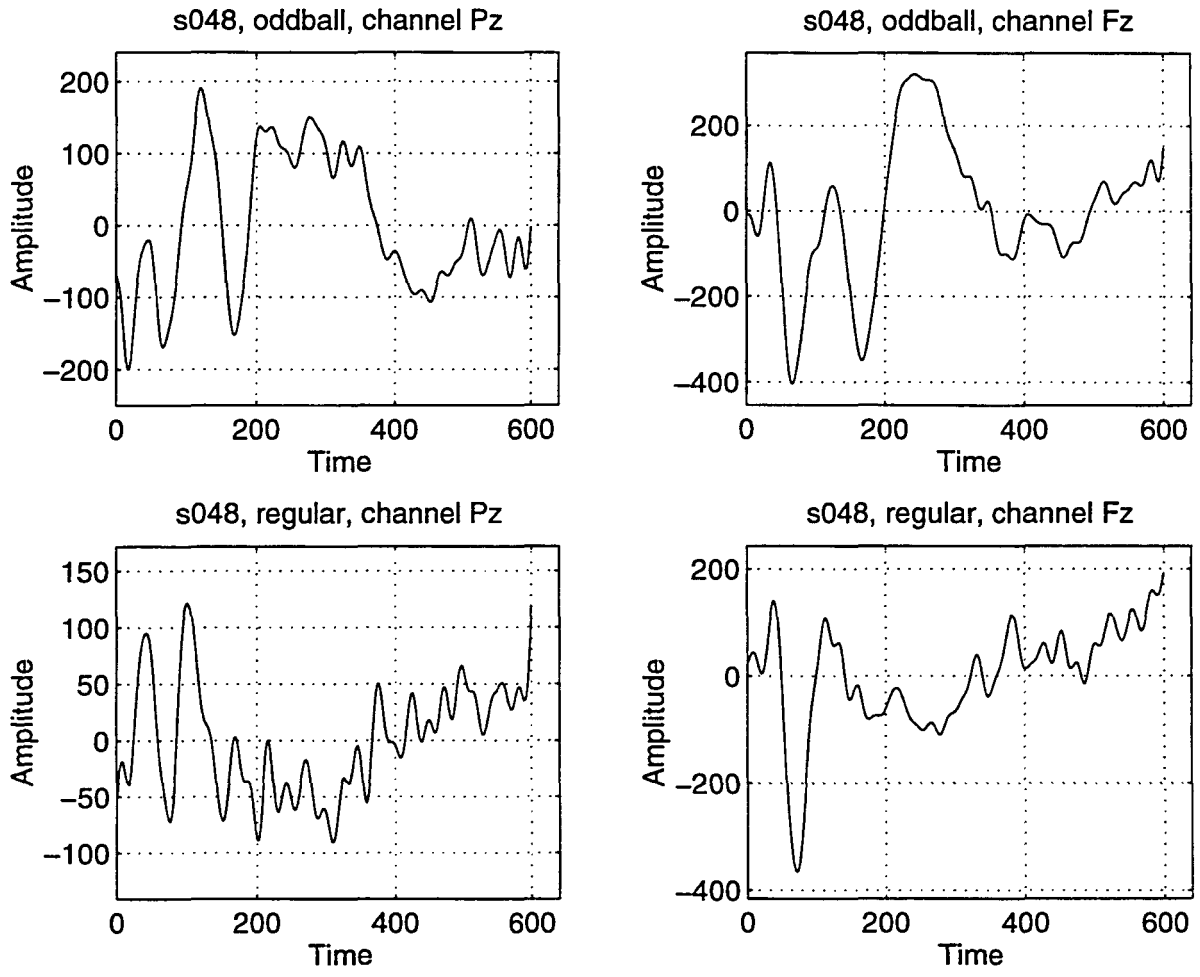

Figure A.1: Patient ID: s048, Alzheimer's

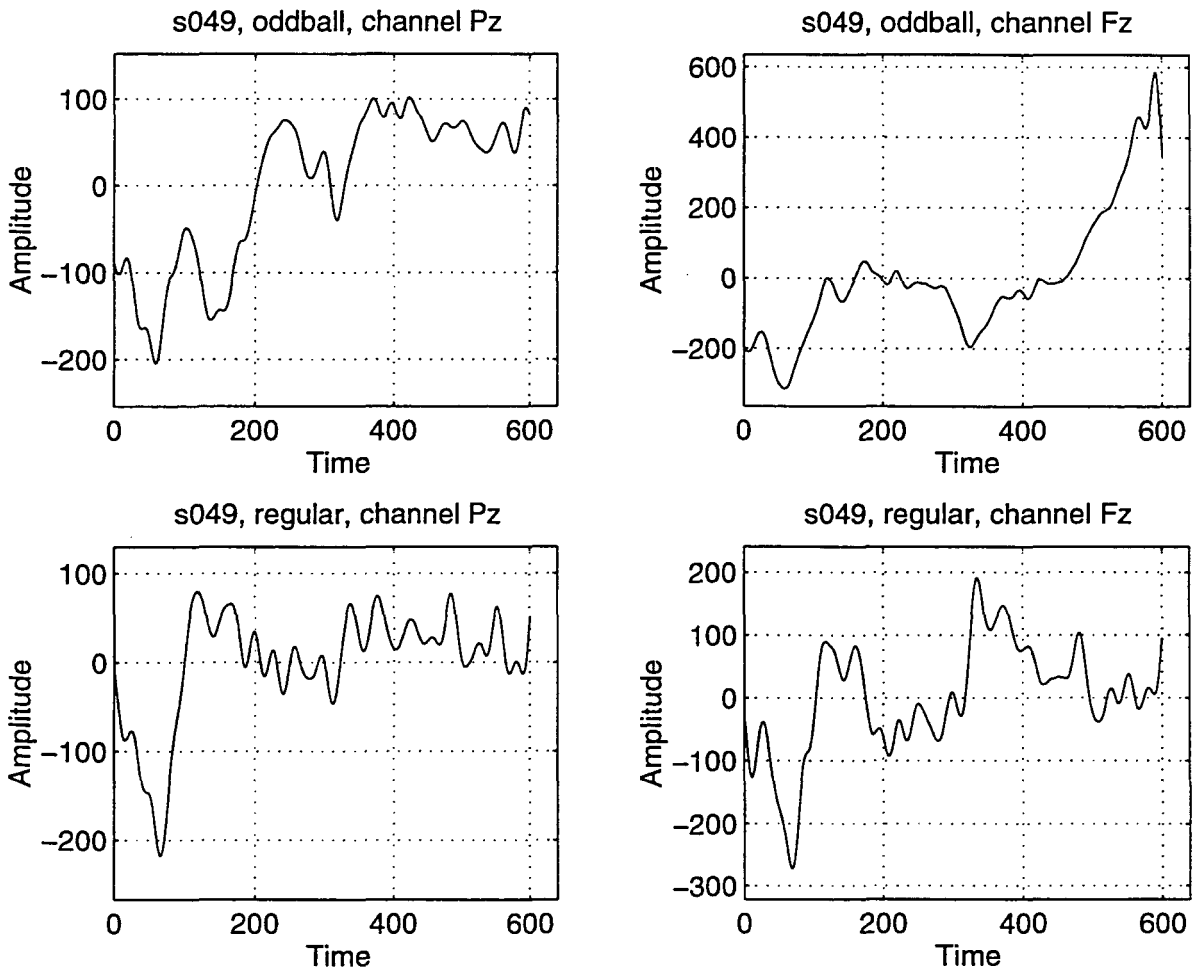

Figure A.2: Patient ID: s049, Alzheimer's

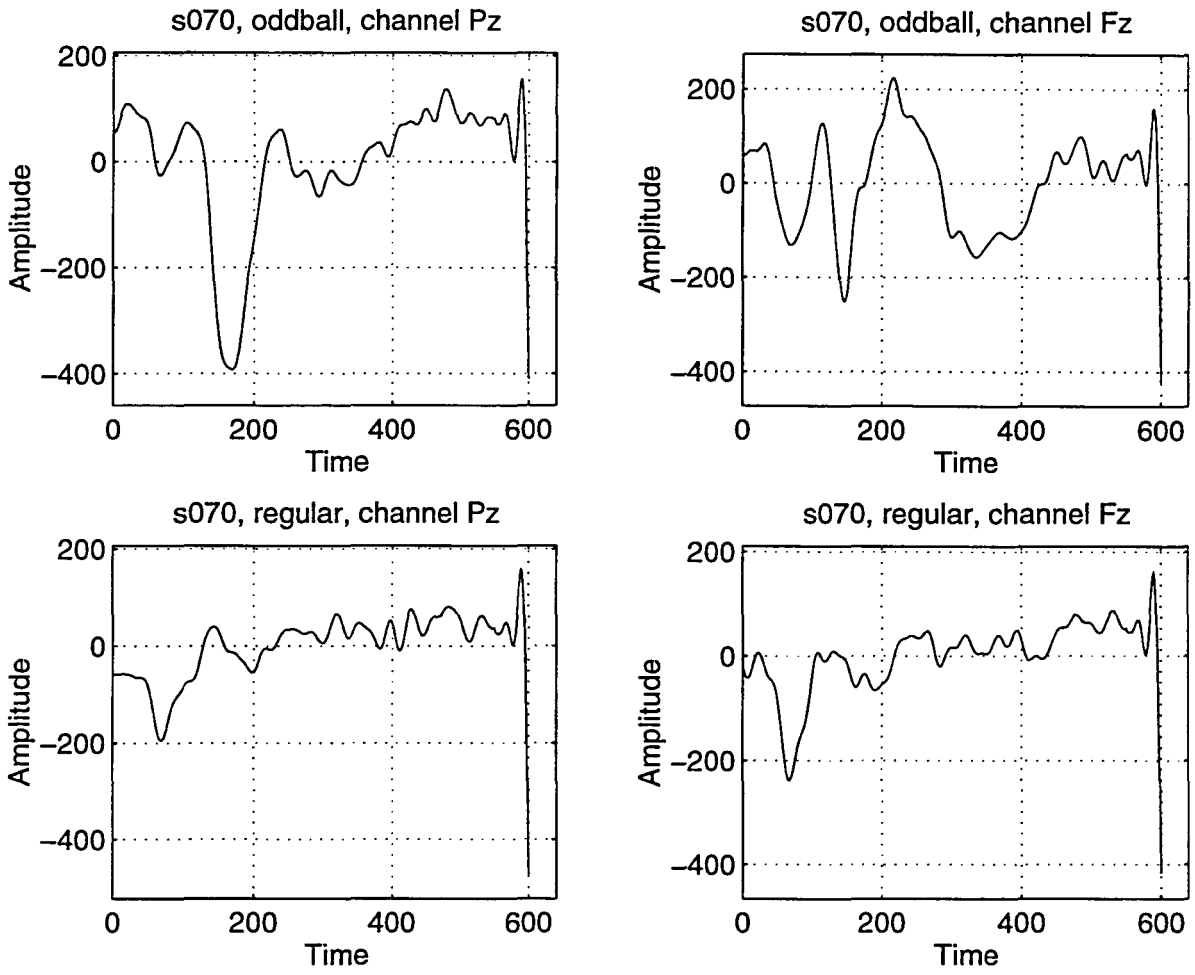

Figure A.3: Patient ID: s070, Normal

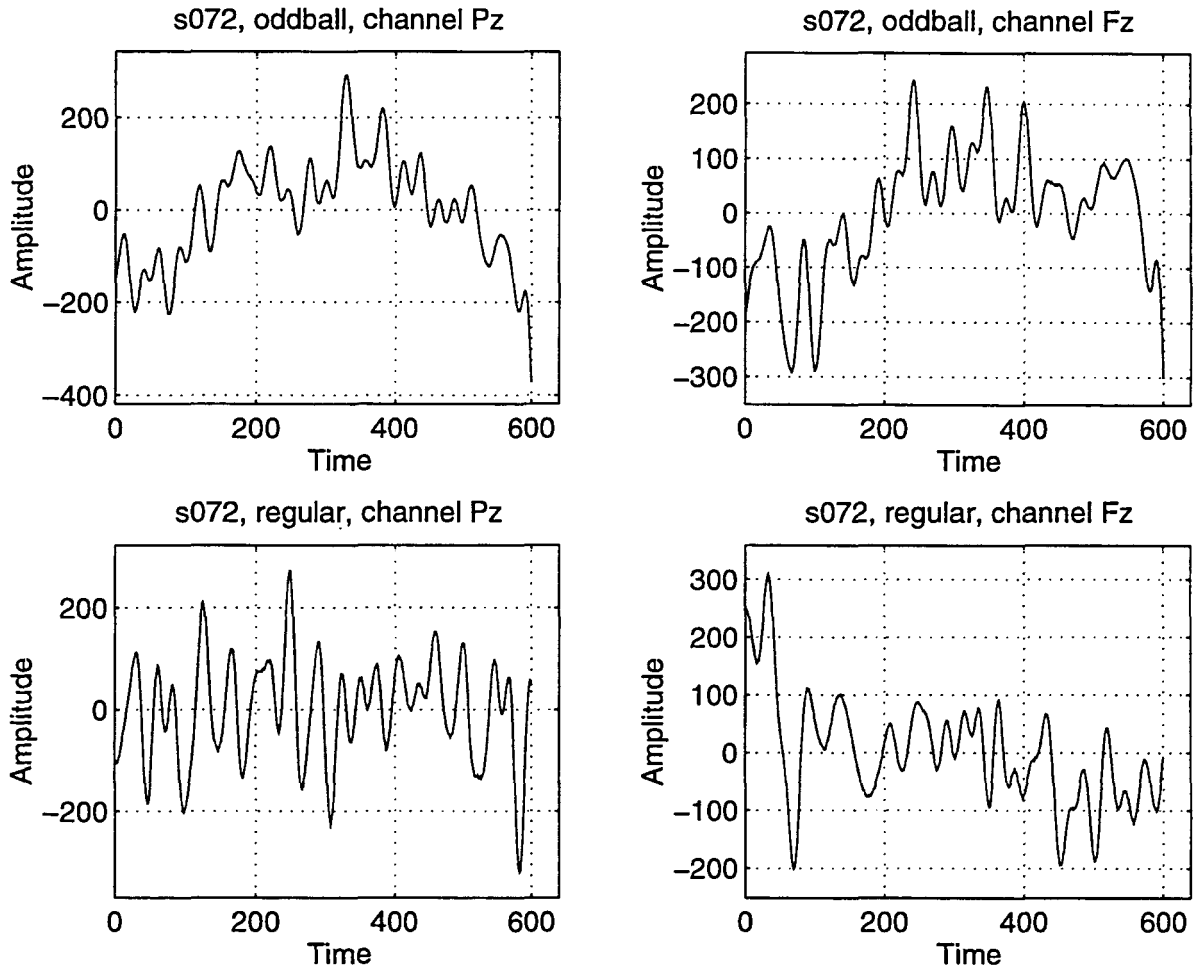

Figure A.4: Patient ID: s072, Alzheimer's

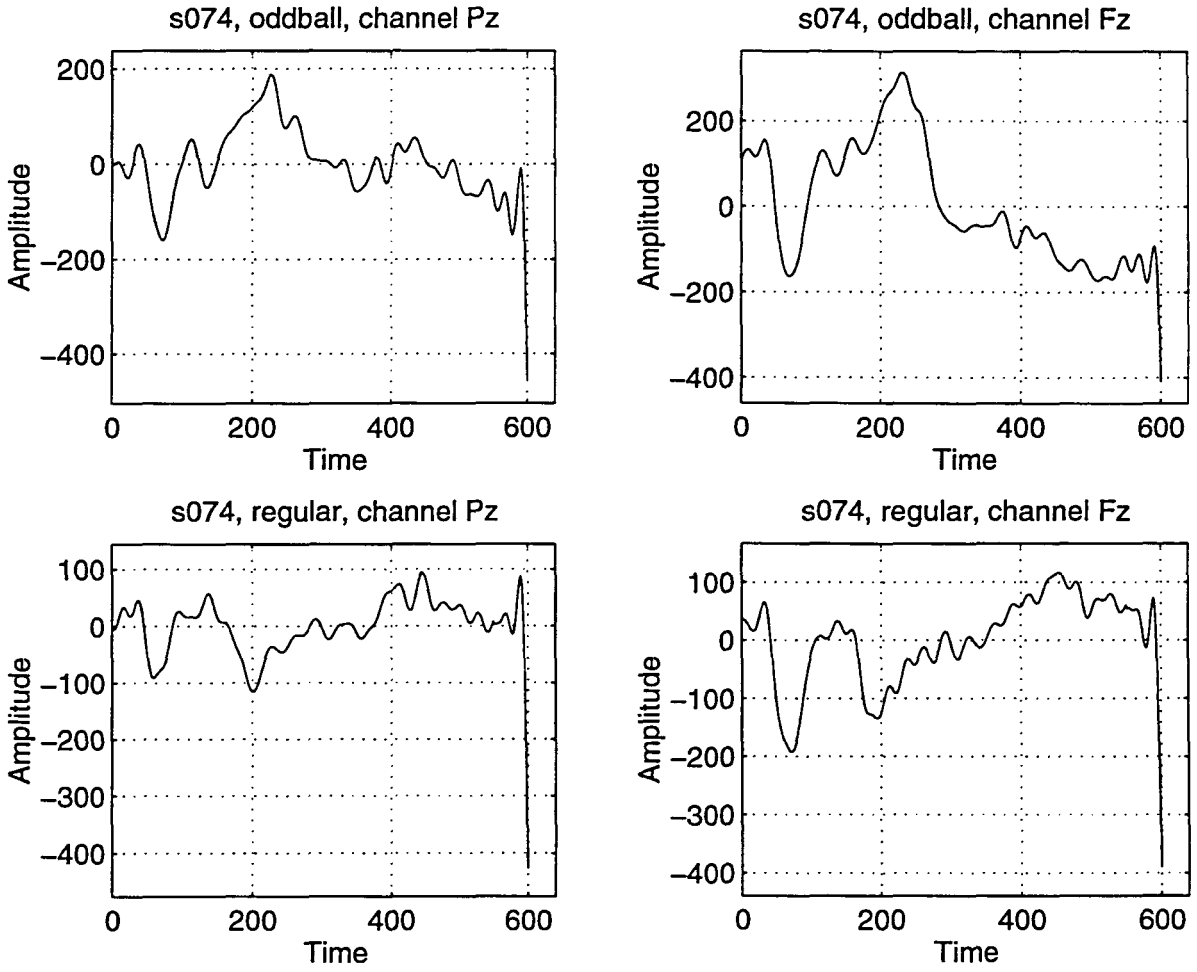

Figure A.5: Patient ID: s074, Normal

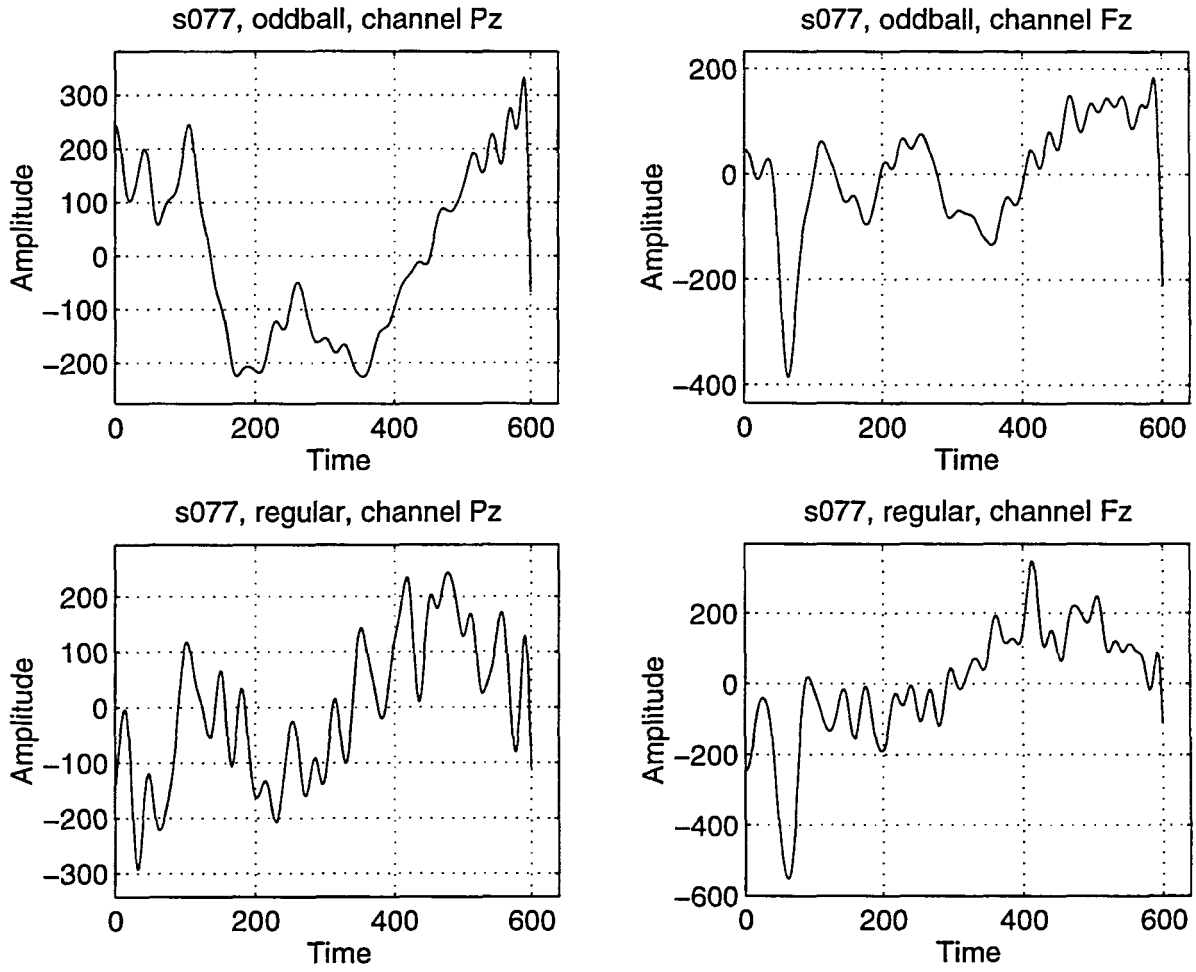

Figure A.6: Patient ID: s077, Normal

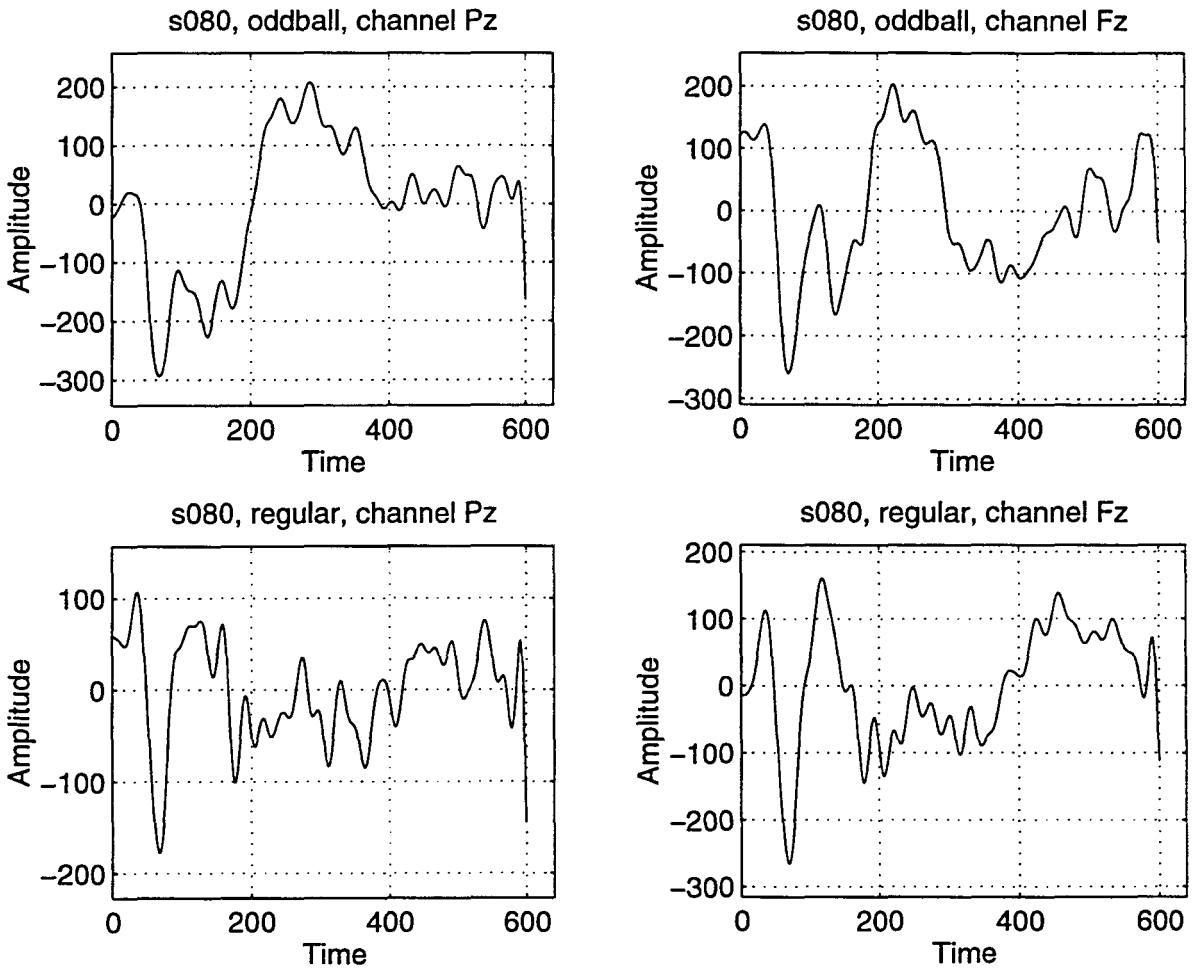

Figure A.7: Patient ID: s080, Normal

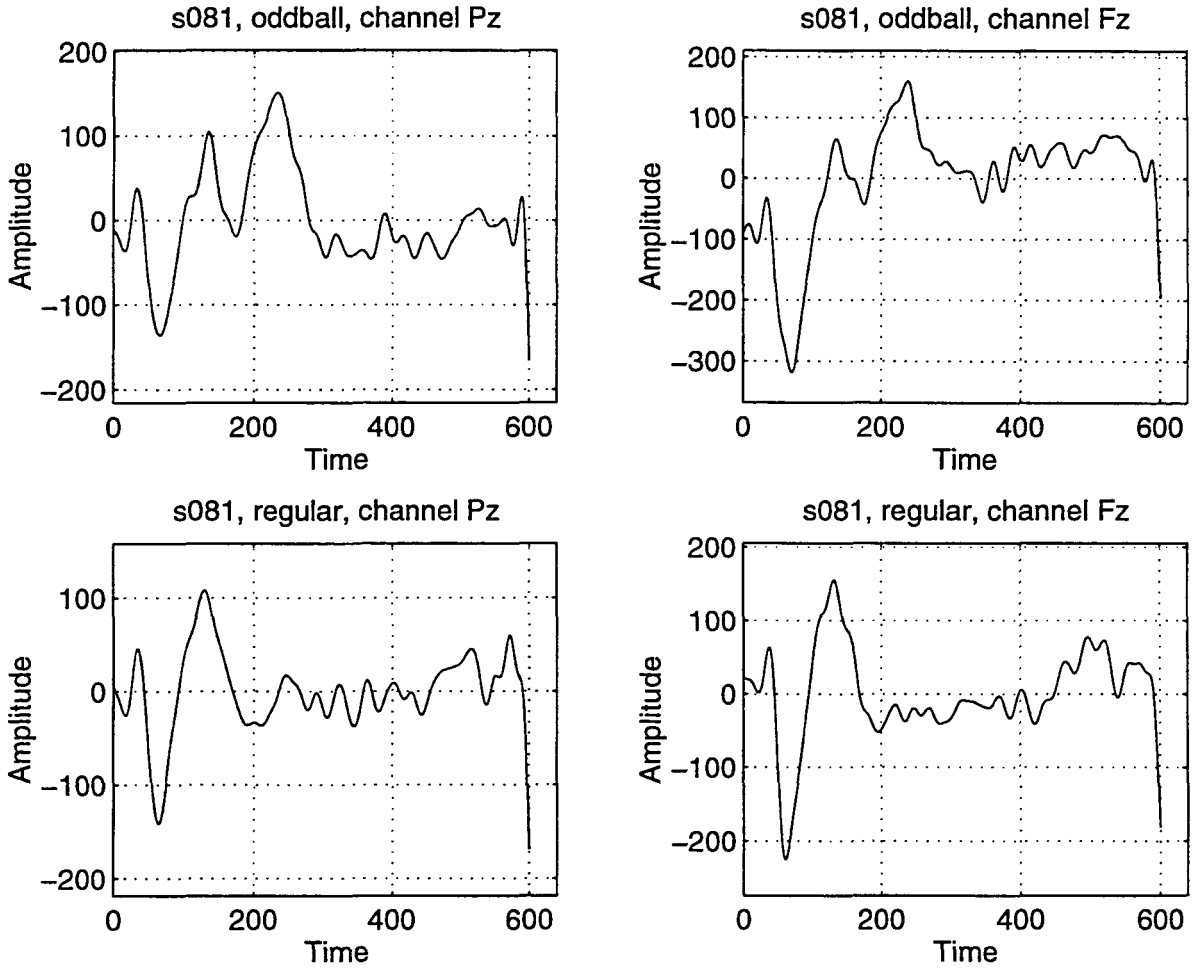

Figure A.8: Patient ID: s081, Normal

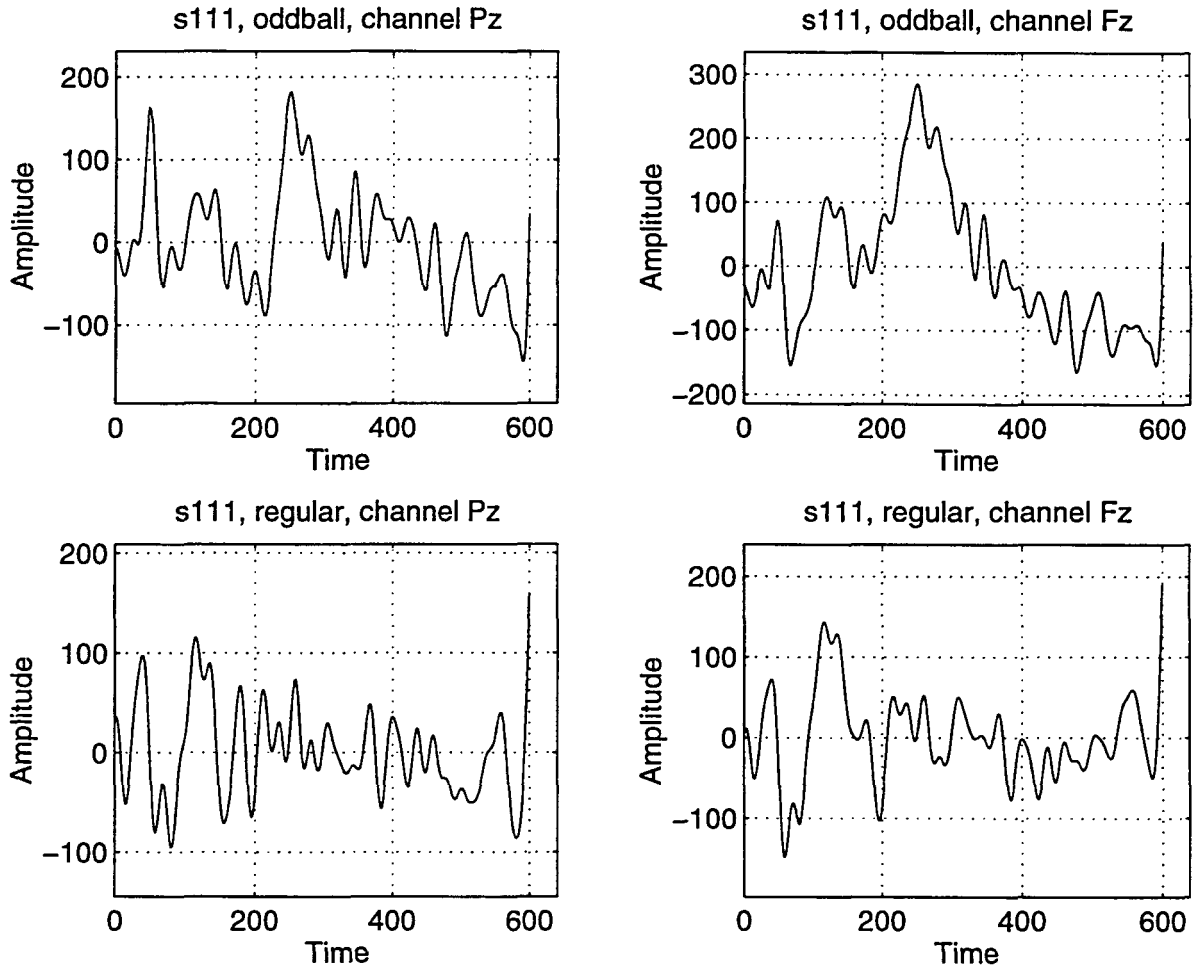

Figure A.9: Patient ID: s111, Alzheimer's

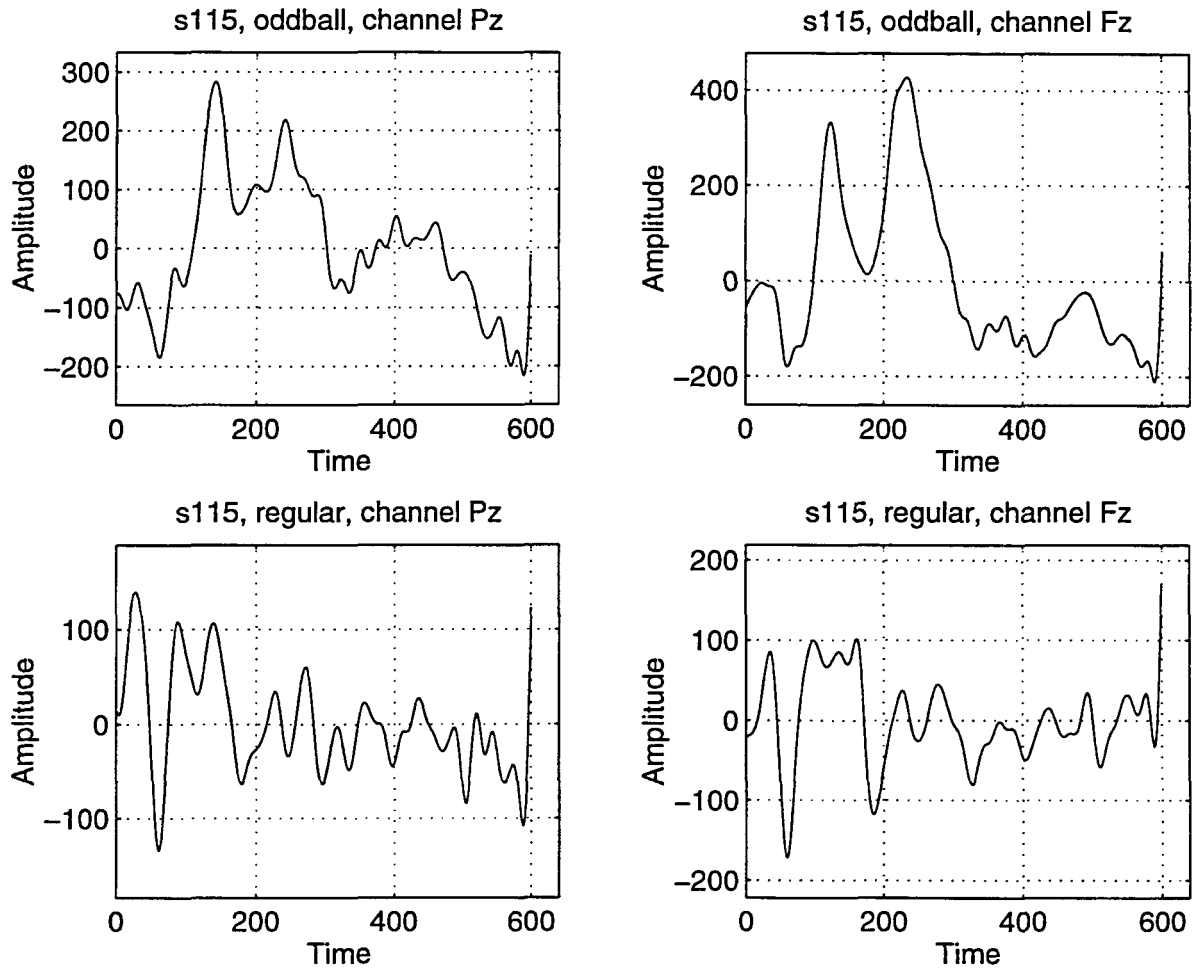

Figure A.10: Patient ID: s115, Alzheimer's

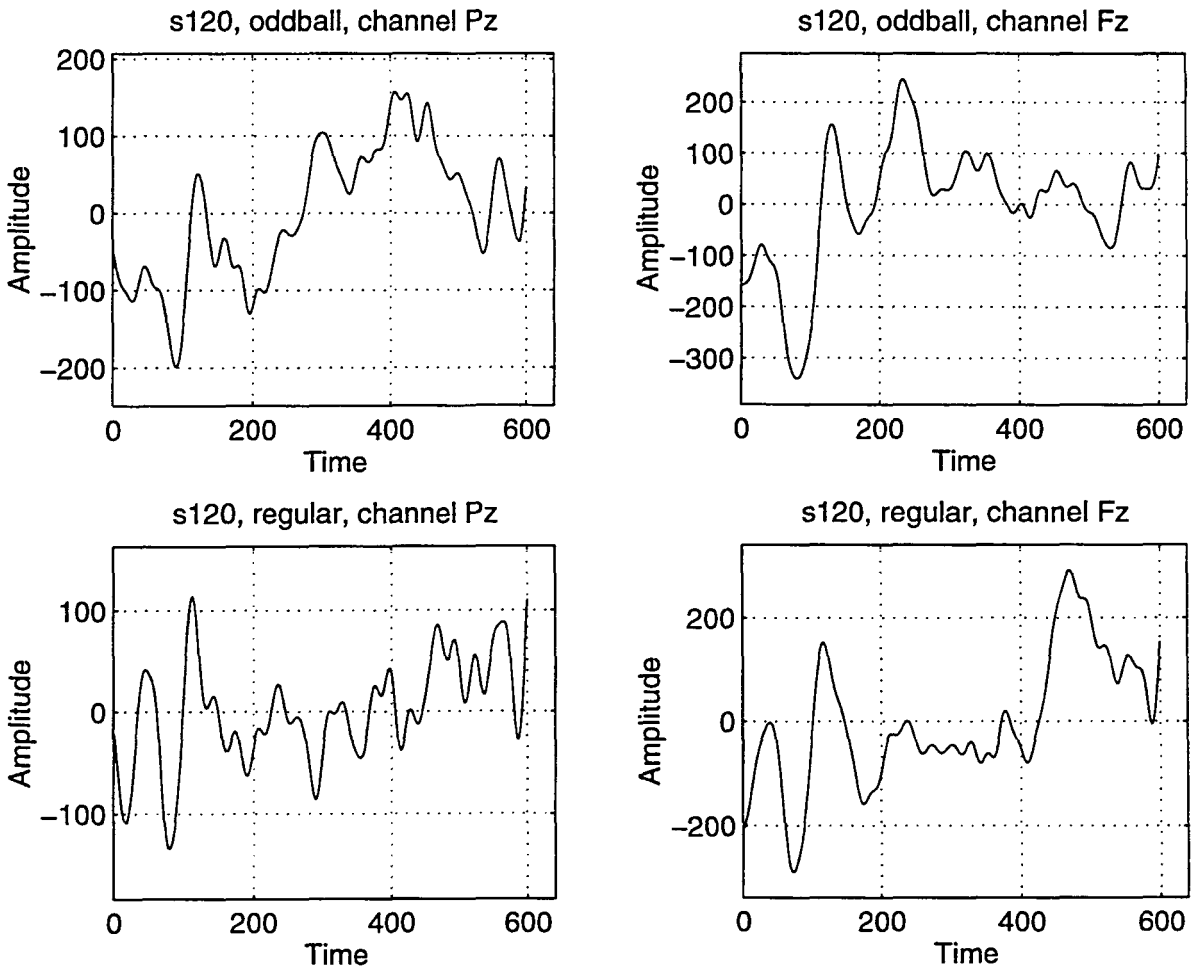

Figure A.11: Patient ID: s120, Alzheimer's

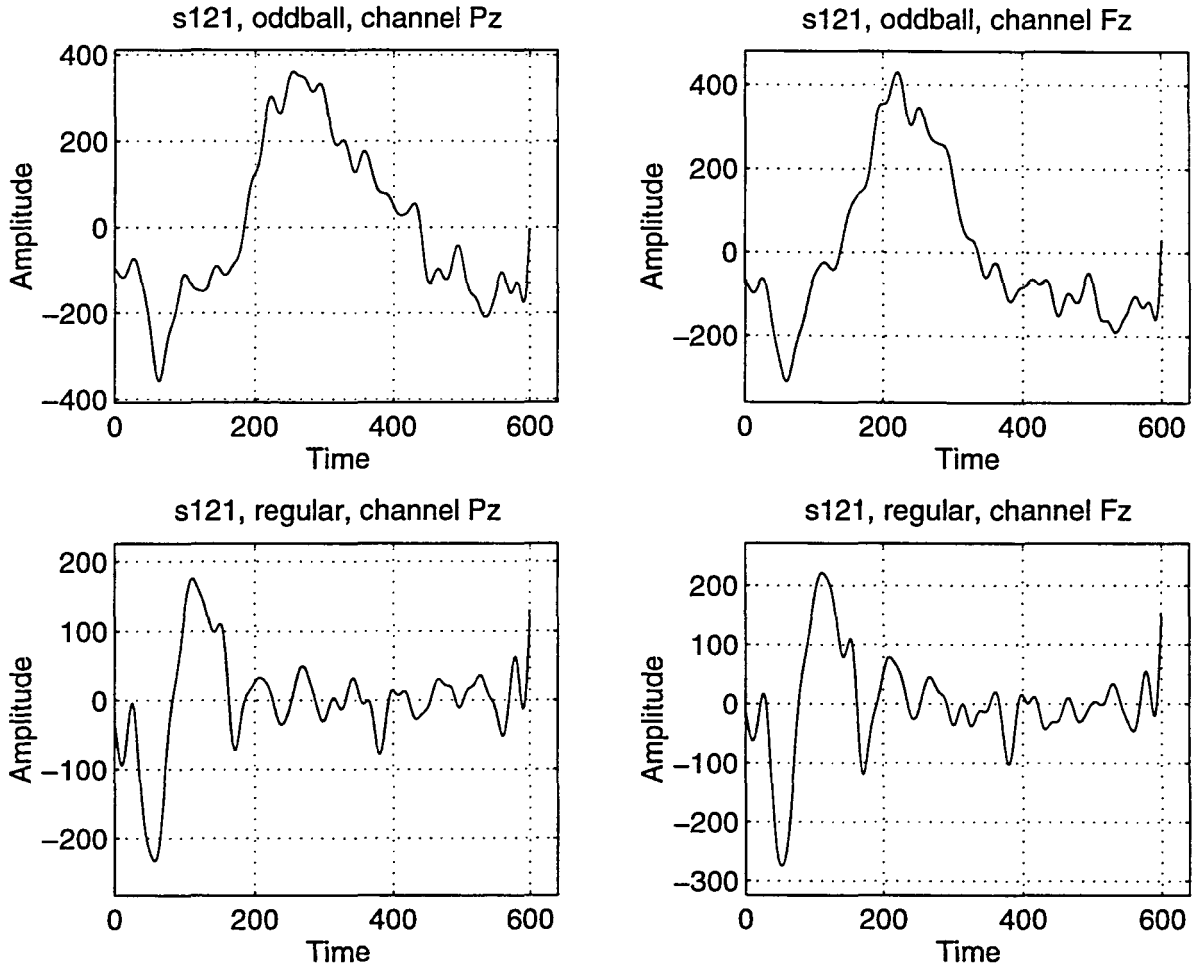

Figure A.12: Patient ID: s121, Normal

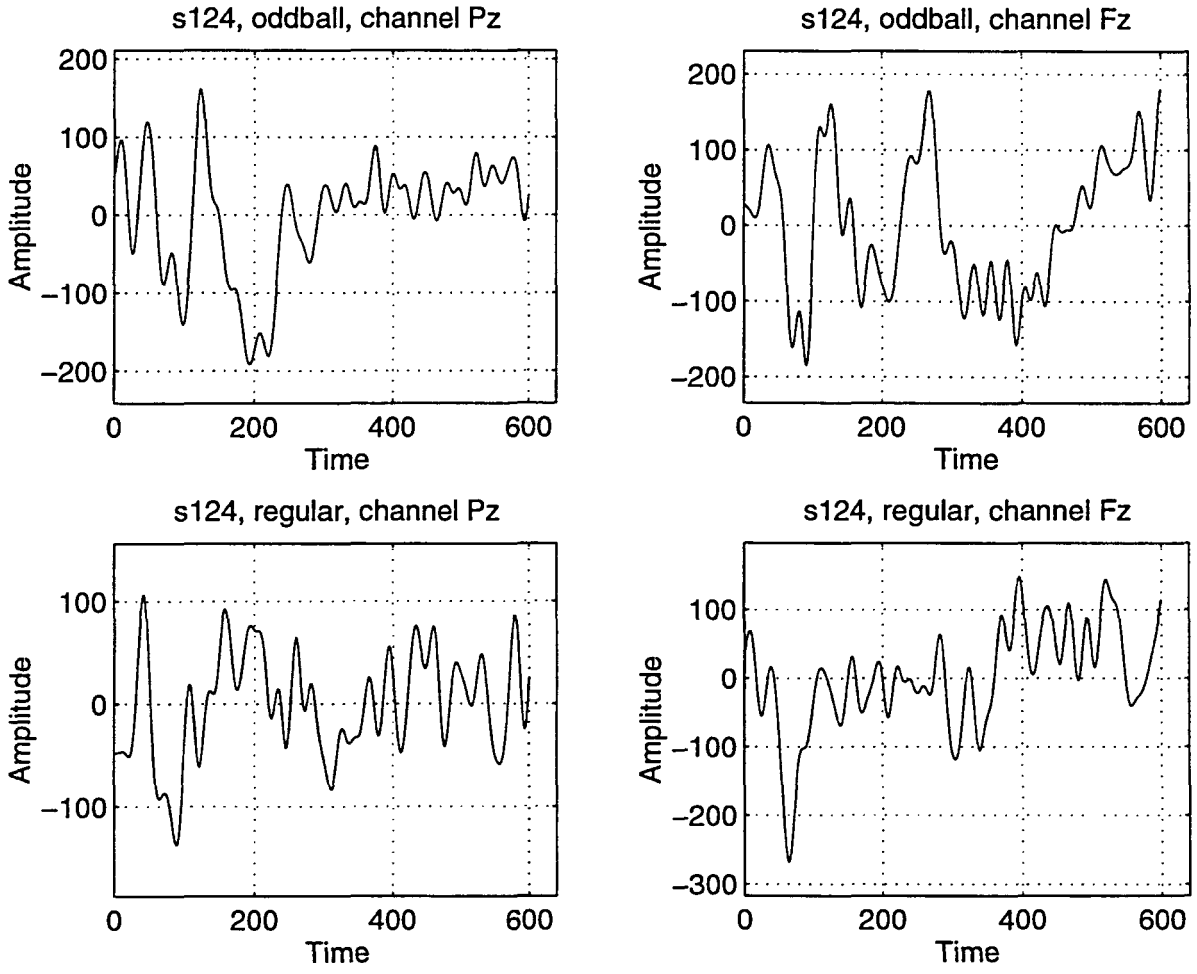

Figure A.13: Patient ID: s124, Normal

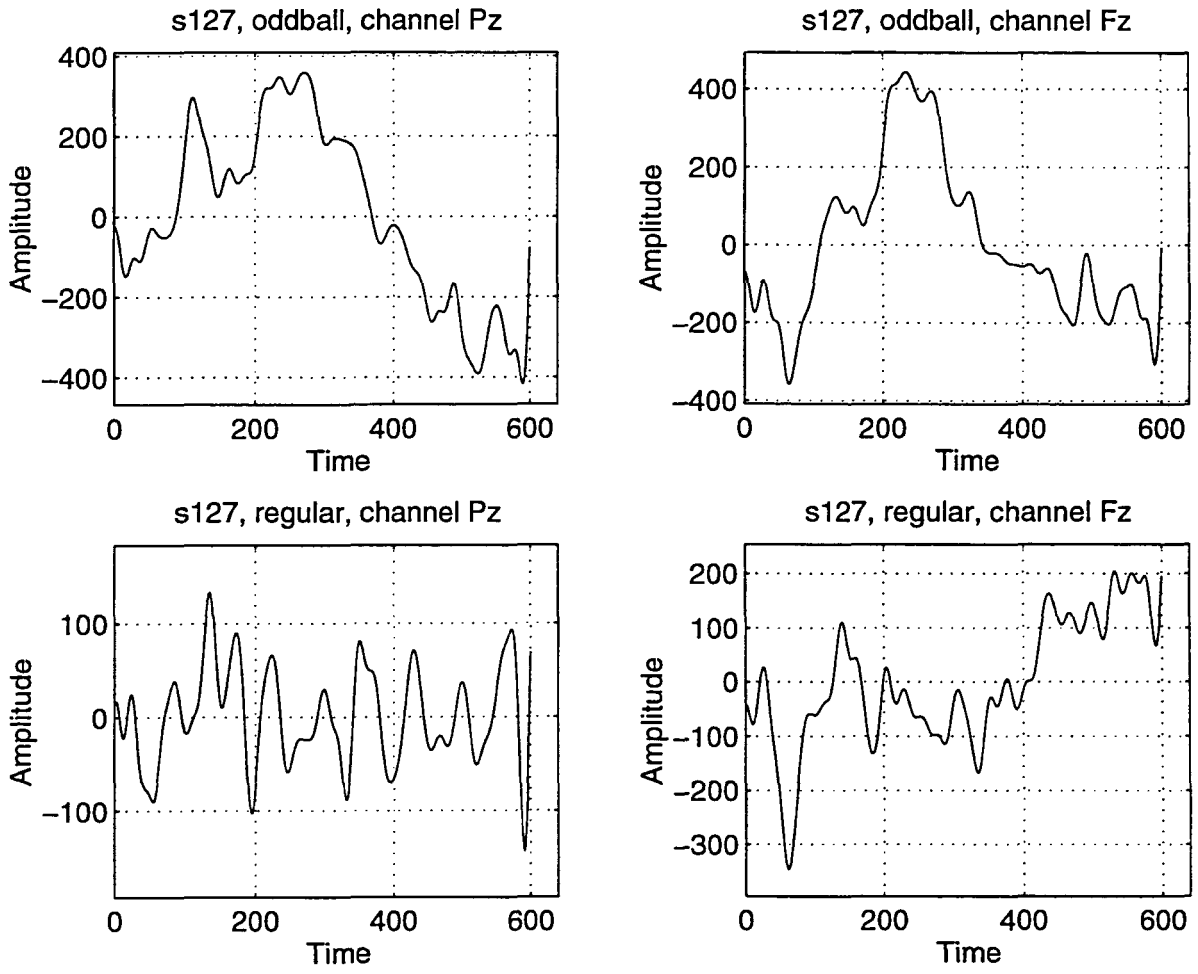

Figure A.14: Patient ID: s127, Alzheimer's

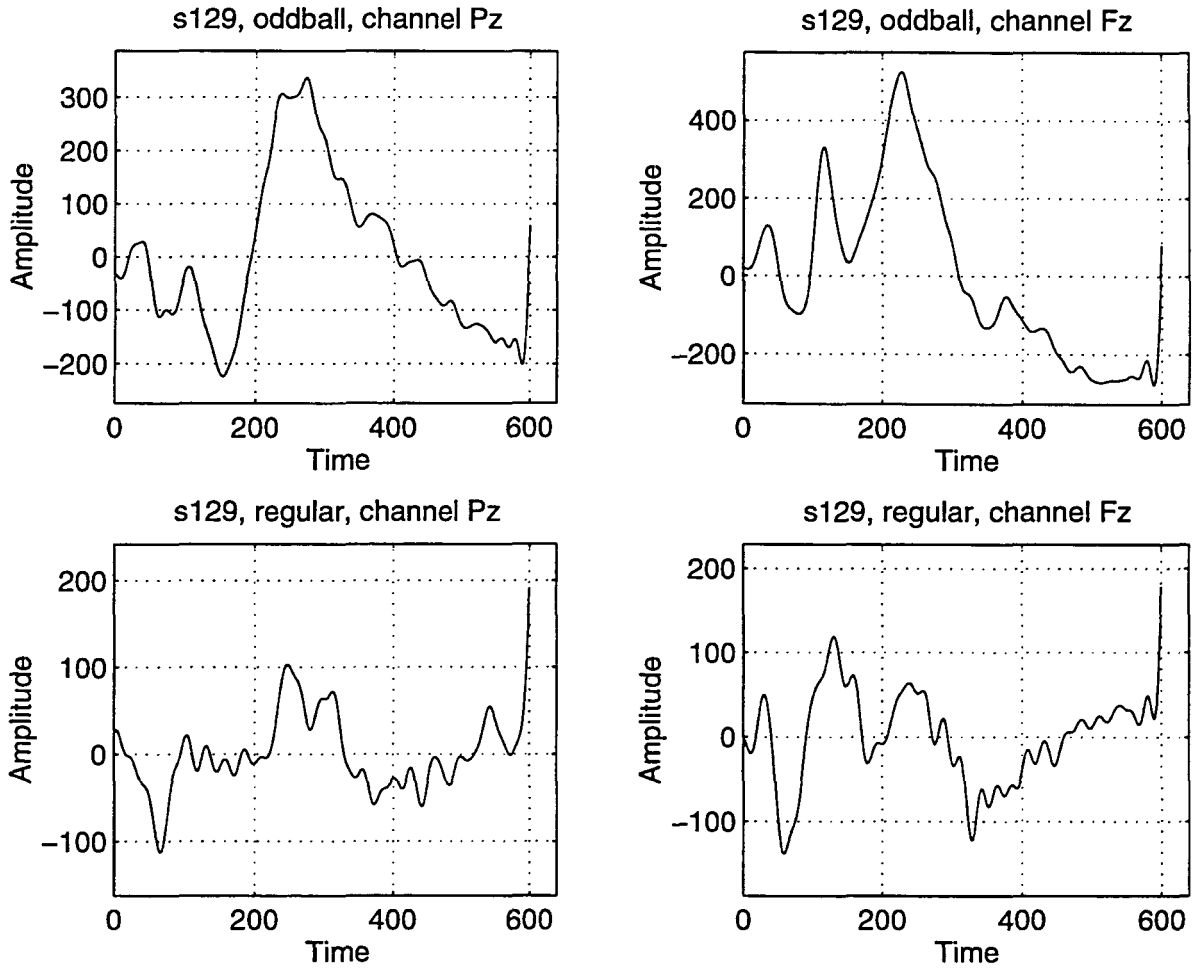

Figure A.15: Patient ID: s129, Alzheimer's

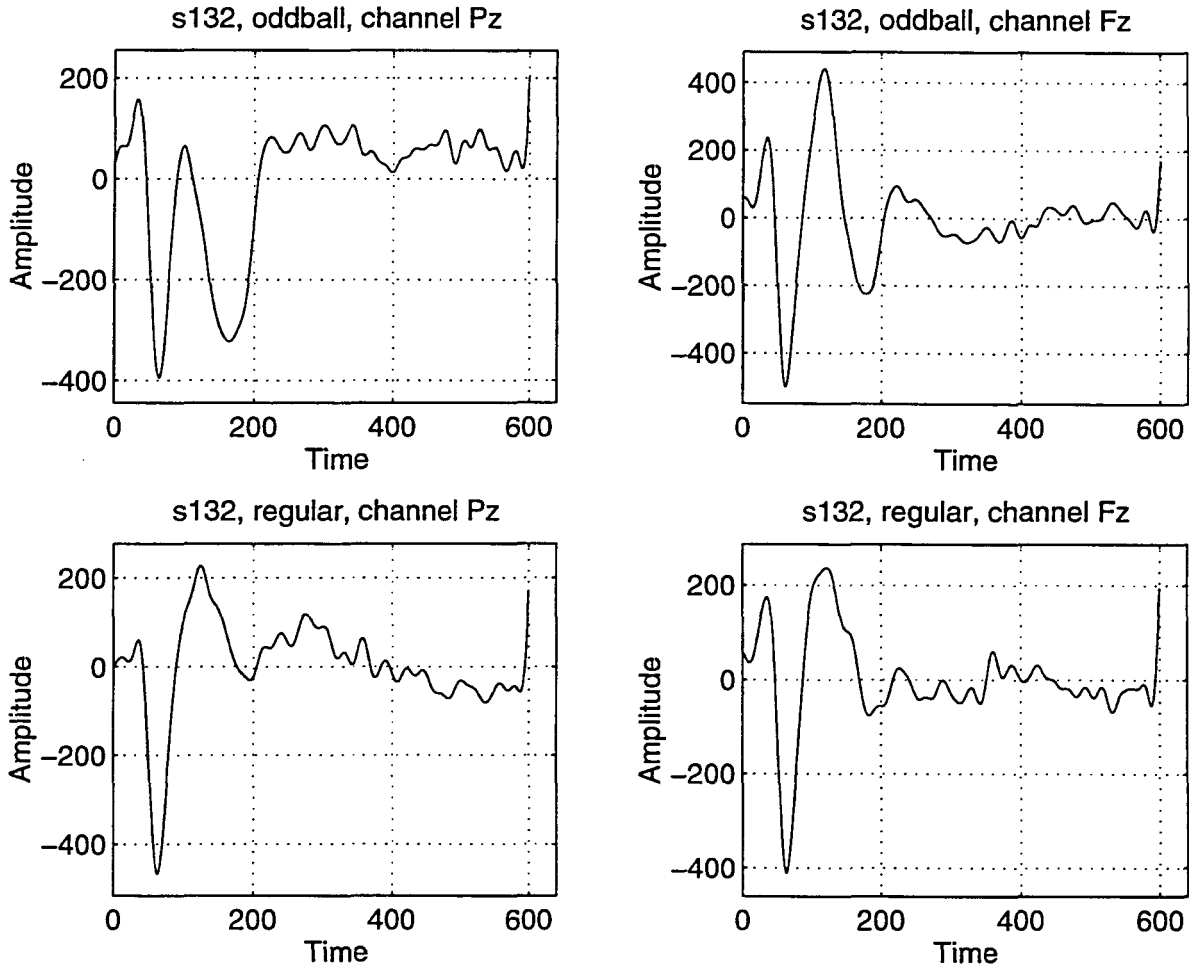

Figure A.16: Patient ID: s132, Normal

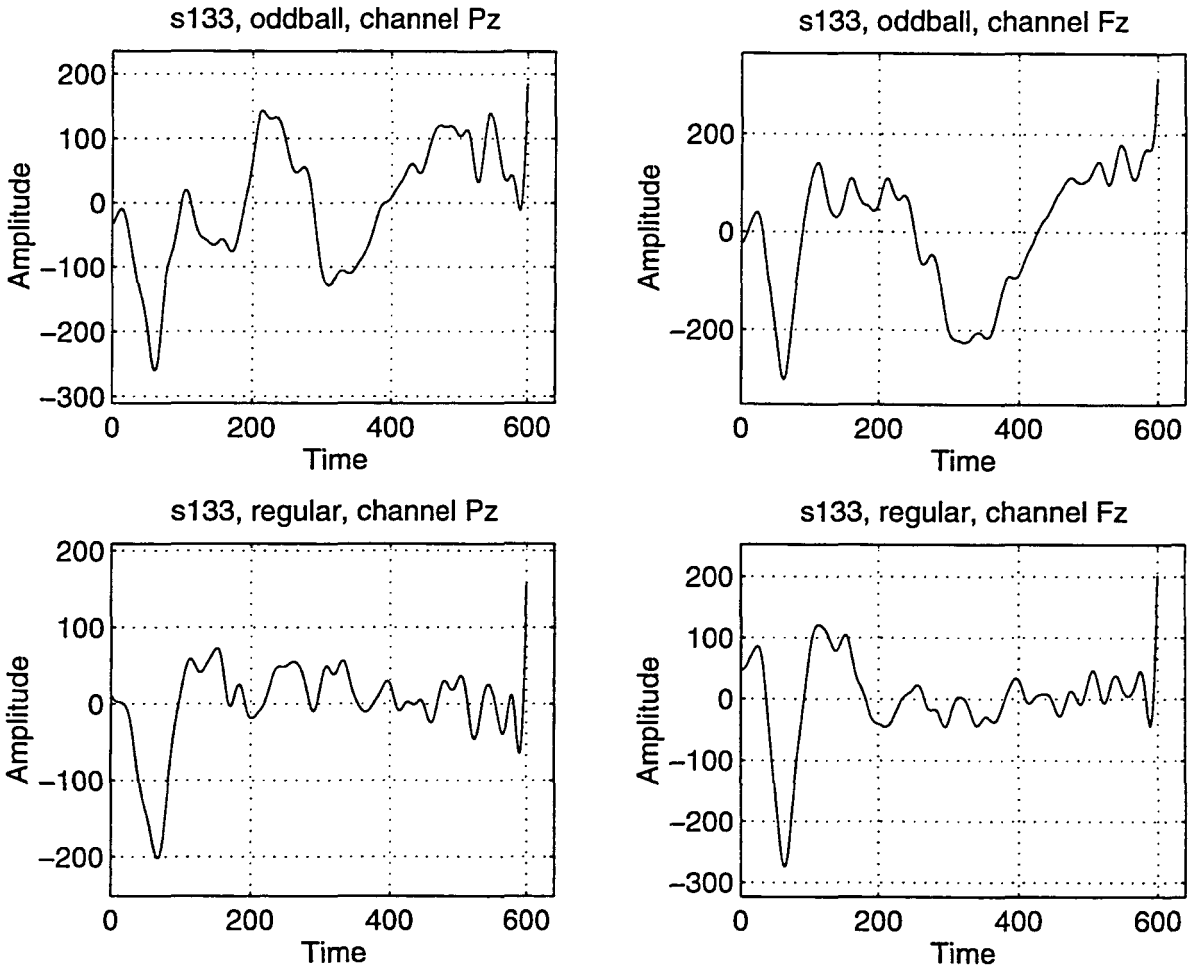

Figure A.17: Patient ID: s133, Normal

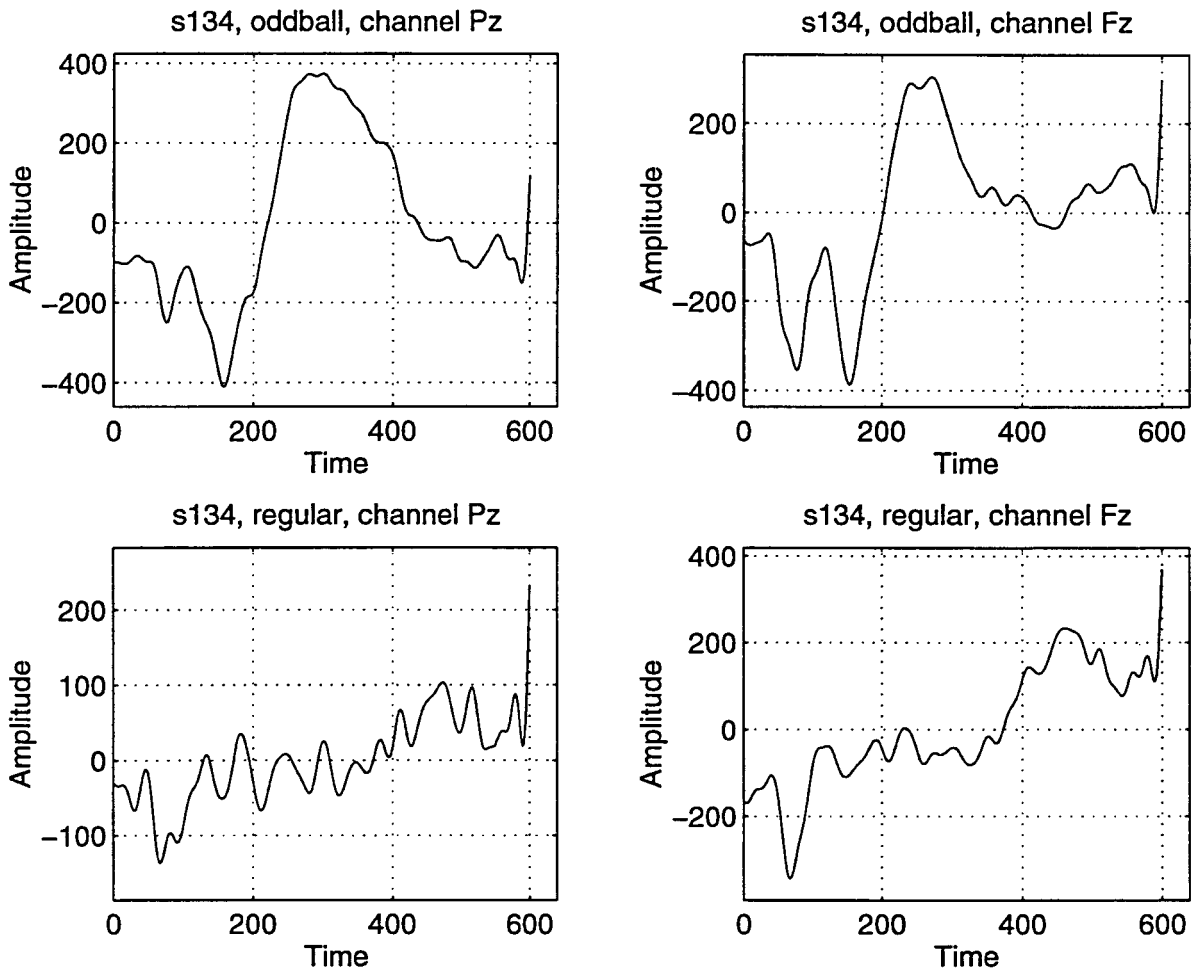

Figure A.18: Patient ID: s134, Normal

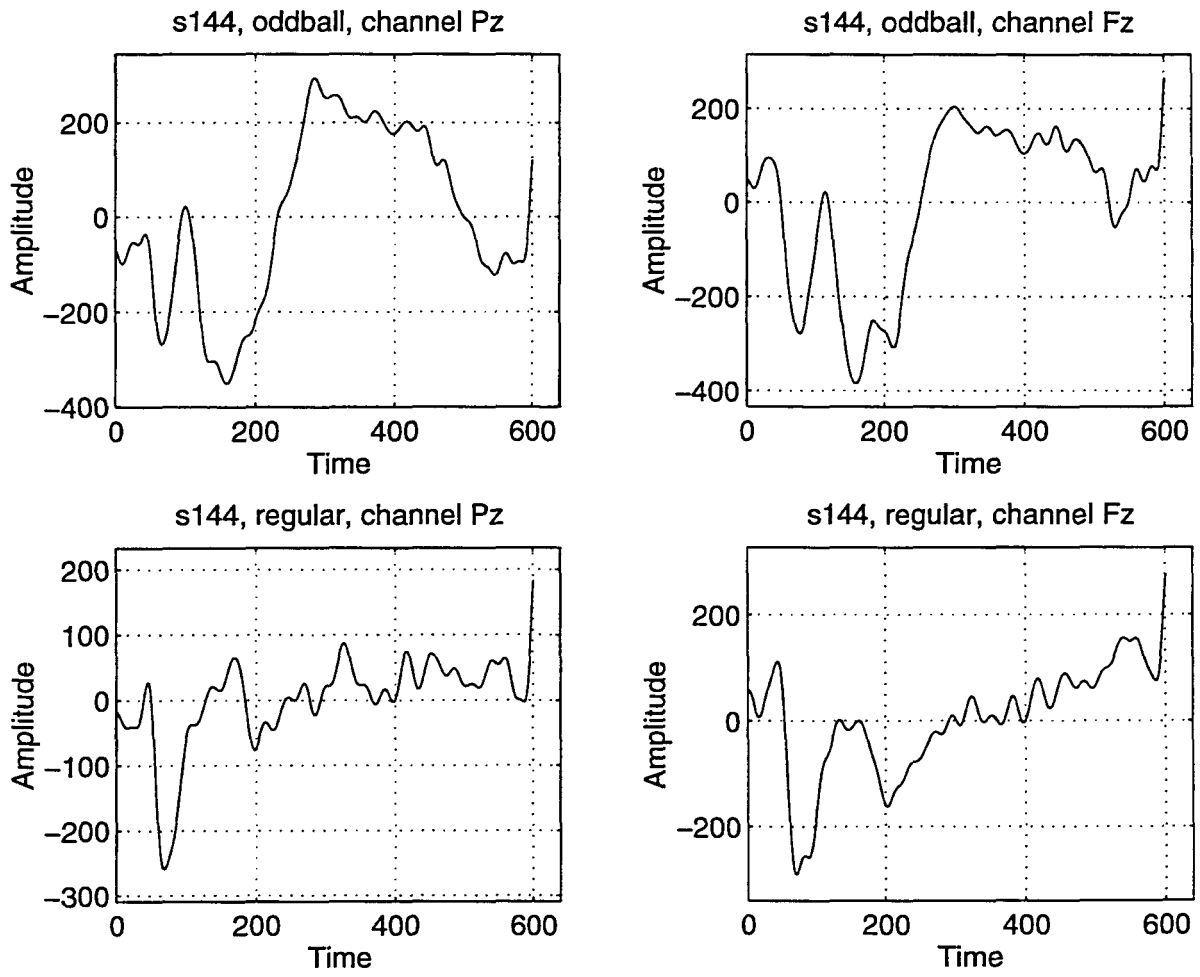

Figure A.19: Patient ID: s144, Normal

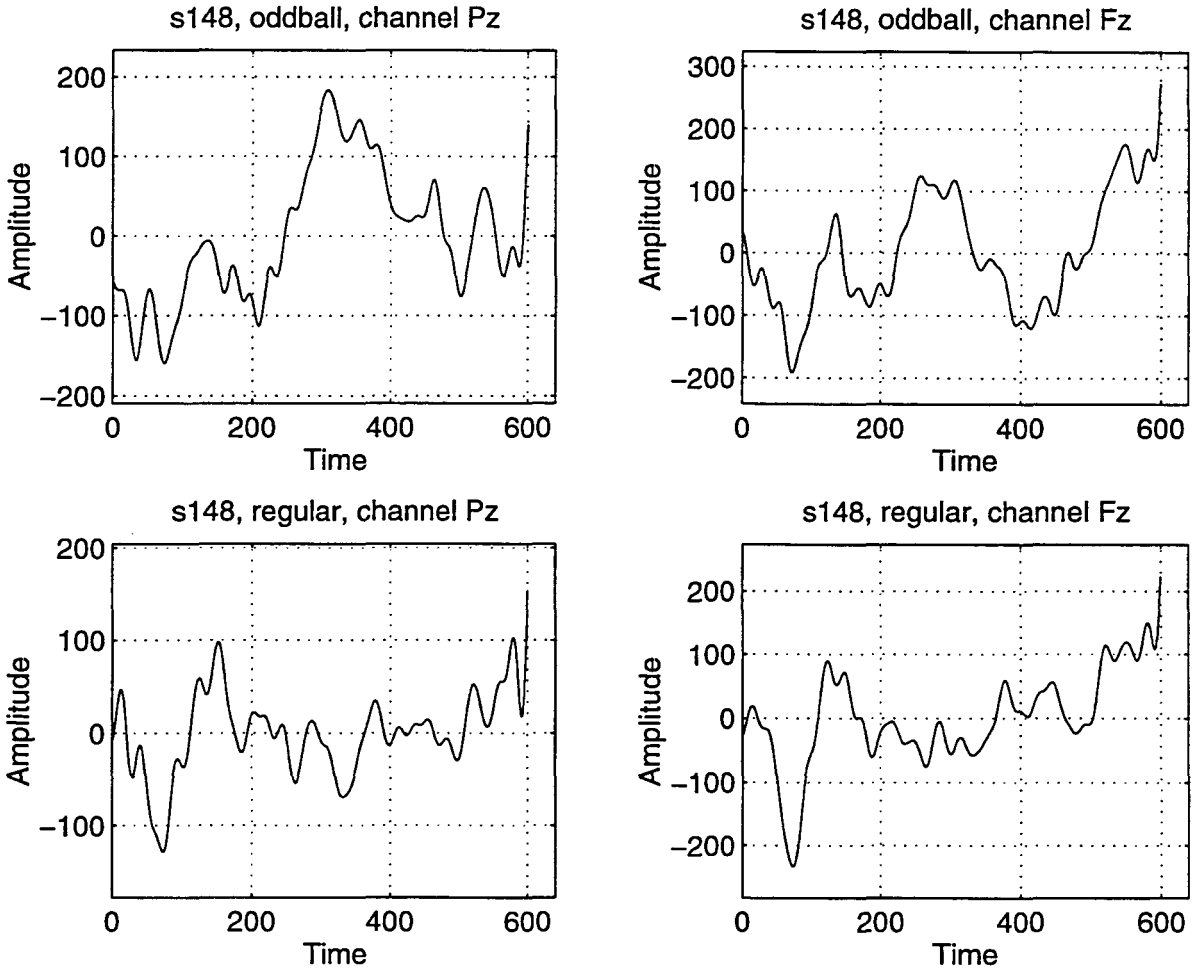

Figure A.20: Patient ID: s148, Normal

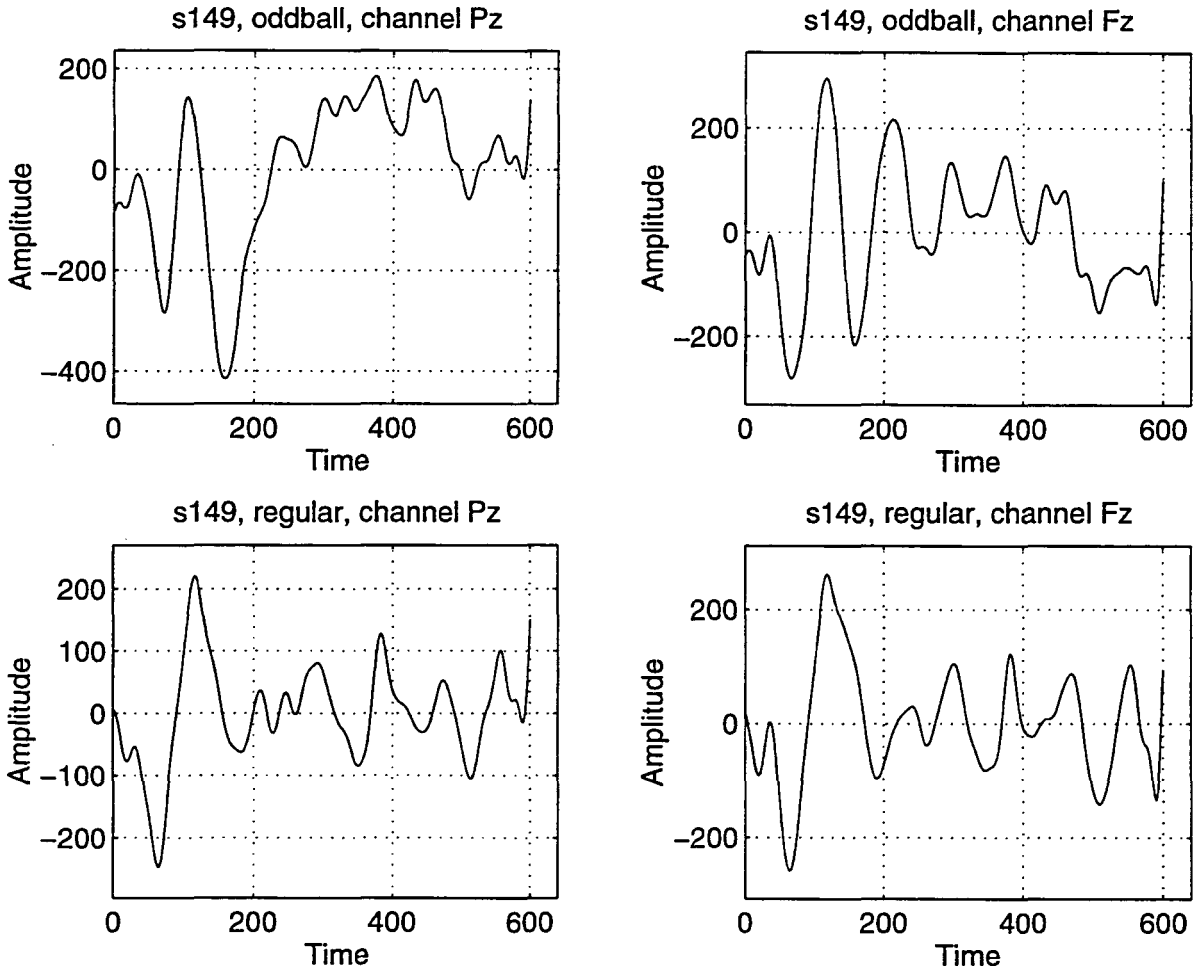

Figure A.21: Patient ID: s149, Alzheimer's

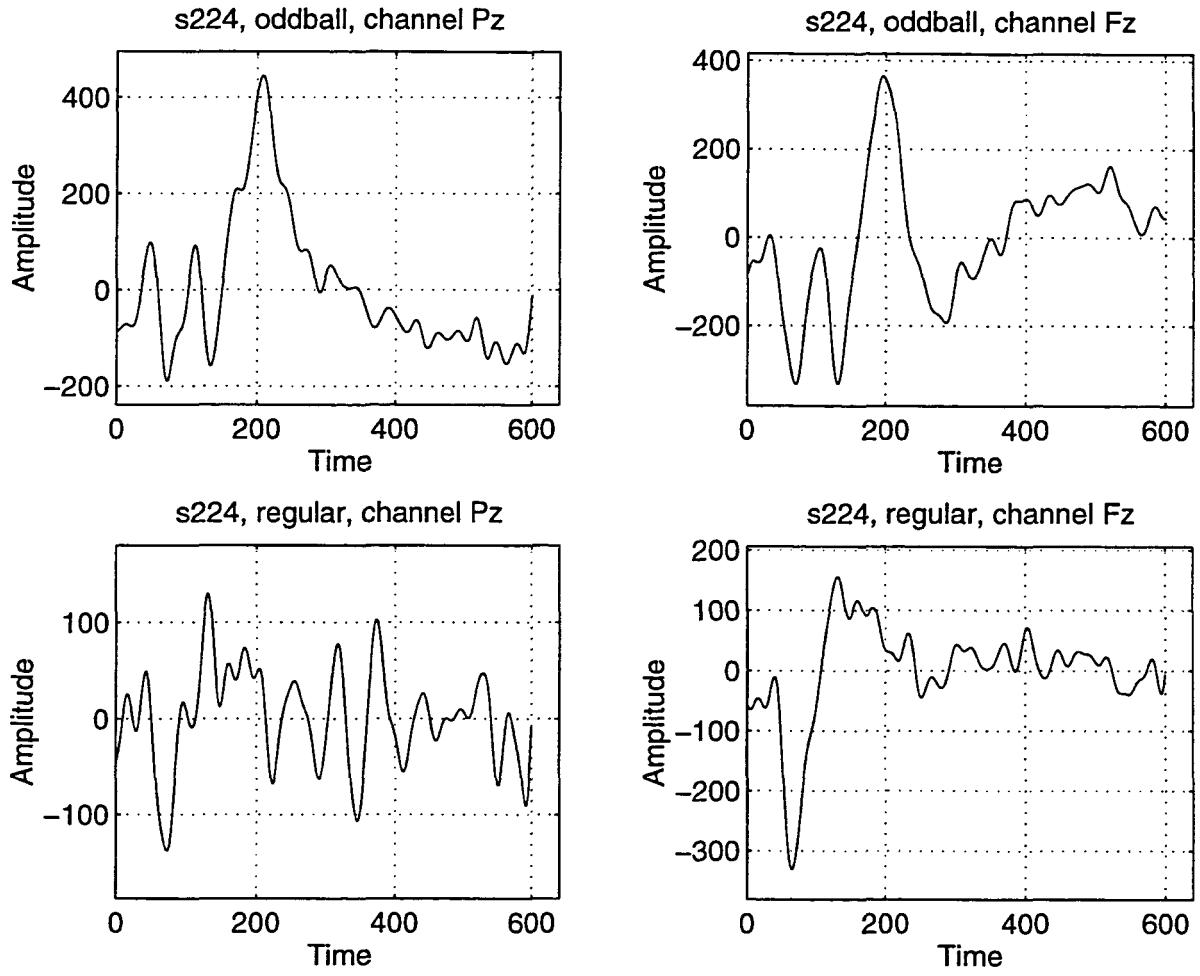

Figure A.22: Patient ID: s224, Normal

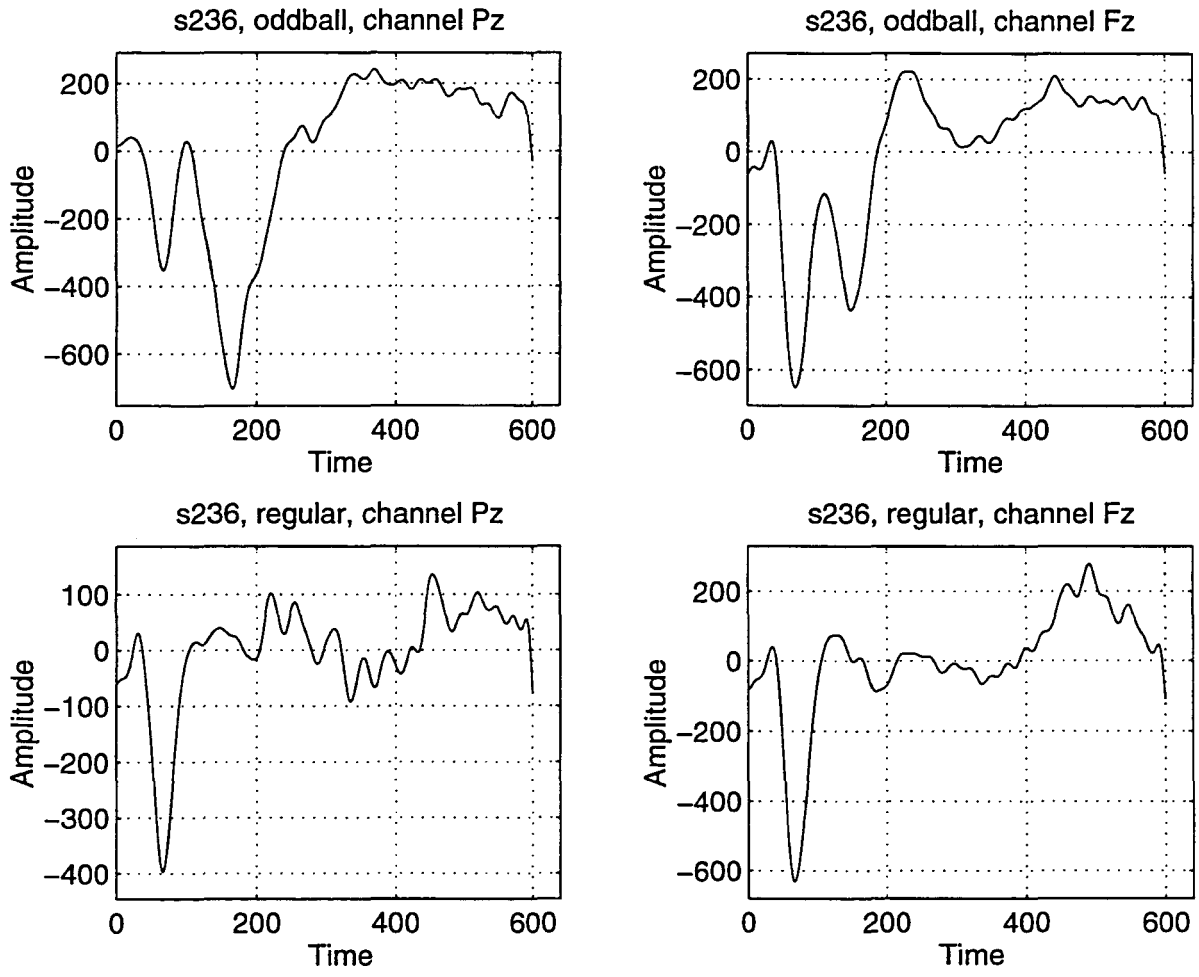

Figure A.23: Patient ID: s236, Normal

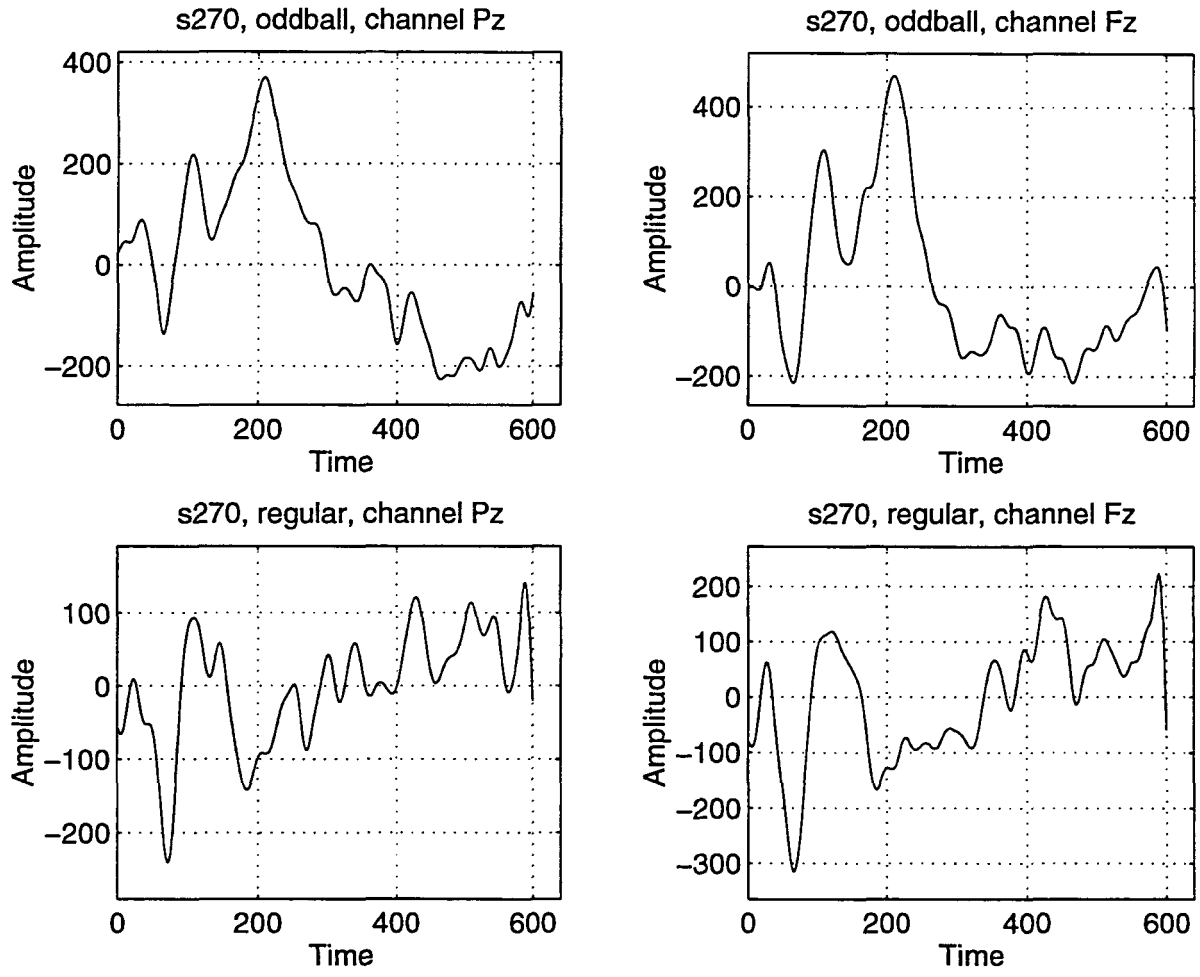

Figure A.24: Patient ID: s270, Alzheimer's

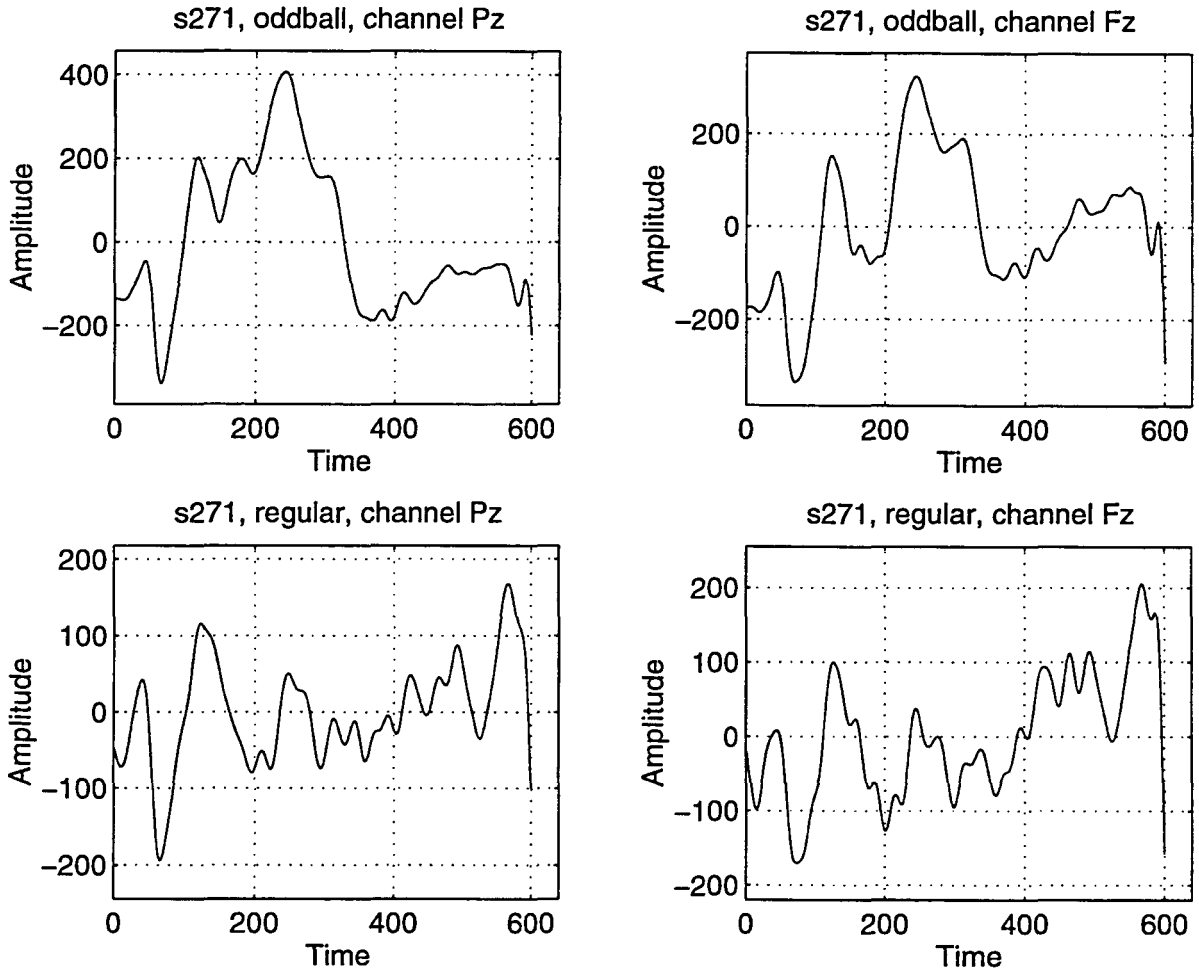

Figure A.25: Patient ID: s271, Alzheimer's

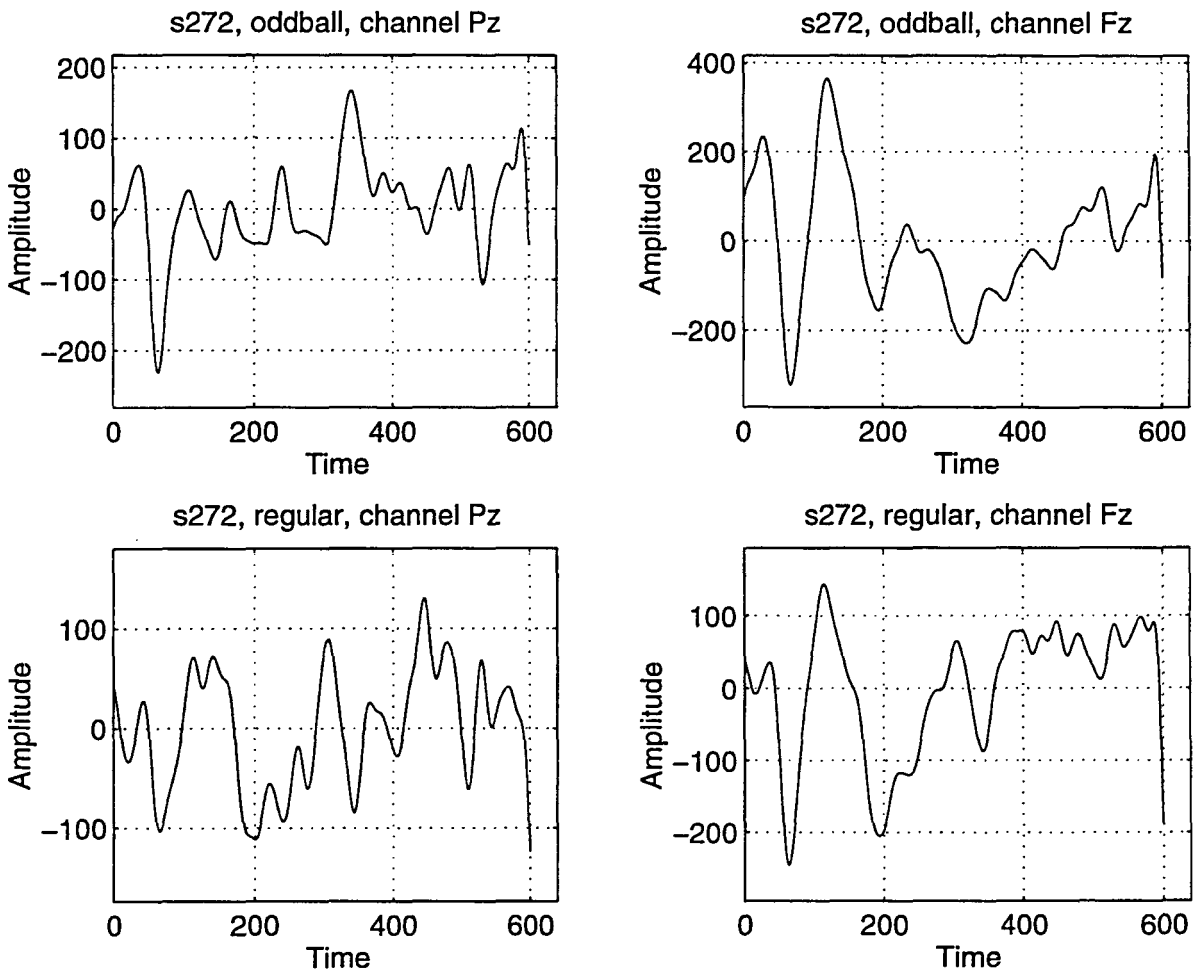

Figure A.26: Patient ID: s272, Alzheimer's

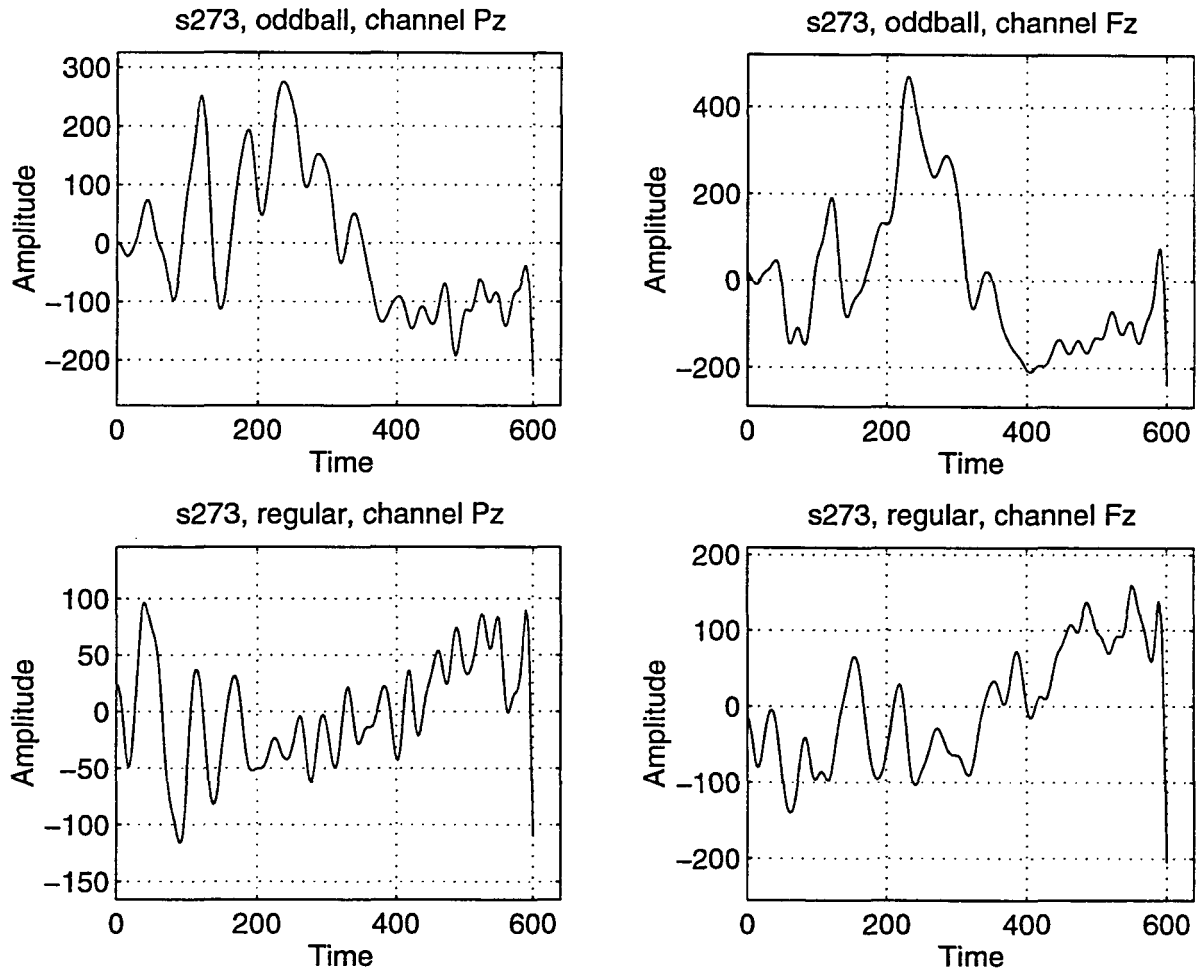

Figure A.27: Patient ID: s273, Alzheimer's

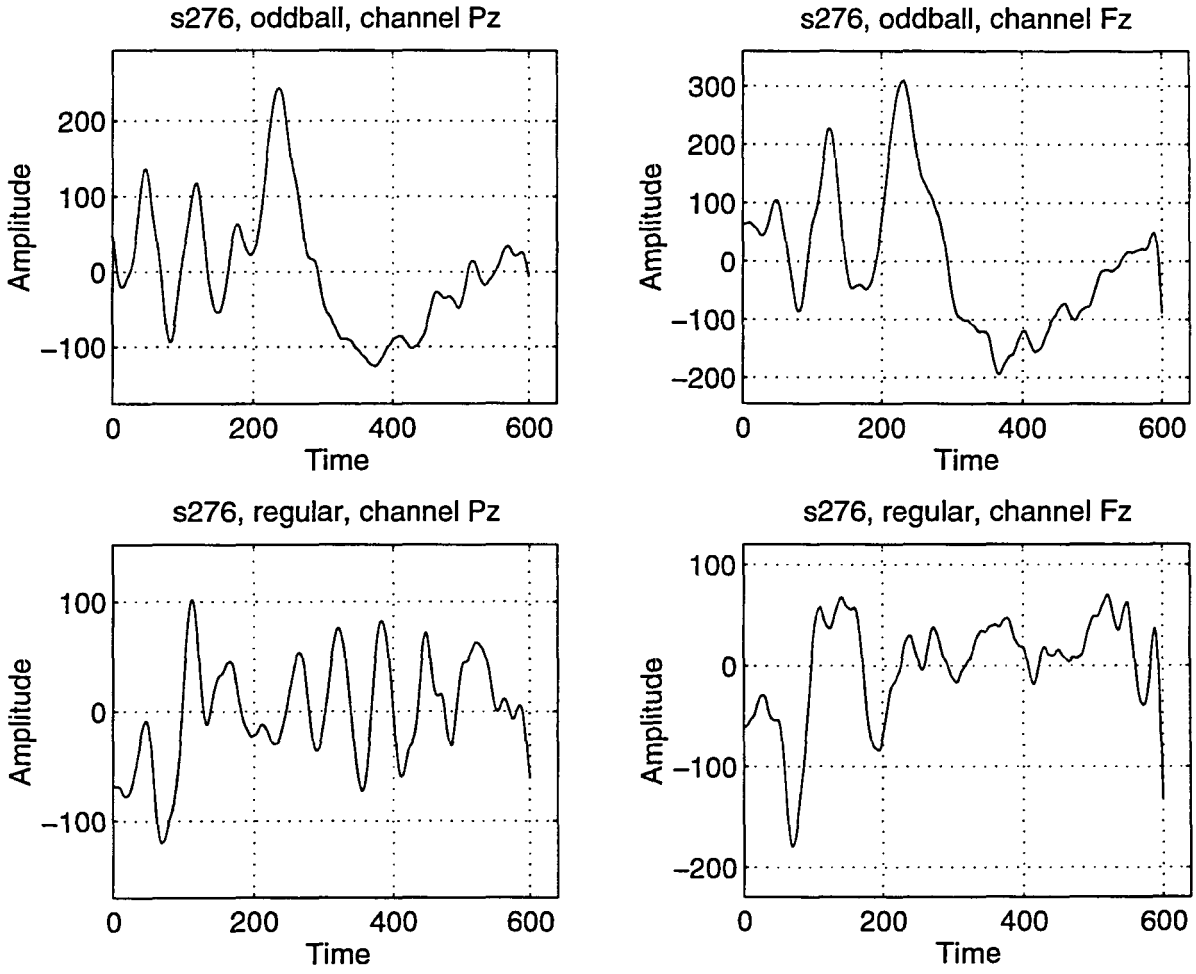

Figure A.28: Patient ID: s276, Alzheimer's

APPENDIX B. CONTINUOUS WAVELET TRANSFORMS OF THE ERPs

This appendix presents the 3-D plots of the continuous wavelet transforms of the four signals that are acquired from each patient, namely, responses to oddball and regular tones recorded from Pz and Fz. The time-frequency representations are calculated at 80 frequency and 120 translation values. The TFRs corresponding to the first 5 frequency values are removed since they did not contain any information. The plots given in this appendix are the sampled versions of these TFRs. Every other frequency value and every third translation value ars shown it the plots. The transformed signals are sampled to reduce the printing time of these plots, and the original transformed signals are available from the authour.

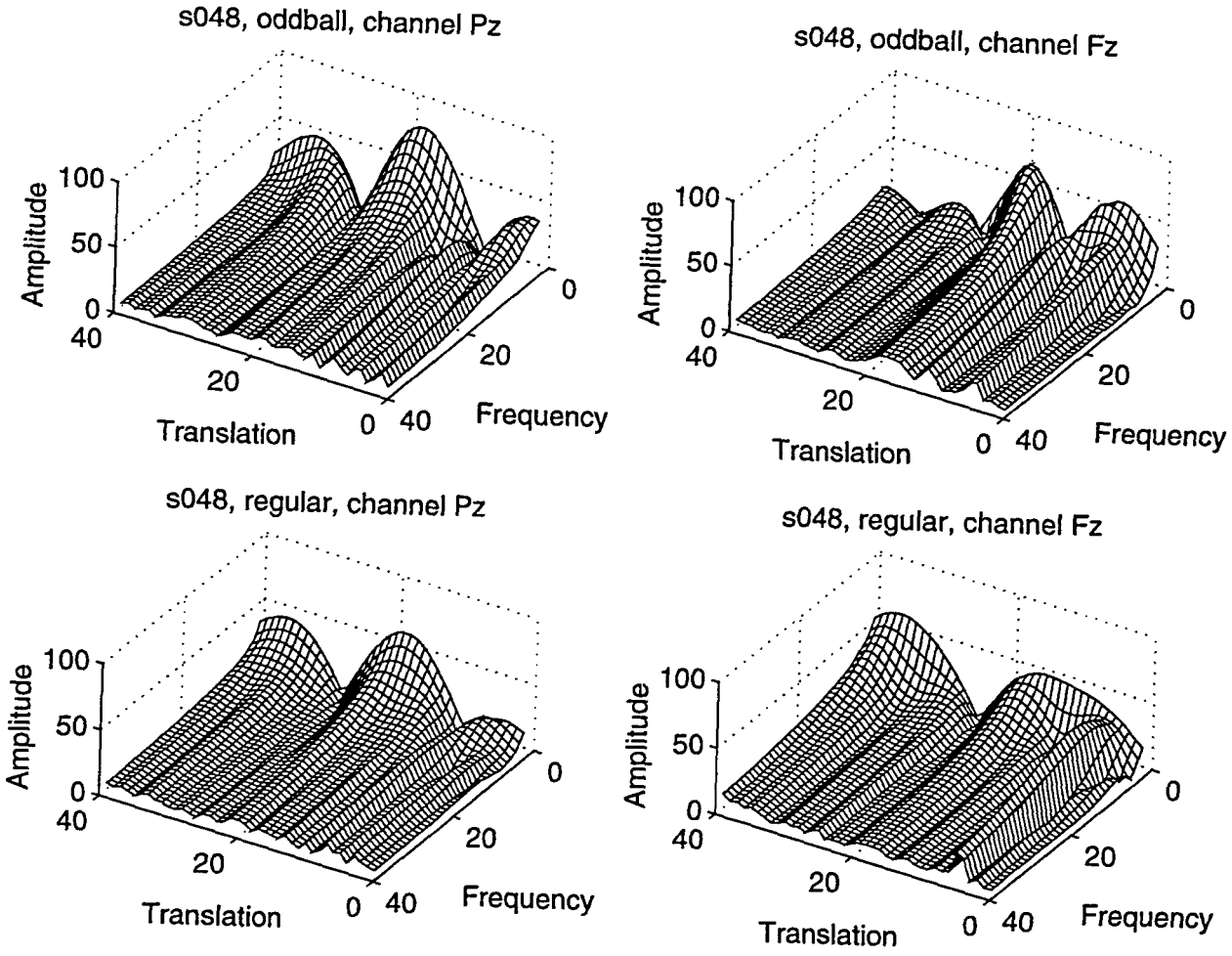

Figure B.1: Patient ID: s048, Alzheimer's

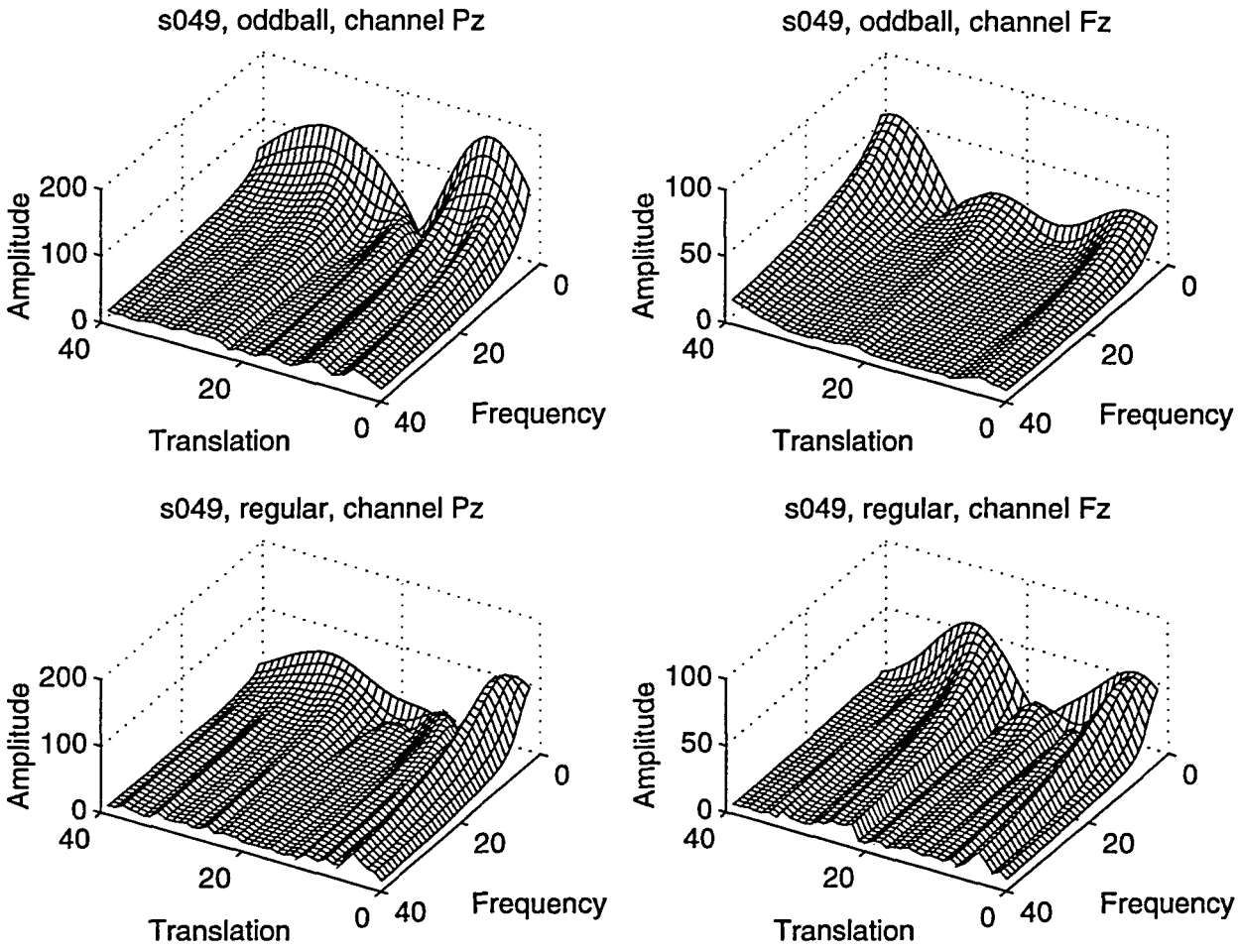

Figure B.2: Patient ID: s049, Alzheimer's

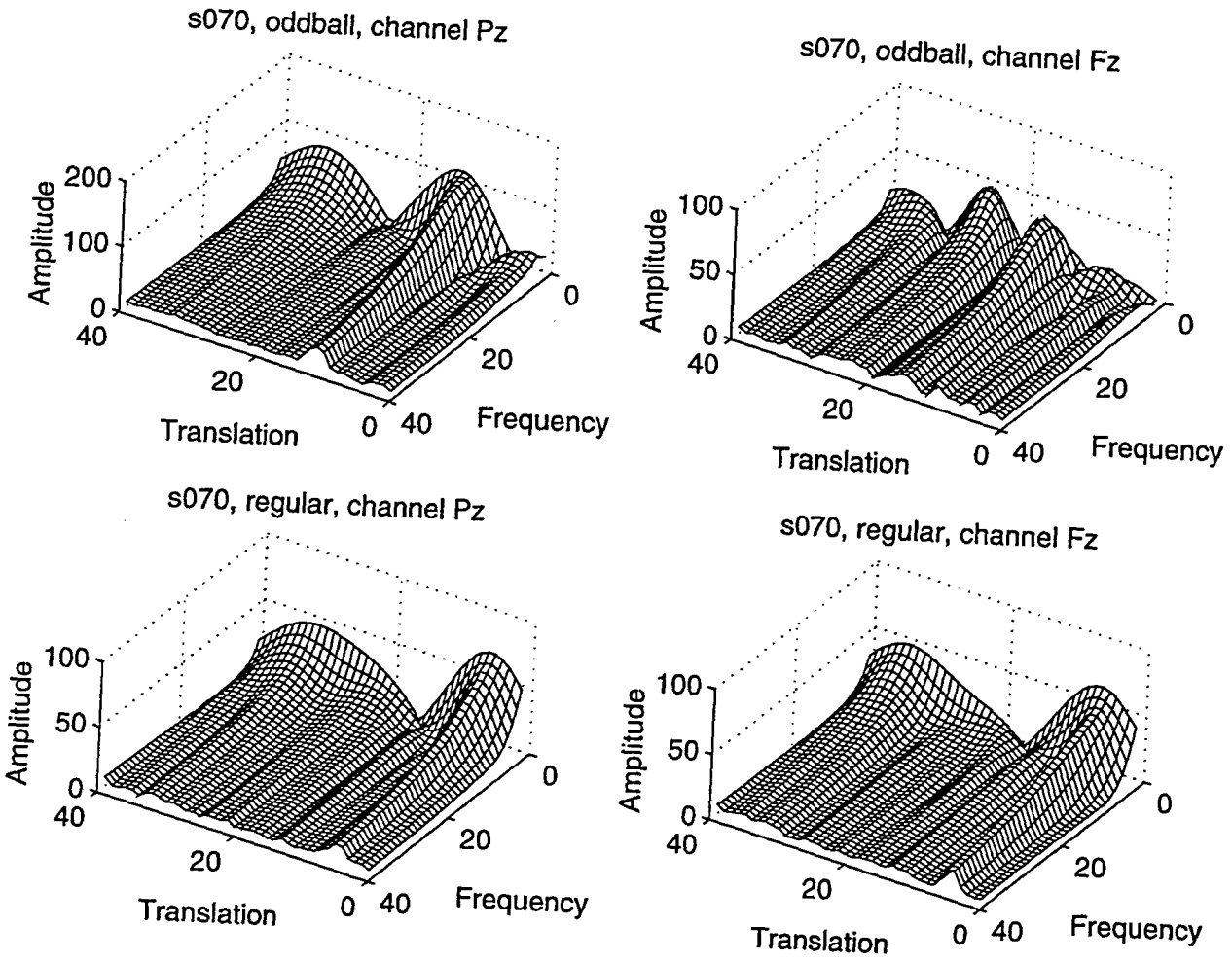

Figure B.3: Patient ID: s070, Normal

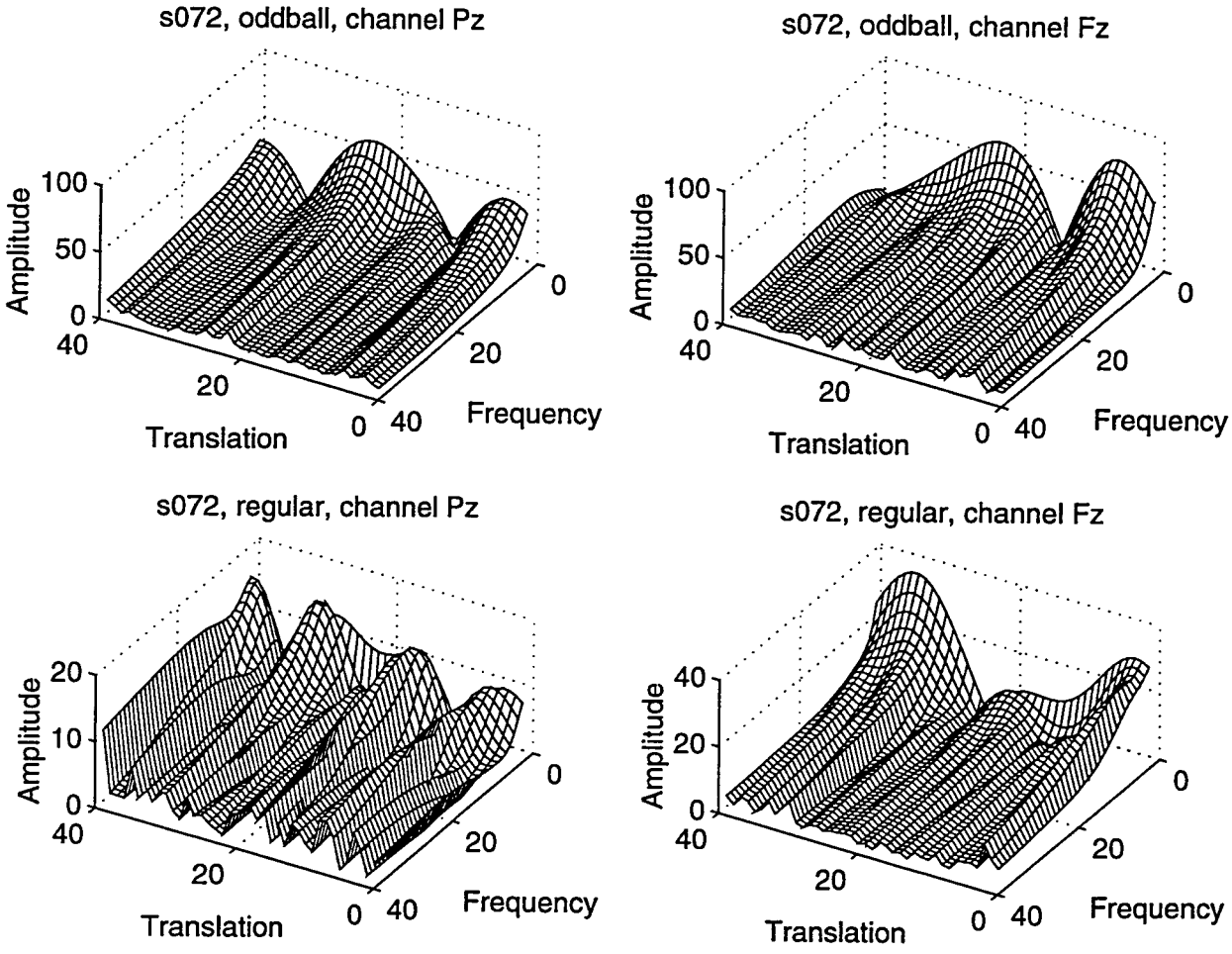

Figure B.4: Patient ID: s072, Alzheimer's

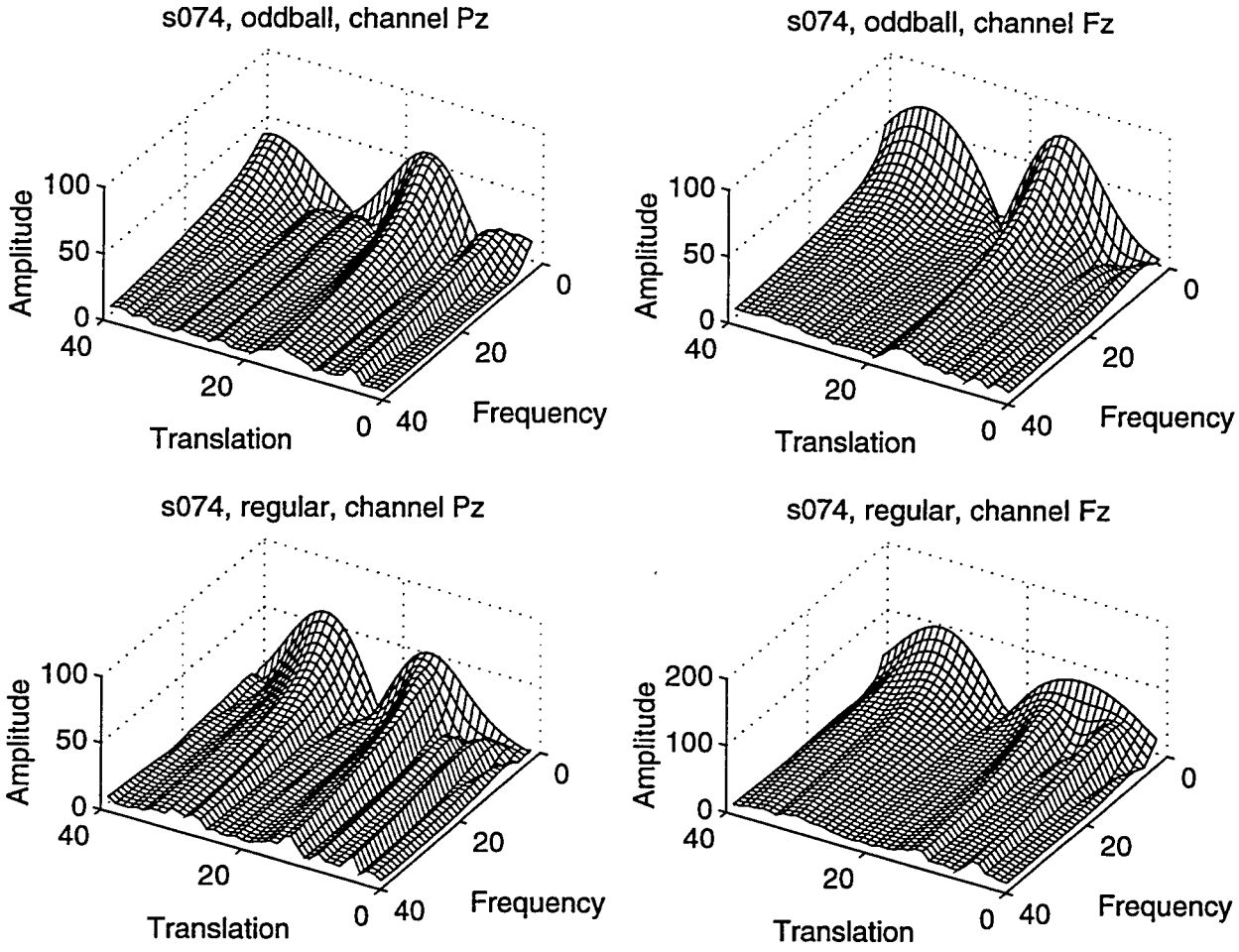

Figure B.5: Patient ID: s074, Normal

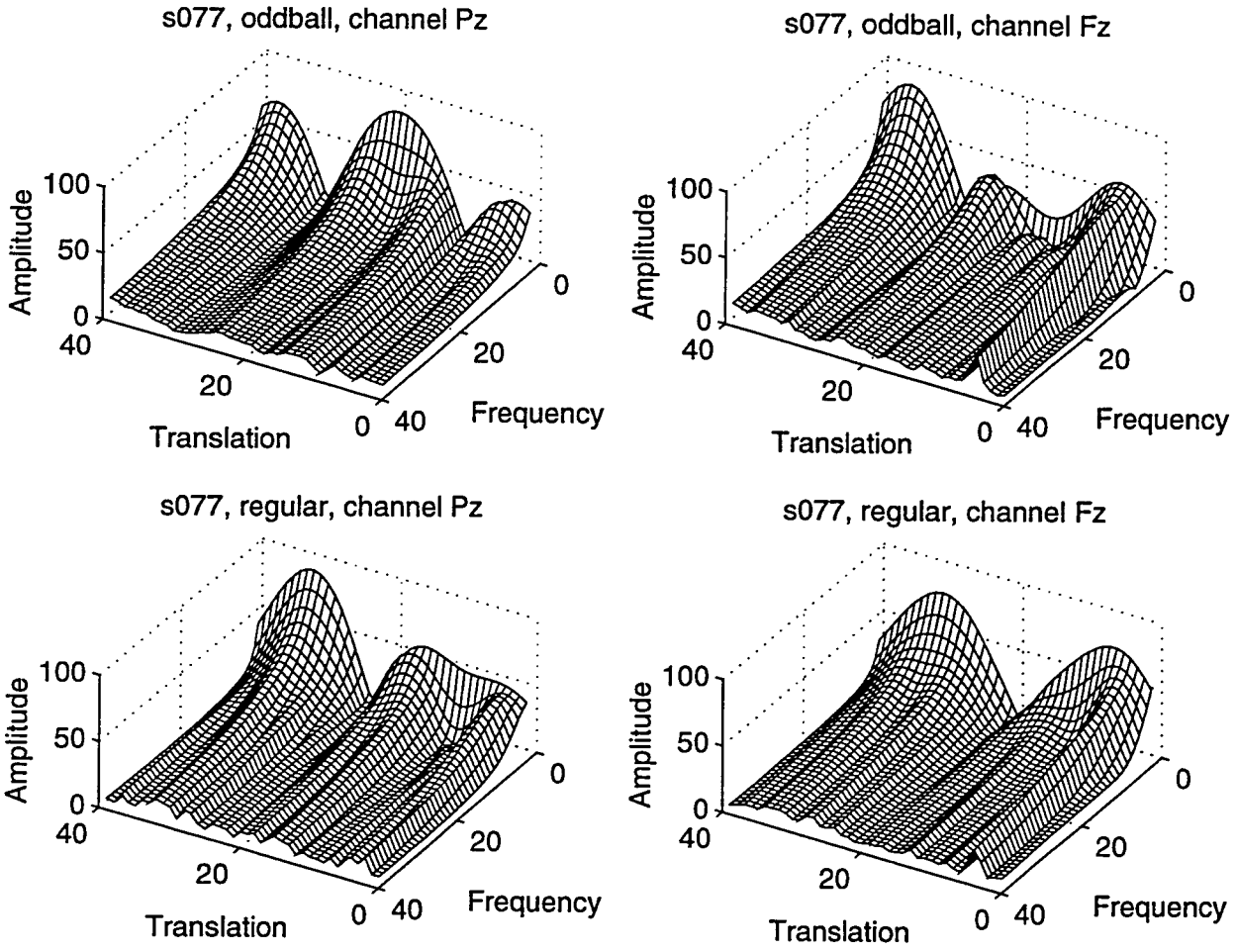

Figure B.6: Patient ID: s077, Normal

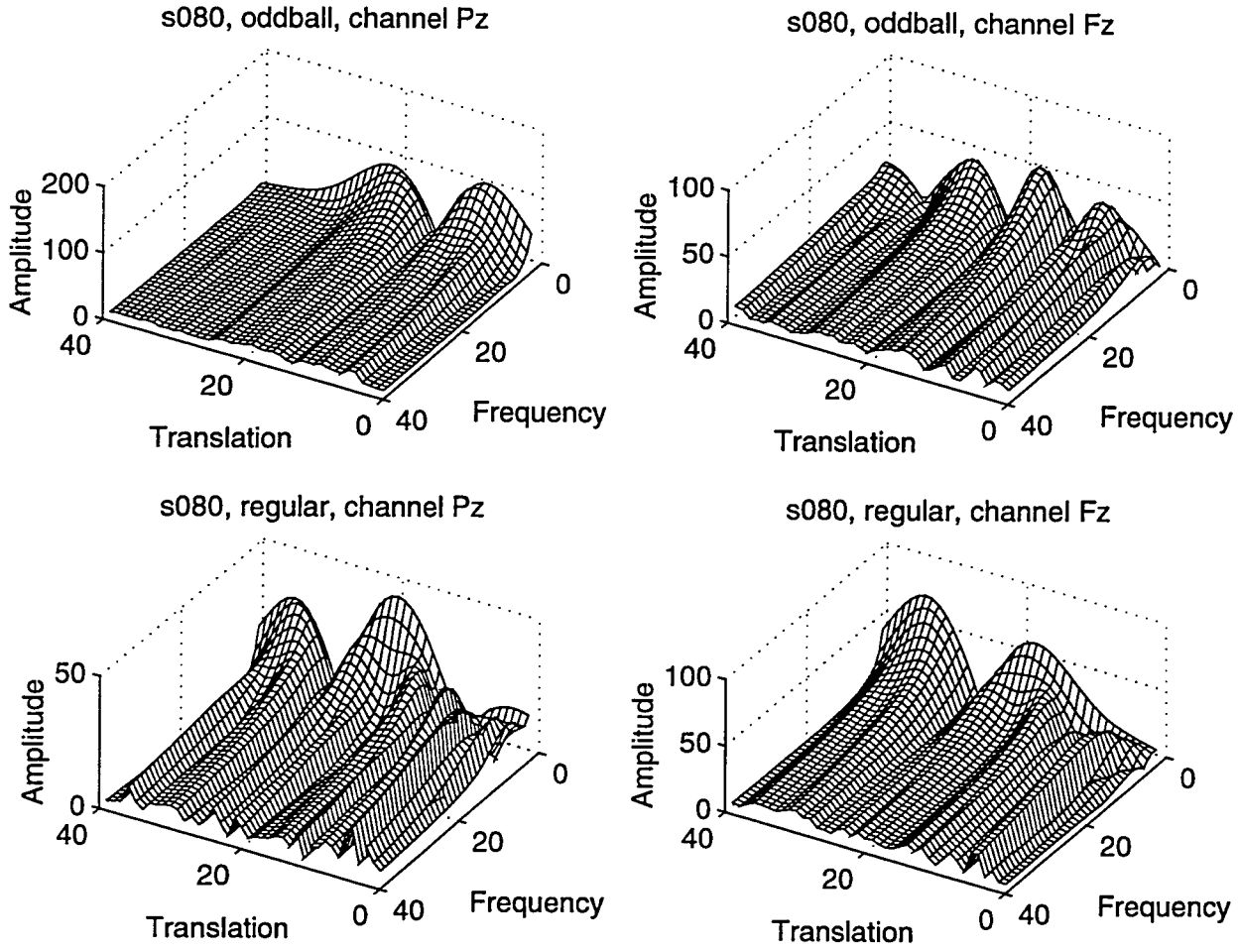

Figure B.7: Patient ID: s080, Normal

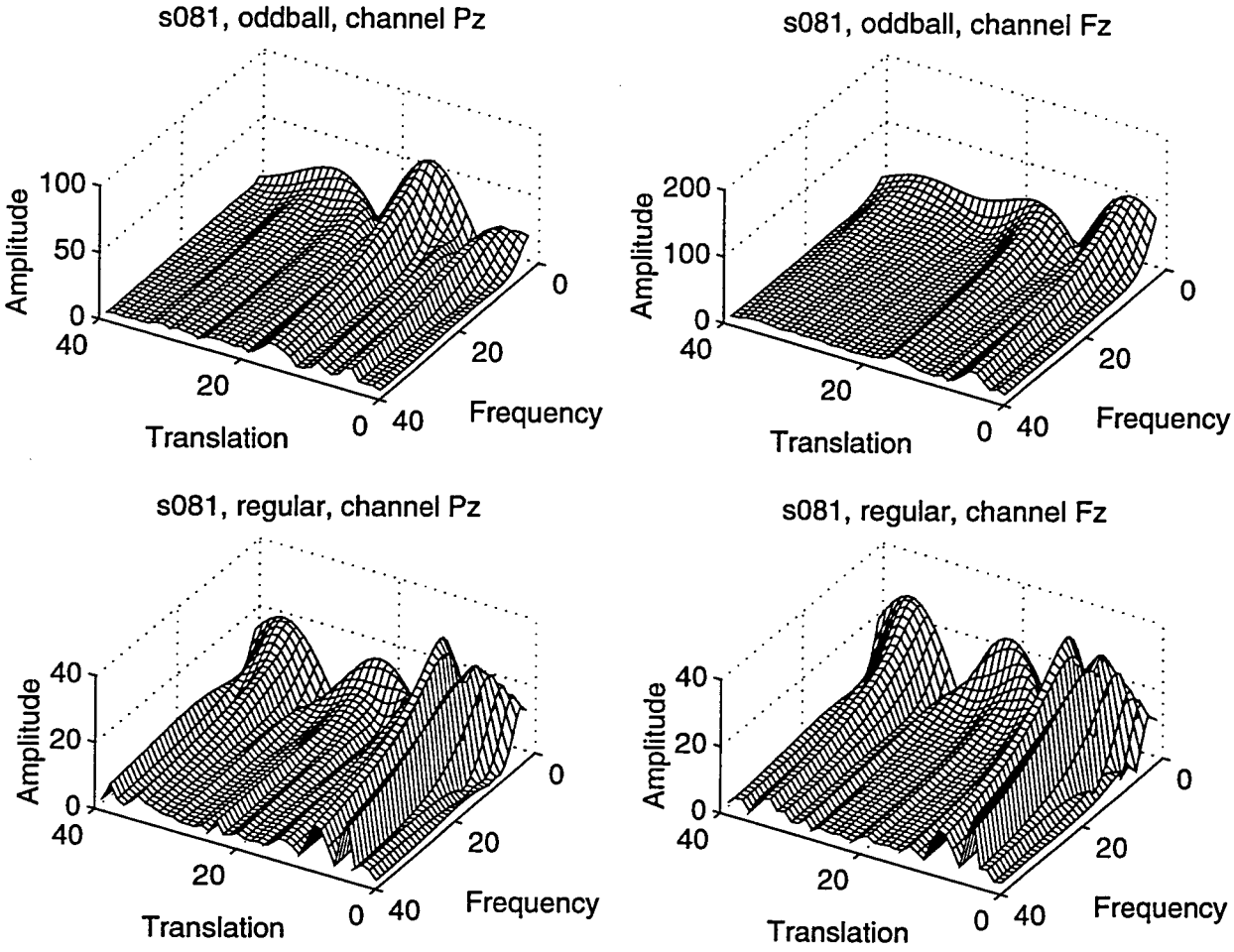

Figure B.8: Patient ID: s081, Normal

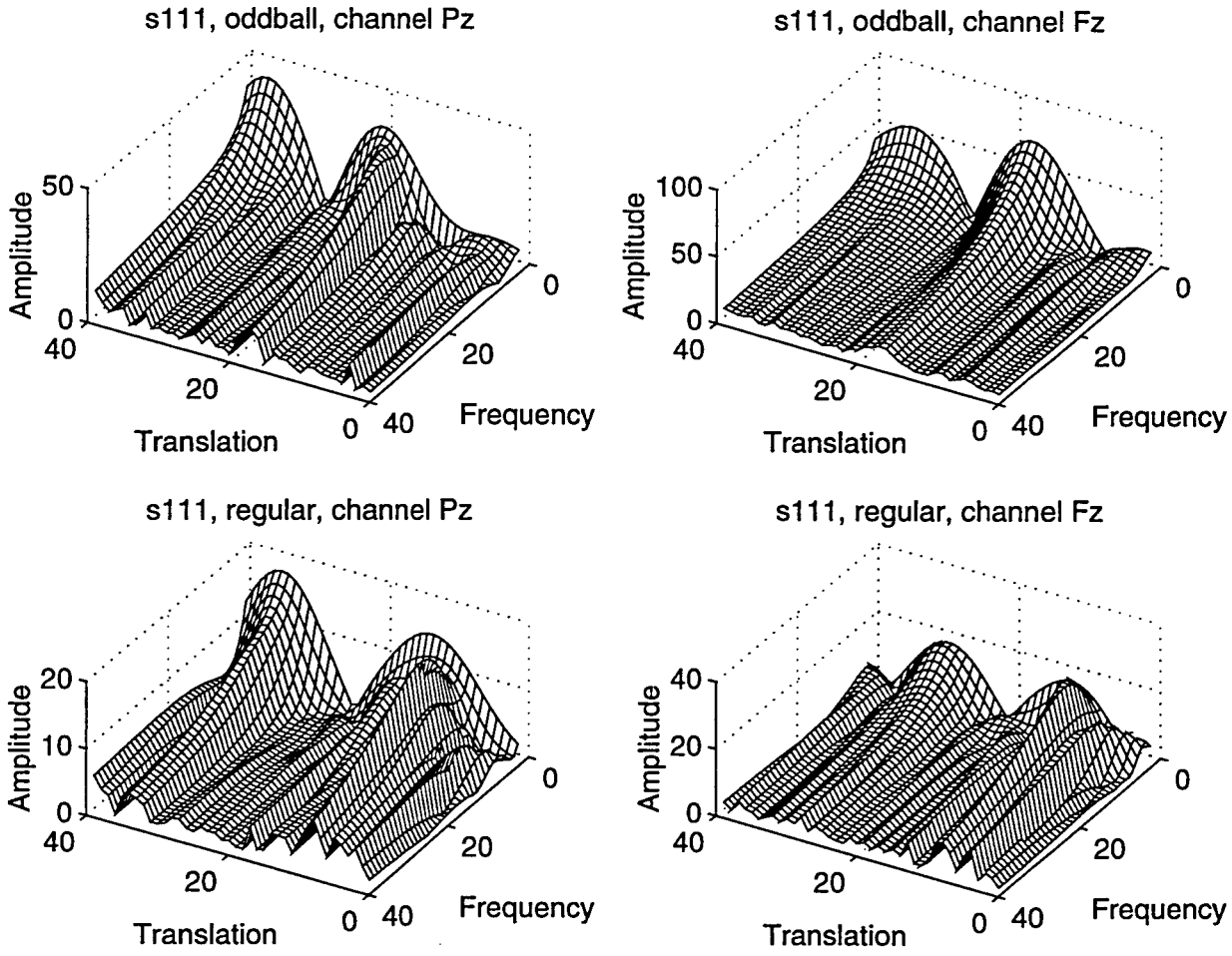

Figure B.9: Patient ID: s111, Alzheimer's

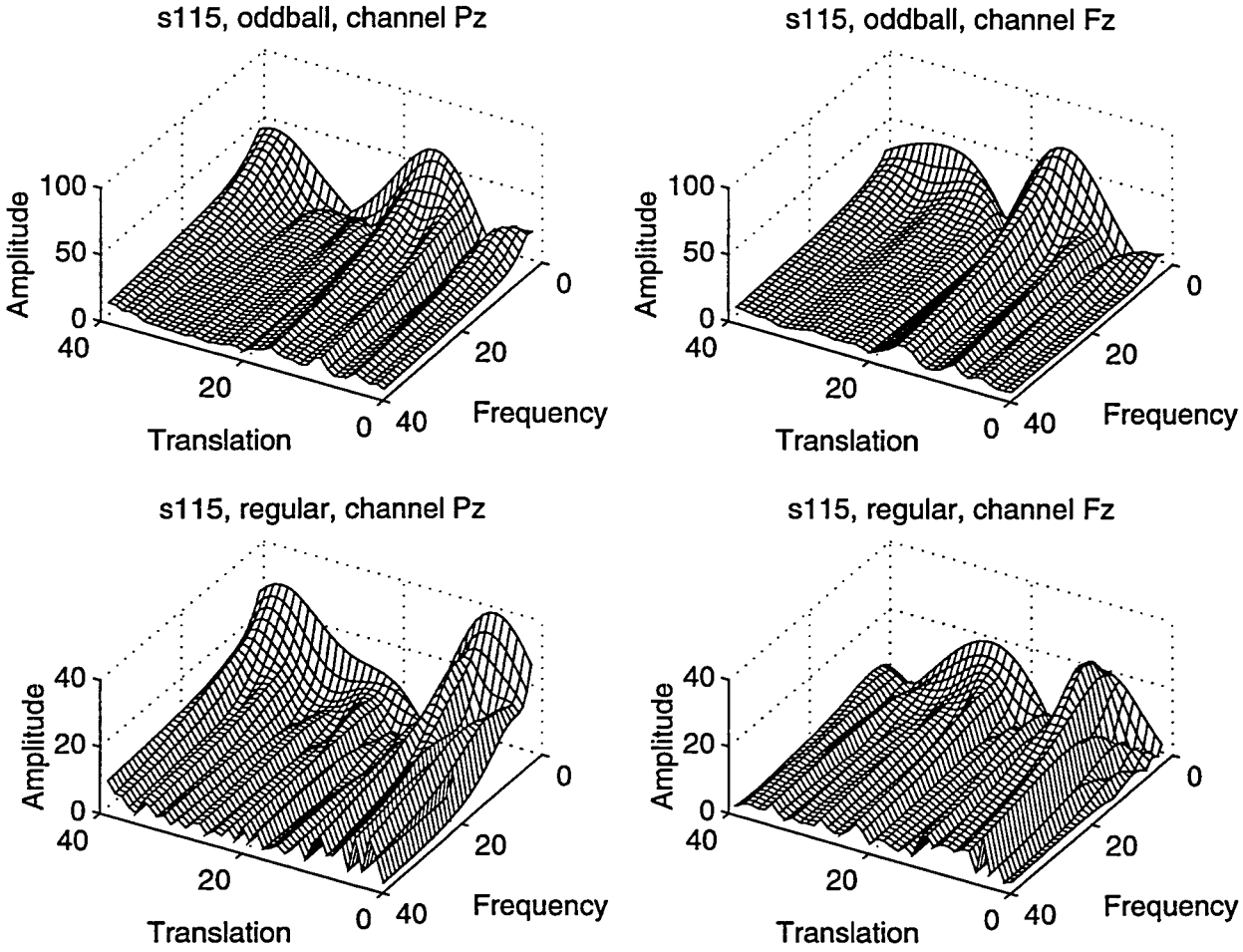

Figure B.10: Patient ID: s115, Alzheimer's

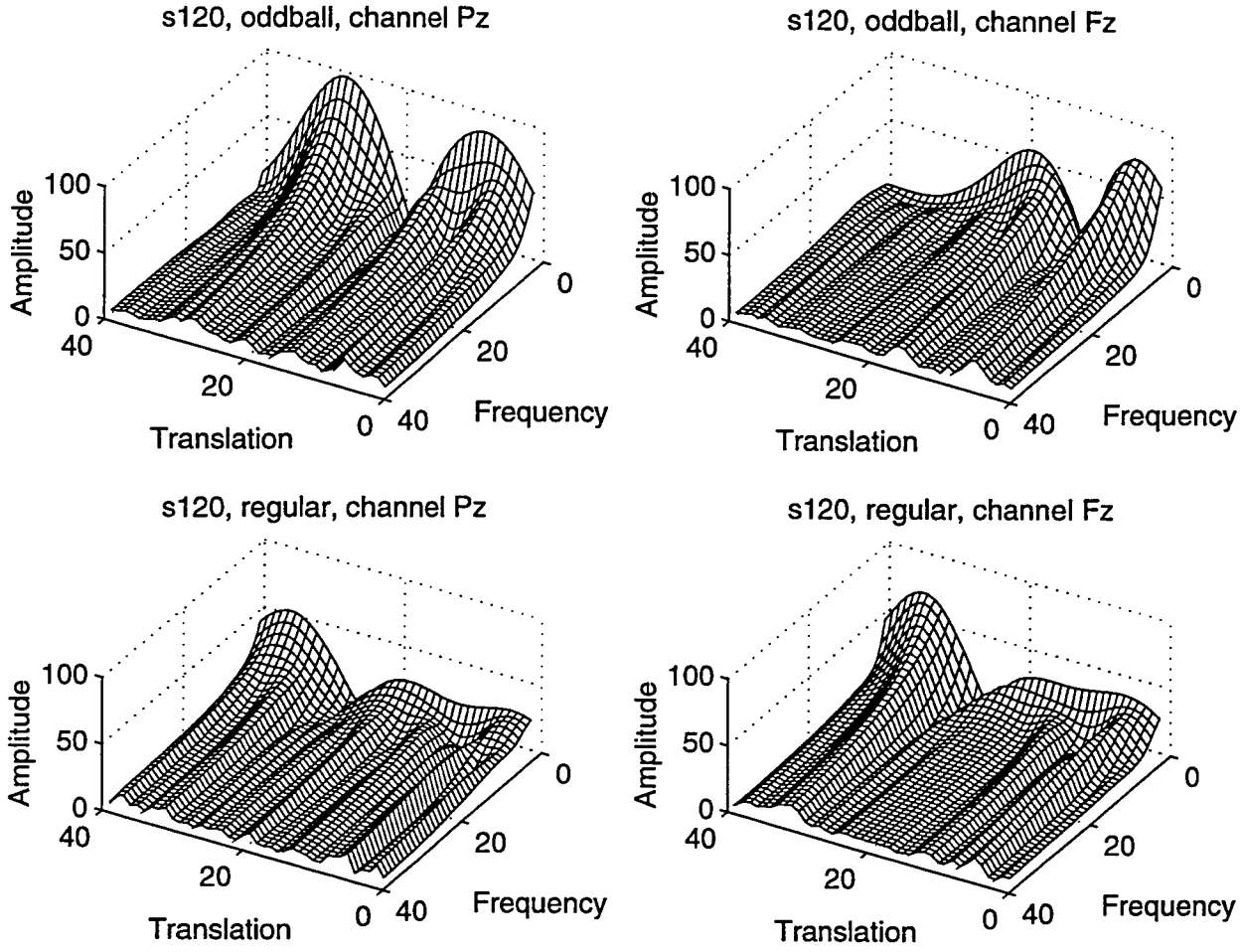

Figure B.11: Patient ID: s120, Alzheimer's

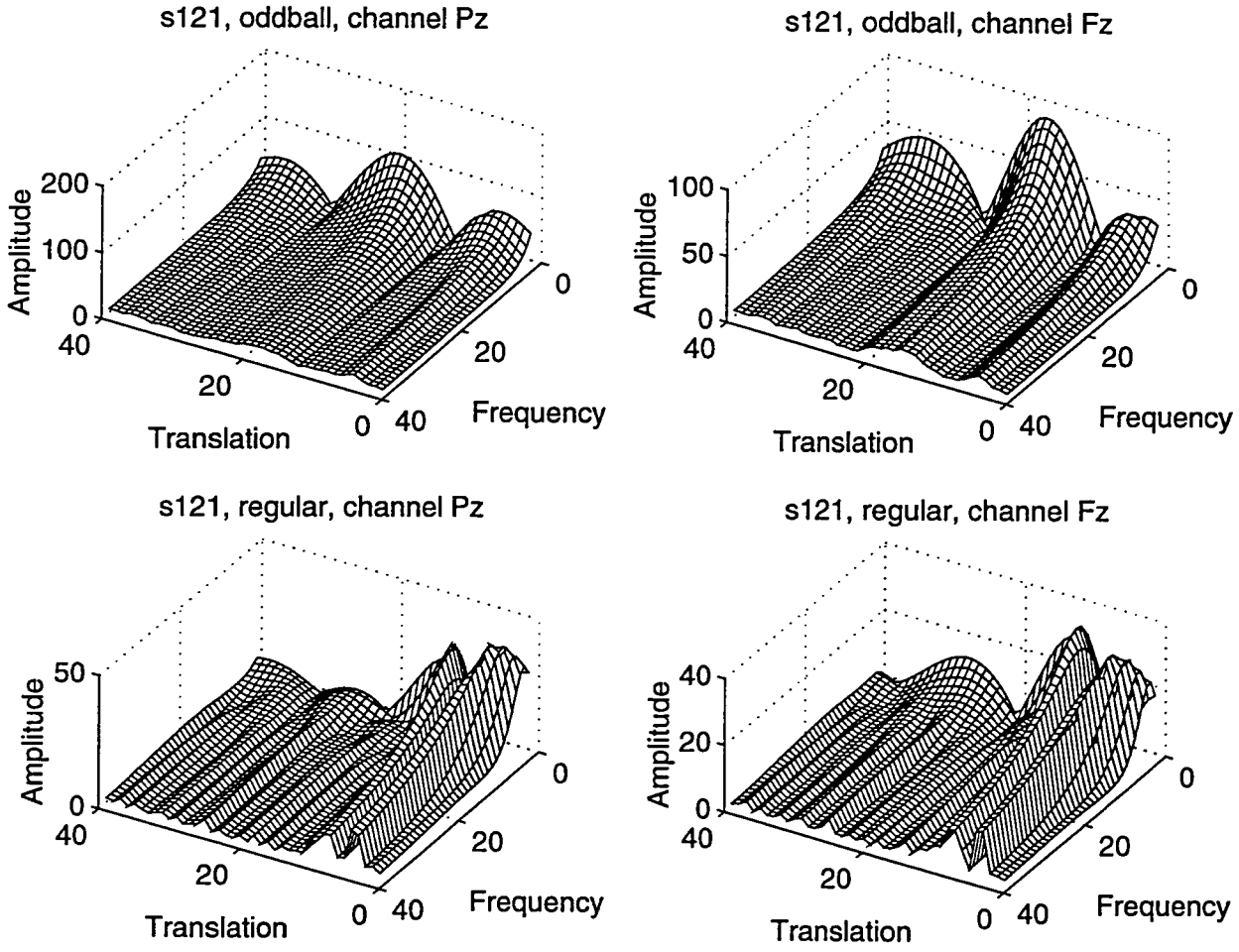

Figure B.12: Patient ID: s121, Normal

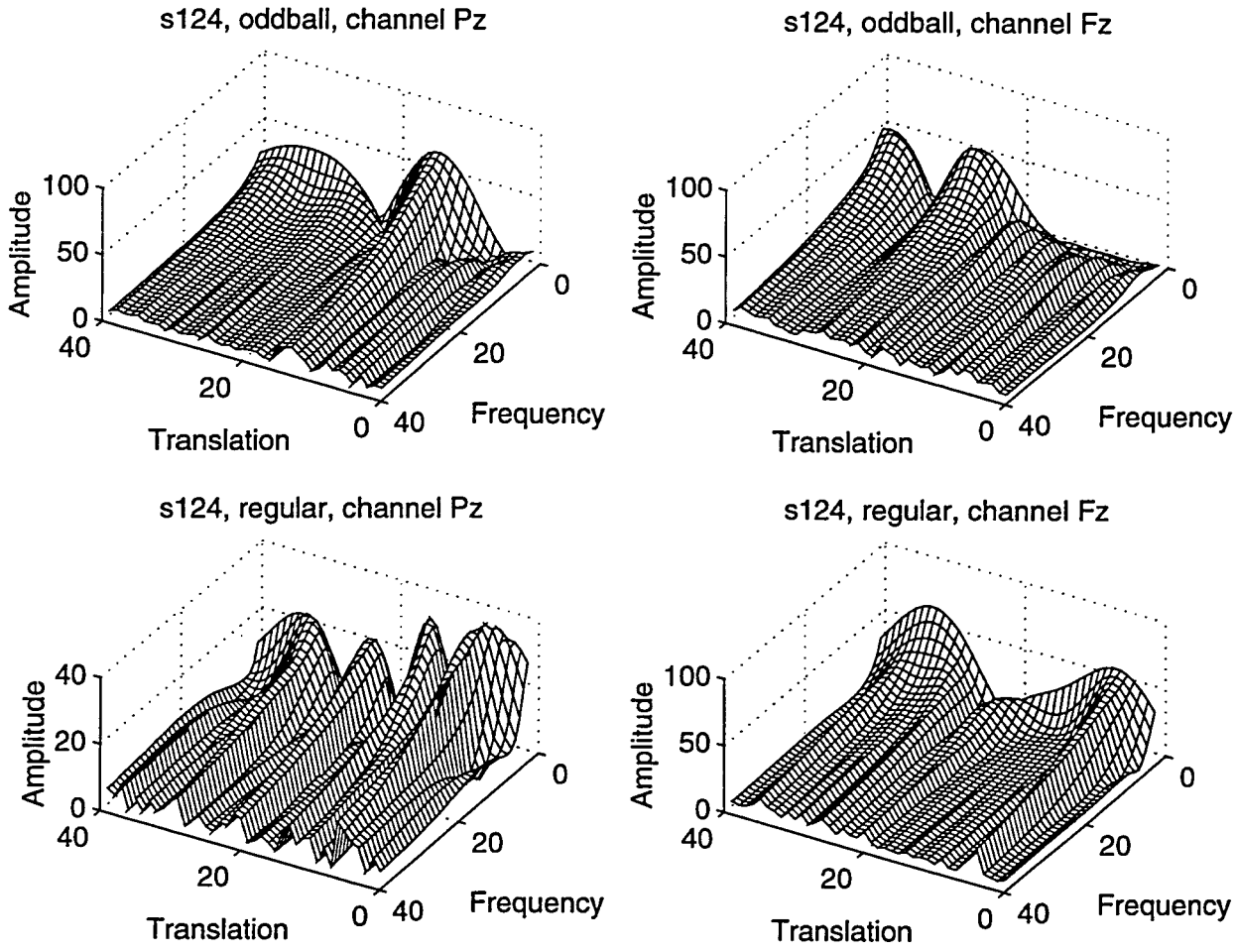

Figure B.13: Patient ID: s124, Normal

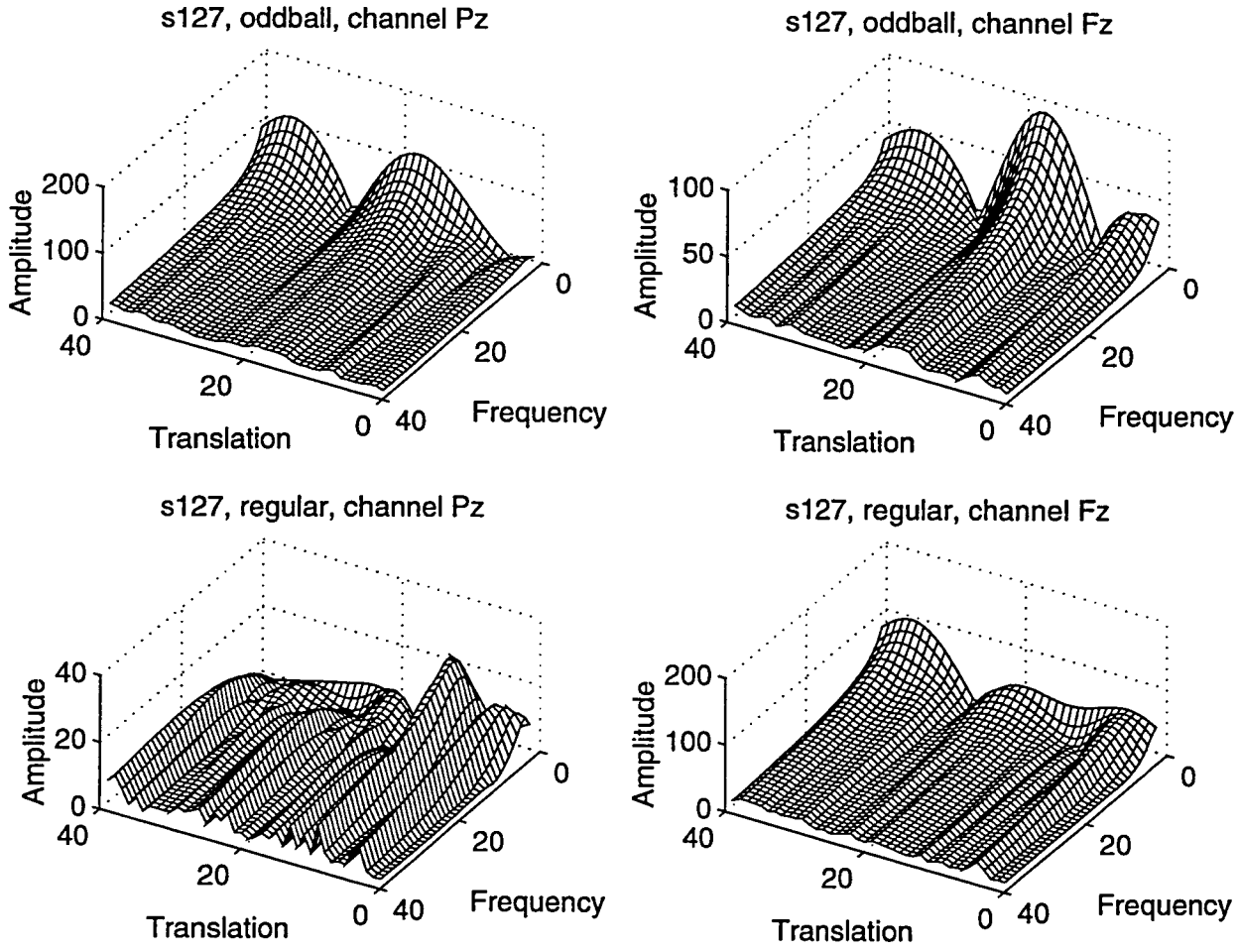

Figure B.14: Patient ID: s127, Alzheimer's

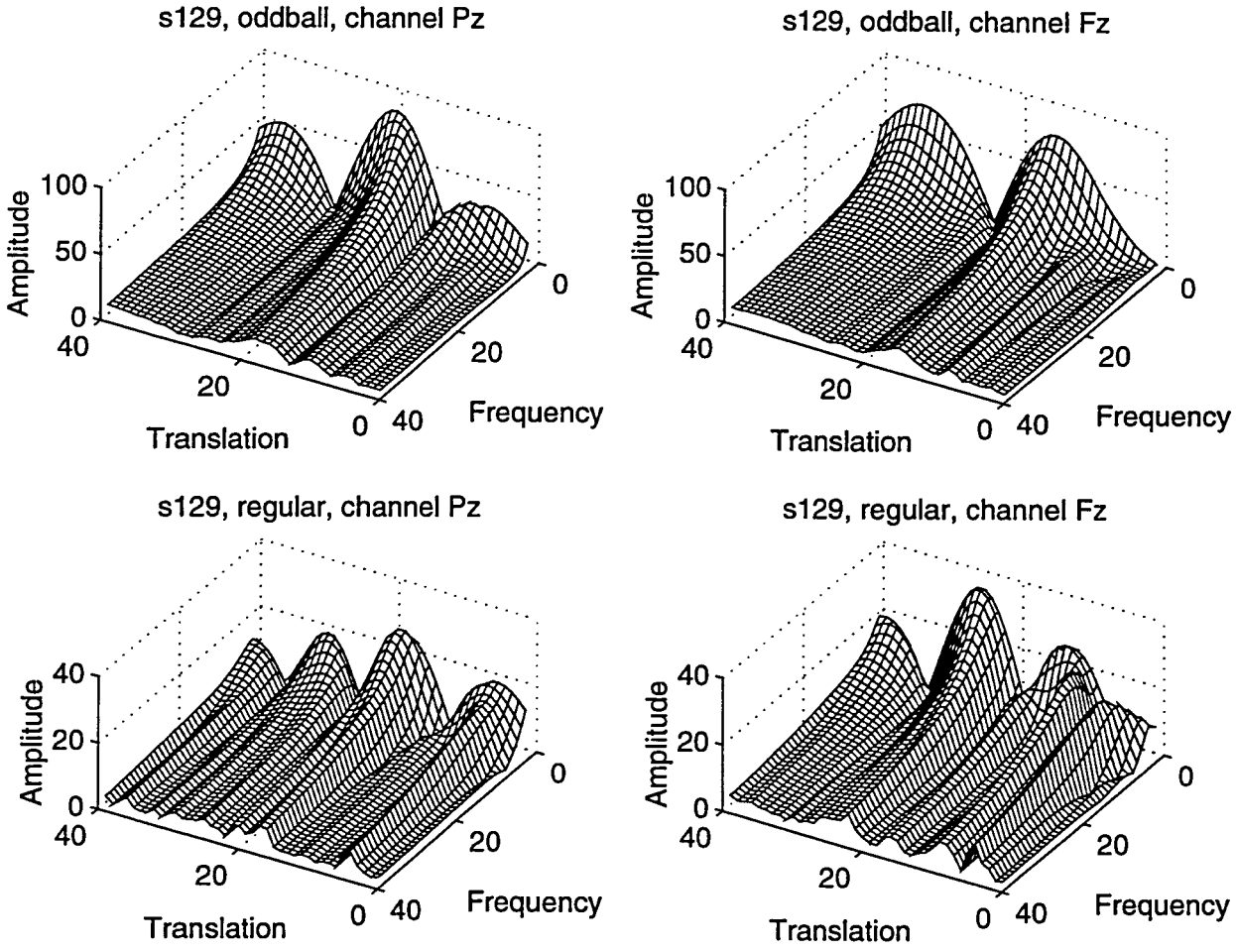

Figure B.15: Patient ID: s129, Alzheimer's

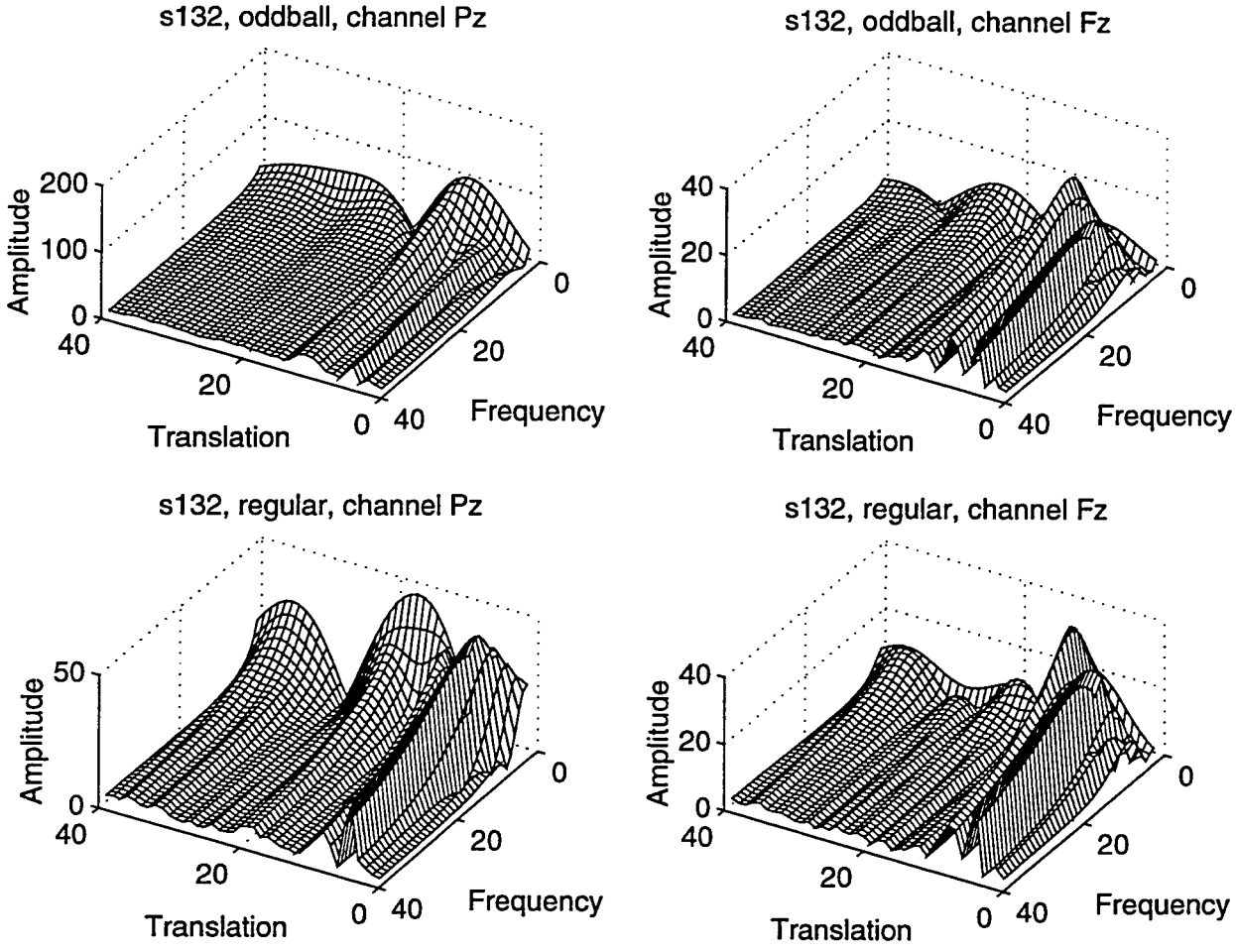

Figure B.16: Patient ID: s132, Normal

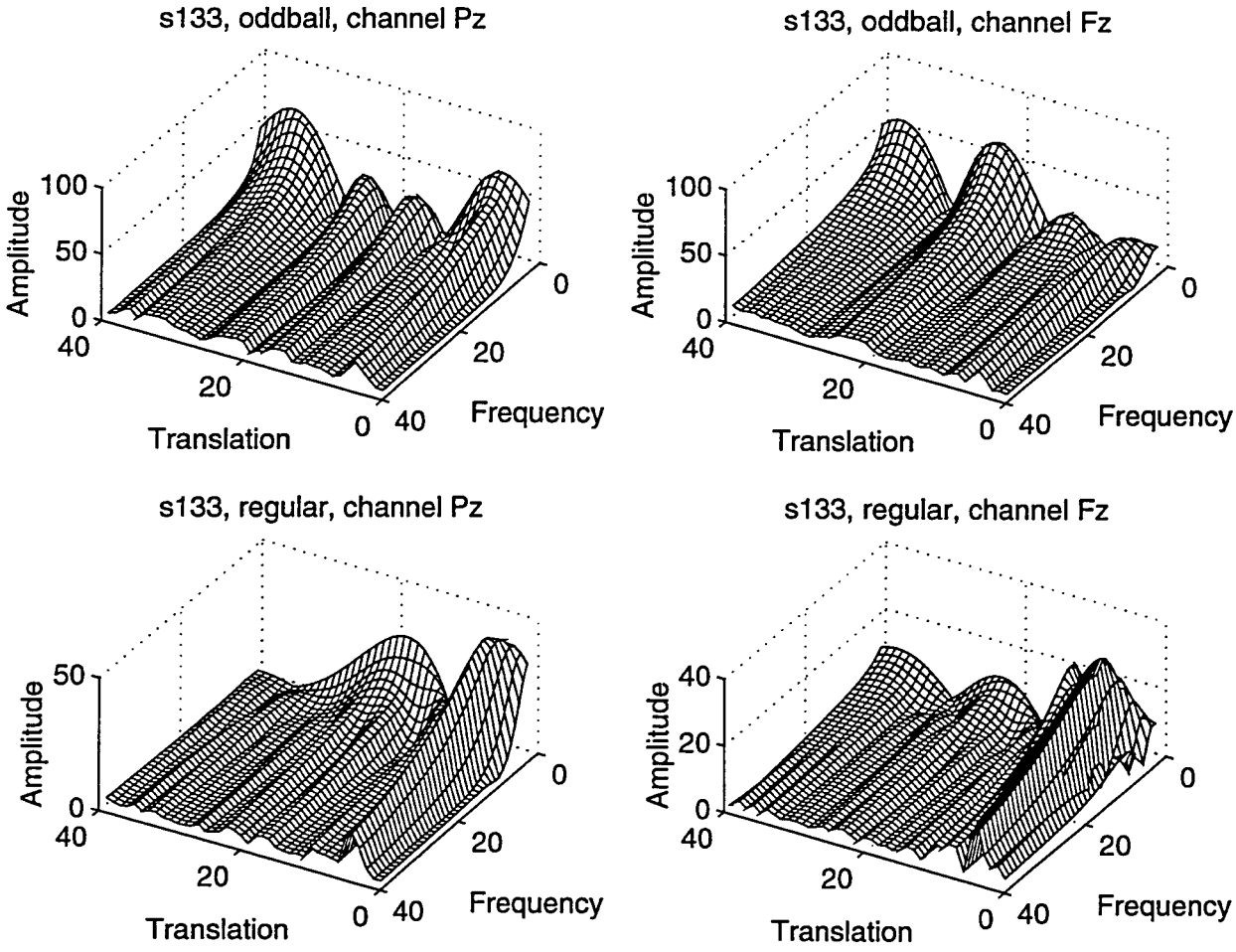

Figure B.17: Patient ID: s133, Normal

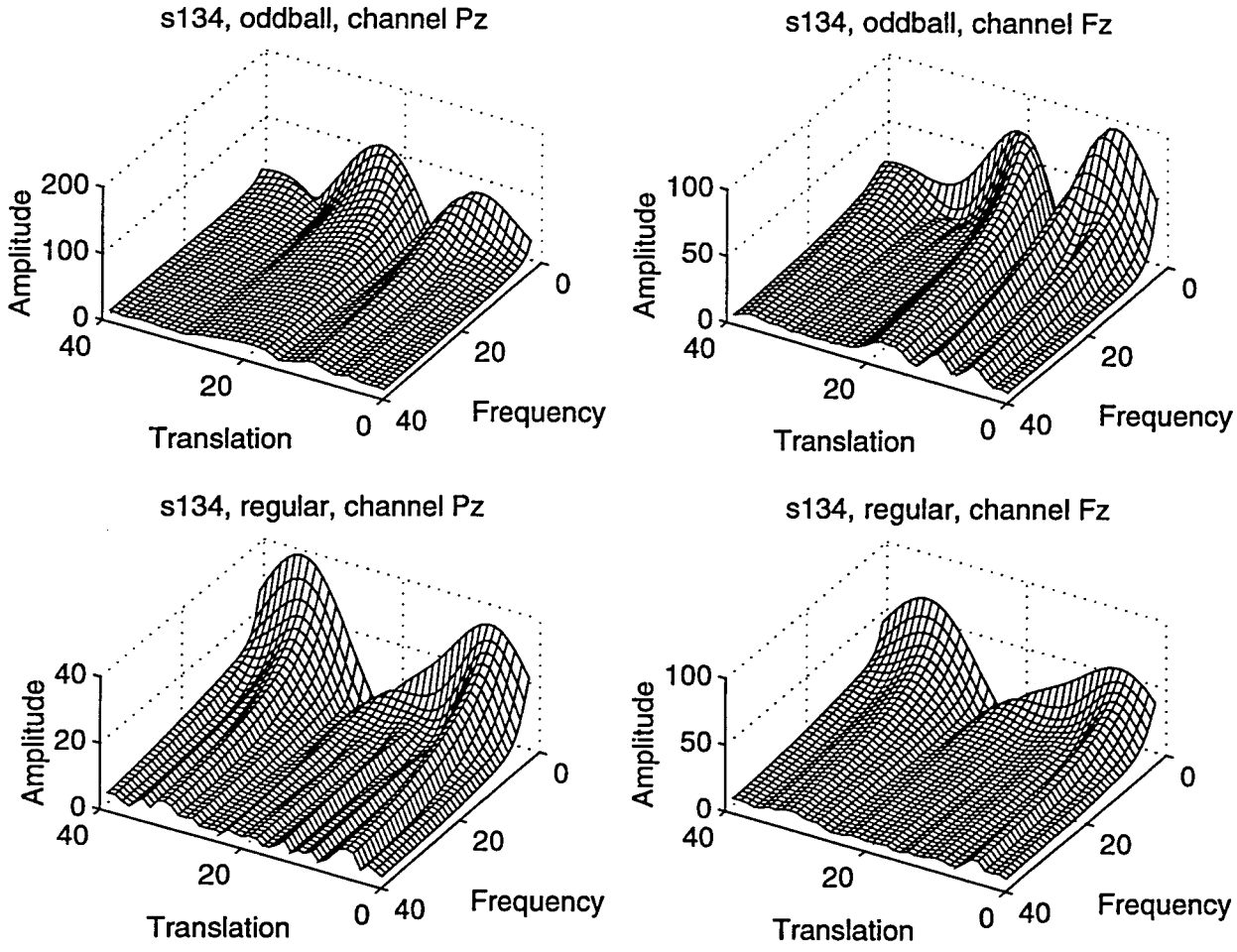

Figure B.18: Patient ID: s134, Normal

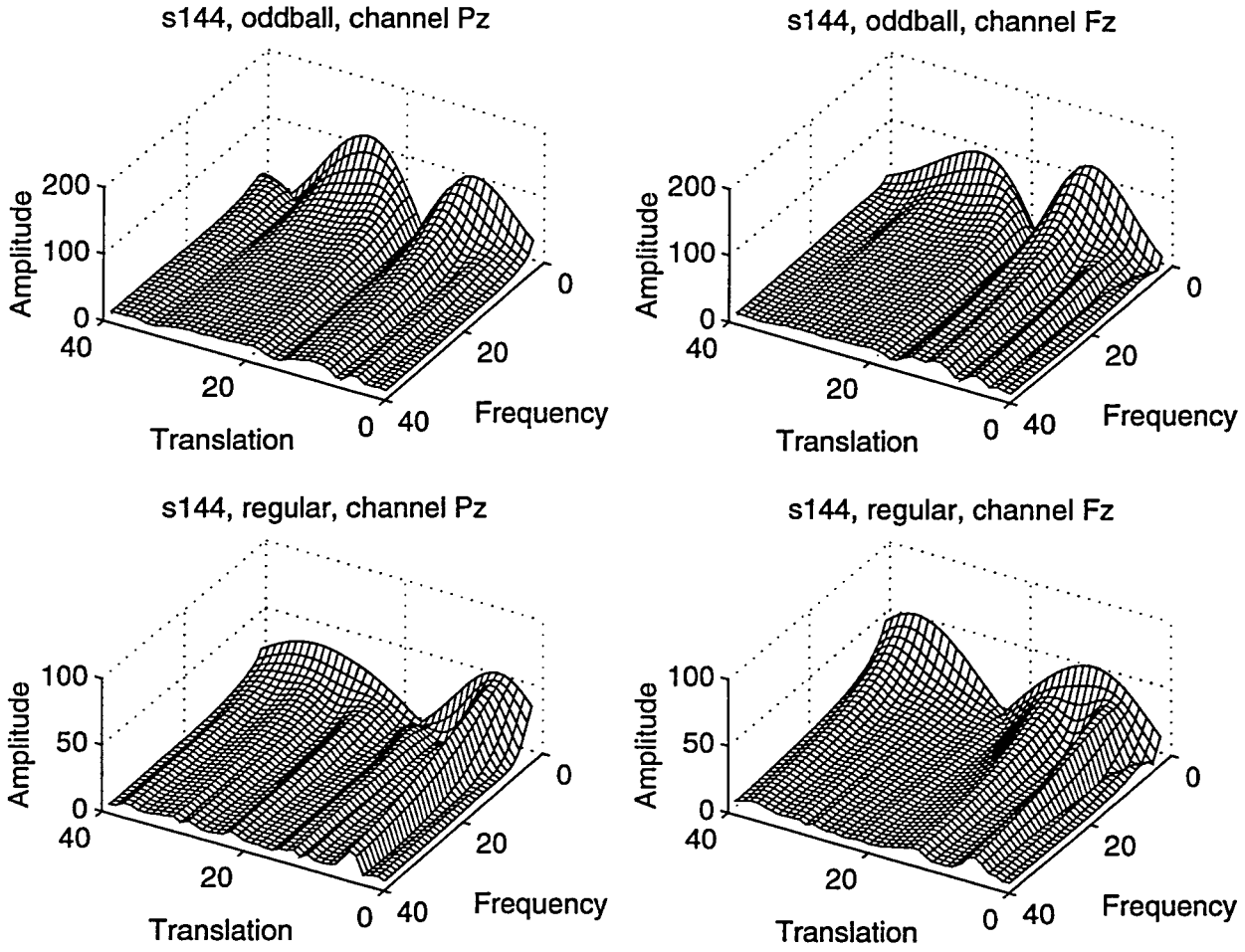

Figure B.19: Patient ID: s144, Normal

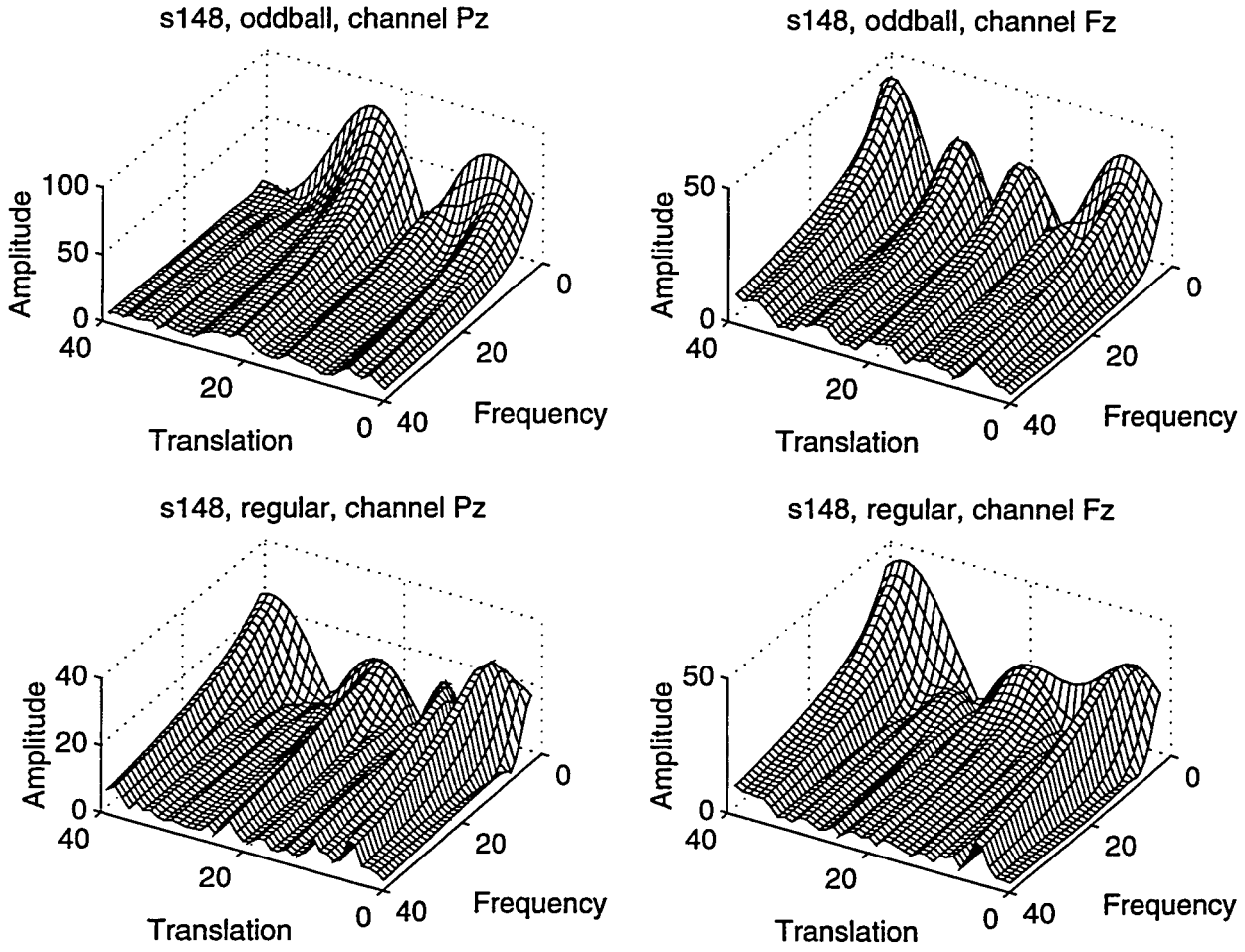

Figure B.20: Patient ID: s148, Normal

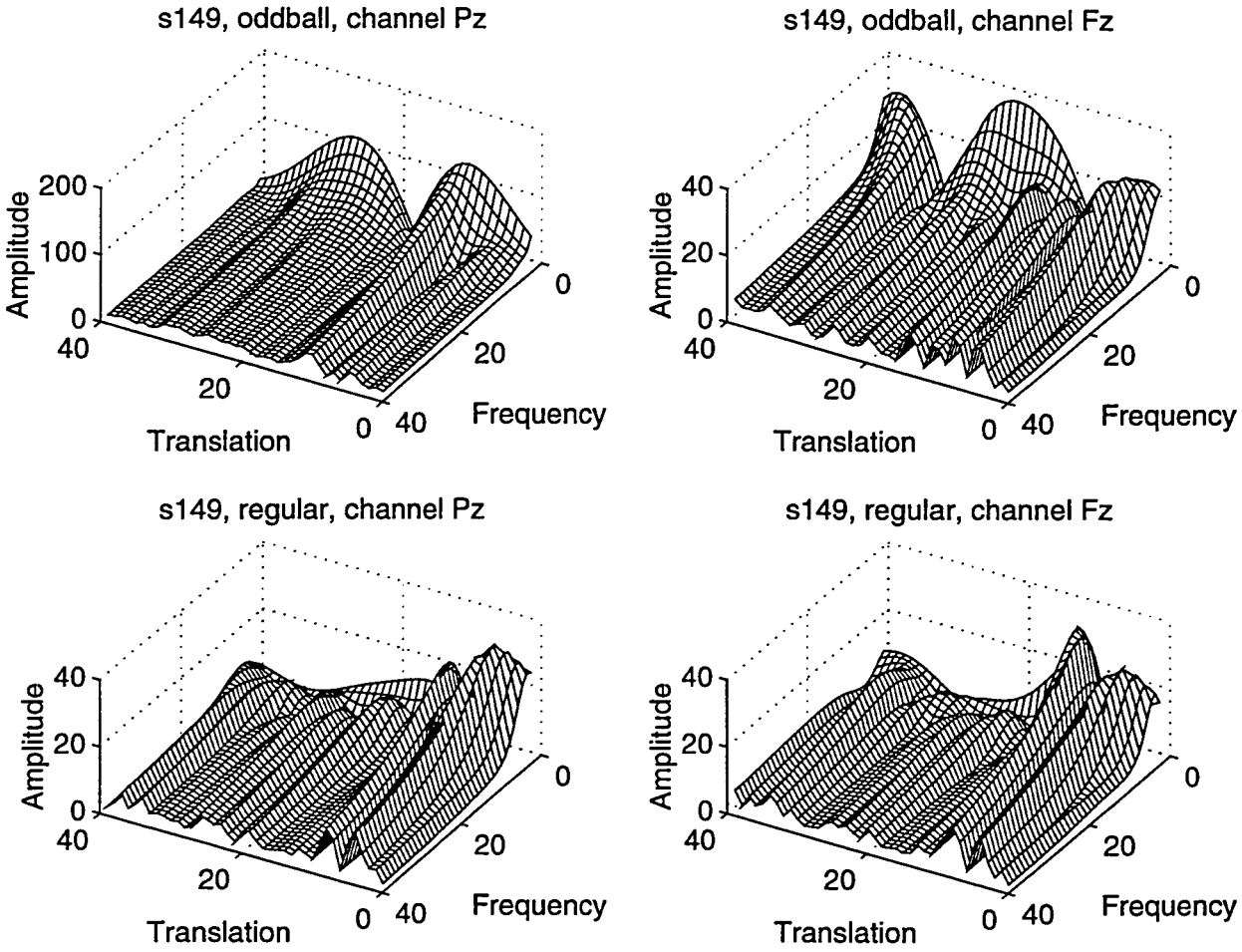

Figure B.21: Patient ID: s149, Alzheimer's

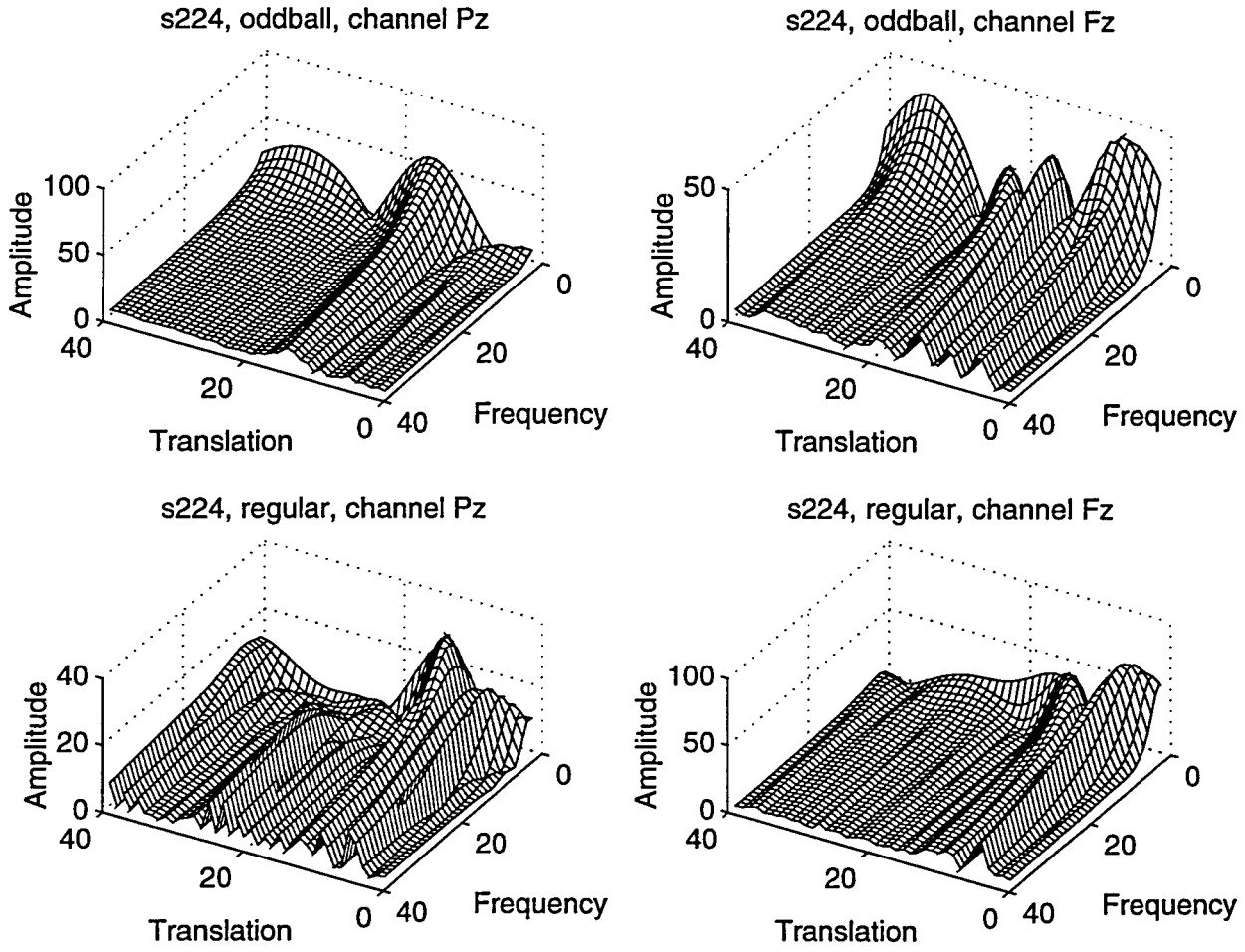

Figure B.22: Patient ID: s224, Normal

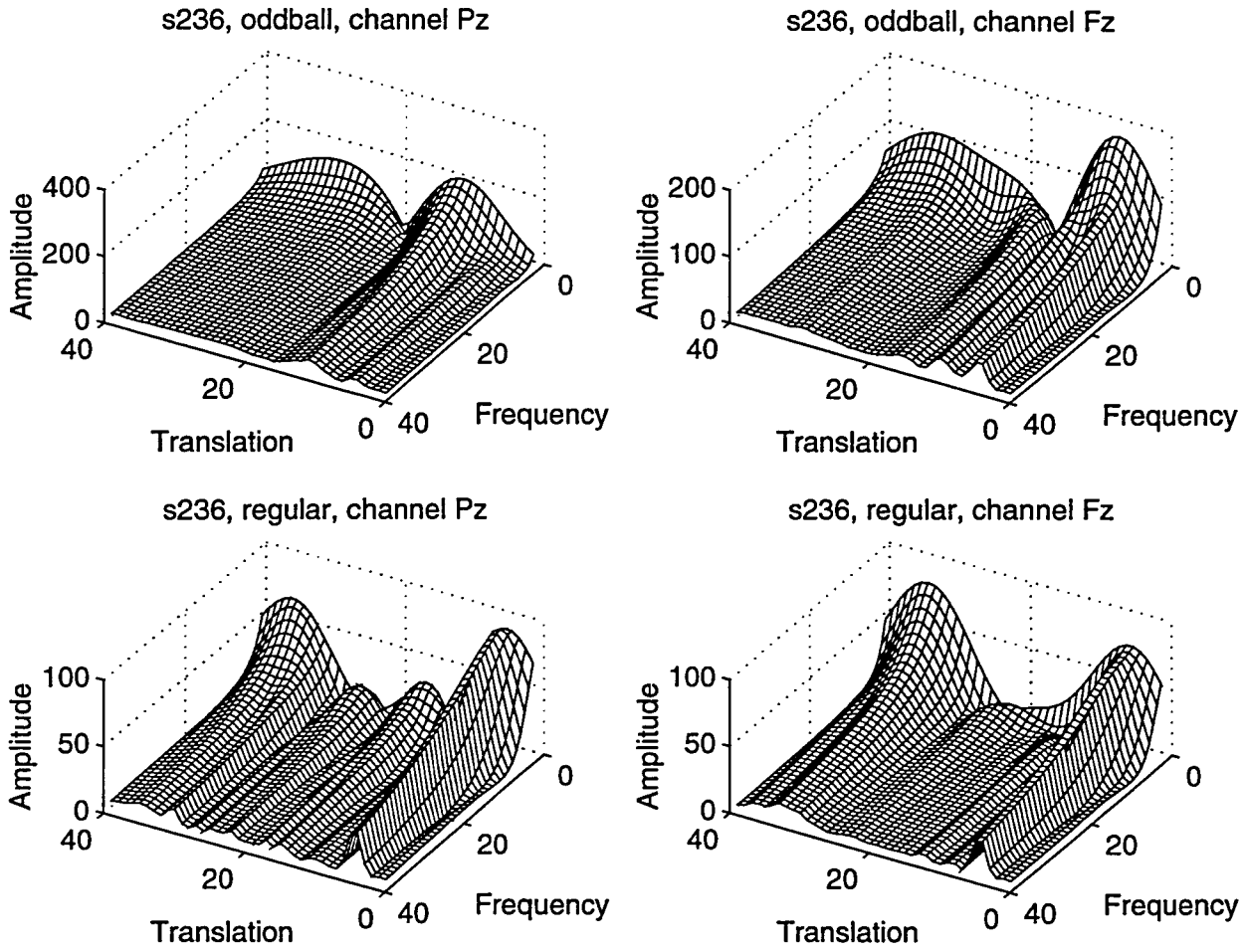

Figure B.23: Patient ID: s236, Normal

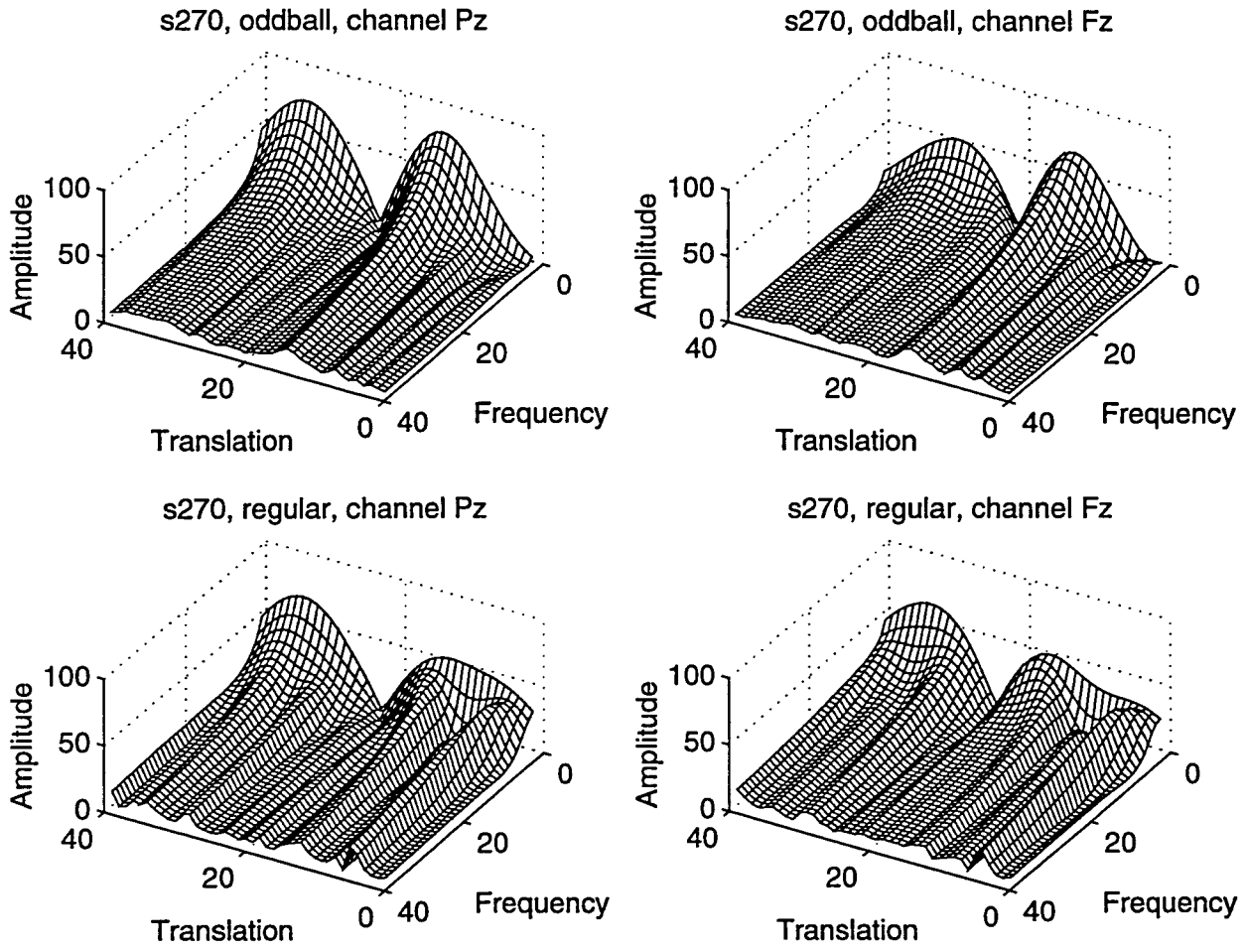

Figure B.24: Patient ID: s270, Alzheimer's

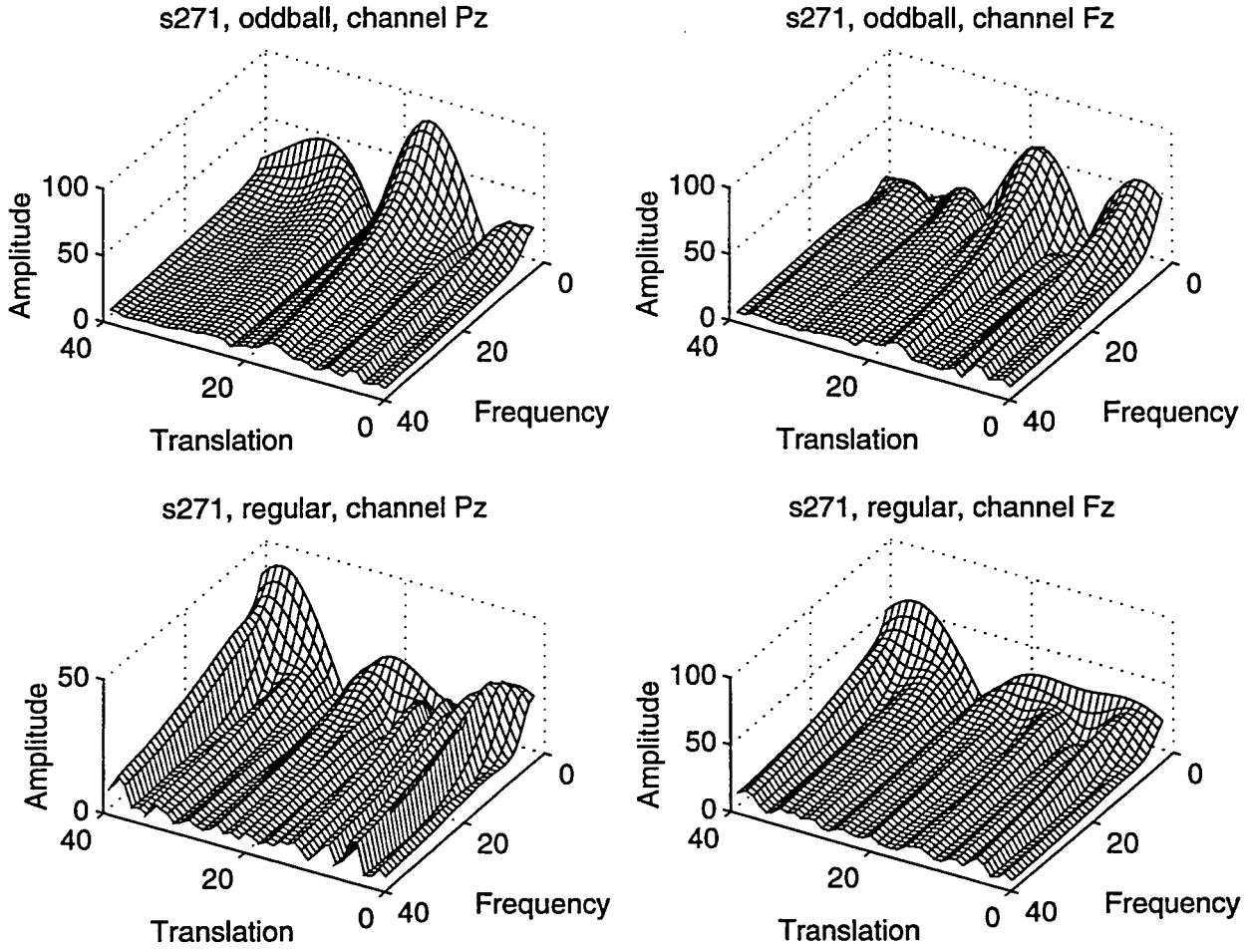

Figure B.25: Patient ID: s271, Alzheimer's

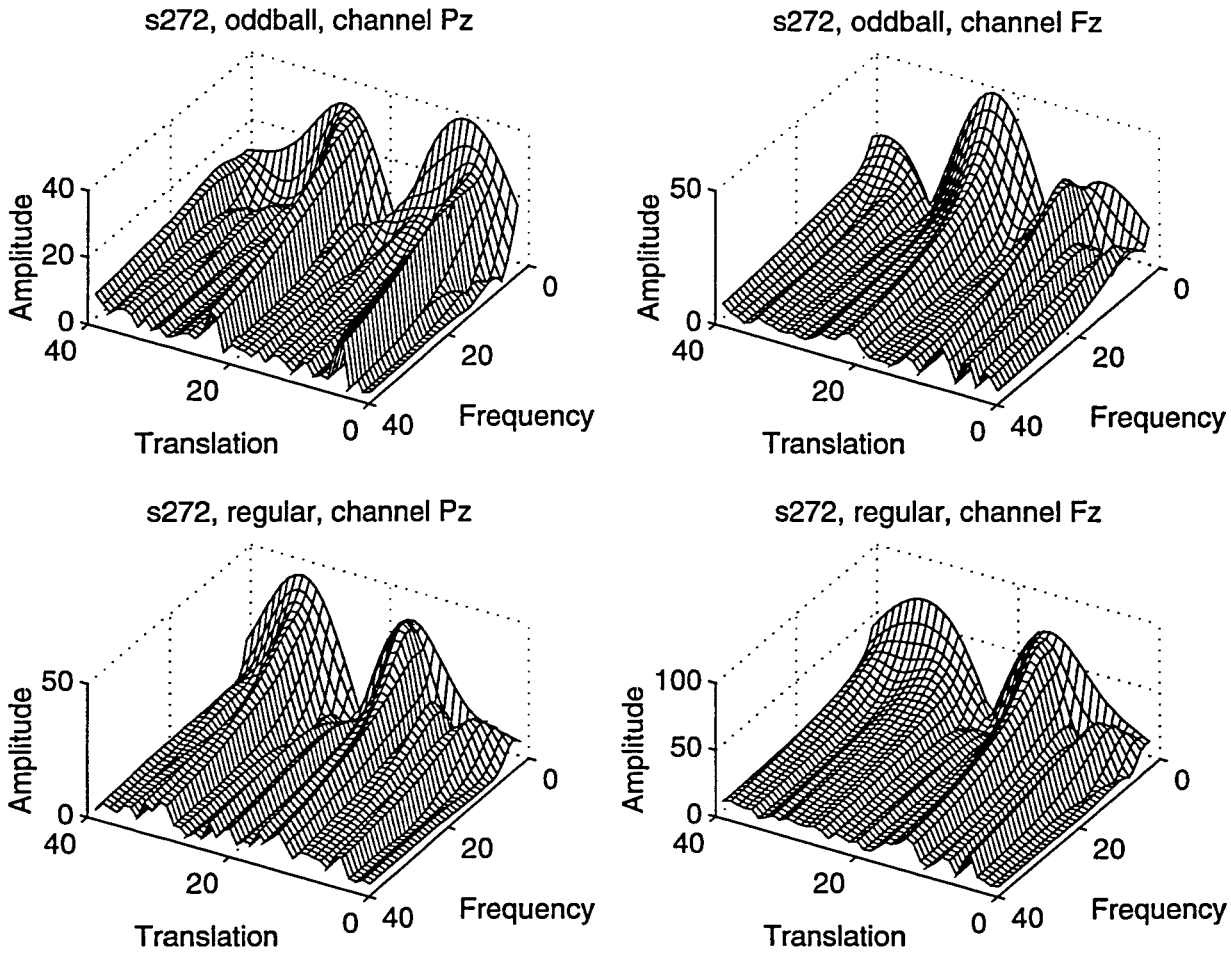

Figure B.26: Patient ID: s272, Alzheimer's

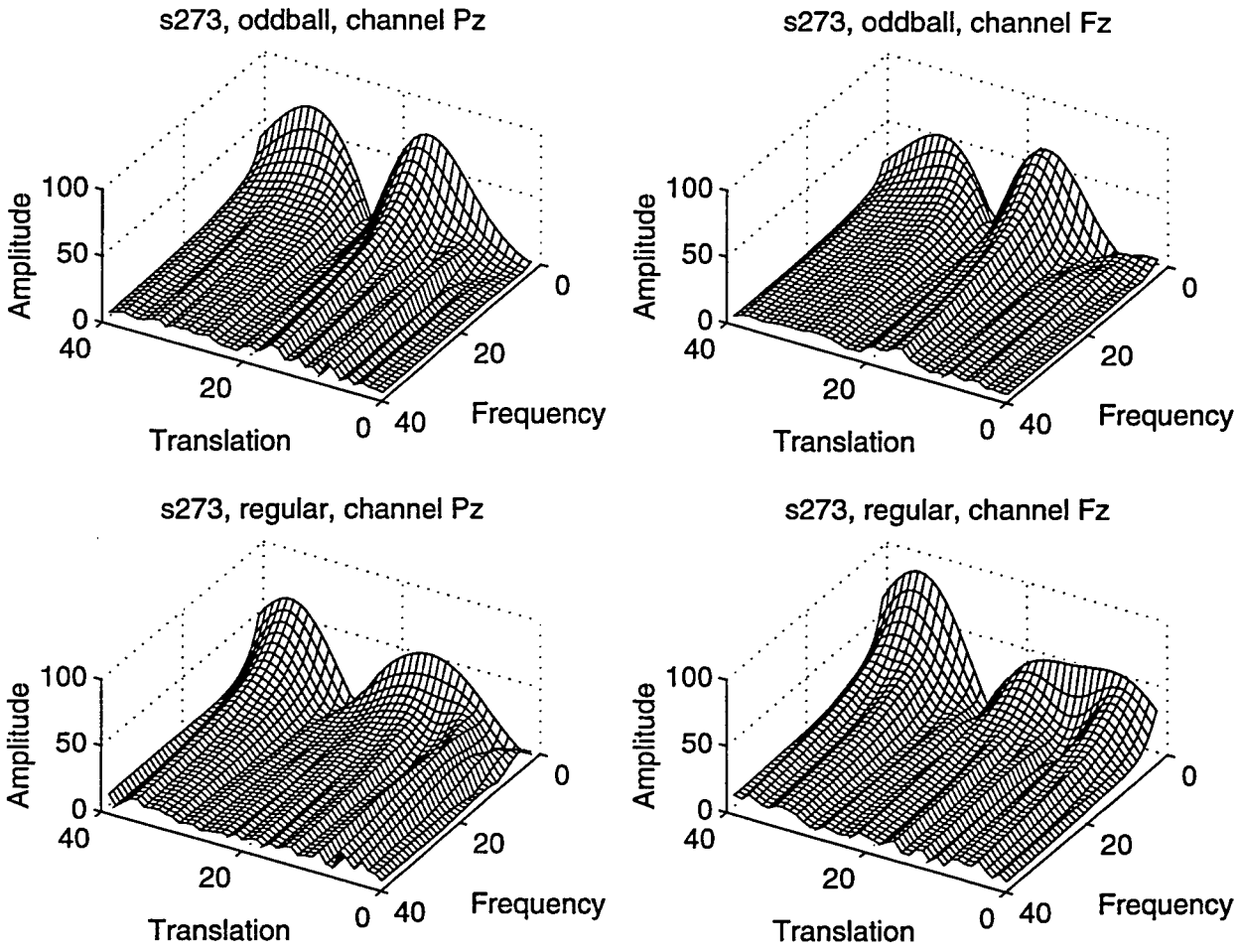

Figure B.27: Patient ID: s273, Alzheimer's

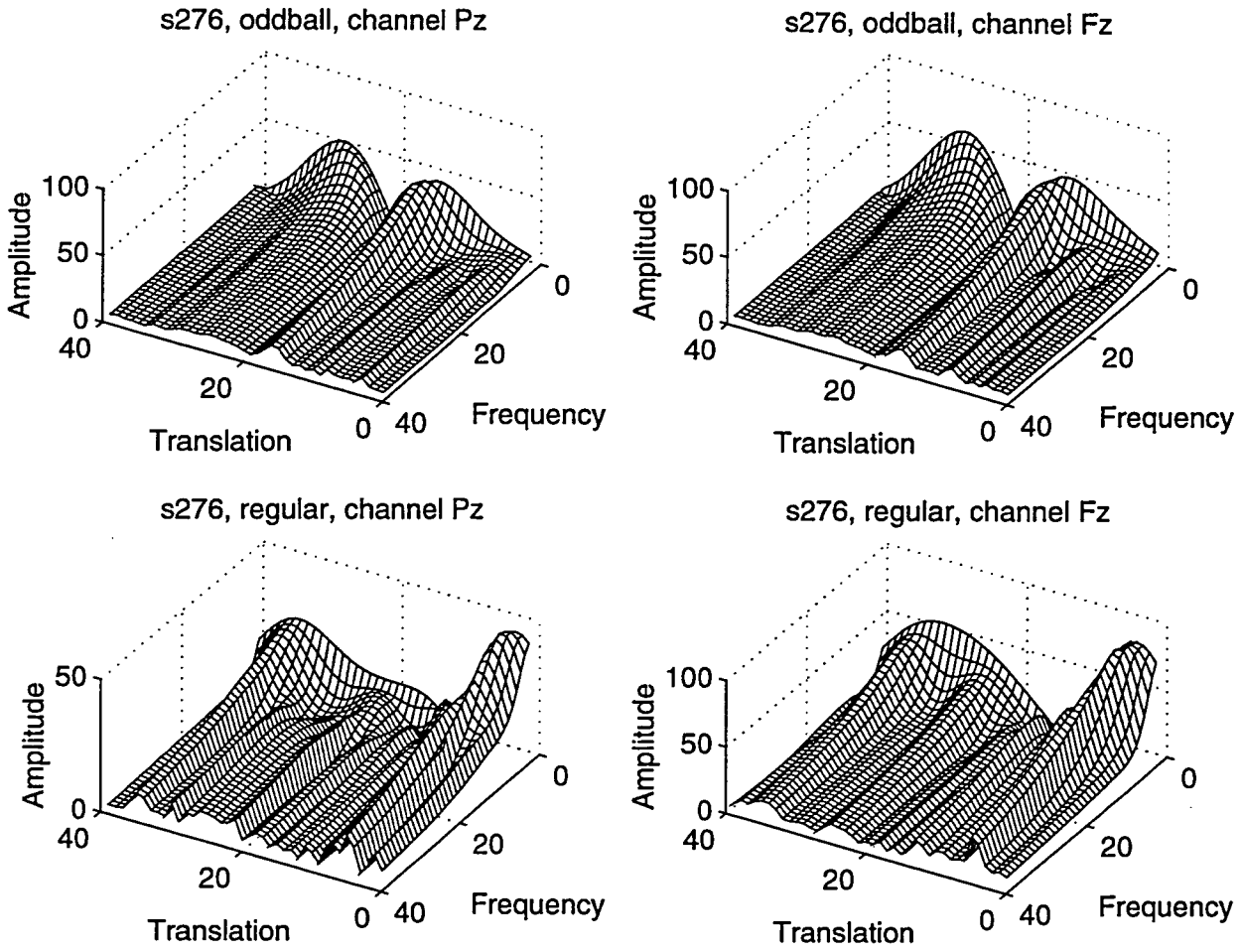

Figure B.28: Patient ID: s276, Alzheimer's

APPENDIX C. DISCRETE WAVELET TRANSFORM SIGNALS

The discrete wavelet transform of the signals as used by the k-means clustering algorithm and the multilayer perceptron neural network are presented in this appendix. As in the previous appendices four plots are given for each patient corresponding to the two channels (Pz and Fz) and two tones (oddball and regular). The first 150 of the 640 samples of the transform are given in the figures, since the other samples do not provide any additional information due to the oversampling of the original signals.

The DWTs are displayed in augmented time format as described in Chapter 4. Seven layers of decomposition are made since there are 640 samples. The first five samples correspond to the lowest frequency analyzed. The next 10 points analyzes the next *octave* (twice the previous frequency), and the next 20 samples analzes the third octave and so forth. The last 320 samples correspond to the highest frequency analyzed with this procedure.

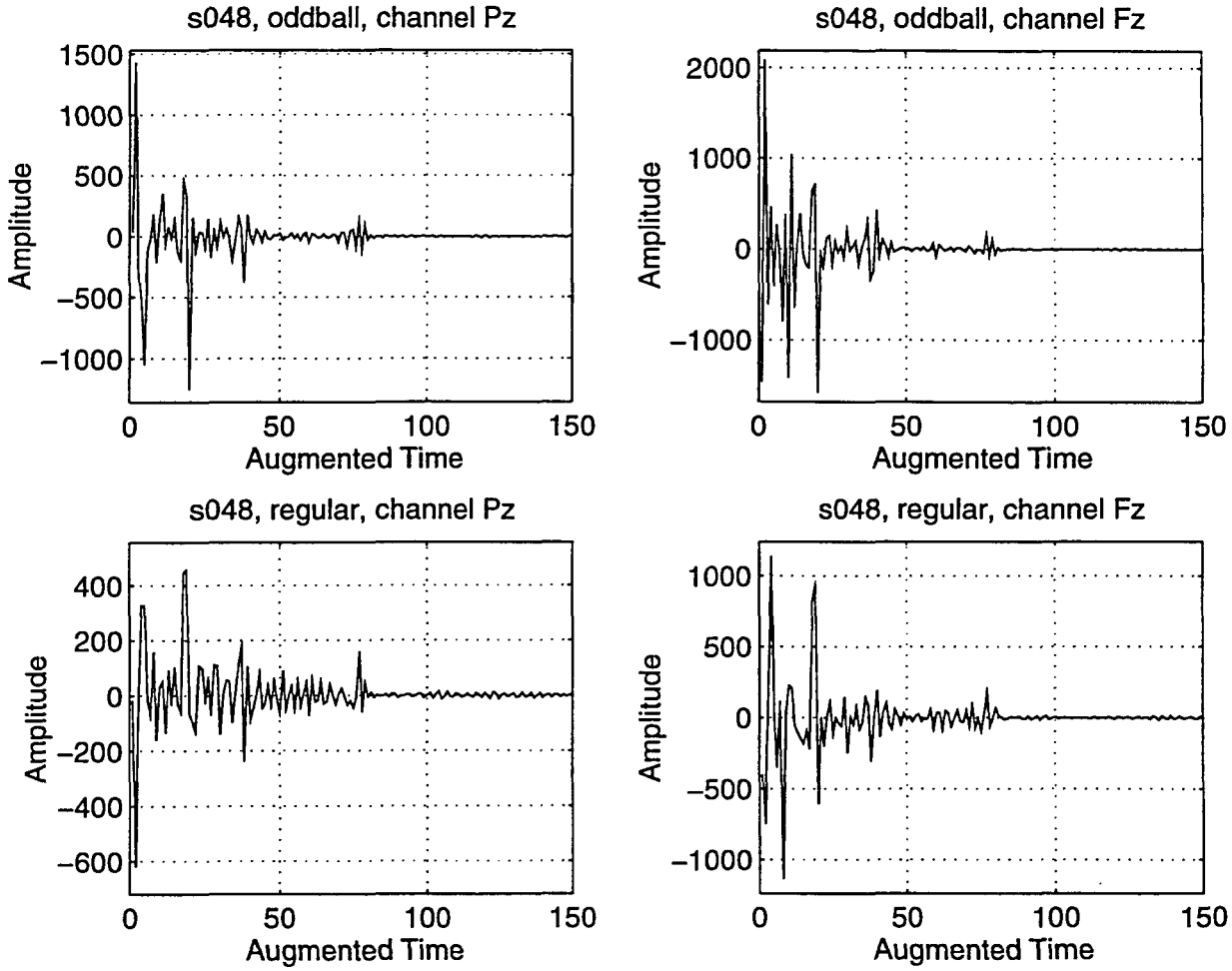

Figure C.1: Patient ID: s048, Alzheimer's

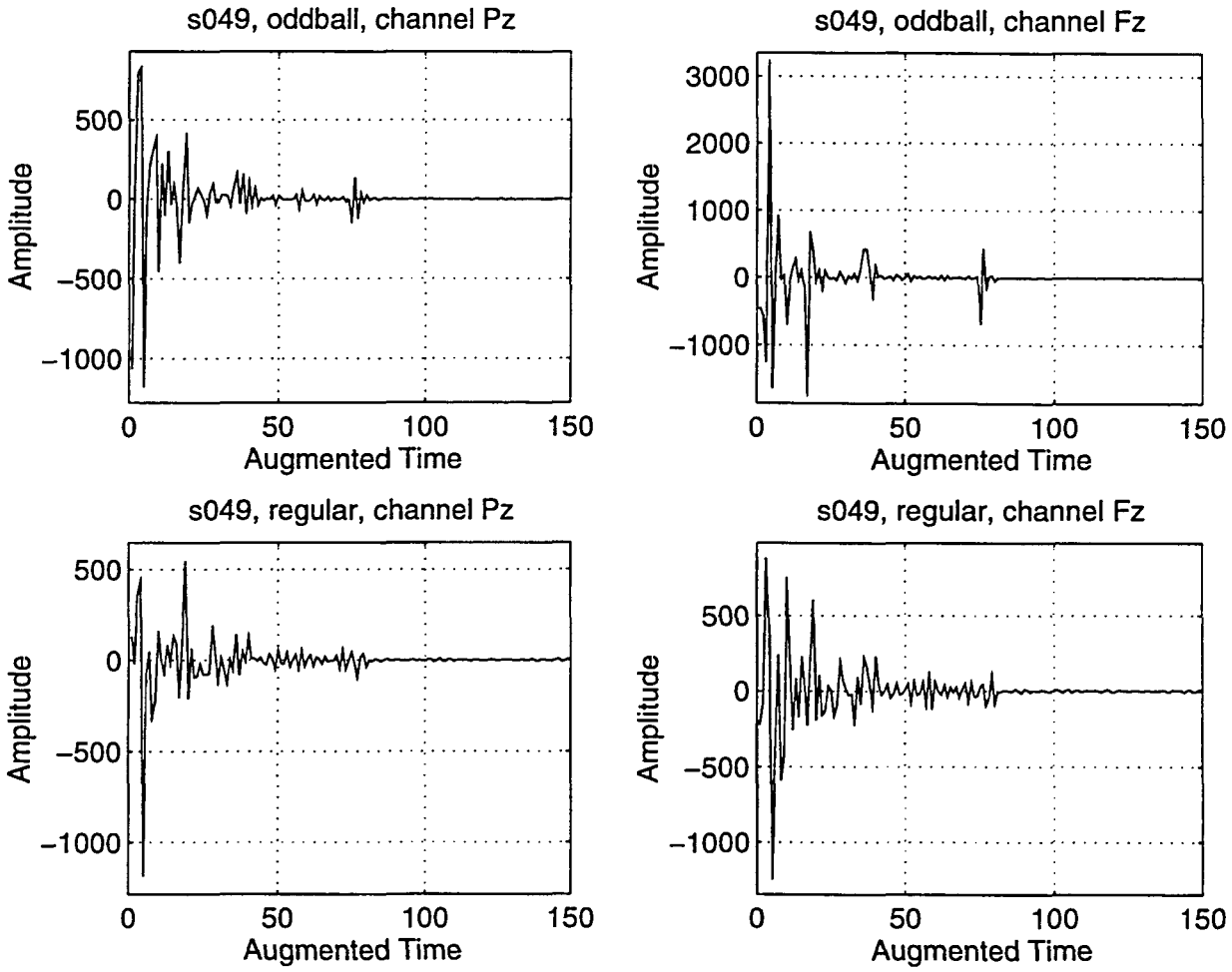

Figure C.2: Patient ID: s049, Alzheimer's

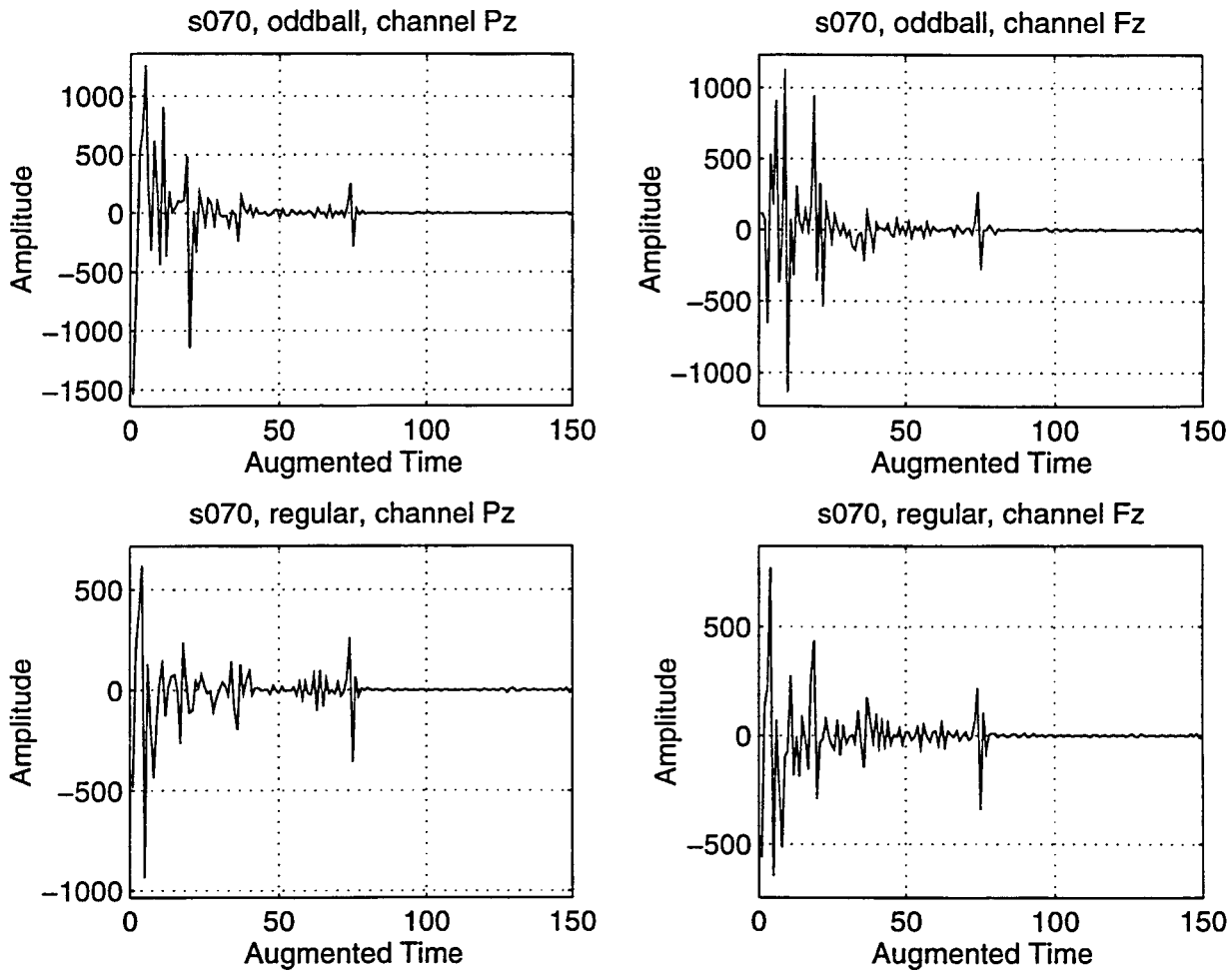

Figure C.3: Patient ID: s070, Normal

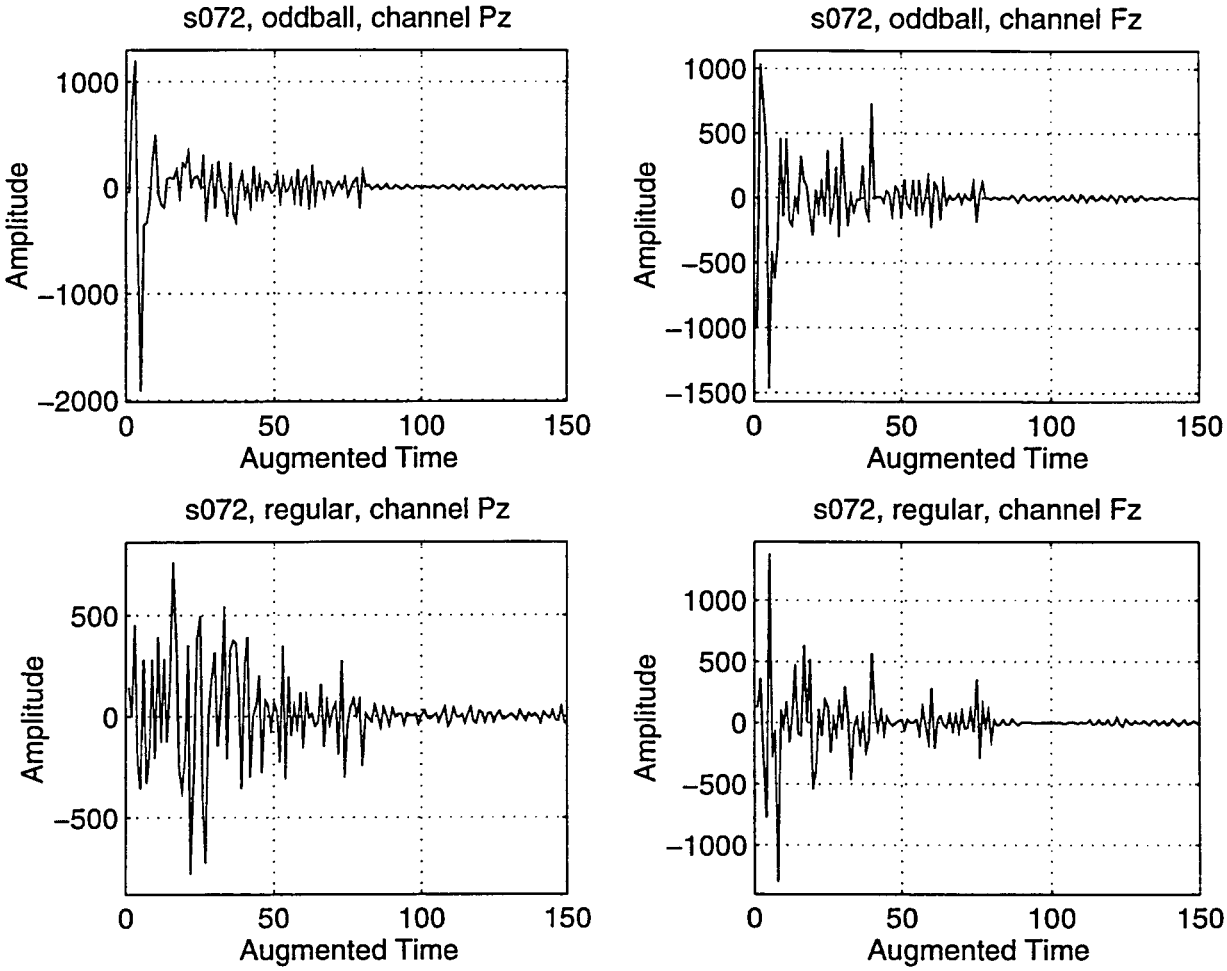

Figure C.4: Patient ID: s072, Alzheimer's

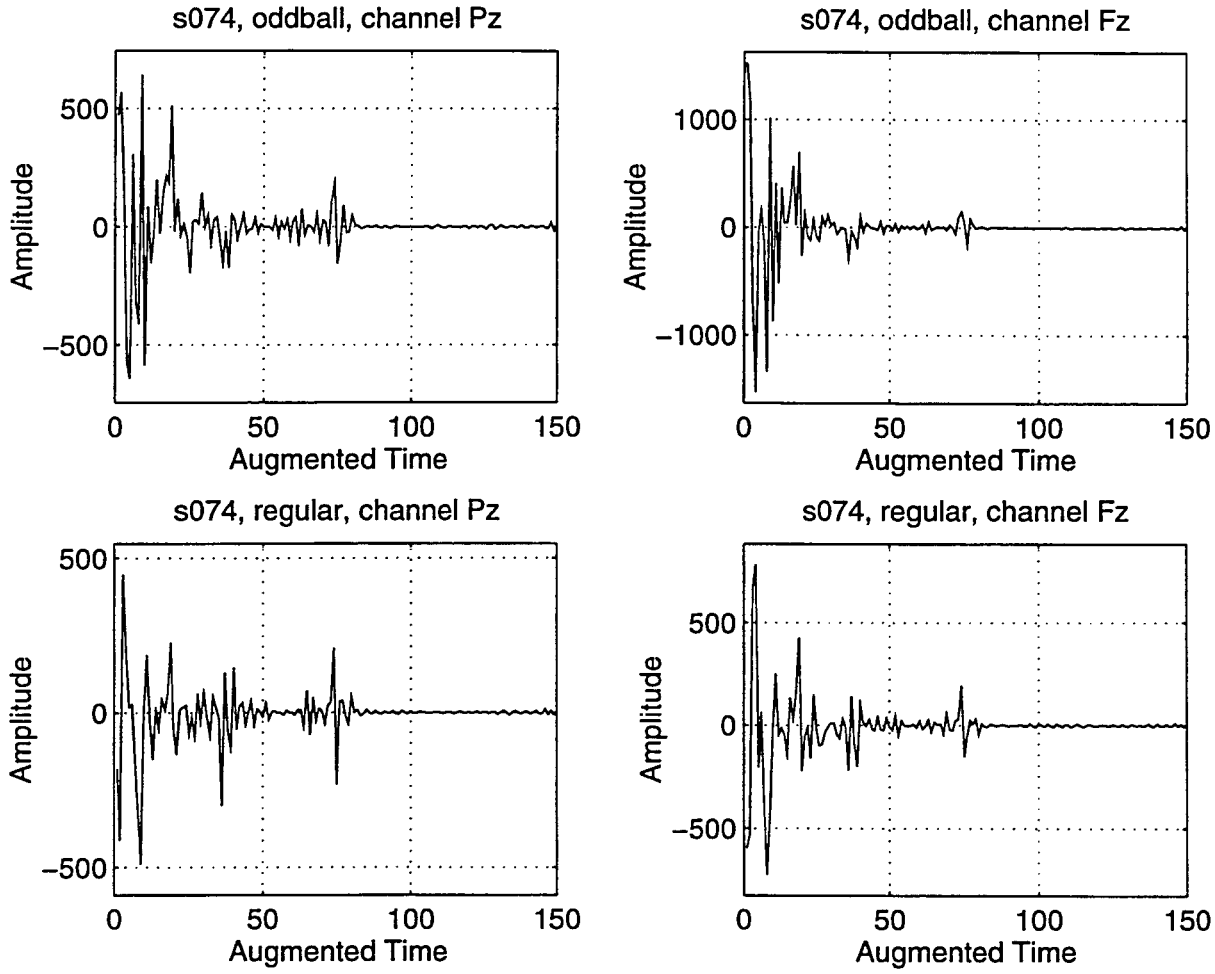

Figure C.5: Patient ID: s074, Normal

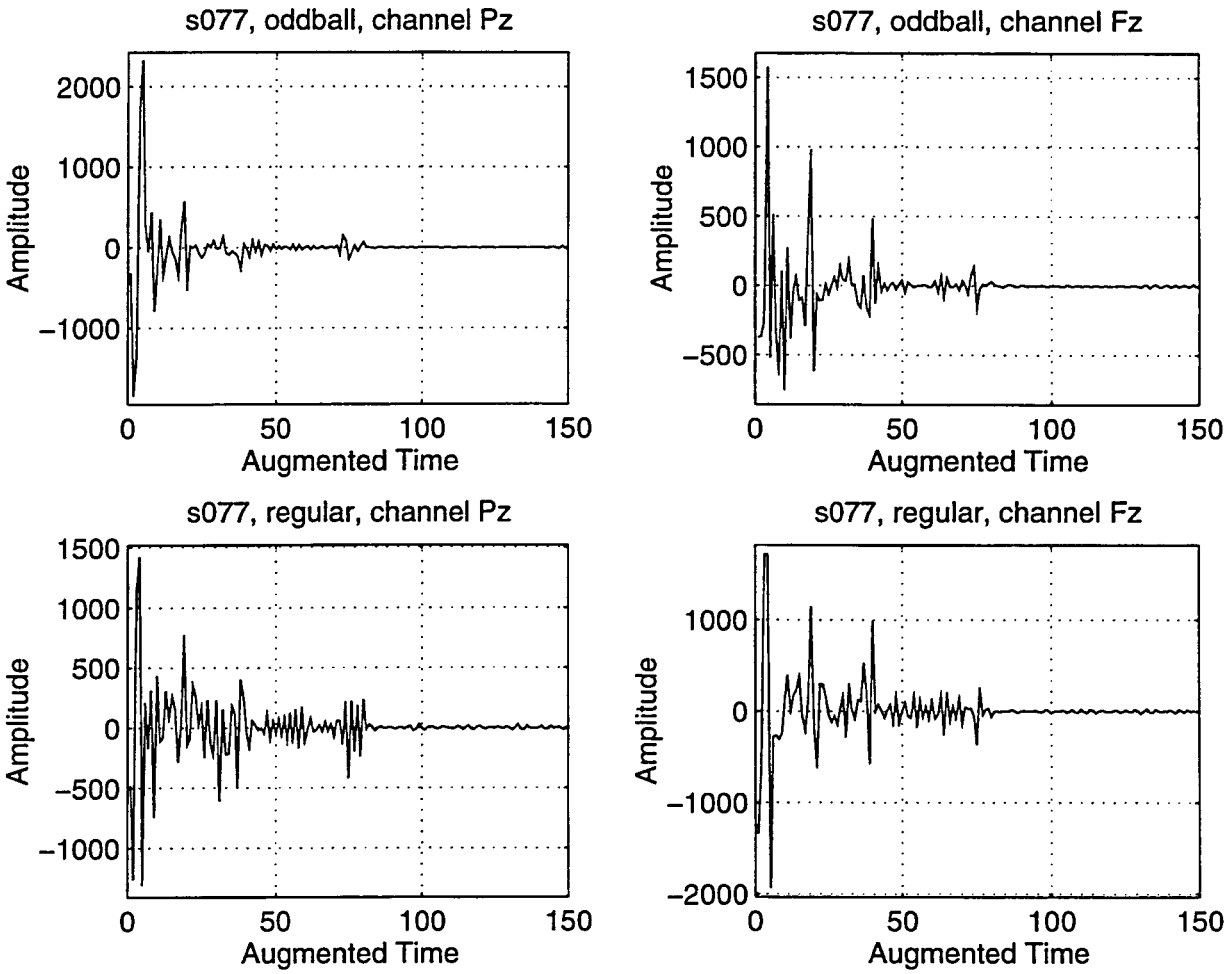

Figure C.6: Patient ID: s077, Normal

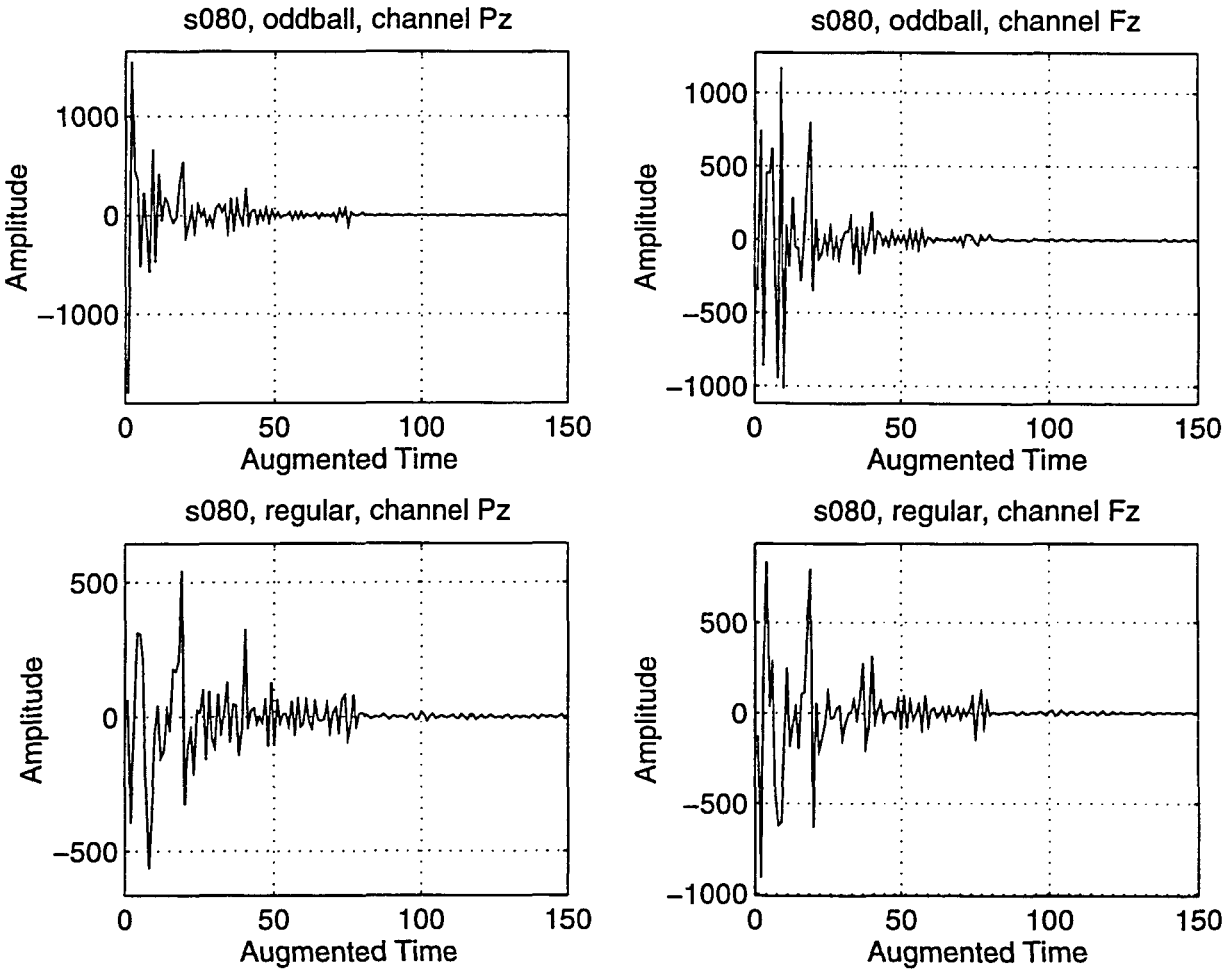

Figure C.7: Patient ID: s080, Normal

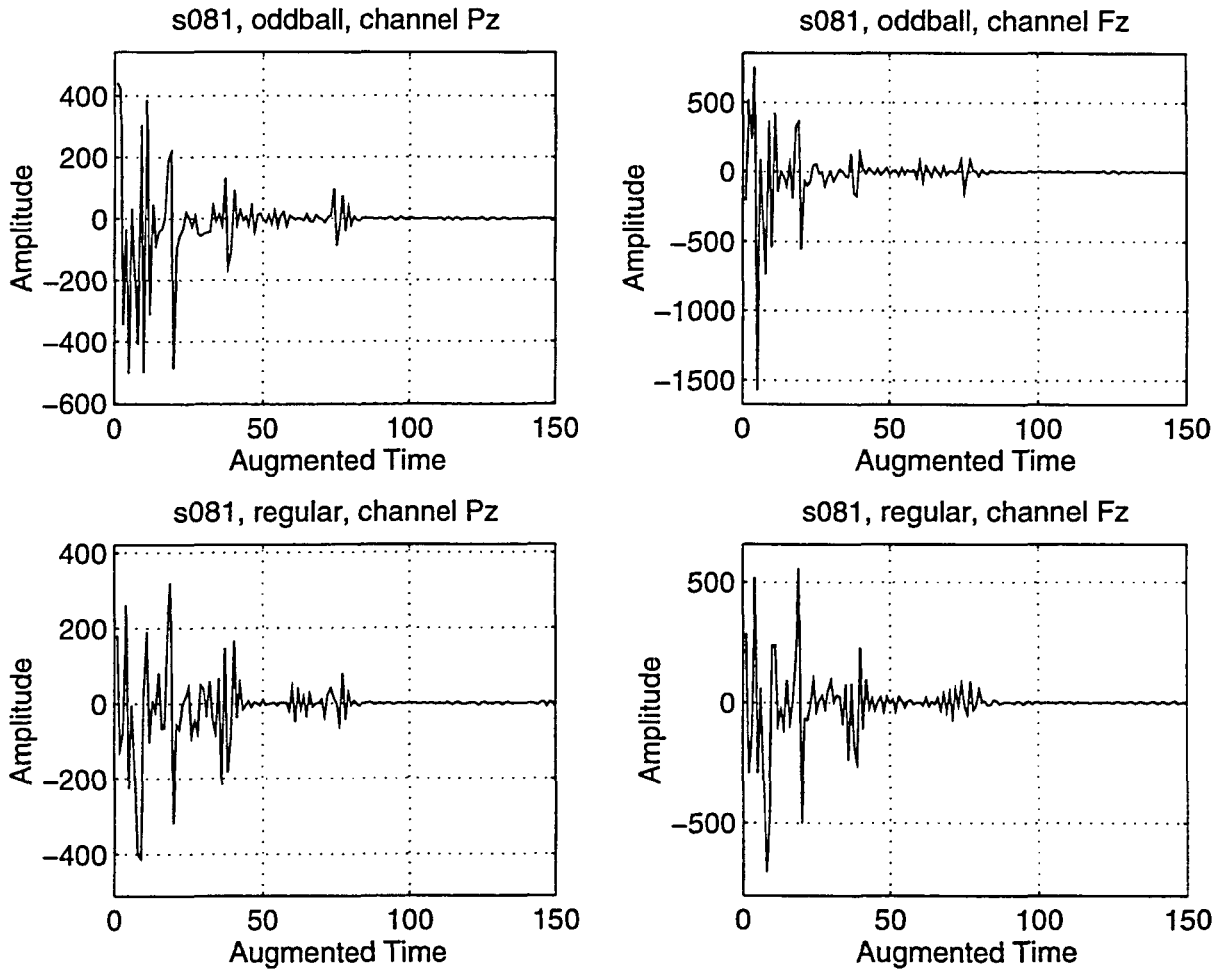

Figure C.8: Patient ID: s081, Normal

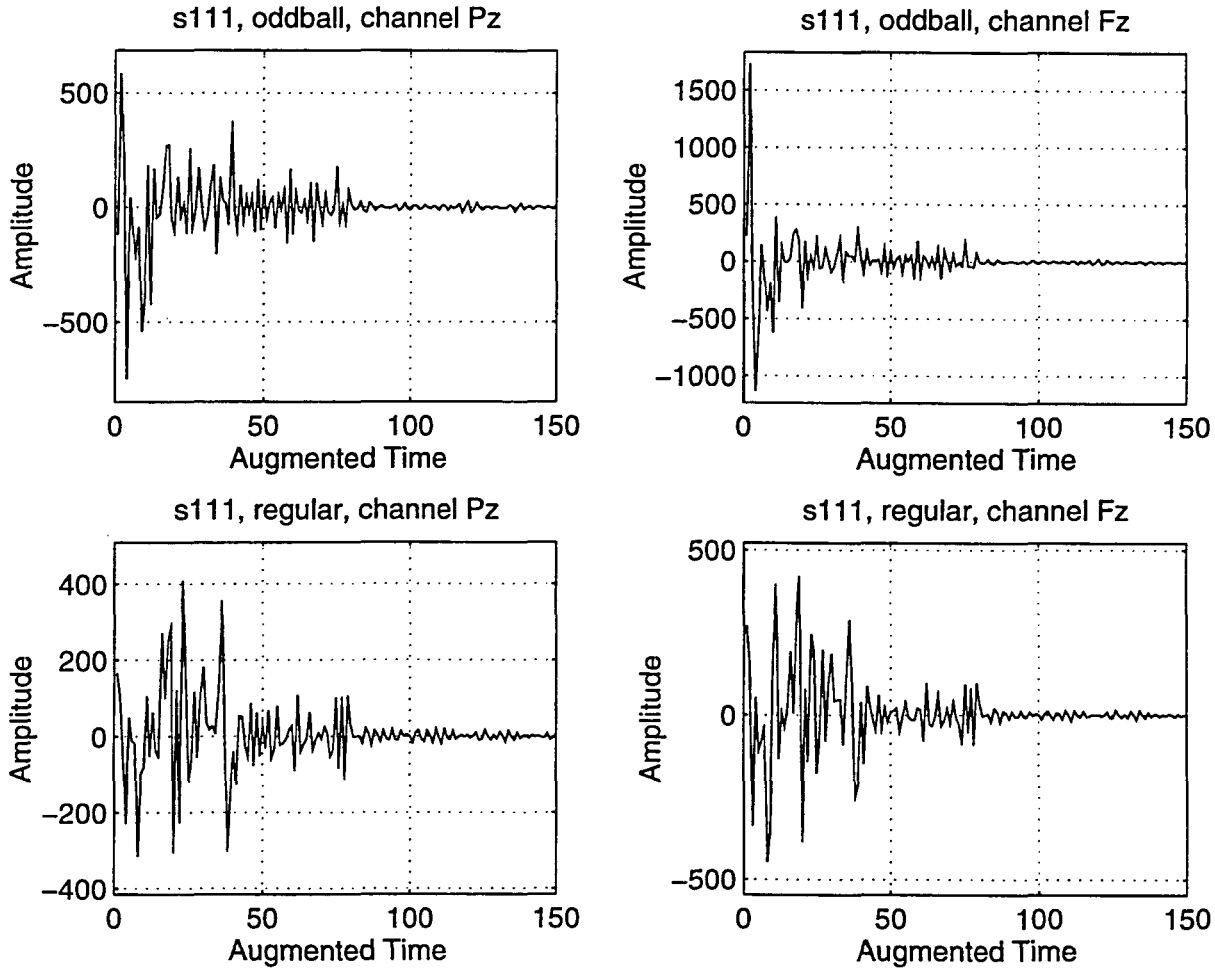

Figure C.9: Patient ID: s111, Alzheimer's

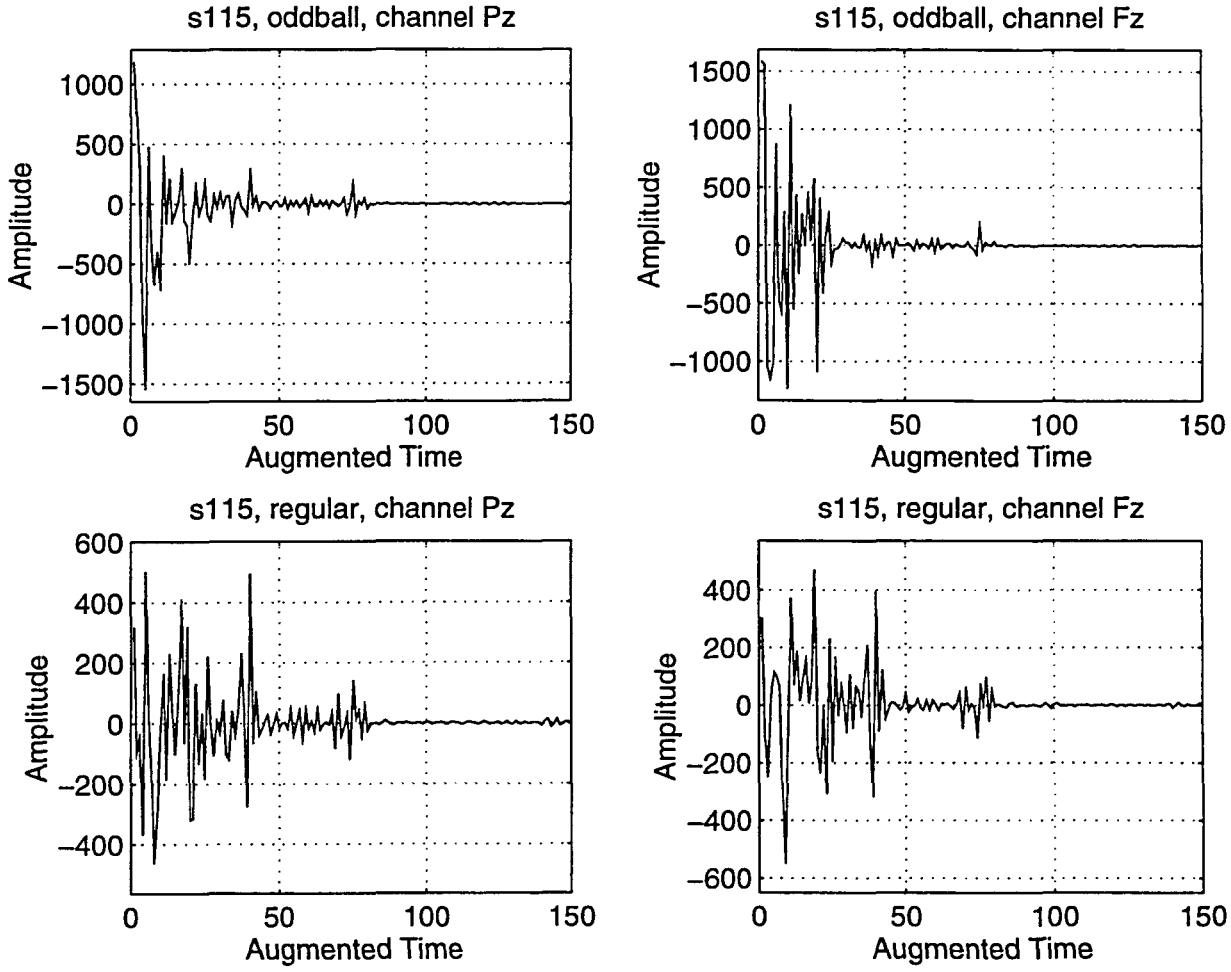

Figure C.10: Patient ID: s115, Alzheimer's

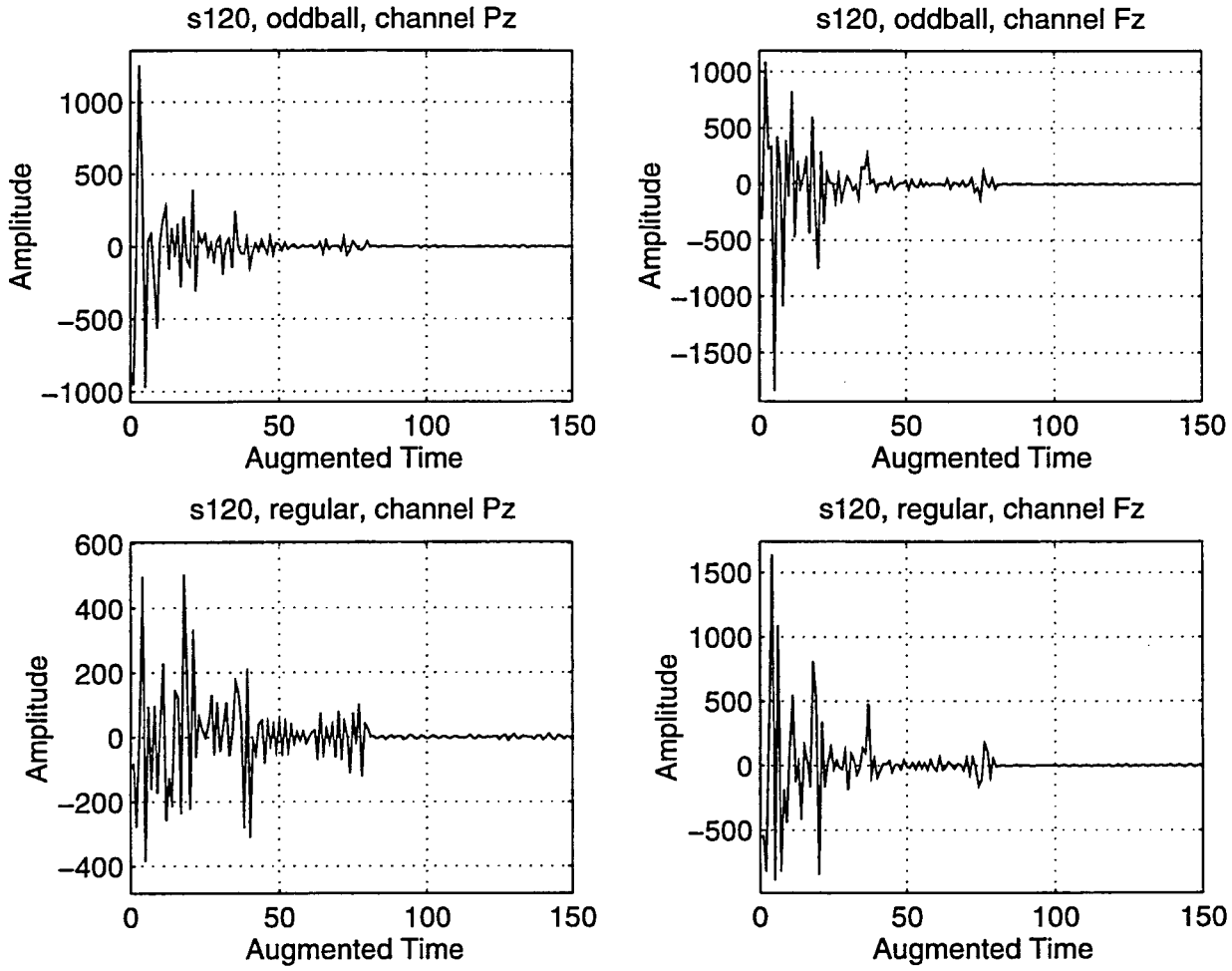

Figure C.11: Patient ID: s120, Alzheimer's

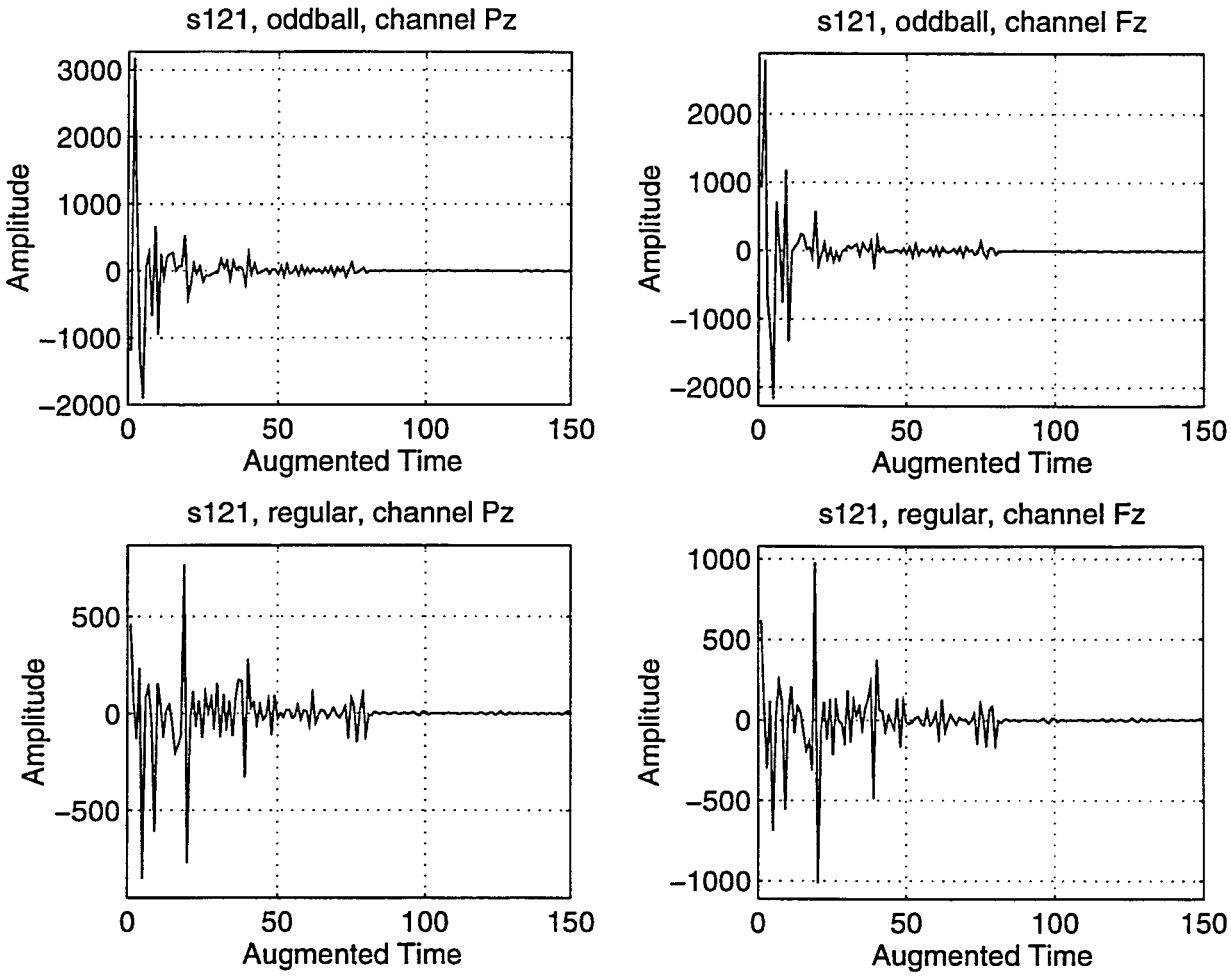

Figure C.12: Patient ID: s121, Normal

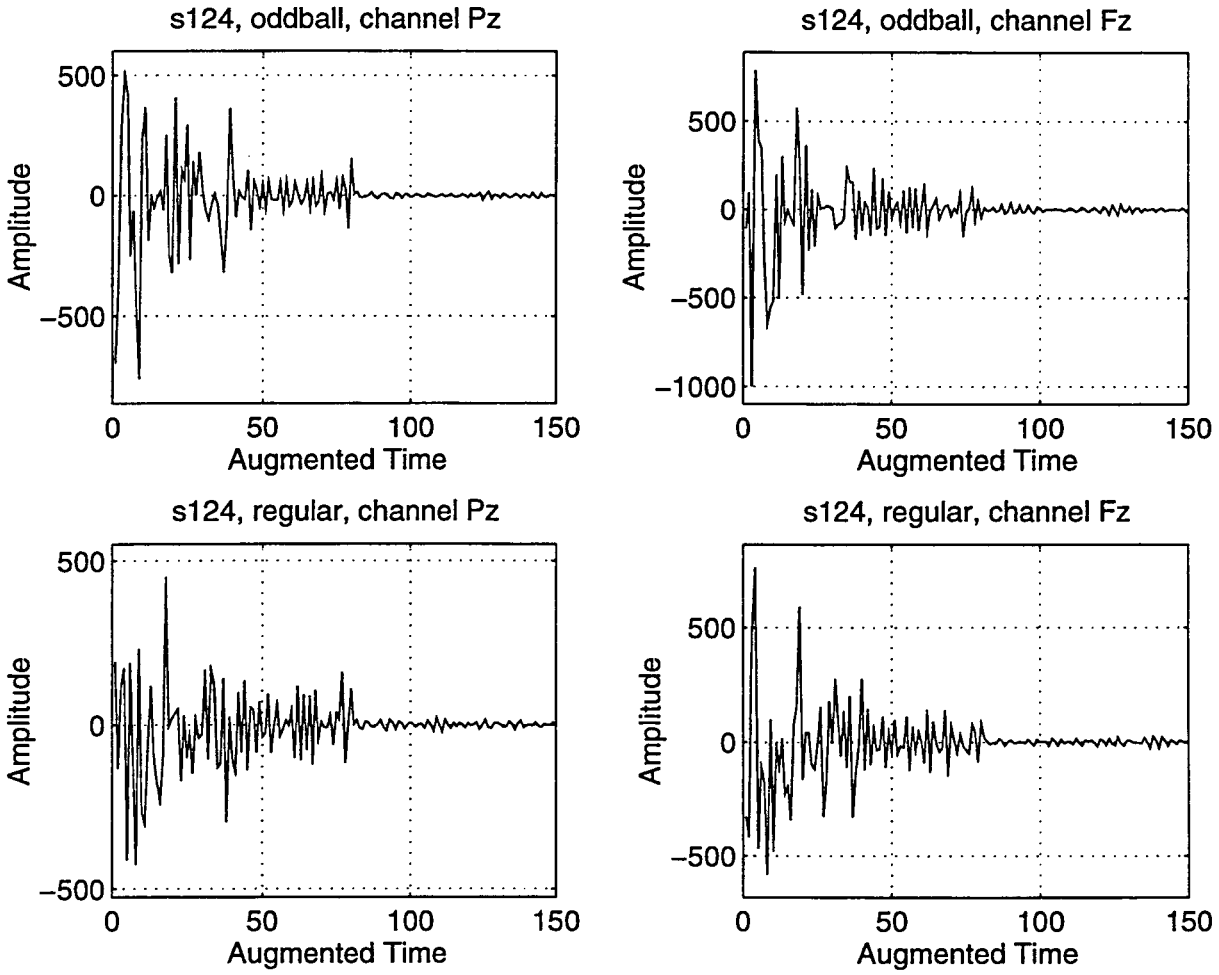

Figure C.13: Patient ID: s124, Normal

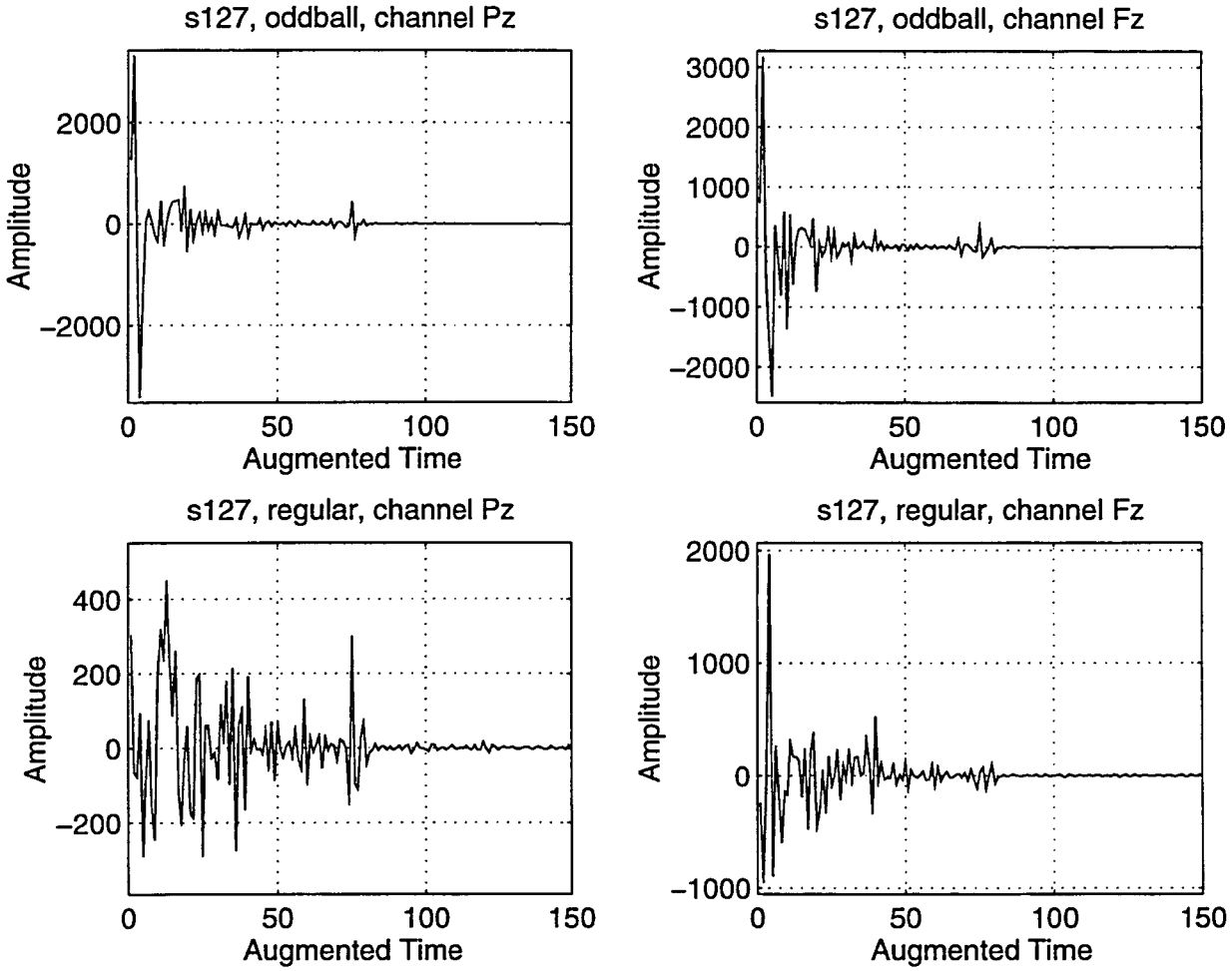

Figure C.14: Patient ID: s127, Alzheimer's

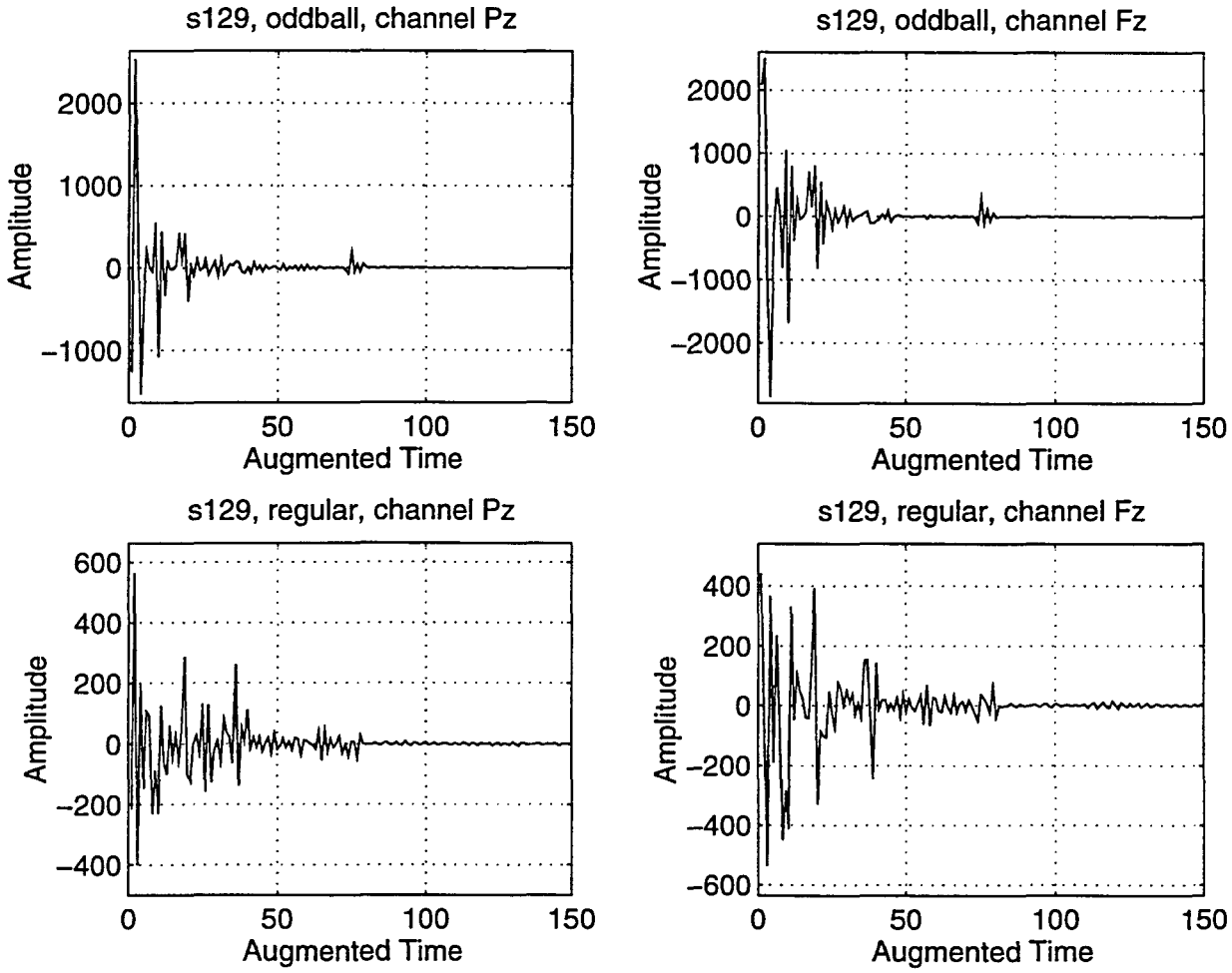

Figure C.15: Patient ID: s129, Alzheimer's

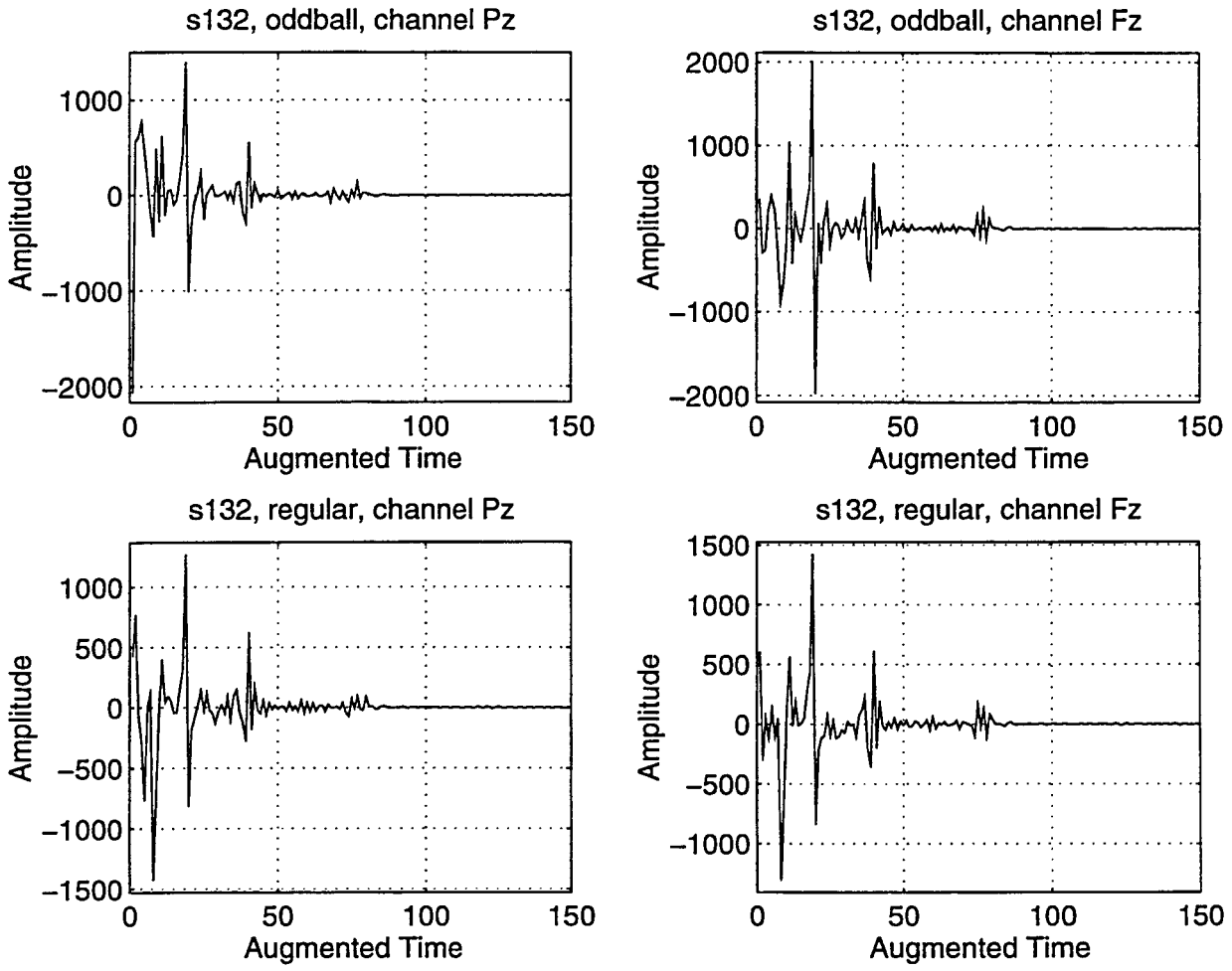

Figure C.16: Patient ID: s132, Normal

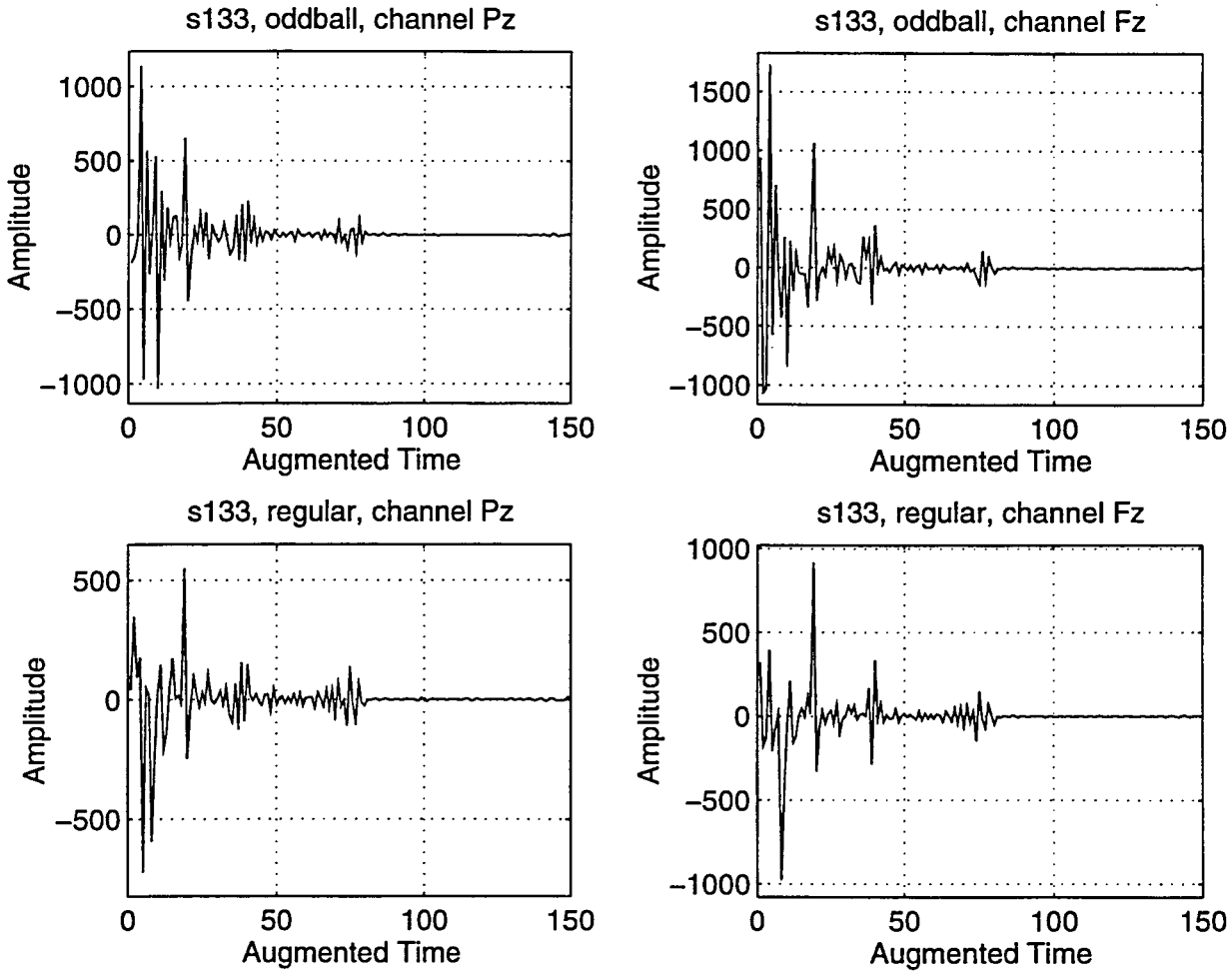

Figure C.17: Patient ID: s133, Normal

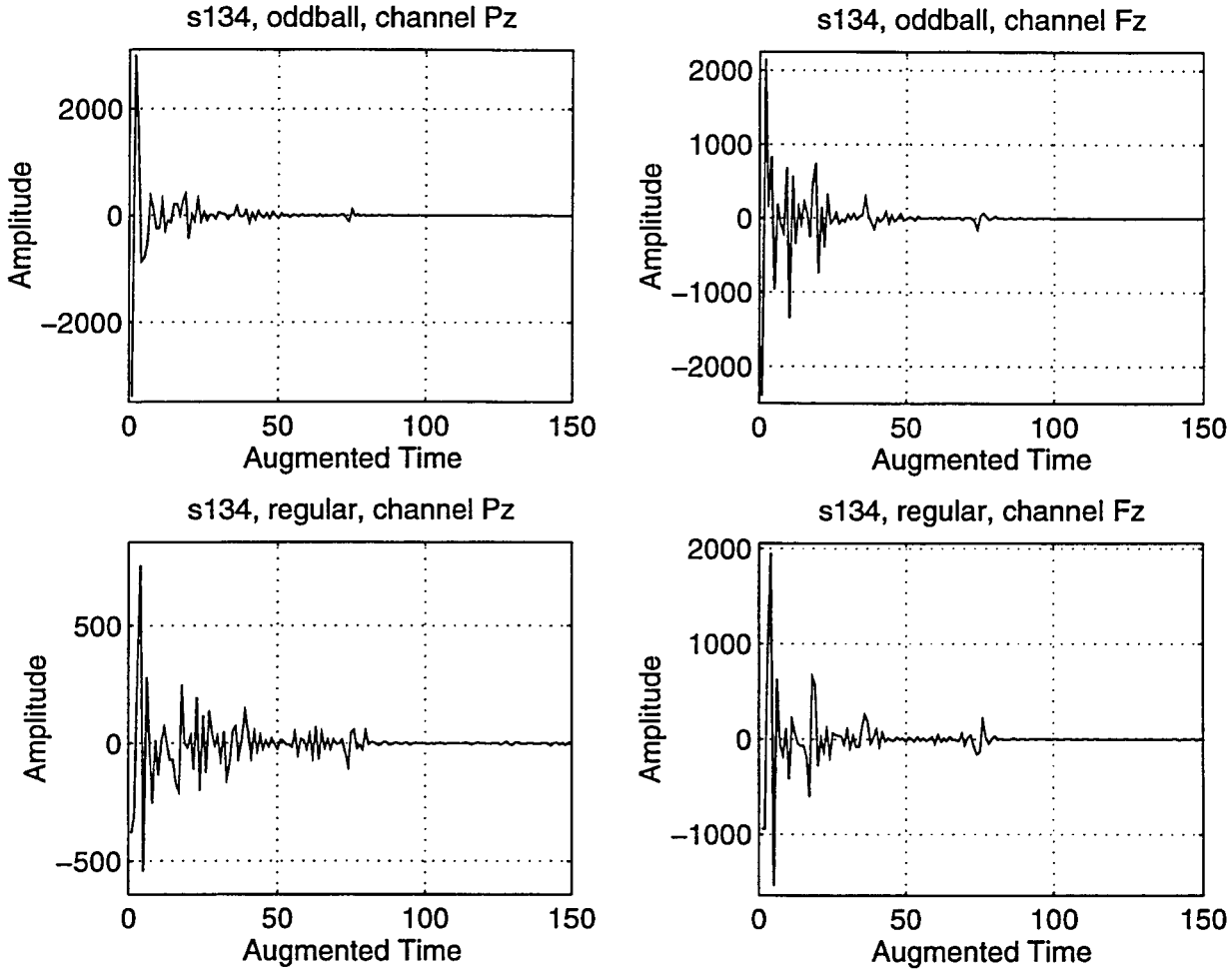

Figure C.18: Patient ID: s134, Normal

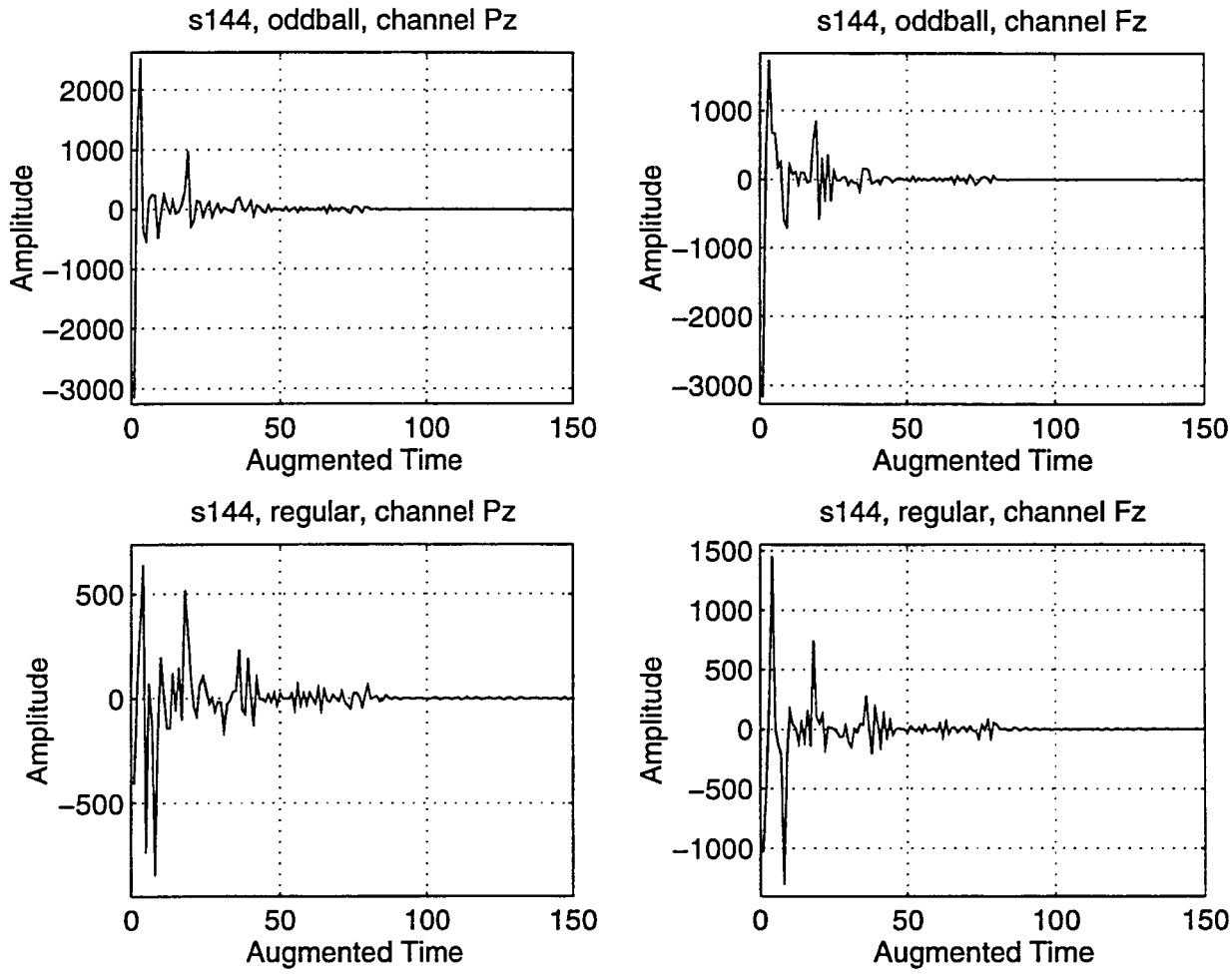

Figure C.19: Patient ID: s144, Normal

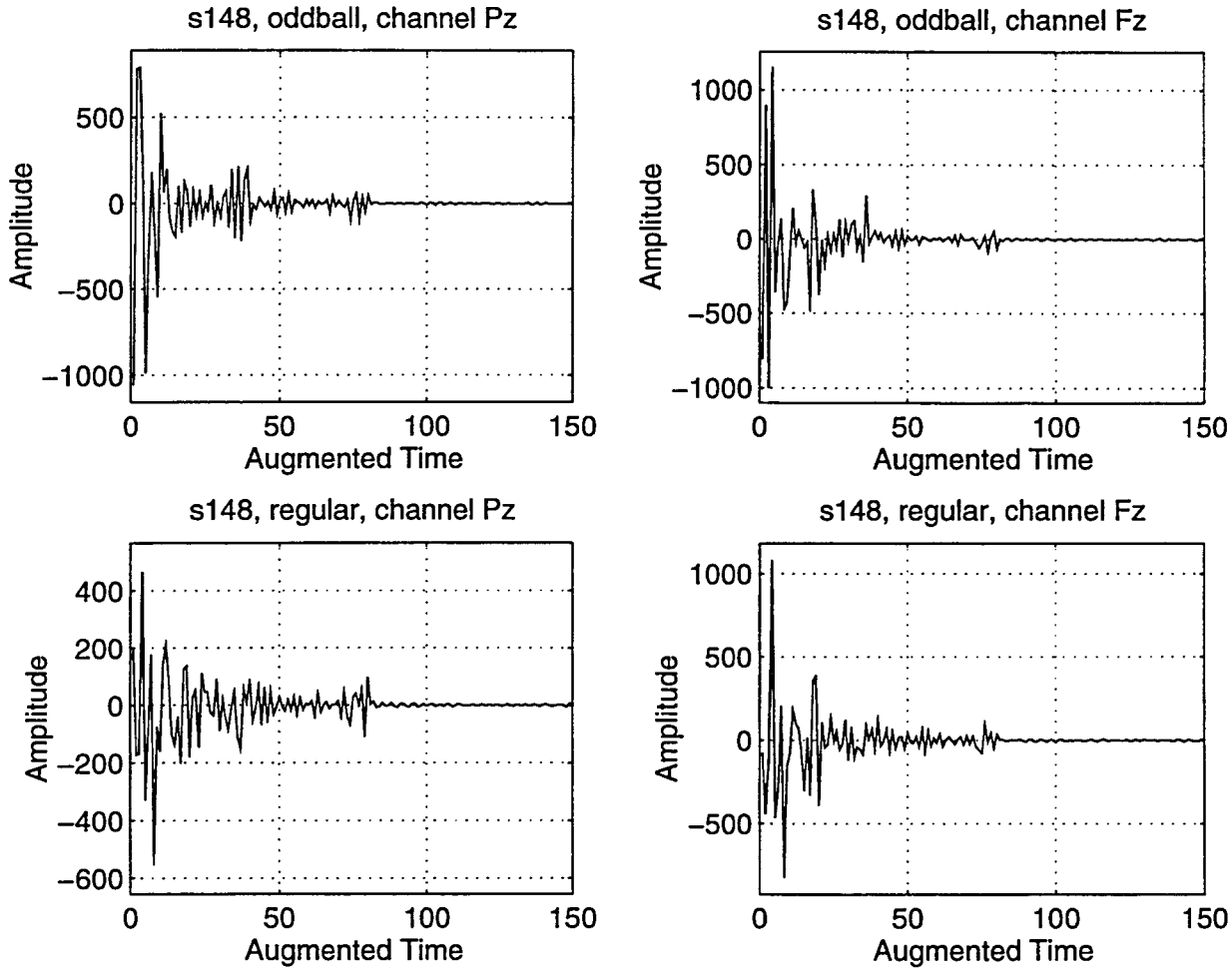

Figure C.20: Patient ID: s148, Normal

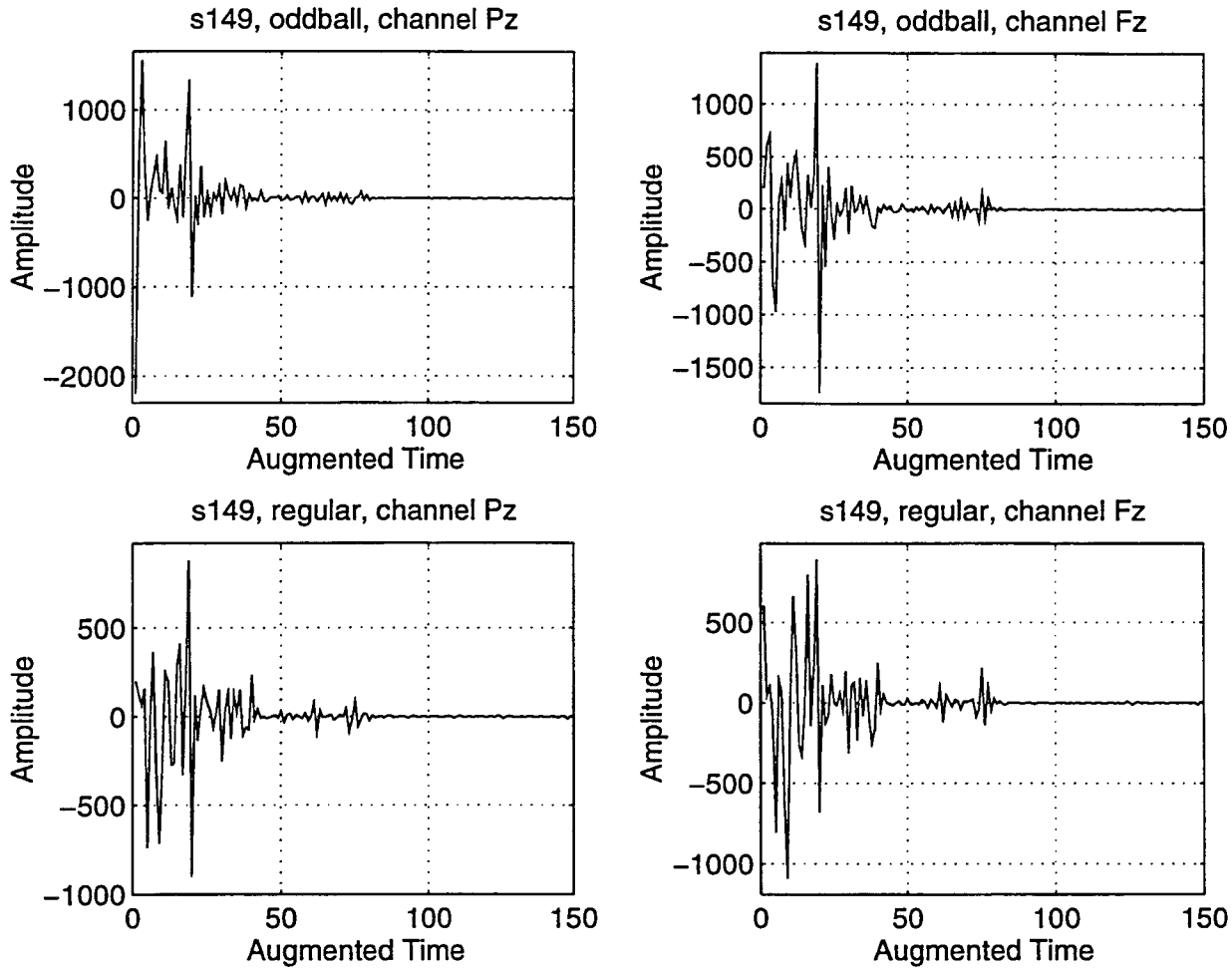

Figure C.21: Patient ID: s149, Alzheimer's

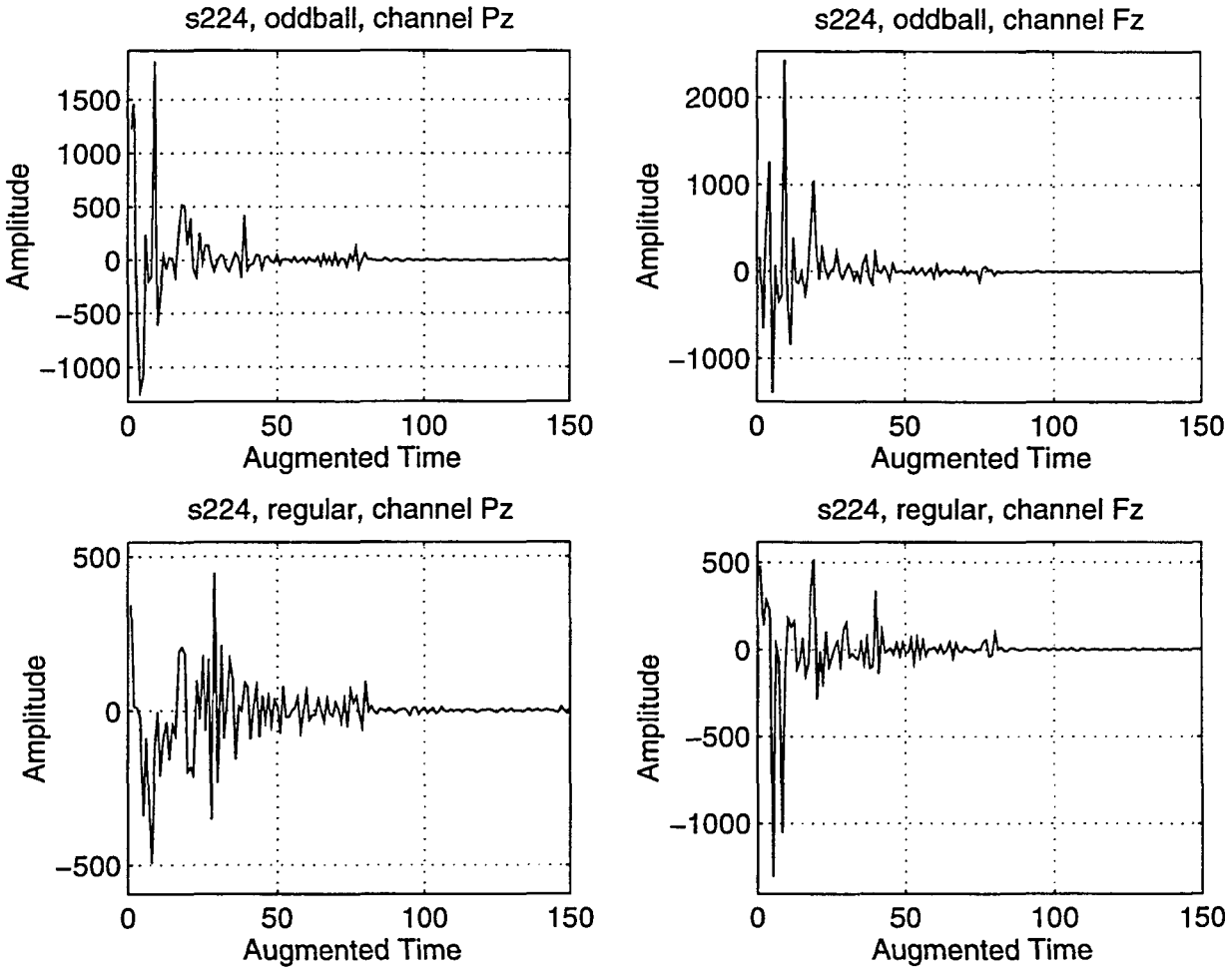

Figure C.22: Patient ID: s224, Normal

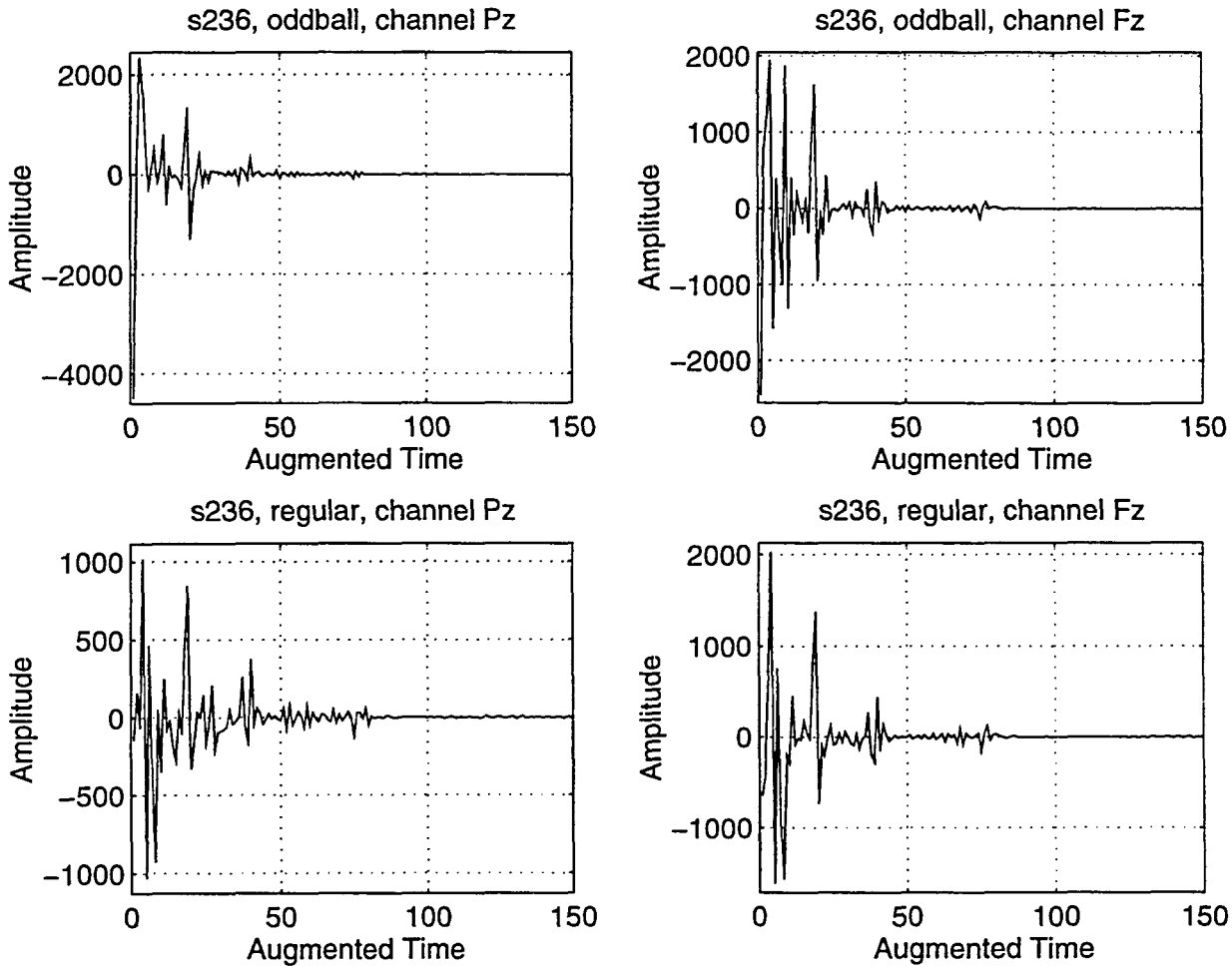

Figure C.23: Patient ID: s236, Normal

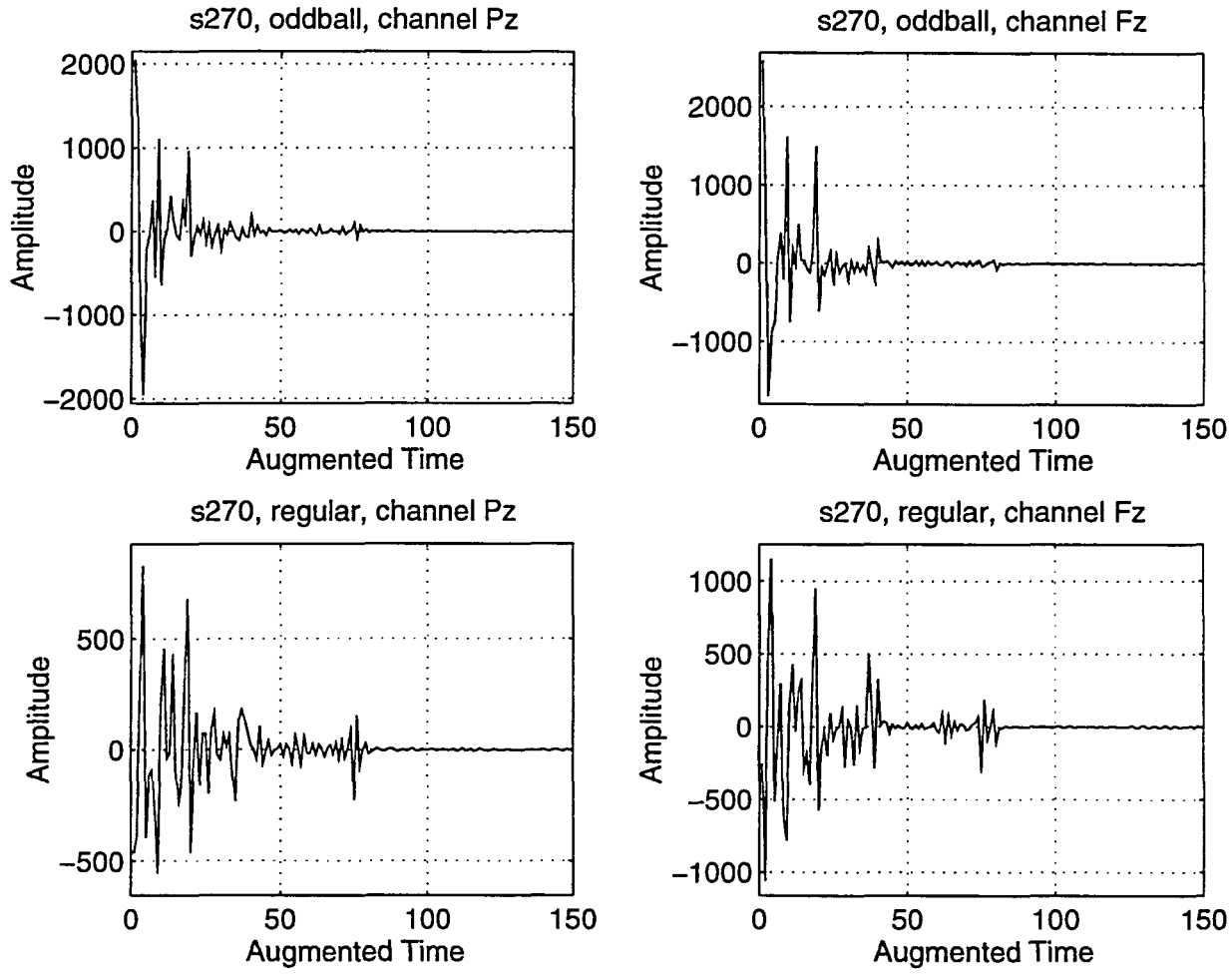

Figure C.24: Patient ID: s270, Alzheimer's

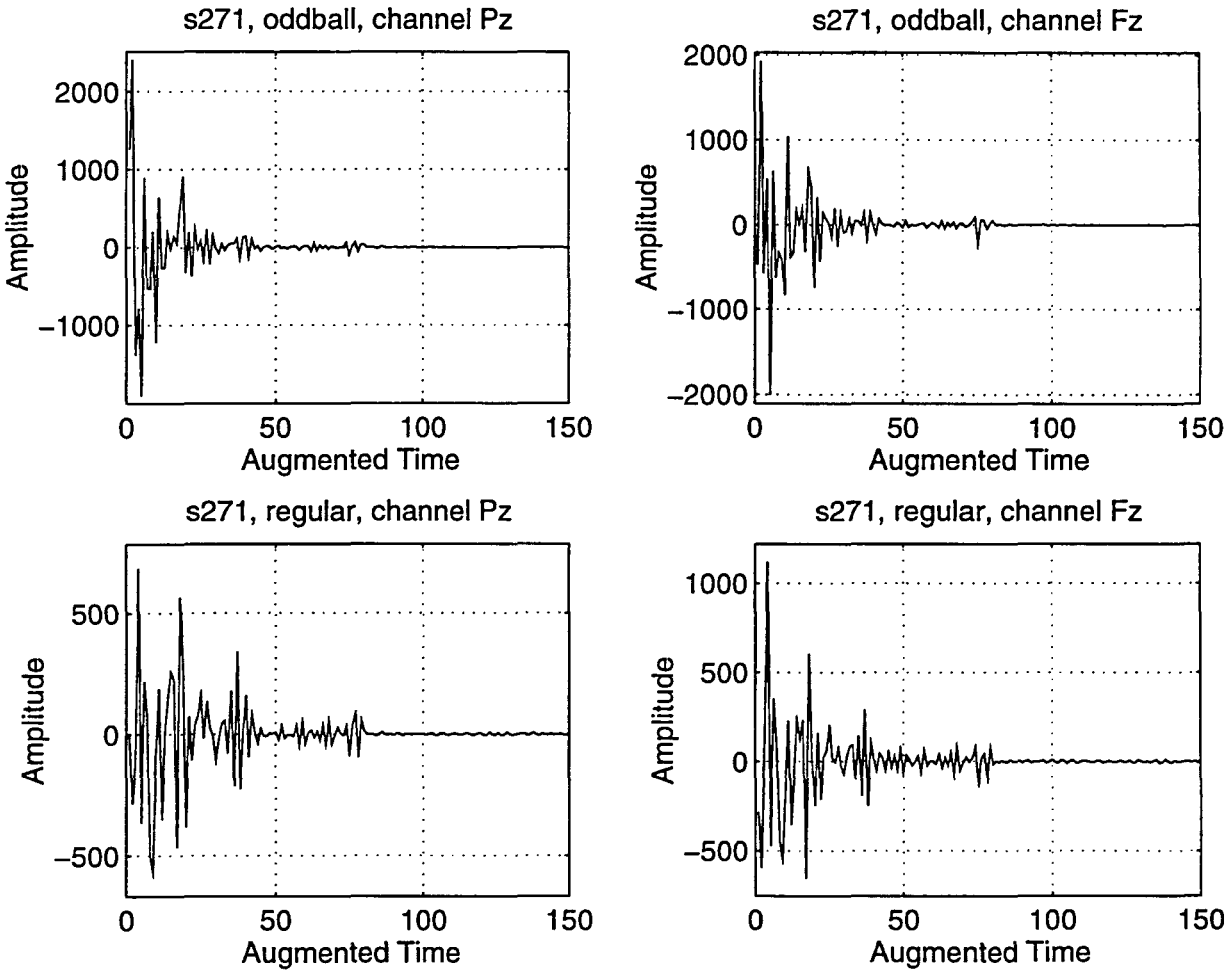

Figure C.25: Patient ID: s271, Alzheimer's

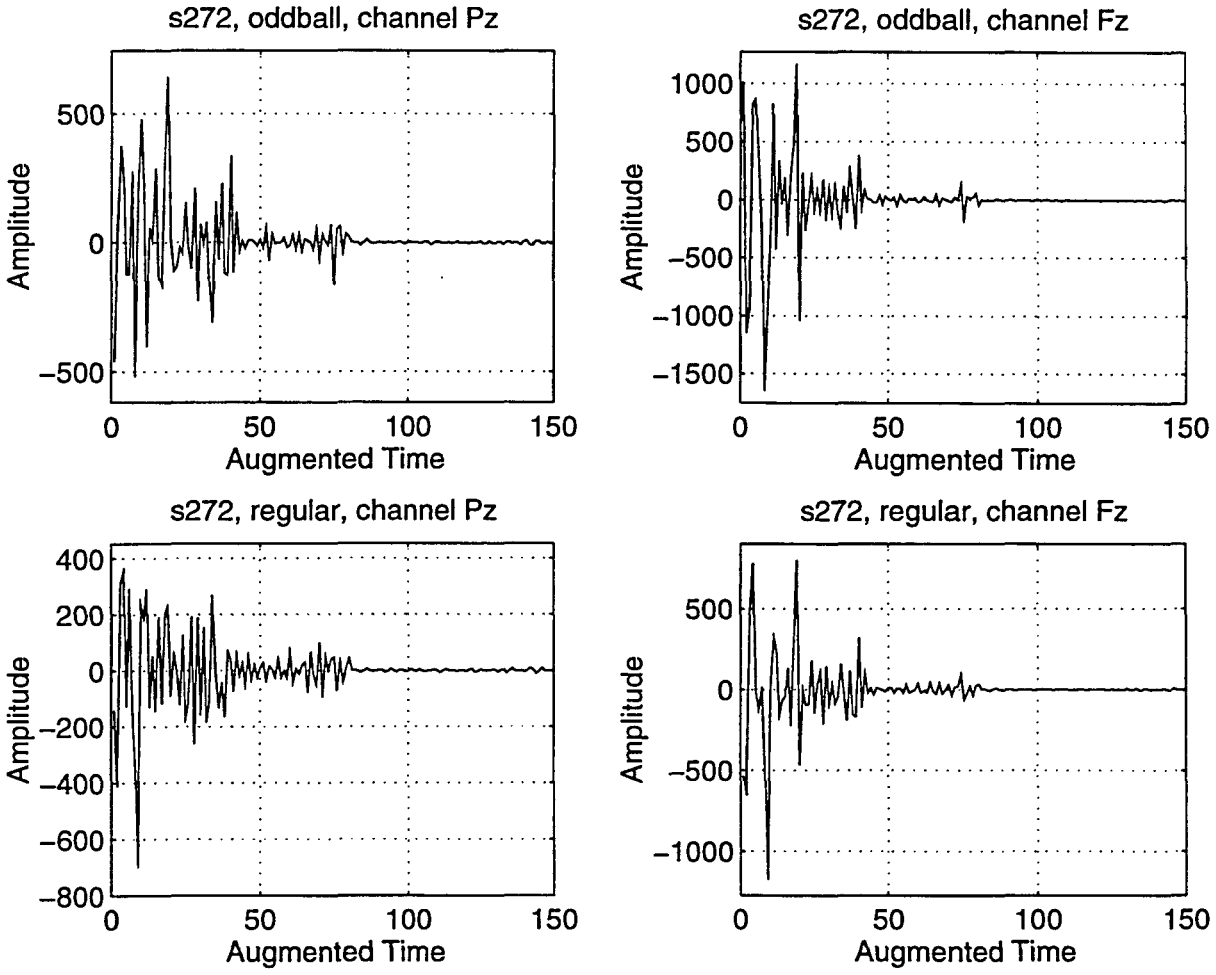

Figure C.26: Patient ID: s272, Alzheimer's

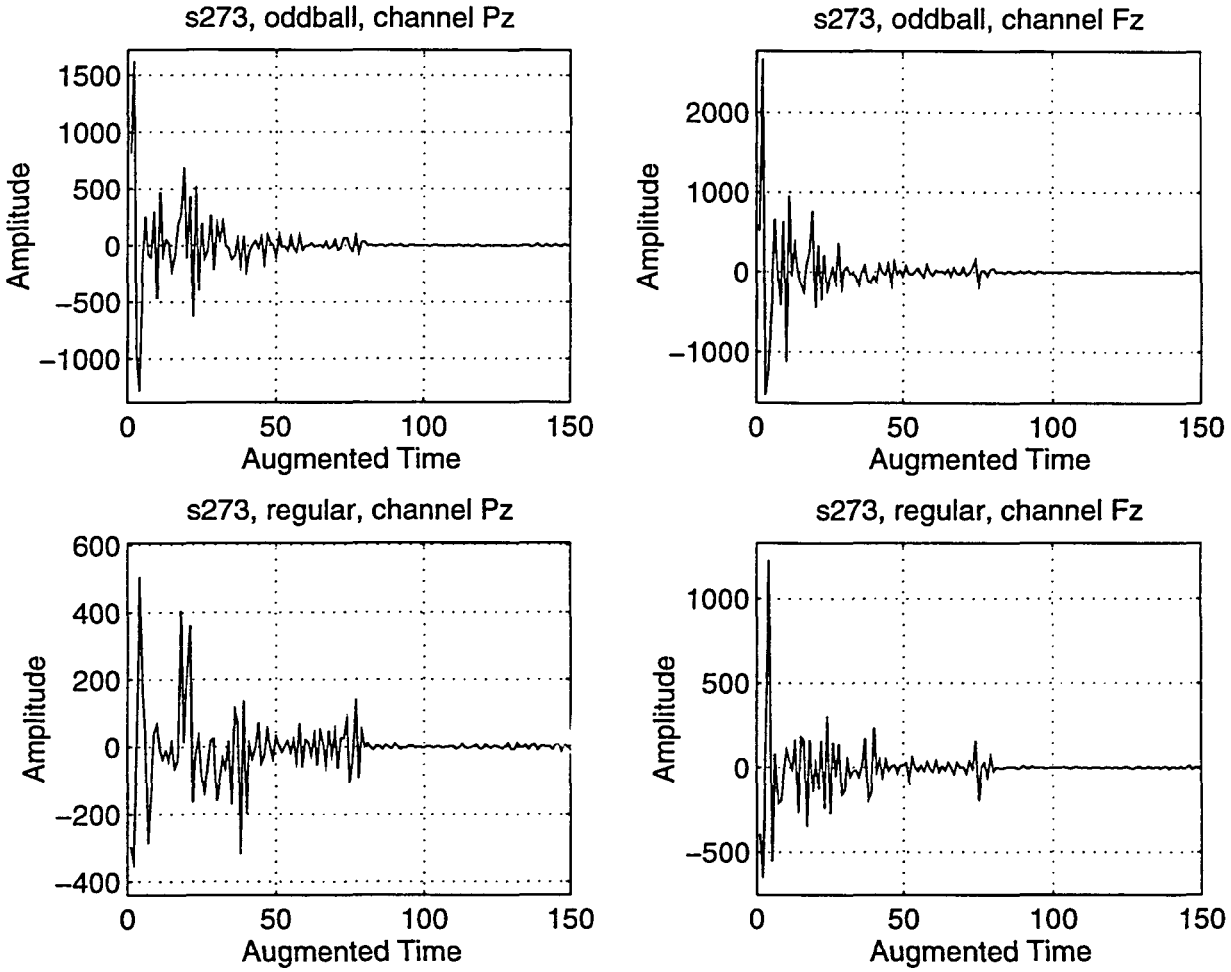

Figure C.27: Patient ID: s273, Alzheimer's

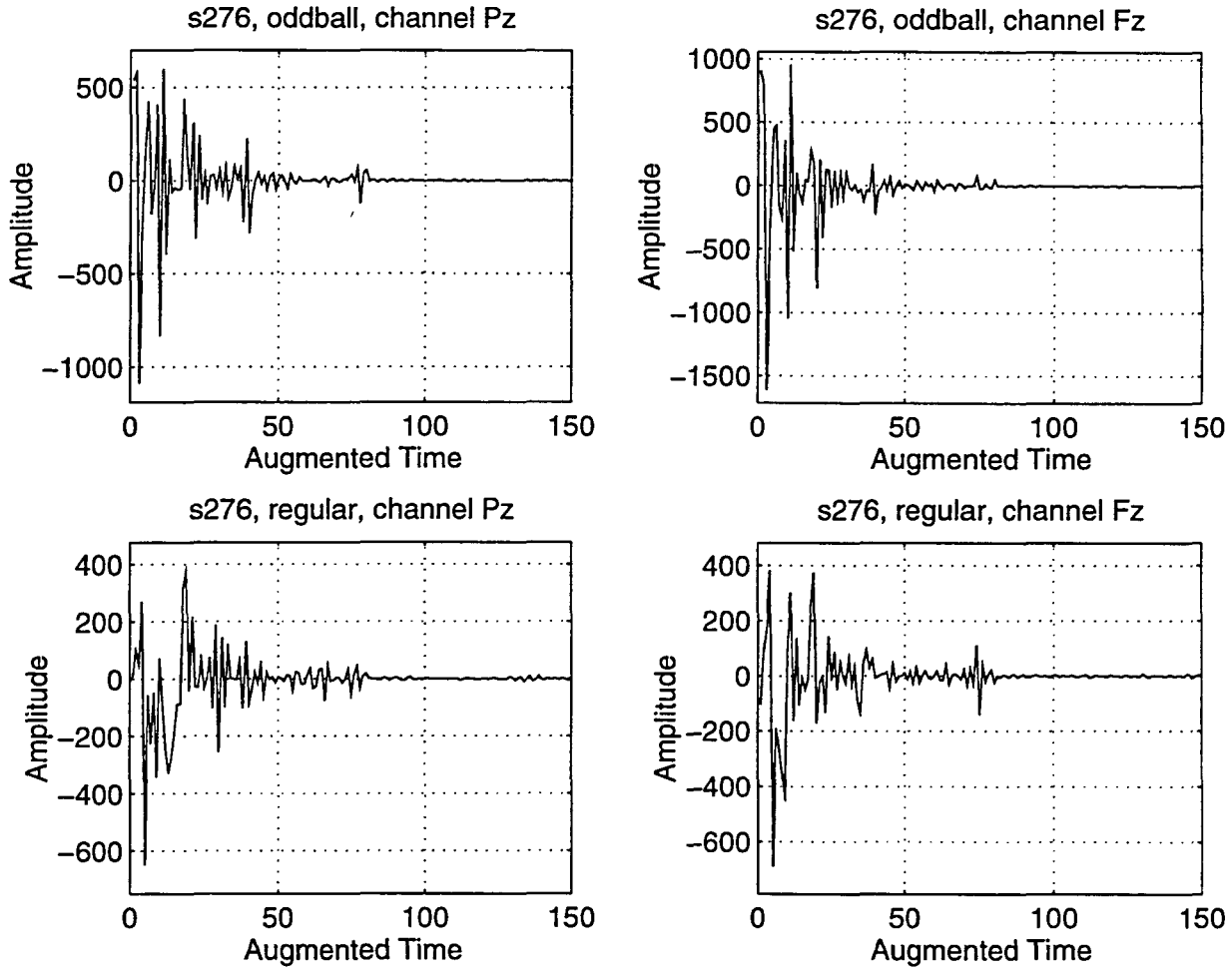

Figure C.28: Patient ID: s276, Alzheimer's

APPENDIX D. CONTINUOUS WAVELET TRANSFORM

```
%CONTINUOUS WAVELET TRANSFORM SCHEME
%02/12/1995
%ROBI POLIKAR

close all
clear
load s048wt.pzo

plot(f)


figure

timestepsize=input('Please enter the time step size: ');
scalestepsize=input('Please enter the scale step size: ');
```

```
filesize=750;
numberofsteps=750/timestepsize;
integral=zeros(1,numberofsteps);
scale=input('Please enter the max scale: ');
WT=zeros(scale/scalestepsize,numberofsteps);
%to=-255;
%t1=-255:256;
to=-(filesize/2)+1;
t1=(-filesize/2)+1:filesize/2;
a=1;
counter=1;
for s=1:scalestepsize:scale;
  %numberofsteps=round(512/stepsize(s));
    for n=1:numberofsteps
        morlet=exp(-i*((t1-to)/(0.2*s))-(((t1-to)/(0.2*s)).^2)/2);
        ff=f.*(morlet);
        integral(n)=sum(ff);
        to=to+timestepsize;
     end
       WT(counter,:)=(1./sqrt(s))*integral;
       s
       to=-(filesize/2)+1;
       integral=0;
       counter=counter+1;
```

```
end


mesh(abs(WT));
figure
mesh(real(WT))
title('CWT OF THE TEST INPUT SIGNAL')
xlabel('Translation')
ylabel('Frequiency')
zlabel('Amplitude')
```

APPENDIX E. DISCRETE WAVELET TRANSFORM

```
%DISCRETE WAVELET TRANSFORM OF THE AVERAGED SIGNALS
%02/22/1995
%by ROBI POLIKAR

%This program computes the discrete wavelet transforms of all the
%averaged signals in the EEG database. The DWT computation is continued
%up to 7 levels. The DWT of each signal is stored in a file having the same
%length with the original signal. Augmented format is used for the
%transformed signals. The actual DWT computation is performed by the
%"fwt" function called from this program. The fwt() and daubechies()
%functions are written by Fritz Keinert.


%pzo:oddball files from lead Pz
%fzo:oddball files from lead Fz
%pzr:regular files from lead Pz
%pzr:regular files from lead Fz
```

```
clear

poddfiles = ['s048a1v.pzo';'s049a1v.pzo';'s070a1v.pzo';'s072a1v.pzo';...
's074a1v.pzo';'s077a1v.pzo';'s080a1v.pzo';'s081a1v.pzo';'s111a1v.pzo';...
's115a1v.pzo';'s120a1v.pzo';'s121a1v.pzo';'s124a1v.pzo';'s127a1v.pzo';...
's129a1v.pzo';'s132a1v.pzo';'s133a1v.pzo';'s134a1v.pzo';'s144a1v.pzo';...
's148a1v.pzo';'s149a1v.pzo';'s224a1v.pzo';'s236a1v.pzo';'s270a1v.pzo';...
's271a1v.pzo';'s272a1v.pzo';'s273a1v.pzo';'s276a1v.pzo'];

foddfiles = ['s048a1v.fzo';'s049a1v.fzo';'s070a1v.fzo';'s072a1v.fzo';...
's074a1v.fzo';'s077a1v.fzo';'s080a1v.fzo';'s081a1v.fzo';'s111a1v.fzo';...
's115a1v.fzo';'s120a1v.fzo';'s121a1v.fzo';'s124a1v.fzo';'s127a1v.fzo';...
's129a1v.fzo';'s132a1v.fzo';'s133a1v.fzo';'s134a1v.fzo';'s144a1v.fzo';...
's148a1v.fzo';'s149a1v.fzo';'s224a1v.fzo';'s236a1v.fzo';'s270a1v.fzo';...
's271a1v.fzo';'s272a1v.fzo';'s273a1v.fzo';'s276a1v.fzo'];


pregfiles = ['s048a1v.pzr';'s049a1v.pzr';'s070a1v.pzr';'s072a1v.pzr';...
's074a1v.pzr';'s077a1v.pzr';'s080a1v.pzr';'s081a1v.pzr';'s111a1v.pzr';...
's115a1v.pzr';'s120a1v.pzr';'s121a1v.pzr';'s124a1v.pzr';'s127a1v.pzr';...
's129a1v.pzr';'s132a1v.pzr';'s133a1v.pzr';'s134a1v.pzr';'s144a1v.pzr';...
's148a1v.pzr';'s149a1v.pzr';'s224a1v.pzr';'s236a1v.pzr';'s270a1v.pzr';...
's271a1v.pzr';'s272a1v.pzr';'s273a1v.pzr';'s276a1v.pzr'];
```

```
fregfiles = ['s048a1v.fzr';'s049a1v.fzr';'s070a1v.fzr';'s072a1v.fzr';...
's074a1v.fzr';'s077a1v.fzr';'s080a1v.fzr';'s081a1v.fzr';'s111a1v.fzr';...
's115a1v.fzr';'s120a1v.fzr';'s121a1v.fzr';'s124a1v.fzr';'s127a1v.fzr';...
's129a1v.fzr';'s132a1v.fzr';'s133a1v.fzr';'s134a1v.fzr';'s144a1v.fzr';...
's148a1v.fzr';'s149a1v.fzr';'s224a1v.fzr';'s236a1v.fzr';'s270a1v.fzr';...
's271a1v.fzr';'s272a1v.fzr';'s273a1v.fzr';'s276a1v.fzr'];

[nrow ncol]=size(poddfiles); %nrow=28, for each set

for i=1:nrow,

fnamePo = poddfiles(i,1:find(poddfiles(i,:) == '.')-1);
fnameFo = foddfiles(i,1:find(foddfiles(i,:) == '.')-1);
fnamePr = pregfiles(i,1:find(pregfiles(i,:) == '.')-1);
fnameFr = fregfiles(i,1:find(fregfiles(i,:) == '.')-1);

out_fnameFo = [fnameFo,'dt.fzo']
out_fnamePo = [fnamePo,'dt.pzo']
out_fnameFr = [fnameFr,'dt.fzr']
out_fnamePr = [fnamePr,'dt.pzr']

[h,g]=daubechies(4); % Use a daubechies wavelet with four vanishing moments
```

```
%**LOAD THE Pz CHANNEL FOR  ODDBALL FILES**

eval(['load ' poddfiles(i,:)]);
oddp=eval(fnamePo);

f1=[oddp oddp(1:40)];
W1=fwt(h,g,f1);

savestr=sprintf('save %s W1 -ascii', out_fnamePo);
eval(savestr);
clear oddp W1 f1

clearstr=sprintf('clear %s',fnamePo);
eval(clearstr);

%**************************************

%**LOAD THE fz CHANNEL FOR  ODDBALL FILES**
eval(['load ' foddfiles(i,:)]);
oddf=eval(fnameFo);

f2=[oddf oddf(1:40)];
W2=fwt(h,g,f2);
```

```
savestr=sprintf('save %s W2 -ascii', out_fnameFo);
eval(savestr);
clear oddf W2 f2

clearstr=sprintf('clear %s',fnameFo);
eval(clearstr);

%*****************************************

%**LOAD THE pz CHANNEL FOR  REGULAR FILES**

eval(['load ' pregfiles(i,:)]);
regp=eval(fnamePr);


f3=[regp regp(1:40)];
W3=fwt(h,g,f3);

savestr=sprintf('save %s W3 -ascii', out_fnamePr);
eval(savestr);
clear regp W3 f3

clearstr=sprintf('clear %s',fnamePr);
eval(clearstr);
```

```
%*******************************************

%**LOAD THE fz CHANNEL FOR  REGULAR FILES**

eval(['load ' fregfiles(i,:)]);
regf=eval(fnameFr);

f4=[regf regf(1:40)];
W4=fwt(h,g,f4);

savestr=sprintf('save %s W4 -ascii', out_fnameFr);

eval(savestr);
clear regf W4 f4

clearstr=sprintf('clear %s',fnameFr);
eval(clearstr);

%*******************************************

i
end; ** the loop over i
```

APPENDIX F. K-MEANS ALGORITHM

```
%03/01/1993
%K-MEANS ALGORITHM FOR EEGDATAS
%ROBI POLIKAR

%This program clusters the given EEG signals into two classes
%by using the k-means algorithm. All the signals from each electrode
%are concatenated to produce a matrix of 28*600. The program that
%creates the matrix is creatematrix.m .

clear
load oddp.dat
W1=oddp(2,:);  %Initialize the cluster centers
W2=oddp(1,:);
W1N=zeros(1,200);
W2N=zeros(1,200);
iteration=0;
```

```
while 1


  for i=1:28
   distance1(i)=dist(oddp(i,:),W1'); %Calculate the distance between the
   distance2(i)=dist(oddp(i,:),W2'); %signal and cluster center

   if (distance1(i) < distance2(i))  %minimum distance determines the class
       class(i)=1;
       cluster1=[cluster1; oddp(i,:)]; %signals in this cluster gathered
       end

   if (distance1(i) > distance2(i))
       class(i)=2;
       cluster2=[cluster2; oddp(i,:)];
       end
  end

  N1=nnz(class(:)-2); %number of signals in cluster 1
  N2=nnz(class(:)-1); %number of signals in cluster 2

  W1N=1/N1 *(sum(cluster1)); %calculate new cluster centers
  W2N=1/N2 *(sum(cluster2));
```

```
  if(W1N==W1 & W2N==W2), break, end;
  W1=W1N;
  W2=W2N;
  iteration=iteration+1;
  iteration
  cluster1=[];
  cluster2=[];
end
```

APPENDIX G. CREATE MATRIX

```
%CONVERT THE FILES INTO MATRICES FOR EACH ELECTRODE COMBINATION
%02/24/1995
% by ROBI POLIKAR

%This program concatenates the 28 files from each electrode combination
%to create one matrix per electrode combination. This matrix is used in
%k-means clustering algorithm and neural network classification with
%multilayer perceptron backpropagation


%pzo:oddball files from lead Pz
%fzo:oddball files from lead Fz
%pzr:regular files from lead Pz
%pzr:regular files from lead Fz
```

```
poddfiles = ['s048a1vd.pzo';'s049a1vd.pzo';'s070a1vd.pzo';'s072a1vd.pzo';...
's074a1vd.pzo';'s077a1vd.pzo';'s080a1vd.pzo';'s081a1vd.pzo';'s111a1vd.pzo';...
's115a1vd.pzo';'s120a1vd.pzo';'s121a1vd.pzo';'s124a1vd.pzo';'s127a1vd.pzo';...
's129a1vd.pzo';'s132a1vd.pzo';'s133a1vd.pzo';'s134a1vd.pzo';'s144a1vd.pzo';...
's148a1vd.pzo';'s149a1vd.pzo';'s224a1vd.pzo';'s236a1vd.pzo';'s270a1vd.pzo';...
's271a1vd.pzo';'s272a1vd.pzo';'s273a1vd.pzo';'s276a1vd.pzo'];

foddfiles = ['s048a1vd.fzo';'s049a1vd.fzo';'s070a1vd.fzo';'s072a1vd.fzo';...
's074a1vd.fzo';'s077a1vd.fzo';'s080a1vd.fzo';'s081a1vd.fzo';'s111a1vd.fzo';...
's115a1vd.fzo';'s120a1vd.fzo';'s121a1vd.fzo';'s124a1vd.fzo';'s127a1vd.fzo';...
's129a1vd.fzo';'s132a1vd.fzo';'s133a1vd.fzo';'s134a1vd.fzo';'s144a1vd.fzo';...
's148a1vd.fzo';'s149a1vd.fzo';'s224a1vd.fzo';'s236a1vd.fzo';'s270a1vd.fzo';...
's271a1vd.fzo';'s272a1vd.fzo';'s273a1vd.fzo';'s276a1vd.fzo'];


pregfiles = ['s048a1vd.pzr';'s049a1vd.pzr';'s070a1vd.pzr';'s072a1vd.pzr';...
's074a1vd.pzr';'s077a1vd.pzr';'s080a1vd.pzr';'s081a1vd.pzr';'s111a1vd.pzr';...
's115a1vd.pzr';'s120a1vd.pzr';'s121a1vd.pzr';'s124a1vd.pzr';'s127a1vd.pzr';...
's129a1vd.pzr';'s132a1vd.pzr';'s133a1vd.pzr';'s134a1vd.pzr';'s144a1vd.pzr';...
's148a1vd.pzr';'s149a1vd.pzr';'s224a1vd.pzr';'s236a1vd.pzr';'s270a1vd.pzr';...
's271a1vd.pzr';'s272a1vd.pzr';'s273a1vd.pzr';'s276a1vd.pzr'];

fregfiles = ['s048a1vd.fzr';'s049a1vd.fzr';'s070a1vd.fzr';'s072a1vd.fzr';...
's074a1vd.fzr';'s077a1vd.fzr';'s080a1vd.fzr';'s081a1vd.fzr';'s111a1vd.fzr';...
```

```
's115a1vd.fzr';'s120a1vd.fzr';'s121a1vd.fzr';'s124a1vd.fzr';'s127a1vd.fzr';...
's129a1vd.fzr';'s132a1vd.fzr';'s133a1vd.fzr';'s134a1vd.fzr';'s144a1vd.fzr';...
's148a1vd.fzr';'s149a1vd.fzr';'s224a1vd.fzr';'s236a1vd.fzr';'s270a1vd.fzr';...
's271a1vd.fzr';'s272a1vd.fzr';'s273a1vd.fzr';'s276a1vd.fzr'];



[nrow ncol]=size(poddfiles); %nrow=28, for each set

for i=1:nrow,


fnamePo = poddfiles(i,1:find(poddfiles(i,:) == 'a')-1);
fnameFo = foddfiles(i,1:find(foddfiles(i,:) == 'a')-1);
fnamePr = pregfiles(i,1:find(pregfiles(i,:) == 'a')-1);
fnameFr = fregfiles(i,1:find(fregfiles(i,:) == 'a')-1);


% LOAD THE Pz CHANNEL FOR  ODDBALL FILES

eval(['load ' poddfiles(i,:)]);
oddp=[oddp; eval([fnamePo 'a1vd'])];
```

```
%****************************************

% LOAD THE fz CHANNEL FOR  ODDBALL FILES
eval(['load ' foddfiles(i,:)]);
oddf=[oddf; eval([fnameFo 'a1vd'])];

%*****************************************
% LOAD THE pz CHANNEL FOR  REGULAR FILES

eval(['load ' pregfiles(i,:)]);
regp=[regp; eval([fnamePr 'a1vd'])];


%*****************************************

% LOAD THE fz CHANNEL FOR  REGULAR FILES

eval(['load ' fregfiles(i,:)]);
regf=[regf; eval([fnameFr 'a1vd'])];

%*****************************************
i
end; % the loop over i
```

```
oddp=oddp(:,(1:200));
oddf=oddf(:,(1:200));
regp=regp(:,(1:200));
regf=regf(:,(1:200));


save oddp.dat oddp -ascii
save oddf.dat oddf -ascii
save regp.dat regp -ascii
save regf.dat regf -ascii
```

APPENDIX H. TRAINING MULTILAYER PERCEPTRON WITH BACKPROPAGATION

```
%BACKPROPAGATION MLP, EEGDATAS
%03/02/1995
%ROBI POLIKAR

%This program trains a multilayer perceptron with backpropagation
%learning rule using Matlab's built-in function.


clear
load oddp.dat
P=oddp(:,(1:150))';
P1=[P(:,1) P(:,3) P(:,4) P(:,8) P(:,10) P(:,13) P(:,15) P(:,16) P(:,17)
P(:,19) P(:,20) P(:,21) P(:,24) P(:,26)];

%P1 is the randomly chosen an arbitrary training data. "class" is the
%cortresponding classes vector.
```

```
class=[1 0 ; 0 1 ; 1 0 ; 0 1 ; 1 0 ; 0 1 ; 1 0 ; 0 1 ; 0 1 ; 0 1 ;...
0 1 ; 1 0 ; 1 0 ; 1 0 ];

% [1 0] shows Alzheimer's disease, [0 1] shows normal patients.

class=class';
R=size(P,1);
Q=size(P,2);
[S1]=15; %number of hidden layer nodes
[S2]=size(class,1);

[W1,B1]=rands(S1,R); %randomly inialized weights
[W2,B2]=rands(S2,S1);

disp_freq=20;
max_epoch=1000;
err_goal=0.01;     % error goal
lr=0.12;           % learning rate
lr_inc=1.25;       % increase in learning rate
lr_dec=0.6;        % decrease in learning rate
momentum=0.95;     % momentum
err_ratio=1.04;    % error ratio
```

```
TP=[disp_freq,max_epoch,err_goal,lr,lr_inc,lr_dec,momentum,err_ratio];

[NW1,NB1,NW2,NB2,TE,TR]=trainbpx(W1,B1,'logsig',W2,B2,'logsig',P1,class,TP);

outputlayer1=logsig(NW1*P1,NB1);
outputlayer2=logsig(NW2*outputlayer1,NB2);
[a b]=max(outputlayer2);

save W1trial1.wgt NW1 -ascii  %save the calculated weights
save W2trial1.wgt NW2 -ascii
save B1trial1.wgt NB1 -ascii
save B2trial1.wgt NB2 -ascii

% TESTING

classtest=[1 1 2 1 2 2 2 2 1 1 1 2 2 1 1 2 2 2 2 2 1 2 2 1 1 1 1 1 ];

%classtest is the vector that holds the correct classes for each signal.
% "1" represents Alzheimer's disease, "2" represents normal patients

outputlayer1=logsig(NW1*P,NB1); %Use all 28 signals to test
outputlayer2=logsig(NW2*outputlayer1,NB2);
[a b1]=max(outputlayer2);
[a b]=max(classtest);
```

```
classification=b-b1;

% "b1" holds the network classification, "b" holds the correct
% classification. The difference between them is zero for correctly
% classified signals and plus/minus 1 for incorrectly classified signals.
```

APPENDIX I. FAST (DISCRETE) WAVELET TRANSFORM SCHEME

```
%This program computes the discrete wavelet transform by subband
%coding scheme. The code belongs to Frizt Keinert. Used with permission.


function w = fwt(h,g,f,larg)

%  w = fwt(h,g,f,l) = fast wavelet transform of f through l levels
%
%  Computes the fast wavelet transform of a vector f through l levels,
%  using the wavelets defined by the coefficients h, g. g and l are
%  optional arguments.
%
%  If g is missing, it is calculated by taking the h coefficients in
%  reverse, with alternating sign.
%
%  If the last number in h or g is an integer, it is assumed that it
```

```
%  is not a part of the coefficients, but the subscript of the first
%  element of h or g. (This is a kludge to get around the MATLAB
%  restriction of having all arrays start with index 1).
%
%  The length of f must be a number m*2^power, where power >= l. If l
%  is not given, the maximum permissible l is computed and used.
%
%  The results are stored in a vector w as :
%
%              l     l-1     l-2
%       w = [H f; GH   f; GH   f; ...; GHf; Gf],
%
%  where H is the filter defined by the coefficients h,
%  and G is the filter defined by the coefficients g.

% Fritz Keinert
% June 22, 1993

% fill in optional arguments, if necessary

if (nargin < 2)
    error('FWT: must have at least two arguments');
end
```

```
if (nargin == 2) % arguments are h, f
    f = g;
    g = qmf(h);
    larg = power2(length(f));
end

if (nargin == 3)
    [d1,d2] = size(f);
    if (d1 * d2 == 1)        % arguments are h, f, larg
larg = f;
f = g;
g = qmf(h);
    else                % arguments are h, g, f
larg = power2(length(f));
    end
end

%  make sure we can do l levels of decomposition

n = length(f);
lmax = power2(n);
if (larg > lmax)
    error('FWT: l is too large for data vector length');
end
```

```
% determine lengths, offsets of h, g

hlen = length(h);
hmin = 0;
if (isint(h(hlen)))
    hmin = h(hlen);
    hlen = hlen - 1;
    h = h(1:hlen);
end

glen = length(g);
gmin = 0;
if (isint(g(glen)))
    gmin = g(glen);
    glen = glen - 1;
    g = g(1:glen);
end

len = max(hlen,glen);

% flip h, g to use with filter program (which flips them back)

h = h(hlen:-1:1);
```

```
g = g(glen:-1:1);

% Do the decomposition through larg levels

w = f;
if (len > 2)
    f(n+len-2) = 0;
end;

for l = 1:larg
    if (gmin ~= 0)
f(1:n) = shiftleft(f(1:n),gmin);
    end
    for i=1:len-2
f(n+i) = f(i);
    end
    d = filter(g,1,f);
    if (hmin ~= gmin)
f(1:n) = shiftleft(f(1:n),hmin-gmin);
for i=1:hlen-2
    f(n+i) = f(i);
end
    end
    s = filter(h,1,f);
```

```
    f(1:n/2) = s(hlen:2:hlen+n-2);
    w(n/2+1:n) = d(glen:2:glen+n-2);
    n = n/2;
end

w(1:n) = f(1:n);
```

APPENDIX J. COMPUTATION OF THE DAUBECHIES WAVELET COEFFICIENTS

```
%This program computes the Daubechies wavelet coefficients with any
%given number of vanishing moments. Daubechies wavelets with 4 vanishing
%moments were used in thsi study. The following modified code belongs to
%Fritz Keinert. Used with permission.

function [h,g] = daubechies(N)

%DAUBECHIES -  returns coefficients for Daubechies wavelet
%
%  [h,g] = daubechies(n)
%
%  Calculates the coefficients for the Daubechies wavelet with N
%  vanishing moments. The coefficients are normalized so that
%  sum(h) = sqrt(2).

%  JCK: September 9, 1991
```

```
%  Copyright (c) 1991 Jeffrey C. Kantor
%  minor modifications by Fritz Keinert, November 15, 1993

if round(N)~=N | (N<1),
  error('Order must be an integer >=1');
end

% Compute coefficients of polynomial P(y) of order N-1, store
% in vector pN is ascending order

p = 1;
pN = [p];
for j=1:N-1,
  p = (N+j-1)*p/j;
  pN = [pN,p];
end

% Now compute polynomial abs(Q(z))^2, z = exp(iw)

f = [-1 2 -1]/4;
fc = f;
q2 = pN(1);
for k=1:N-1,
  q2 = [0,q2,0] + pN(k+1)*fc;
```

```
  fc = conv(fc,f);
end

% q2 is a polynomial with coefficients in ascending order of
% powers ranging from 1-N to N-1. The coefficients of z^-k and
% z^k are the same. Roots are eigenvalues of a companion matrix.

r = eig([-q2(2:length(q2))./q2(1);eye(2*N-3),zeros(2*N-3,1)]);

% Construct q from the roots outside the unit disk

q = poly(r(find(abs(r)>1)));

% Normalize phase so that q(0) is real

q0 = q(length(q));
q = q*(conj(q0)/abs(q0));

% Everything should be real, so drop the imag parts

q = real(q);

% Now compute h by multiplying with (0.5(1+z))^N
```

```
for k = 1:N,
  q = conv(q,[1/2 1/2]);
end

% Normalize, since the normalization was lost in the factoring
% and reconstruction of q

h = sqrt(2)*q/sum(q);

% Report result in ascending order

h = h(length(h):-1:1);

% Compute the quadrature mirror filter from h

g = qmf(h);
```