The Fractal Characterisation of Phonetic Elements of Human Speech

by

Paul McDowell

A Masters Thesis submitted in partial fulfilment for the award of

Master of Philosophy

of the Loughborough University of Technology

July 1995

V8912545

Abstract

The use of fractal techniques and fractal dimensions as a means of speech characterisation and speech recognition is a relatively new concept and as such very few papers have addressed the possibilities of it's use and associated advantages and disadvantages over conventional methods.

This thesis demonstrates that fractal techniques can effectively be used as a method of broad recognition of phonetic elements in human speech. Three distinct fractal methods have been used to associate fractal dimensions with speech: The Box Counting method, The Divider or Richardson method and the Minkowski-Bouligand disc method.

Speech has been recorded by myself and another male and female speaker to provide a database of phonetic recordings that could be experimented on. The three fractal techniques were emulated by means of software programs written in a high level language.

Acknowledgements

I gratefully acknowledge the assistance and invaluable help and guidance throughout the course of this project of my supervisor, Dr. S. Datta.

I would also like to express my gratitude to Steven Foley and Susan Leech for their help in providing speech samples.

This thesis is dedicated to the memory of Andrew David Hindley.

Table of Contents

Chapter 1

Introduction

An introduction to any subject or topic related to the concept of fractals or fractal dimensions would be wholly inadequate and incomplete without some mention of the man who created a field of science around fractals in his own right. Benoit Mandelbrot [1], a French mathematician born in 1924, coined the term, *fractal*, in a paper of his that attempted to describe certain natural patterns and objects using a new and revolutionary concept.

Mountains, clouds and coastlines are prime examples of natural objects that exhibit the phenomena that Mandelbrot terms *self similar* and *scale invariant*, i.e., that at all scales of observation they appear to exhibit the same properties. His own loose definition of the term *fractal* is as follows:

'A fractal is a shape made of parts similar to the whole in some way'

In order to characterise one fractal object from another, the so called *fractal dimension*, D, of the object can be calculated. The fractal dimension is a real number which falls between the limits of 1 and 3 and can be calculated in a number of ways.

The use of a fractal dimension has been derived from the fact that the conventional topological dimension, i.e. 1, 2 or 3, has been found to be too general to be of use in many areas of science and nature when trying to distinguish between similar objects that cannot be described exactly using Euclidean mathematics. In short, the fractal dimension gives a measure of the degree of irregularity or roughness for an object.

1.1 Fractals and Speech

The fact that such complicated structures such as those described can be characterised by a single number has led to work being carried out in the area of acoustic and speech science. Speech waveforms themselves are highly irregular patterns which can be quantified using fractal mathematics.

In 1986 Pickover and Khorsani [2] attempted to characterise whole sentences using fractal mathematics and concluded that, irrespective of speaker or sentence, the fractal dimension, D, for speech samples in the range 10 ms to 2 seconds remained constant at about 1.66.

In 1991 Petros Maragos [3] published a paper which examined the problem in more depth. In it he proposes that the degree of turbulence in speech can be measured using fractal mathematics. He concludes that D is not strictly constant for speech in the time scales that he uses and that certain phonemes, such as unvoiced fricatives and affricates, have consistently much higher fractal dimensions than for vowel sounds and that voiced fricatives can be distinguished and segmented from vowel sounds.

It is important to note at this point the scope and accuracy of characterisation or recognition that can reasonably be expected using fractal techniques and fractal dimensions. As stated previously the laws of fractal dimensions bounds us to measurements of between 1 and 3, however as we are only concerned with all possible waveforms between dc (1 dimensional) and white noise (2 dimensional), then clearly we can only expect our measurements to exist between those limits. If the intention was to suppose that it would be possible to recognise every word of the English language then we would need a method of calculating fractal dimensions consistently to many decimal points for each word. To suppose an individual even utters every word of the English language as consistently as that is unrealistic and as calculation of the fractal dimension itself involves linear regression techniques and best line fitting to a series of measured points then clearly this level of recognition is seemingly impossible.

Disregarding the possibility of such a level of recognition then we have to examine more closely the actual features of speech that make up our words. At a simplistic broad level that could merely mean distinguishing between vowels and consonants. If this were successful then we may go on to propose that diphthongs could be distinguished from pure vowels or fricative consonants distinguishable from plosives. Perhaps, further still, that vowel sounds could be distinguished in terms of position of the tongue and that all the consonants could be distinguished by place and manner of articulation, voiced or unvoiced.

Even this level of characterisation would require a very consistent algorithm and testing procedure if one were to attempt it using fractal dimension techniques.

1.2 Aim of this Research

The purpose of this research project was to follow on from the work carried out by Pickover and Khorsani [2] and by Maragos [3] and to investigate whether classes of phonemes or phonetic elements of the English language could be consistently segmented using fractal dimension techniques and whether elements in those classes could themselves be distinguished.

Speech wave forms of the time scale 30 ms to 70 ms have been extensively examined, this time scale covering the duration of usual phonetic utterances. Three fractal techniques, the Box Counting method, the Minkowski Bouligand method and the

Divider or Richardson method have been described and used to quantify the phonetic elements.

The speech elements tested were provided by myself and by another male and a female speaker to gain a measure of correlation and to investigate the speaker independence of the system.

1.3 The Structure of this Thesis

The structural content of this thesis is as follows:

In Chapter 2 the previous work carried out on the subject of speech and fractals is reviewed and discussed.

In Chapter 3 the mechanisms of speech production are discussed in length in terms of the function of the vocal organs. The methods by which the phonetic elements are linguistically classified and organised is then also described.

Chapter 4 gives a detailed description of the definition and origins of fractals and the fractal dimension. The mathematics for calculating the three fractal methods described is also then given.

Chapter 5 gives an account of all the experimental procedures and tests carried out.

Chapter 6 discusses all the results obtained for the three methods and for each speaker.

Chapter 7 provides an analytical review of the results.

Finally, Chapter 8 gives an assessment of the research discussed in this thesis, providing a conclusion to the work and a discussion on possible future work.

References

[1] Mandelbrot B.B. 1982 'The Fractal Geometry of Nature' W.H. Freeman and Company New York

[2] Pickover C., Khorsani A. 1986 'Fractal Characterization of Speech Waveform Graphs' Computer & Graphics Vol.10 No. 1, pp 51-61

[3] Maragos P. 1991 'Fractal Aspects of Speech Signals : Dimension and Interpolation' Proceedings IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP) Vol. 1 pp 417-424

[4] Feder J. 1988 'Fractals' Penum Press New York and London

Chapter 2

Review

2.1 Introduction

The purpose of this chapter is to review and discuss the related work that has been previously carried out in the area of speech recognition. The actual work on speech and fractals will then also be discussed.

2.2 A Brief History of Speech Recognition

Since the turn of the century, when it was thought that speech signals and any other acoustic waves or electromagnetic waves could be described as a collection of simple sinusoids, research has addressed the complex and difficult problem of producing a machine that could recognise or interpret speech and in response perform a valid task.

Primarily, technical difficulties aside, the problem for the engineer and scientist lay in the fact that different people speak in different ways and even the same speaker will vary his or her pronunciations under different conditions. Thus, the early research tended to focus on isolated word recognition using a limited vocabulary and a limited number of speakers in a good acoustic environment.

2.2.1 Isolated Word Recognition

In the 1950's a number of machines had been developed that worked principally by dividing the speech up into a number of frequency bands. These bands could then be processed and compared with existing stored speech patterns.

The first use of a digital computer in recognition was introduced in 1960 by Denes and Mathews as reported by Wayne [1] who used time normalisation techniques to improve the efficiency of their system. Programs were then also written on computers in the early part of that decade that could recognise formant frequencies in vowels.

The 1960's also saw the development of the special purpose hardware devices for speech recognition of isolated words. One such device was developed by IBM which was developed as a marketable product being of minimal size and cost.

In 1972 the first commercial isolated word speech recognisers appeared, developed by Scope Electronics and Threshold Technology. Their devices were capable of operating with success rates of up to 99% in laboratory conditions.

In 1977 the concept of syntax was introduced by Coler as reported by Wayne [1] in an attempt to limit the number of possible commands that a recogniser had to distinguish at any point in time which made it possible to have larger vocabularies. This was followed by a system developed by Bell that could recognise speech spoken from a number of independent speakers over the telephone.

2.2.2 Continuous Speech Recognition

While these isolated word systems were being developed research was also turning increasingly to the more difficult problem of continuous speech recognition. It was collectively agreed that ultimately, because of the way humans generally converse, continuous word recognition systems would be the natural and most desirable form of solution to the speech recognition problem.

In 1966 one such system was developed by Otten as reported by Wayne [1] using syllabic units, prosodic and Markov models to represent the structure of speech dialogue. Other systems involved phonetic segmentation and a success rate of up to 80% was recorded by Reddy and Vicens as reported by Wayne [1] using a 16 word vocabulary. Further systems were then developed which used the distinctions in phonemes such as voiced or unvoiced, turbulent or non turbulent and high or low as a signature which could then be matched by a computer program to existing patterns.

In 1971 the Advanced Research Projects Agency (ARPA) commissioned a group, Speech Understanding Research (SUR) consisting of five initial contractors, to begin a five year multi million dollar project which would focus on bringing advances in artificial intelligence and computational linguistics to the task of having the machine comprehend the full linguistic structure and producing the valid response. The project called for success rates over 90% in ideal conditions using sentences comprised of a 1000 word vocabulary. In 1976 the results were unveiled and a number of the groups had produced systems which met or even exceeded the specifications, notably the Carneige-Mellon University with their 'Harpy' and 'Hearsay II' [1] systems and Bolt Beranek and Newman Inc. with their 'Hear What I Mean' (HWIM) [1] system. Harpy's success was largely due to a new concept of incorporating knowledge about the acoustic properties of speech sounds and phonetic composition of words using a special network. The network structure permitted very rapid examination of many alternative word sequences before selecting the best fitting sequence as the sentence spoken by the user.

In 1973 IBM researchers developed a system known as 'ARCS' which segmented contiguous speech into short transitional elements called transemes. Such transemes capture some of the coarticulatory effects of flowing speech thus being more detectable than normal phonetic units. In their later work IBM researchers included both the transeme and conventional phonetic segmentation, working together in a dual classifier. They reported 89% correct labelling of general phonetic categories and 64% correct identification of specific phone labels. The 'ARCS' system was the first to perform automatic recognition of continuous speech based on a substantial command language.

Several systems have been specifically developed for recognising sequences of numbers and certain words, usually for security purposes. In 1976, at the Bell Laboratories, Rabiner and Sambur as reported by Wayne [1] developed a system that could detect boundaries between adjoining digits by finding unvoiced portions of the speech as well as observing significant dips in energy.

2.2.3 Word Spotting Systems

As the progress towards continuous speech developed, other simpler systems that rely on only detecting key information-carrying words have also been developed. Such key words are usually stressed and well articulated and coarticulation with surrounding words is minimised thus easing the problem. However, the conversations that are to be analysed for word spotting are spoken over noisy communication channels by arbitrary speakers thus making the task still somewhat difficult. Such systems must detect words in the context of any other words without being sensitive to speaker differences. Early work on word spotting was concerned with detecting linguistically invariant units such as phonemes or phonetic classes and matching phonetic strings with those analysed in the incoming speech.

Dialog Systems as reported by Wayne [1] developed a word spotter with the goal of 90% detection of key words and up to 5 false detections per hour for use over telephone lines and radio links. The technique used was based on the extension of an algorithm originally designed for isolated word recognition and to detect half-syllable sized units whose sequence made up the single word 'Kissinger'. Tests with nine speakers reading a news script gave 90%-95% detection rates with 4-6 false detections per hour against the limited number of test voices. Further research in this area showed that it was in fact more difficult to detect the word with a wider range of speakers and that detecting other shorter words was much less successful.

2.3 Related Work

At the beginning of this research project, in October 1992, a literature survey was carried out that revealed three papers had been written concerned with or relating to aspects of fractals and speech [2, 3, 4]. A further reference was also located as a chapter of a PhD thesis [5].

2.3.1 The Fractal Characterization of Speech Waveform Graphs

The first known paper was produced by Cliff Pickover and Al Khorsani [2] in 1985 at IBM's Thomas J. Watson Research Centre in New York state. Prompted in part by the earlier work of a colleague, Richard Voss [6] and his work on flicker noise, they experimented with real and synthetic speech data in the time range of 10 milliseconds to 2 seconds. This time scale represents the area in which important prosodic (pitch, amplitude, stress and intonation) and phonetic events occur. The aim of their work was to use a fractal method as a means of producing quantitative data to compare speech signals to gain more of an understanding of speech itself and thus to improve synthetic speech.

The speech signals studied were digitally recorded at a sampling rate of 10 kHz and then graphically displayed in a polar form by mapping the amplitude of the speech wave into radius and time into angle. This had the effect of producing a so called *speech island* [2] which emphasised the texture of the edge of the waveform and facilitated comparisons with other closed curve natural objects. From these graphical displays they concluded that speech was indeed fractal because increasing the magnification of the waveform revealed more and more detail. A second conclusion, that speech exhibits self similarity (in a statistical sense), was also drawn.

Using the Box Counting [8] method to compute fractal dimensions and a normalisation technique to eliminate differences in gain, whole sentences were examined as spoken by men, women and machine. It was found that for human speech, the fractal dimension, $D \approx 1.66$ averaging sentence utterances for both male and female speakers of 2 seconds duration regardless of pitch. In comparison it was found that the digital speech synthesiser used gave a fractal dimension, D, of ≈ 1.57. This led to the conclusion that a fractal technique was not particularly useful for comparing the two. However, it was found that D did differ significantly for vocal stress in vowels and also in nasal phonetic elements between human and machine.

Pickover's and Khorsani's [2] overall conclusion to their work was that the fractal approach to speech characterisation provided a versatile, reliable method for qualitatively and quantitatively examining speech. Although not useful for comparing human and synthetic speech there was evidence to suggest that it could be useful for improving the production of synthetic stressed and nasal phonetic elements.

2.3.2 Fractal Aspects of Speech Signals : Dimension and Interpolation

Following on from the work of Pickover and Khorsani [2], Petros Maragos [3] wrote a paper in 1991 which examined the link between speech and it's associated fractal dimensions in much greater depth. The aim of his research was to conceptually equate the amount of turbulence in a speech waveform with its fractal dimension.

Turbulence and turbulent flow has long been the subject of much discussion and research by mathematicians investigating the possibility of quantifying turbulence using fractal methods. Mandelbrot [7], in particular, has inferred that several geometrical aspects of turbulence are fractal.

Using the Minkowski-Bouligand [8] method to calculate the fractal dimension, Maragos [3] sampled words at 30 kHz and then experimented on time scales of between 1/15 - 6 ms resulting in the calculation of a *Local Fractal Dimension* as opposed to a *global dimension* that Pickover and Khorsani [2] were estimating for whole sentences. His reason for recording at such a high sampling rate was to preserve the fragmentation of the sampled signal as close as possible to that of the continuous time signal.

In conclusion to his findings, Maragos states that there is evidence to suggest that the fractal dimension of a phonetic element's speech waveform is indeed linked to the amount of turbulence associated with the production of that sound. His results show that unvoiced fricatives such as /f/, /θ/ and /s/, affricates, stops (during their turbulent phase) and some voiced fricatives such as /z/ have a high fractal dimension (1.6 $\Rightarrow$ 1.9) at all time scales. Vowels at small time scales (< 0.1 ms), however, have a small fractal dimension (1.0 $\Rightarrow$ 1.3) which is also consistent with the absence of turbulence in their production.

Summarising his work Maragos found that fractal dimension estimates could be affected by several factors including the time scale of waveform, the algorithm used for calculating D and also the speaking state, i.e. loud or whispered. Therefore no particular importance could be assigned to the absolute D estimate but only on their average range and relative difference. He does go on to say that a fractal method could be used successfully as a method of segmentation and distinguishing between certain phonetic elements.

2.3.3 The Fractal Dimension of Fricative Speech Sounds

A second paper was also produced in 1991, written by Hendrik Boshoff [4] at the University of Stellenbosch, South Africa. His paper concentrates on determining the fractal dimension of purely fricative speech sounds using Hurst's rescaled range analysis [8, 9]. His reason for using this method was that he claimed the Box Counting [8] method did not yield a consistent value of dimension over various time scales though no reason was given.

Rescaled range analysis is based on the observation of natural patterns that occur over a period of time. Temperature records, river discharges and rainfall are examples of records that can be characterised by an exponent H, the Hurst [9] exponent. Hurst found that the observed rescaled range, *R/S*, where *R* is the range of observations and *S* the standard deviation could be described by the relation:

$$R/S = (\tau/2)^H$$

where τ is the period of time of the observation.

The fractal dimension D is then related to the Hurst exponent by the following equation:

$$D = 2 - H$$

In the case of Boshoff's [4] experiments the observations he made were the recorded time samples of the speech phonemes sampled at 16 kHz.

Drawing on results made from a considerable database of fricative elements, Boshoff was able to conclude that, generally, voiced fricatives tend to have a lower fractal dimension than their unvoiced counterparts. This finding is consistent with the turbulence theory put forward by Maragos in his paper. The individual values of dimension found lie, on average, between 1.2 and 1.8 which suggests that this fractal method alone is not sufficient to allow fricative recognition, the spread and overlap being too great.

2.3.4 Fractal Dimension Segmentation of Speech Signals

As part of his research in Data Compression Techniques at Cranfield Institute of Technology in 1991, M. Gadallah [5] investigated a fractal method for segmenting speech signals.

Part of the aim of his research was to investigate the possibility of characterising and hence recognising a word using a template matching system scheme. This effectively is a feature extraction method of recognition.

The work of Richard Voss [6] and his study of the characterisation of noise sequences is referred to here again. Voss suggested that a relationship between the fractal dimension, D, and the spectral tilt of a power spectrum, B could be given by the equation:

$$D = 1 + (3 - B)/2, \qquad \text{for } 1 < B < 3$$

implying that the fractal dimension is correlated with the spectral tilt. The principle method to compute D was to fit a straight line to the power spectrum of a speech signal and measure the gradient (or the tilt).

Using non overlapping speech frames of 20 ms, Gadallah initially obtained a recognition accuracy of only 62.3 % but by introducing further parameters connected with the average power as part of the feature extraction phase, a recognition accuracy of 91.7 % was obtained.

In conclusion, Gadallah states the fractal technique for characterising speech frames by feature extraction, although only a preliminary investigation, showed significant advantages over the conventional filter bank (FB) or linear prediction coefficient (LPC) methods.

References

[1] Lea A. Wayne 1986 'Trends In Speech Recognition' Speech Science Publications Apple Valley, Minnesota

[2] Pickover C., Khorsani A. 1986 'Fractal Characterization of Speech Waveform Graphs' Computer & Graphics Vol.10 No. 1, pp 51-61

[3] Maragos P. 1991 'Fractal Aspects of Speech Signals : Dimension and Interpolation' Proceedings IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP) Vol. 1 pp 417-424

[4] Boshoff H.F.V., Grotepass M. 1991 'The Fractal Dimension of Fricative Speech Sounds' Communication and Signal Processing Proceedings of the IEEE South African Symposium pp 12-16

[5] Gadallah M.E. 1991 'Data Compression Techniques for Isolated and Connected Word Recognition' PhD Thesis Cranfield Institute of Technology Dept of Applied Computing and Mathematics pp 203-215

[6] Voss R. 1985 'Random Fractals : Characterization and Measurement. In Scaling Phenomena in Disordered Systems'
Proceedings Nato A.S.I. Geilo, Norway

[7] Mandelbrot B.B. 1982 'The Fractal Geometry of Nature'
W.H. Freeman and Company New York

[8] Feder J. 1988 'Fractals'
Penum Press New York and London

[9] Hurst H.E. 1965 'Long Term Storage : An Experimental Study'
Constable London

Chapter 3

Speech Production and Classification

3.1 Introduction

The sound waves of speech can justifiably be said to be among the most complicated to be found in the natural world. Extreme changes in sound quality follow each other with great rapidity, indicating that the speech mechanism, viewed as a generator of sound, must work in a complicated manner and be capable of operating in a wide variety of ways.

3.2 The Mechanics of Speech

The vocal organs of the human body are the lungs, the windpipe (or the *trachea*), the larynx (containing the vocal cords), the throat (or the *pharynx*), the nose and the mouth [1]. Together these organs form an intricately shaped tube extending from the lungs to the lips. One part of the tube, lying above the larynx, is called the *vocal tract* [1], and consists of the pharynx, mouth and nose. The shape of the vocal tract can be varied extensively by moving the tongue, the lips and other parts of the tract.

The source of energy for speech production is the steady stream of air that comes from the lungs as we exhale. Under normal respiratory conditions it is inaudible. It can be made audible by setting the air into rapid vibration by way of the vocal cords. By using the laryngeal muscles the vocal cords can be brought together forming a barrier across the airway which leads from the lungs through the trachea to the pharynx and the mouth. When the vocal cords are open, the air stream passes into the vocal tract. When closed, The fundamental frequency has a number of significant functions as part of the human communication system. Fore mostly it is a carrier of intonation which is a vital element in recognising speech. The larynx also controls *voice quality* [2] and the production of voiced and unvoiced sounds, known as *voice switching* [2]. It does not however transmit any linguistic information as such. This act is performed by the vocal tract which modifies the sound the larynx carries.

Though the fundamental frequency is continually changing during conversation, the components of the larynx tone are always harmonics of the fundamental. The effect of the resonances of the vocal tract is to produce a peak in the output at the harmonics which are closest to the natural resonance of the tract. This ensures that the spectrum of the resulting sound has the same envelope. This leads to the fact that a range of sounds with different fundamentals can be heard to have a certain sameness of quality. The resonances of the vocal tract are known as *formants* and their frequencies as *formant frequencies* [3]. The property of the vocal tract is such that its acoustic performance can be readily changed enabling perceptible differences in formant structure. Every configuration of the vocal tract has its own set of characteristic formant frequencies.

These changes are a result of movement of the tongue and lips etc. This is properly known as articulation and the individual movements of the tongue, lips, palate etc. is called *articulatory movements* [3]. The body of the tongue can be moved backwards and forwards, up and down which changes the length and cross section of the vocal tract thus producing different resonances. The lips and cheeks are capable too of changing the shape of the vocal tract and hence the kind of speech sound produced. The lips can be rounded or spread to various degrees but also closed to stop air flow altogether. The teeth also affect the vocal tract's shape. They can be used to restrict or stop the air flow by placing them close to the lips or the tip of the tongue. It is therefore the acoustic characteristic of the whole vocal tract which modifies the wave generated at the larynx and hence determines the type of sound spoken.

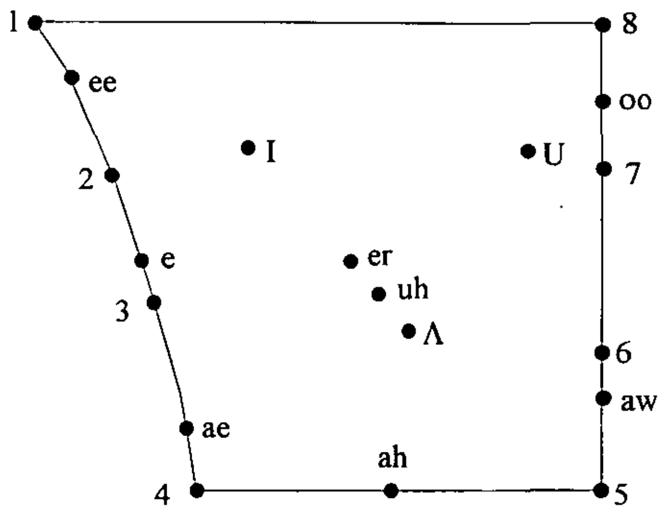
3.3 Classification of Phonemes

3.3.1 Vowels

For the purposes of description, the tongue positions for making vowel sounds are compared with the positions used for making a number of reference or *cardinal vowels* [3]. The cardinal vowels are a set of standard reference vowels whose quality is defined independently of any language. They form a reference point against which the quality of any vowel can be measured. Of course a strict definition of the term, *cardinal vowel*, is not possible since the quality of such a sound can only be perceived when it is correctly spoken.

The position of the tongue for the eight cardinal vowels can be described using a *vowel quadrilateral* [3] as shown in fig 3.1.

Fig 3.1 The Vowel Quadrilateral



All the tongue positions of the cardinal vowels are along the outer limits of tongue movement. If the position of the tongue moves towards the centre of the mouth the quality of the sound becomes more neutral and "uh" -like.

When the tongue is near the roof of the mouth (or the *palate*), the sound produced is called a close vowel. When the tongue is low, at the bottom of the mouth, the vowel is called open. For example the sound /ee/ is thus a close front vowel and /oo/ a close back vowel. The sounds /aw/ and /ah/ are open back and open front vowels respectively.

In terms of lip position, front vowels are usually made with spread lips and back vowels with rounded lips. As the tongue is lowered to more open positions the lips tend to become unrounded.

The vowel quadrilateral is only useful for describing the principle or *pure vowels* [1,2,3] of the English language. There is, however, another group of vowels known as *diphthongs* [1,2,3] which have their own distinction. The sound quality of a diphthong changes noticeably from its beginning to its end in a syllable. The tongue movements associated with these sounds are roughly the movements of positions assumed for pure vowels and these sounds therefore give rise to a more or less rapid switching from one set of formants to another. The actual movement of the tongue position is known as a *glide* [2] in a given direction because a diphthong does not actually involve a complete change to a new vowel.

3.3.2 Consonants

The sounds discussed so far all consist of a fundamental frequency plus a series of harmonics that are exact multiples of this fundamental frequency. Because of the regularly repeating character of the type of wave motions when components of this form are added together, such sounds are said to be *periodic* [2, 4]. Although periodic sounds may differ from each other in terms of quality, periodicity produces a certain character recognised by the ear and brain. Periodic sounds of this nature are called *tones* [2] and have a definite pitch which is easily recognisable.

Sounds which are without any harmonic basis, i.e. in which the component frequencies are not related to each other are known as *aperiodic* [2, 4]. Generally speaking there is no simple numerical relationship between the frequencies, and the waveform of such sounds show no obvious repeating pattern. Aperiodic sounds are hence classified by the ear and brain as *noises*. The essential difference between a noise and a tone is the random nature of the air particle movement in a noise and the regular, patterned character of the movement of a tone. Random movement is set up by forcing air to flow through a relatively narrow gap giving rise to the phenomenon known as *turbulence* [2, 4]. In speech for example, to pronounce an aperiodic sound such as /s/ a narrow constriction is made between the tongue and the teeth ridge and air is compelled to flow rapidly into this gap.

There is a second class of sounds which the ear recognises as noises. If the ear is struck by a single disturbance which is of short duration and not repeated, this too will be heard as a noise. In response to such a noise the ear drum is forced in and out just once unlike the response to a tone where it is forced in and out repeatedly.

The noises which occur in speech include both the single disturbance or *transient* type, known as *plosive consonant* sounds, and the longer lasting type in the *fricatives* and *affricates*. All of these require the setting up of a noise source at some point along the vocal tract. Whereas the source of periodic sound in speech is always the larynx, the noise source for consonants may be situated at many different places in the tract, the actual location referred to as the point of articulation. The noise generated for /p/ and /b/ is at the lips, for /f/ and /v/ at the lower lip, for /θ/ and /th/ at the upper teeth, for /t/ and /d/ at the teeth ridge likewise almost for /s/ and /z/. /sh/ and /zh/ have the noise generator a little further back, /k/ and /g/ noise is generated at the soft palate and for /h/ in the larynx alone.

When a consonant noise is voiced, the tone generator in the larynx is working simultaneously with a noise generator at some other point in the vocal tract. The sound /z/ when it is voiced, for example, calls for the operation of the periodic sound generator and the noise generator at the alveolar ridge. Such sounds are therefore a mixture of tone and noise. As such their waveforms have no strict repeating pattern because of the noise component, but a degree of periodicity is evident which enables differences of pitch in these sounds to be heard.

Whichever of the three techniques are used to produce sound, the resonances of the vocal tract still modify the character of the basic sounds produced whether they be by hiss, plosion or vocal cord vibration.

For the purposes of description the English consonants can be described by their *place-of-articulation* [3] their *manner-of-articulation* [3] and further still by whether they are *voiced* or *unvoiced*. The significant places-of-articulation in English are the lips *(labial)*, the teeth (*dental*), the gums (*alveolar)*, the palate (*velar*) and the glottis (*glottal)* [1]. The categories of manner-of-articulation are *plosive, fricative, nasal, semi-vowel* and *liquid.*

Table 3.1 Classification of English Consonants by Place and Manner of Articulation

Place of articulation	Manner of articulation				
	Plosive	Fricative	Semi-vowel	Liquids	Nasal
Labial	p b		w		m
Labio-Dental		f v			
Dental		θ th			
Alveolar	t d	s z	y	l r	n
Palatal		sh zh			
Velar	k g				ng
Glottal		h			

3.3.3 The Nasal Consonants

The articulation of nasal consonants involves the lowering of the soft palate so that the air passage which leads out through the naso-pharynx and the nostrils is open. The acoustic result of opening of what is effectively a side-branch of the vocal tract is to change the resonance characteristics of the tract and to introduce an anti-resonance or filtering effect. The frequency band affected by this filtering effect is between 800 and 2000 Hz.

In nasal consonants the oral branch of the vocal tract is closed off during the time when the nasal branch is open. The nostrils are less effective in radiating sound into the air and consequently the overall intensity of nasal consonants is lower than that of vowels with which they are associated.

A comparison of the nasal consonants shows that there is very little in the resonance of formant bars to differentiate them. The feature which is important for distinguishing the three nasal consonants from each other is in the rapid changes in formant frequency. The labial articulation, /m/, produces a transition of the second formant from or towards a lower frequency, depending upon whether the consonant is initial or final. The alveolar, /n/, involves little transition and the velar, /ng/, from or towards a higher frequency.

3.3.4 The Liquid Consonants

The consonant /r/ is a glide from a neutral vowel towards a following vowel. It is formed by turning the tip of the tongue upwards towards the hard palate.

The Alveolar, /l/, is by strict definition a lateral voiced consonant. It is made by putting the tip of the tongue against the gums and allowing the air to pass either side of the tongue.

3.3.5 The Semi-Vowels

The semi-vowels are the /w/ and /y/ voiced sounds. They are produced by keeping the vocal tract briefly in a vowel-like position, and then changing it rapidly to the position required for the following vowel in the syllable. Consequently the semi-vowels must always be followed by a vowel in whatever syllable they are used. The alveolar, /y/, is formed by putting the tongue close frontal position, as for an /ee/ sound, holding it there briefly and then changing to what ever vowel follows. Forming the labial, /w/, is similar except the lips are first close rounded as for an /oo/ sound.

Summary

Summarising this chapter, a detailed description of how the vocal organs of the body operate to produce audible sound has been given in addition to what the raw components are that the sound is comprised of. Methods of articulation are then discussed and how they can change the sound produced to form either vowels, plosive sounds or fricatives. Finally the classification of English phonemes is described and tabulated.

References

[1] Fletcher H. 1953 'Speech and Hearing in Communication'
Van Nostrand London New York Toronto

[2] Fry D.B. 1979 'The Physics of Speech'
Cambridge University Press

[3] P. B. Denes, E. N. Elliot 1963 'The Speech Chain'
Bell Telephone Laboratories

[4] Fant G. 1970 'Acoustic Theory of Speech Production'
Moulton & Co. N.V. The Hague

Chapter 4

Fractals and Dimension

4.1 Introduction

The introduction of the word *fractal* to the scientific community and the world at large can only be attributed to one man, Benoit Mandelbrot [1]. Since the introduction of its concept in his classical 1975 paper, '*Fractal objects : Form, Chance and Dimension*' [2], fractal models have been successfully applied to describe and understand the geometry of countless natural phenomenon and geometries ranging from particle trajectories and hydrodynamic flow to landscape structures and biological studies.

The purpose of this chapter is to address the fundamental questions frequently posed by the uninitiated when first encountering the subject:

1) What is a fractal ?
2) What is a fractal dimension ?
3) How can fractal dimensions be calculated and what use are they ?

When these have been adequately answered, the question of how fractals can be applied to speech science will also be addressed.

4.2 Beginnings

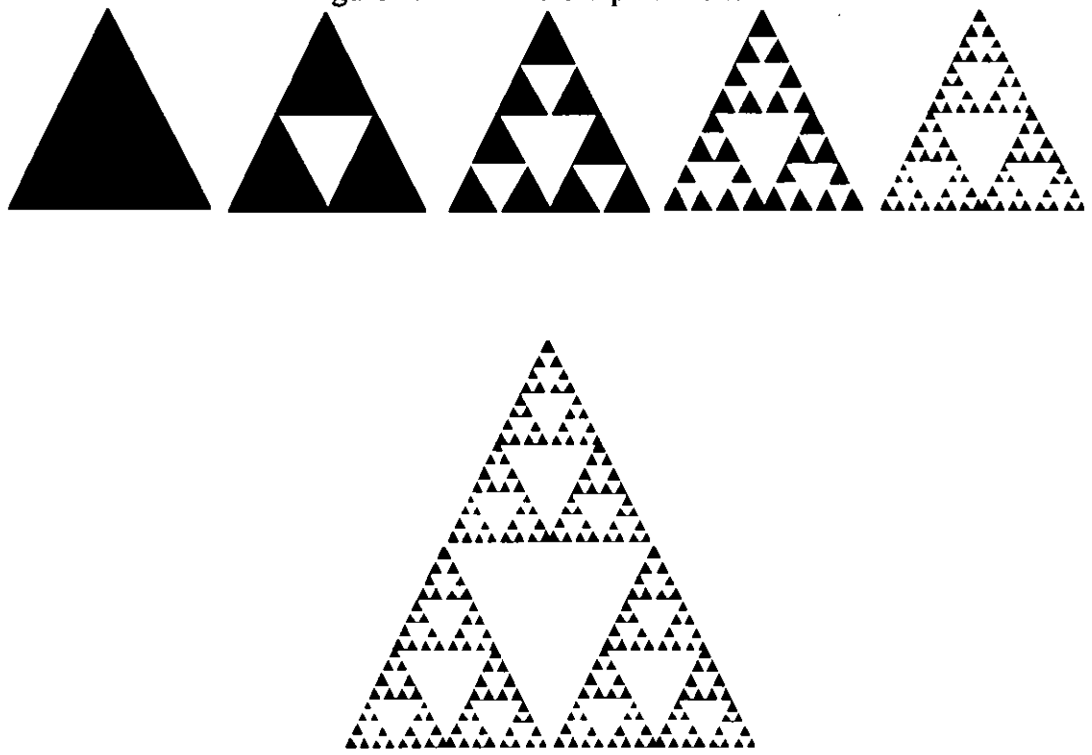
Standard geometry has long been an established tool for man to mathematically model simple, Euclid objects such as cubes, pyramids, squares, circles and lines. Yet the shapes that nature produces have proven time and time again to be extremely non linear, so varied in fact as to warrant the term '*geometrically chaotic*' [1]. Not surprisingly early mathematicians struggled greatly to apply the standard geometric techniques to describe such natural patterns, finishing up with models that, at the time, could only be labelled as *pathological* and *monstrous* [1].

Mandelbrot, with his ingenious concept, has given us a tentative bridge between the extremes of linear geometric order of Euclid and the geometric chaos of roughness and fragmentation. It has become known as *'the fractal geometry of nature and chaos'* [1].

To give a broad definition : Mathematical and natural fractals are shapes whose roughness and fragmentation neither tend to vanish nor fluctuate up and down but remains essentially unchanged under all levels of examination and scrutiny [1]. This is also commonly referred to as *self-similar* [3], a common fractal term which is often used in conjunction with the terms *scaling* or *scale invariance* [3].

The fractal figure below, fig 4.1, is a classic example of a construction that exhibits self similarity and has scale invariance. It is known as the *Sierpinski Gasket* [3].

Figure 4.1 The Sierpinski Gasket



The basic process of construction is to divide a given black triangle into four sub triangles, and then delete the middle fourth. This process is then carried out on the next stage and so on. The main diagram is clearly just a construction of the final stages shown above it. If one imagines viewing the gasket at any scale the pattern would remain wholly unchanged.

So what makes such patterns and the like of them so special ? The key to fractal geometry's effectiveness is that simplicity of their construction itself. The algorithms that generate such fractal shapes are typically extraordinarily short, yet their outputs often appear to involve structures of great complex intricacies and richness. The simple reason for this is that fractal patterns are derived from algorithms containing iterative processes, i.e. the basic instructions are simple and their effects can be followed easily. Hence, fractal geometry gives us a method of describing and explaining the chaotic patterns of nature as merely the culmination of many simple steps.

4.3 Fractal Dimensions

Having loosely established what a fractal is and what they can be used for, the methods by which one fractal object can be compared to another is examined. There are various numbers associated with fractals which can be used to compare them. These are generally known as *fractal dimensions* [1,3]. They are important because they can be defined in terms of real-world data, and can be measured approximately by means of experiment.

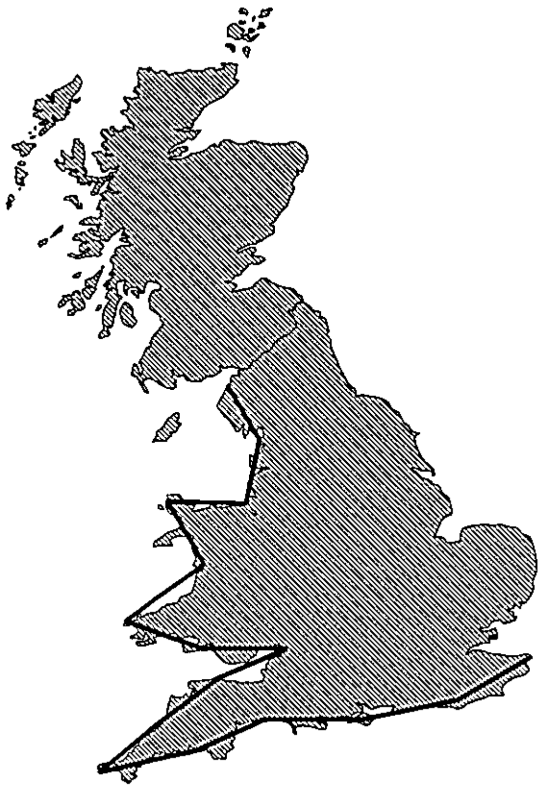
Essentially the fractal dimension is an attempt to quantify how densely the fractal object occupies the metric space in which it lies. They can thus be considered as sets of points embedded in space. To explain this point more clearly, consider these examples of standard sets. The set of points that make up a line in ordinary Euclidean space has the topological dimension of 1 and a fractal dimension, D, of 1 also.

Similarly the set of points that form a surface in Euclidean space has the topological dimension of 2 and the fractal dimension, D, of 2. Finally a ball or sphere has a topological dimension of 3 and a D of 3. However, the resemblance between the two dimensions ends there as the topological dimension is always an integer whereas the fractal dimension need not necessarily be so. This is essentially where the term, *fractal dimension*, is derived from. Curves therefore for which D exceeds the topological dimension 1 are called *fractal curves* [1] and coastlines and similar natural objects are called *fractal patterns* [1].

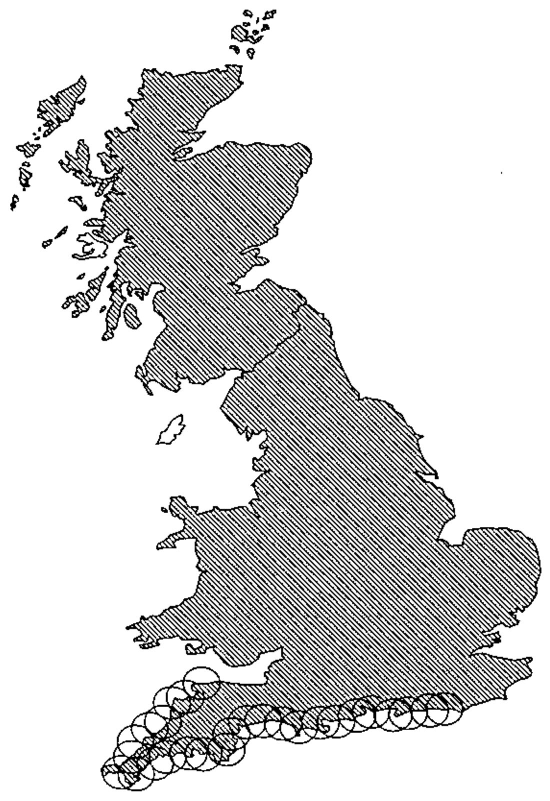
4.4 Calculations

For simplicities sake, we can begin our discussion by considering the same question as Mandelbrot [4] posed in his 1967 paper, 'How long is the coast of Britain ?' It is obvious that its length is at least equal to the distance measured along a set of straight lines between its beginning and end. However the coastline is irregular and winding and there is no question it is much longer than the straight lines between it's end points. There are of course many more accurate ways of determining the length. Ultimately though, all measurements lead to the same conclusion, that the coastline's length is in fact very large if not indeterminate, so much so that it is best considered as being infinite. Consider the possible methods of measurement as outlined below :

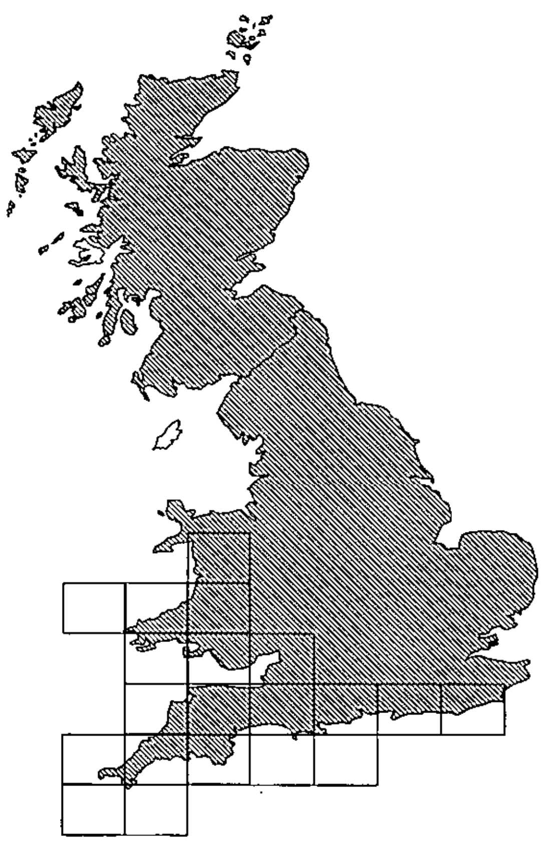
1. By using a set of dividers of length η, walking them along the coastline and counting the number of steps it takes to circumscribe the coast. The approximate length of the coast will be the length of the set dividers multiplied by the number of steps, L(η).



2. By attempting to cover the entire coastline with circular discs of radius η. In other words, considering all the points of both land and sea for which the distance to the coastline is no more than η. This in effect forms a kind of tape of width 2η which covers the entire coastline. The approximate length of the coastline can then be calculated by measuring the surface area of the tape and dividing it by 2η.



3. By covering the entire coastline with a grid with cells of size η by η and counting the number of cells that the coastline intersects. The approximate length can then estimated by multiplying the number of cells by η.



There are of course other methods using similar techniques which would work just as well. However, all of the methods described above have one very common similarity. That decreasing the size, η, of the measuring device, be it divider length, disc radius or grid cell size will result in a more accurate estimation of the coastline if one were to repeat the experiment. Further, no matter how small the measuring device is made, the estimation in length increases unbound. More and more detail of the coast could be measured as the device length decreased (or the scale of the map increased). This leads to the conclusion that, theoretically, the length of the coast of Britain or of any coastline for that matter is infinite. In practice the limiting factor is naturally the resolution of what we can set our measuring device to, not to mention the effects of the crashing sea and ebbing tide !

Having discussed the three methods given we can turn our attention once again to Mandelbrot and his definition of the *fractal dimension* [1,3]. Following on from the work carried out by the mathematician, Lewis Fry Richardson [5], Mandelbrot suggested that the relationship between the measuring device length, η, and the number of steps, N, for the device to estimate the length of a coastline could be expressed by the parameter D, the fractal dimension.

$$N(\eta) \quad = \quad \lambda\, \eta^{-D} \tag{1}$$

where λ is a constant. Multiplying by η gives the estimation of the coastline length $L(\eta)$ as

$$L(\eta) \quad = \quad \lambda\, \eta^{1-D} \tag{2}$$

Rearranging (2) and taking logs of both sides gives the equation

$$D \log \eta \;=\; \log \eta - \log L(\eta) \tag{3}$$

From (3) we can obtain the *fractal dimension* from the slope of coastline length against divider length.

$$D \;=\; \lim_{\eta \to 0} \,[1 - \{\log L(\eta) \,/\, \log \eta\}] \tag{4}$$

The value of the exponent D seems to depend upon the coastline that is chosen, and different pieces of the same coastline, if considered separately, may produce different values of D. However, its value seems to be independent of the method chosen to estimate the length. Mandelbrot thus concluded that despite the fact that D is not strictly an integer, it should be interpreted as a dimension, or to be precise, as a *fractal dimension*.

4.5 Hausdorff Dimension

If we accept that the length of any coastline or of any other fractal pattern we care to measure is in fact infinite, what conclusions of measurement can be drawn when comparing such fractal patterns with each other ? The basis of a solution to this problem was provided by Felix Hausdorff [6], a German writer, philosopher and mathematician who produced a paper in 1919 on dimensions and measurement.
It is inspired by the seemingly intuitive method for calculating the length of a linear polygon, made by simply adding the sides of its lengths together. Transforming the lengths is not necessary since the sides are raised to the power of one, the Euclidean dimension of a straight line. Similarly, the surface area of a closed linear polygon's interior is calculated by adding the sides of the squares that pave it, raised to the power two, the Euclidean dimension of a plane.

If we now consider a polygonal approximation of a coastline or any other fractal pattern consisting of small pieces of length η and raised to the power D, a quantity called the *approximate measure in the dimension D* [1] can be obtained. From equation (1) above, the number of sides is given by

$$N(\eta) = \lambda\,\eta^{-D} \qquad (1)$$

The approximate measure in the dimension D is thus

$$L(\eta^{D}) = N(\eta)\,\eta^{D} \qquad (5)$$

$$L(\eta^{D}) = \lambda\,\eta^{-D}\,\eta^{D} = \lambda \qquad (6)$$

The result is important because it shows that the approximate measure in dimension D is independent of the size of the polygonal constructs, η. So, however we choose to measure our fractal pattern, with regards to the size of our measuring device, the fractal dimension should remain unchanged.

4.6 Fractals Dimensions and Speech

Mandelbrot has established a method by which we can measure and compare the chaotic and fragmented patterns of nature against each other and with respect to theoretical fractal curves. The question that this whole research project has addressed is what use, if any, is this in terms of speech ?

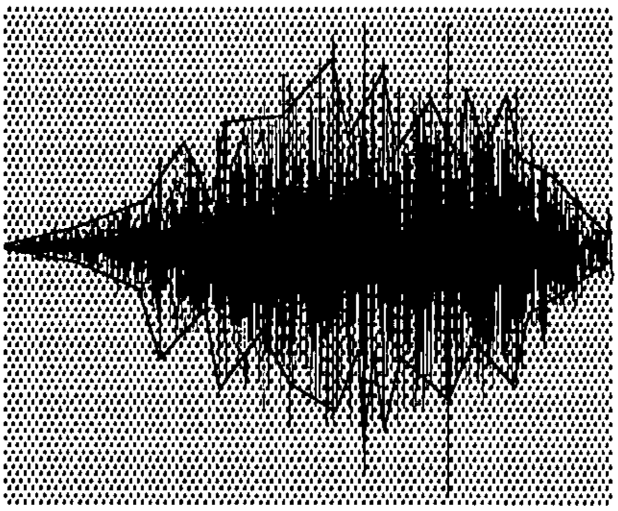
As discussed in the previous chapter, the mechanisms of speech can introduce varying amounts of air turbulence in order to facilitate the pronunciation of the numerous phonetic elements of English. Unvoiced fricatives such as /s/ and /f/ have more turbulence associated with their production than their voiced counterparts such as /z/ and /v/. Likewise unvoiced plosives have a higher turbulence than voiced plosives. Vowel sounds on the other hand have little or no turbulence associated with their production.

The initial aim of this research project was to establish the possibility of equating the amount of turbulence in the production of a phonetic element with its fractal dimension.

To understand how this could be achieved we must return to the methods described above for calculating fractal dimensions and examine how the techniques can be applied to speech waveforms.

The first method, which we will call the Richardson Method [5], is shown in fig 4.2 below as it could be applied to a different fractal pattern. The fractal pattern shown is not a coastline but a graphical representation of the digitally recorded phonetic element, /s/, an unvoiced fricative consonant.

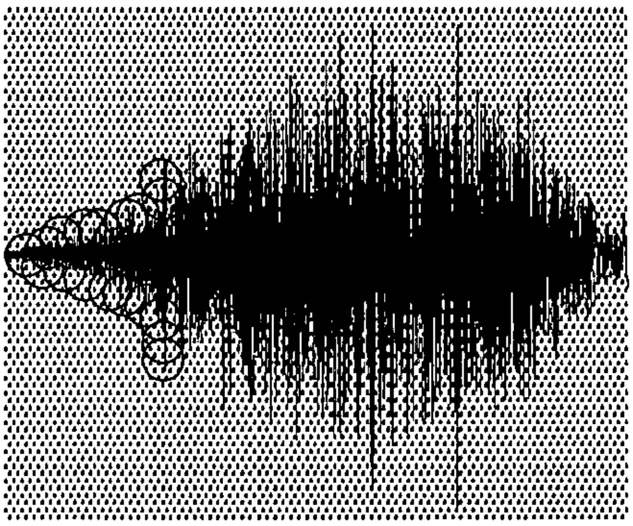
Fig 4.2 The Richardson Method



By using smaller and smaller 'divider length' sizes, a more accurate approximation of the fractal pattern can be established. By plotting the log of the sizes of divider length used against the log of the number of dividers to approximate the waveform, the fractal dimension of the pattern can be obtained from the slope of the graph.

The application of the second method, properly known as the Minkowski-Bouligand [8] technique, is shown in fig 4.3 below on the same waveform.

Fig 4.3 The Minkowski Bouligand Method



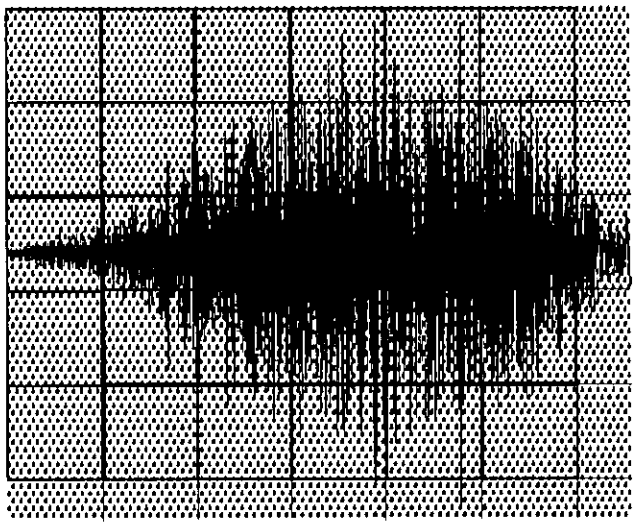
The same principle applies for this technique only in this method we measure the area that the circular discs produce and plot the log of this value against the log of the radius of the discs used.

The equation for calculating the fractal dimension using this method is a variation of (4) where A is the area produced by the discs.

$$D_{MB} = \lim_{\eta \to 0} [2 - \{\log A(\eta) / \log \eta\}] \qquad (7)$$

Finally the application of the third method, commonly known as the Box Counting [8] method is shown in fig 4.4.

Fig 4.4 The Box Counting Method



The same principle again only in this method the log of the number of boxes intersected by the waveform is plotted against the log of the size of the boxes.

The equation here again is another variation of (4) where N is the number of boxes that the pattern intersects.

$$D_{Box} = \lim_{\eta \to 0} [\log N(\eta) / \log 1/\eta] \qquad (8)$$

Hence we have three different methods by which phonetic elements can be individually examined in terms of their fractal dimension. The figures obtained from calculation can then be compared with those obtained for other elements in similar groups and with other elements in different linguistic groups.

Summary

As stated in the opening paragraphs, the purpose of this chapter was primarily to introduce the reader to the concept of fractals and to their associated dimensions, how they can be calculated and their possible application . In doing so the concept of self similarity, geometric chaos and scale invariance has also been introduced.

This has hopefully been achieved by way of demonstrating how one could measure a coastline or any fractal pattern for that matter and deduce that its length does in fact theoretically tend to infinity. Three methods of calculating fractal dimensions have then been shown using the Richardson method, the Minkowski-Bouligand method and also using the Box Counting method, and how this figure theoretically remains constant irrespective of the size of measuring device used to deduce it.

Finally the application of fractal dimensions on speech waveforms has been demonstrated.

References

[1] Mandelbrot B.B. 1977 'Fractals Form, Chance and Dimension' W.H. Freeman and Company San Francisco

[2] Mandelbrot B.B. 1975 'Les Objects Fractals : Forme, Hazard et Dimension' Paris : Flamirion

[3] Fleischmann M. 1989 'Fractals in the Natural Sciences'
Princeton University Press New Jersey

[4] Mandelbrot, B.B. 1967 'How Long is the Coast of Britain ? Statistical Self-similarity and Fractional Dimension' Science 155, 636-638

[5] Richardson, L.F. 1961 'The Problem of contiguity: an appendix of statistics of deadly quarrels.' General Systems Yearbook 6, 139-187

[6] Hausdorff F. 1919 'Dimension und aeusseres Mass'
Mathematics Annalen 79, 157-179

[7] Le Mehaute A. 1990 'Fractal Geometries'
Penton Press London

[8] Feder J. 1988 'Fractals' Penum Press
New York and London

Chapter 5

Experimental Procedures

5.1 Introduction

Having established the methodologies for calculating fractal dimensions, attention can be turned to the problem of how to characterise phonetic element speech waveforms. The purpose of this chapter is to describe the methods and approaches used to obtain and test the speech data.

5.2 Data Acquisition

Research in the area of speech analysis naturally requires a large selection of examples because of the vast differences that can occur in the pronunciation of words from one speaker to another and even from the same speaker. For that reason a large database of phonetic elements was compiled by myself and with the aid of two volunteers.

In the initial stages of the project a diverse range of speech examples were experimented on from single words to complete sentences. No satisfactory conclusions could be drawn as to the significance of the results that were obtained for whole words and sentences basically because of the limited range over which a valid fractal dimension could exist. A different approach was therefore sought. It was decided that a fractal analysis technique could possibly only be of any use for examining the phonetic elements that our words are actually comprised of, i.e. vowels, fricative and plosive elements etc.

Rather than trying to record the phonetic elements directly from microphone, an extraction process was carried out, using a speech workstation, on the recordings of a number of words which contain the phonetic elements required. The words used to extract these elements relating to phonemes are listed in Table 5.1 below.

Table 5.1 - The Phonemes of English

	Vowels				Consonants			
/ee/	-	heat	*/t/*	-	tee	*/s/*	-	see
/I/	-	hit	*/p/*	-	pea	*/sh/*	-	shell
/e/	-	head	*/k/*	-	key	*/h/*	-	he
/ae/	-	had	*/b/*	-	bee	*/v/*	-	view
/ah/	-	father	*/d/*	-	dawn	*/th/*	-	then
/aw/	-	call	*/g/*	-	go	*/z/*	-	zoo
/U/	-	put	*/m/*	-	me	*/l/*	-	law
/oo/	-	cool	*/n/*	-	no	*/zh/*	-	garage
/Λ/	-	ton	*/ng/*	-	sing	*/r/*	-	red
/uh/	-	the	*/f/*	-	fee	*/y/*	-	you
/er/	-	*bird*	/θ/	-	thin	*/w/*	-	we
/oi/	-	toil						
/au/	-	shout						
/ei/	-	take						
/ou/	-	tone						
/ai/	-	might						

The speech workstation used had the capacity to record signals at various sampling rates up to 40 kHz. 40 kHz was chosen as the sampling frequency to preserve the fragmentation of the sampled signal as close as possible to that of the continuous time speech signal [1].

Each word was recorded a minimum of ten times to establish a database of usable phonetic elements. This process was then repeated with another male and female speaker to establish whether the system was speaker independent and whether the results were gender dependent. The phonetic data files themselves were then produced by manually segmenting the words to obtain the required sounds.

5.3 Normalisation

Because of the graphical nature of the fractal techniques used, it became apparent that a method of amplitude normalisation was required to ensure that the recorded data would be independent of differences in recording levels and the speaker's own loudness level at the time of recording. Each of the data files were thus normalised using a standard deviation and a scaling factor technique to eliminate the differences. The scaling figure is arbitrary and a figure of 5000 was eventually used throughout the experimentation process.

Initially the speech data files were tested in their entirety but again, because of the graphical nature of the system, the results were deemed invalid because, in general, the fricative data files were significantly longer than the plosive data files which in turn were longer than the vowel data files. A form of time normalisation was thus introduced into the test procedure. Instead of testing an entire data file, individual portions of each file from 20 ms up to 70 ms in 10 ms steps were used. Once normalised the speech was then operated on by the three fractal methods.

5.4 Calibration

In order to obtain a measure of accuracy and calibration for the fractal programs written, two files were produced to simulate the extremes of possible data input. The first was a recording of a pure d.c. level which appeared as a single horizontal white line on the graphical display and the other a recording of high amplitude white noise signal which appeared as an oblique white screen, the theory being that the d.c. case would have a fractal dimension of one and the white noise case a fractal dimension of two. The fractal dimension of these two inputs were then examined using the three techniques. For the Box Counting method, grid sizes of 4, 5, 6, 7, 8 and 9 pixels were finally arrived at giving the following results :

horizontal line fractal dimension : 0.998

oblique screen fractal dimension : 2.000

Thus, under the test conditions stated, fractal dimensions of any waveform or object could be measured on a 479 by 639 resolution screen to an accuracy of 2 decimal points using the Box Counting technique.

For the Richardson method and the Minkowski-Bouligand method only the dc simulation could be tested because of the nature of the data input expected by the programs. For the Richardson method, using divider lengths of 2, 4 ,8, 16, 32 and 64 pixels, a result of 0.973 was achieved for the dc case and for the Minkowski Bouligand method, using disc radii of 1 to 10 pixels, a result of 1.081 was obtained for the dc case. Though clearly not as accurate as a the Box Counting algorithm the results obtained from these techniques would still give an appreciable estimate of the fractal dimension of any waveform.

With over a thousand phonetic example to examine using three different techniques, the test process was carried out using a series of batch files capable of running continuously. The results were stored in separate files which were updated after each successive batch file had finished its function.

Summary

In this chapter a description of the process used to record the actual phonetic elements has been given together with an overview of how the files were time and amplitude normalised. A description of how the individual fractal programs were tested along with a measure of the accuracy has also been included.

References

[1] Maragos M. 1991 'Fractal Aspects of Speech Signals: Dimension and Interpolation', Proc IEEE ICASSP Toronto 1, 417-424

Chapter 6.

Experimental Results

6.1 Introduction

The results obtained for the fractal experiments carried out in this research project are tabulated and discussed in the following chapter. Results of experimentation on samples made by myself are referred to as Male Speaker 1 and the results obtained from the other male and the female speaker referred to as Male Speaker 2 and Female Speaker respectively.

6.2 Objectives

The three fractal methods are treated in turn for each speaker beginning with the Box Counting method followed by the Richardson method and lastly the Minkowski-Bouligand method. The X-axis of each table is divided into five equal divisions of milliseconds from 30 ms to 70 ms. A discussion of the results is given at the end of each phonetic category for each method for the three speakers. Where appropriate mean values are given at the foot of certain tables to highlight the points raised in the discussion areas.

6.3 The Box Counting Method

The Box Counting fractal dimension, as described in chapter 4, is calculated by means of a variable grid which intersects the fractal pattern under examination. The dimension is calculated by repeatedly counting the number cells that the pattern intersects as the grid dimension varies. The results obtained using this technique are generally very consistent with the expected theoretical results.

6.3.1 Box Counting Fricative Elements

Tables 6.3.1.1, 6.3.1.2 and 6.3.1.3 below show the results obtained for each speaker for experiments carried out on fricative phonetic elements using the Box Counting Method.

Table 6.3.1.1 Box Counting Fricative Elements for Male Speaker 1

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.55	1.64	1.72	1.77	1.80
/θ/	1.62	1.72	1.78	1.80	1.81
/sh/	1.55	1.63	1.71	1.76	1.78
/s/	1.74	1.76	1.80	1.76	1.80
mean	**1.615**	**1.688**	**1.753**	**1.773**	**1.798**
voiced					
/v/	1.62	1.67	1.71	1.73	1.73
/h/	1.51	1.61	1.67	1.67	1.67
/z/	1.65	1.71	1.75	1.76	1.76
/zh/	1.54	1.64	1.74	1.76	1.77
/th/	1.59	1.67	1.73	1.75	1.76
mean	**1.582**	**1.660**	**1.720**	**1.734**	**1.738**

Table 6.3.1.2 Box Counting Fricative Elements for Male Speaker 2

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.73	1.77	1.79	1.80	1.81
/θ/	1.74	1.77	1.79	1.78	1.80
/sh/	1.72	1.74	1.76	1.77	1.77
/s/	1.73	1.75	1.76	1.78	1.80
mean	**1.730**	**1.758**	**1.775**	**1.783**	**1.795**
voiced					
/v/	1.72	1.73	1.76	1.78	1.77
/h/	1.53	1.65	1.73	1.78	1.71
/z/	1.71	1.77	1.77	1.76	1.79
/zh/	1.66	1.74	1.76	1.77	1.77
/th/	1.74	1.75	1.77	1.78	1.80
mean	**1.672**	**1.728**	**1.758**	**1.774**	**1.768**

Table 6.3.1.3 Box Counting Fricative Elements for Female Speaker

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.71	1.76	1.77	1.79	1.79
/θ/	1.70	1.75	1.79	1.81	1.82
/sh/	1.76	1.77	1.78	1.80	1.82
/s/	1.74	1.77	1.78	1.80	1.83
mean	**1.728**	**1.763**	**1.780**	**1.800**	**1.815**
voiced					
/v/	1.65	1.73	1.77	1.77	1.75
/h/	1.59	1.63	1.65	1.66	1.70
/z/	1.74	1.76	1.78	1.78	1.77
/zh/	1.76	1.76	1.78	1.80	1.80
/th/	1.71	1.76	1.78	1.78	1.75
mean	**1.690**	**1.728**	**1.752**	**1.758**	**1.754**

Taking the results for each speaker by examining the average values shown, there is certainly statistical evidence to suggest that, irrespective of the speaker, unvoiced fricative elements tend to have a slightly greater fractal dimension than the voiced counterparts when examined using the Box Counting method. The difference becomes more accentuated as more of the sample is examined. The highest recorded fractal dimensions in this case were that of the female speaker's unvoiced fricatives, the lowest that of Male Speaker 1's voiced fricatives.

6.3.2 Box Counting Plosive Elements

Table 6.3.2.1 Box Counting Plosive Elements for Male Speaker 1

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.48	1.56	1.62	1.63	1.65
/t/	1.60	1.70	1.73	1.78	1.79
/k/	1.45	1.61	1.67	1.71	1.73
mean	**1.510**	**1.623**	**1.673**	**1.707**	**1.723**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.44	1.53	1.55	1.53	1.55
/g/	1.37	1.43	1.48	1.49	1.50
mean	**1.413**	**1.523**	**1.567**	**1.557**	**1.567**

Table 6.3.2.2 Box Counting Plosive Elements for Male Speaker 2

			Time Scale		
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.45	1.55	1.63	1.64	1.67
/t/	1.75	1.78	1.79	1.79	1.79
/k/	1.62	1.70	1.73	1.73	1.74
mean	**1.607**	**1.677**	**1.717**	**1.720**	**1.733**
voiced					
/b/	1.61	1.67	1.65	1.65	
/d/	1.58	1.64	1.64	1.65	1.67
/g/	1.50	1.63	1.65	1.65	1.66
mean	**1.497**	**1.553**	**1.567**	**1.577**	**1.580**

Table 6.3.2.3 Box Counting Plosive Elements for Female Speaker

			Time Scale		
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.51	1.56	1.60	1.61	1.61
/t/	1.59	1.67	1.73	1.77	1.76
/k/	1.44	1.59	1.68	1.79	1.75
mean	**1.513**	**1.607**	**1.670**	**1.723**	**1.707**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.55	1.63	1.64	1.62	1.63
/g/	1.58	1.63	1.63	1.65	1.64
mean	**1.553**	**1.600**	**1.603**	**1.610**	**1.607**

Taking the collected results for the three speakers again, statistical evidence is apparent that would suggest the unvoiced plosive elements have a greater fractal dimension than voiced. As was the case for the fricative elements the effect becomes more apparent as the extent of the file under examination is increased. The mean values for D are consistently greater across all time scales for both male speakers. However, at the smaller time scales for the female speaker the situation appears to reverse.

6.3.3 Box Counting Vowels

Table 6.3.3.1 Box Counting Vowels for Male Speaker 1

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.50	1.51	1.54	1.53	1.48
/i/	1.54	1.61	1.63	1.61	1.61
/e/	1.42	1.53	1.59	1.59	1.67
/ae/	1.44	1.61	1.61	1.71	1.71
back					
/ah/	1.32	1.39	1.47	1.50	1.53
/aw/	1.22	1.25	1.29	1.34	1.34
/u/	1.32	1.38	1.46	1.50	1.49
/oo/	1.38	1.51	1.57	1.61	1.59
neutral					
/Λ/	1.30	1.39	1.52	1.57	1.60
/uh/	1.43	1.51	1.60	1.64	1.61
/er/	1.43	1.60	1.61	1.62	1.60
diphthong					
/oi/	1.34	1.46	1.54	1.55	1.54
/au/	1.38	1.52	1.60	1.63	1.62
/ei/	1.45	1.55	1.60	1.59	1.56
/ou/	1.35	1.48	1.55	1.59	1.61
/ai/	1.40	1.49	1.58	1.62	1.65
mean	**1.389**	**1.487**	**1.548**	**1.575**	**1.576**

Table 6.3.3.2 Box Counting Vowels for Male Speaker 2

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.52	1.55	1.62	1.60	1.59
/i/	1.60	1.77	1.78	1.78	1.77
/e/	1.54	1.65	1.69	1.70	1.68
/ae/	1.38	1.55	1.64	1.69	1.68
back					
/ah/	1.34	1.40	1.49	1.60	1.64
/aw/	1.31	1.29	1.32	1.34	1.40
/u/	1.40	1.50	1.59	1.63	1.66
/oo/	1.46	1.55	1.56	1.61	1.61
neutral					
/Λ/	1.38	1.41	1.54	1.61	1.67
/uh/	1.35	1.48	1.59	1.63	1.65
/er/	1.43	1.55	1.60	1.63	1.63
diphthong					
/oi/	1.33	1.41	1.49	1.52	1.55
/au/	1.36	1.48	1.60	1.65	1.64
/ei/	1.50	1.53	1.58	1.59	1.55
/ou/	1.33	1.44	1.51	1.53	1.53
/ai/	1.38	1.50	1.55	1.58	1.59
mean	**1.413**	**1.504**	**1.572**	**1.606**	**1.615**

Table 6.3.3.3 Box Counting Vowels for Female Speaker

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.64	1.63	1.62	1.59	1.66
/i/	1.50	1.52	1.53	1.50	1.61
/e/	1.44	1.46	1.46	1.52	1.65
/ae/	1.49	1.58	1.60	1.57	1.51
back					
/ah/	1.33	1.46	1.56	1.60	1.65
/aw/	1.28	1.34	1.41	1.48	1.66
/u/	1.32	1.43	1.51	1.59	1.68
/oo/	1.27	1.28	1.35	1.47	1.65
neutral					
/Λ/	1.45	1.56	1.65	1.64	1.66
/uh/	1.44	1.56	1.60	1.61	1.60
/er/	1.49	1.61	1.67	1.68	1.70
diphthong					
/oi/	1.51	1.56	1.53	1.52	1.58
/au/	1.43	1.57	1.69	1.71	1.68
/ei/	1.51	1.54	1.55	1.52	1.57
/ou/	1.51	1.54	1.53	1.56	1.62
/ai/	1.42	1.60	1.67	1.69	1.67
mean	**1.439**	**1.515**	**1.558**	**1.578**	**1.634**

The Box Counting method used on vowel sounds showed the technique to be unsatisfactory as a means of segmenting the elements into their respective classes. Only at the lowest time scales, particularly for the Female speaker does it appear that the front vowels at least can be distinguished from the back vowels, however, no theoretical explanation can support this observation.

6.3.4 Box Counting Nasal, Liquids and Semi Vowels

Table 6.3.4.1 Box Counting Nasal, Liquids and Semi Vowels for Male Speaker 1

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.54	1.68	1.74	1.68	1.76
/m/	1.43	1.53	1.63	1.65	1.65
/ng/	1.65	1.66	1.67	1.67	1.68
/l/	1.45	1.58	1.62	1.65	1.62
/r/	1.28	1.36	1.38	1.39	1.41
/y/	1.53	1.52	1.54	1.50	1.52
/w/	1.19	1.20	1.22	1.27	1.32

Table 6.3.4.2 Box Counting Nasal, Liquids and Semi Vowels for Male Speaker 2

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.71	1.75	1.76	1.74	1.73
/m/	1.49	1.53	1.50	1.48	1.48
/ng/	1.54	1.58	1.61	1.64	1.66
/l/	1.42	1.57	1.67	1.74	1.71
/r/	1.35	1.41	1.46	1.45	1.46
/y/	1.68	1.73	1.75	1.75	1.74
/w/	1.34	1.33	1.32	1.35	1.37

Table 6.3.4.3 Box Counting Nasal, Liquids and Semi Vowels for Female Speaker

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.52	1.58	1.58	1.57	1.59
/m/	1.45	1.53	1.60	1.64	1.63
/ng/	1.46	1.48	1.51	1.51	1.52
/l/	1.61	1.66	1.67	1.68	1.67
/r/	1.36	1.50	1.55	1.60	1.56
/y/	1.72	1.72	1.72	1.70	1.65
/w/	1.45	1.46	1.47	1.49	1.52

The last three phonetic categories have been grouped together in the manner shown for the simple reason that none of them can be consistently distinguished from the three main phonetic categories, the fricative, plosive and vowel elements. However, the Box Counting technique does highlight some interesting points. Firstly the liquid element, /l/, has a consistently higher dimension than /r/ for each speaker as does the semi vowel /y/ over /w/. The nasal category does not follow such a simple pattern. For the two male speakers the element /n/ is consistently higher than the element /ng/ which in turn is consistently higher than the element /m/. However the situation for the female speaker is not so well defined. At the lower time scales /n/ appears to be the *stronger* category but as the length of sample increases /m/ appears more dominant.

6.4 The Richardson Method

The most striking observation that can be made on all the results obtained for the Richardson Method is the consistency of the fractal values obtained at all time scales. Unlike the Box Counting method the mean values do not vary by more the 0.05 of Fractal dimension. Nor do the values consistently increase from the lowest time scale up to highest.

6.4.1 Richardson Fricative Elements

Table 6.4.1.1 Richardson Fricative Elements for Male Speaker 1

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.77	1.80	1.81	1.80	1.81
/θ/	1.73	1.77	1.78	1.79	1.78
/sh/	1.79	1.78	1.79	1.79	1.79
/s/	1.68	1.66	1.68	1.68	1.67
mean	**1.743**	**1.753**	**1.765**	**1.765**	**1.763**
voiced					
/v/	1.66	1.67	1.67	1.66	1.63
/h/	1.68	1.70	1.70	1.69	1.64
/z/	1.72	1.70	1.69	1.69	1.68
/zh/	1.71	1.80	1.80	1.80	1.78
/th/	1.68	1.68	1.71	1.69	1.67
mean	**1.690**	**1.710**	**1.714**	**1.706**	**1.680**

Table 6.4.1.2 Richardson Fricative Elements for Male Speaker 2

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.74	1.72	1.73	1.71	1.72
/θ/	1.68	1.68	1.69	1.67	1.67
/sh/	1.79	1.77	1.76	1.72	1.71
/s/	1.69	1.68	1.70	1.68	1.68
mean	**1.725**	**1.712**	**1.720**	**1.695**	**1.695**
voiced					
/v/	1.68	1.66	1.66	1.65	1.66
/h/	1.77	1.79	1.81	1.65	1.66
/z/	1.66	1.64	1.62	1.60	1.61
/zh/	1.79	1.78	1.78	1.75	1.72
/th/	1.64	1.64	1.63	1.62	1.62
mean	**1.708**	**1.702**	**1.700**	**1.654**	**1.654**

Table 6.4.1.3 Richardson Fricative Elements for Female Speaker

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.72	1.72	1.73	1.73	1.72
/θ/	1.71	1.71	1.74	1.72	1.70
/sh/	1.77	1.71	1.70	1.69	1.65
/s/	1.65	1.65	1.66	1.68	1.67
mean	**1.713**	**1.700**	**1.708**	**1.705**	**1.685**
voiced					
/v/	1.71	1.75	1.76	1.73	1.72
/h/	1.67	1.66	1.63	1.65	1.66
/z/	1.66	1.65	1.63	1.65	1.64
/zh/	1.77	1.73	1.72	1.71	1.67
/th/	1.64	1.63	1.65	1.64	1.63
mean	**1.690**	**1.684**	**1.678**	**1.676**	**1.664**

Despite the fact that the values obtained do not vary in magnitude with increased time as with the Box Counting method, the overall results do in fact conform to the same theory that unvoiced fricative elements have a higher fractal dimension than the voiced fricative elements. The scale of the difference between the two sub-groups is not so well defined and certain elements are consistently well above or below the average dimension for the group. At the higher time scales both male speakers show a much greater distinction between the two sub-groups than the female speaker.

6.4.2 Richardson Plosive Elements

Table 6.4.2.1 Richardson Plosive Elements for Male Speaker 1

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.68	1.72	1.70	1.68	1.68
/t/	1.72	1.75	1.77	1.76	1.77
/k/	1.72	1.76	1.80	1.79	1.76
mean	**1.707**	**1.743**	**1.757**	**1.743**	**1.737**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.53	1.55	1.54	1.53	1.54
/g/	1.56	1.59	1.63	1.64	1.63
mean	**1.553**	**1.573**	**1.573**	**1.577**	**1.573**

Table 6.4.2.2 Richardson Plosive Elements for Male Speaker 2

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.71	1.70	1.67	1.65	1.61
/t/	1.75	1.72	1.73	1.70	1.68
/k/	1.74	1.75	1.72	1.71	1.68
mean	**1.733**	**1.723**	**1.706**	**1.687**	**1.657**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.66	1.65	1.65	1.64	1.66
/g/	1.64	1.64	1.66	1.67	1.66
mean	**1.583**	**1.587**	**1.593**	**1.583**	**1.600**

Table 6.4.2.3 Richardson Plosive Elements for Female Speaker

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.55	1.58	1.59	1.62	1.62
/t/	1.69	1.67	1.67	1.68	1.67
/k/	1.73	1.78	1.79	1.80	1.79
mean	**1.657**	**1.677**	**1.683**	**1.700**	**1.693**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.64	1.71	1.70	1.69	1.68
/g/	1.68	1.72	1.73	1.73	1.72
mean	**1.633**	**1.667**	**1.660**	**1.667**	**1.653**

The above set of plosive results give the first real indication that the fractal dimension of an utterance, using the Richardson method at least, may in fact be gender dependant. The distinction between the voiced and unvoiced elements for the plosive group is well defined for both male speakers at all time scales and the dimension gap between the sub-groups is consistent throughout. However the results for the female speaker show quite a different situation. The mean results for each time slot are very similar due to the discrepancies that exist between the individual elements. Only /k/ shows any real dominance in the unvoiced group while /t/, /d/ and /g/ follow each other quite closely as do /p/ and /b/.

6.4.3 Richardson Vowels

Table 6.4.3.1 Richardson Vowels for Male Speaker 1

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.44	1.45	1.46	1.45	1.51
/i/	1.57	1.58	1.57	1.53	1.51
/e/	1.56	1.56	1.55	1.53	1.50
/ae/	1.60	1.60	1.57	1.54	1.53
back					
/ah/	1.43	1.47	1.46	1.47	1.43
/aw/	1.40	1.42	1.45	1.52	1.52
/u/	1.45	1.49	1.53	1.55	1.54
/oo/	1.43	1.52	1.54	1.60	1.59
neutral					
/Λ/	1.47	1.51	1.53	1.53	1.52
/uh/	1.50	1.54	1.56	1.53	1.53
/er/	1.52	1.54	1.53	1.51	1.49
diphthong					
/oi/	1.53	1.55	1.57	1.57	1.55
/au/	1.56	1.57	1.56	1.55	1.52
/ei/	1.42	1.50	1.53	1.49	1.48
/ou/	1.41	1.49	1.55	1.55	1.54
/ai/	1.54	1.57	1.57	1.52	1.51
mean	**1.489**	**1.523**	**1.533**	**1.528**	**1.517**

Table 6.4.3.2 Richardson Vowels for Male Speaker 2

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.54	1.51	1.51	1.50	1.58
/i/	1.73	1.72	1.71	1.65	1.59
/e/	1.60	1.60	1.58	1.57	1.52
/ae/	1.54	1.55	1.53	1.52	1.50
back					
/ah/	1.52	1.57	1.59	1.59	1.58
/aw/	1.42	1.46	1.55	1.59	1.60
/u/	1.64	1.67	1.71	1.72	1.73
/oo/	1.61	1.62	1.64	1.70	1.67
neutral					
/Λ/	1.52	1.53	1.53	1.52	1.53
/uh/	1.59	1.62	1.66	1.68	1.65
/er/	1.60	1.60	1.63	1.62	1.59
diphthong					
/oi/	1.54	1.57	1.63	1.65	1.65
/au/	1.55	1.56	1.57	1.61	1.56
/ei/	1.49	1.49	1.48	1.48	1.52
/ou/	1.51	1.57	1.62	1.62	1.62
/ai/	1.59	1.63	1.64	1.64	1.61
mean	**1.562**	**1.579**	**1.599**	**1.604**	**1.594**

Table 6.4.3.3 Richardson Vowels for Female Speaker

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.61	1.60	1.62	1.65	1.69
/i/	1.54	1.64	1.60	1.62	1.64
/e/	1.53	1.58	1.63	1.63	1.63
/ae/	1.53	1.62	1.58	1.58	1.77
back					
/ah/	1.59	1.68	1.70	1.67	1.72
/aw/	1.48	1.61	1.63	1.65	1.69
/u/	1.60	1.72	1.73	1.76	1.78
/oo/	1.42	1.51	1.54	1.61	1.71
neutral					
/Λ/	1.65	1.72	1.75	1.76	1.73
/uh/	1.55	1.58	1.66	1.64	1.62
/er/	1.63	1.65	1.71	1.67	1.64
diphthong					
/oi/	1.59	1.69	1.66	1.66	1.68
/au/	1.62	1.71	1.69	1.71	1.71
/ei/	1.46	1.45	1.61	1.56	1.57
/ou/	1.60	1.63	1.70	1.68	1.65
/ai/	1.61	1.69	1.70	1.69	1.71
mean	**1.653**	**1.630**	**1.657**	**1.659**	**1.684**

The vowel section of results for the Richardson method also indicates a significant difference between the female and male speakers. As with the fricative and plosive elements the results remain essentially constant at each time scale. It is the actual magnitude of the results for each speaker which is the interesting point. The range of dimension for male speaker one and two is in the order of 1.49 to 1.53 and 1.56 to 1.60 respectively while for the female it is 1.63 to 1.68. This difference in scale tends to reiterates the suggestion that any fractal dimension calculated using the Richardson method may well indeed be dependant on the gender of the speaker..

6.4.4 Richardson Nasal, Liquids and Semi Vowels

Table 6.4.4.1 Richardson Nasal, Liquids and Semi Vowels for Male Speaker 1

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.69	1.71	1.70	1.68	1.64
/m/	1.54	1.56	1.53	1.50	1.50
/ng/	1.59	1.58	1.58	1.57	1.58

/l/	1.57	1.58	1.59	1.58	1.57
/r/	1.35	1.43	1.47	1.48	1.50
/y/	1.41	1.40	1.50	1.51	1.49
/w/	1.28	1.37	1.39	1.49	1.41

Table 6.4.4.2 Richardson Nasal, Liquids and Semi Vowels for Male Speaker 2

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.71	1.66	1.64	1.60	1.58
/m/	1.42	1.46	1.49	1.43	1.44
/ng/	1.52	1.53	1.53	1.49	1.49
/l/	1.57	1.70	1.72	1.71	1.67
/r/	1.38	1.44	1.52	1.51	1.48
/y/	1.68	1.66	1.63	1.60	1.57
/w/	1.33	1.41	1.48	1.50	1.52

Table 6.4.4.3 Richardson Nasal, Liquids and Semi Vowels for Female Speaker

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.46	1.56	1.56	1.55	1.54
/m/	1.56	1.64	1.64	1.63	1.62
/ng/	1.46	1.48	1.57	1.56	1.57
/l/	1.63	1.65	1.65	1.60	1.63
/r/	1.61	1.66	1.67	1.67	1.69
/y/	1.70	1.64	1.65	1.61	1.60
/w/	1.51	1.60	1.61	1.59	1.65

Observing the results of the remaining phonetic groups, nasal, liquid and semi-vowels, for the two male speakers it is apparent that the same pattern occurs for this technique as did for the Box Counting method. The nasal element /n/ is stronger at all time scales than the element /ng/ which in turn is greater than /m/. For the liquids and the semi-vowels the situation is quite similar, /l/ greater than /r/ and /y/ greater than /w/. The results recorded for the female speaker show a another digression. In the semi-vowel section /m/ replaces /n/ as the stronger element followed by /ng/ at the longer time scales and /n/ at the shorter. No appreciable difference exists in between /l/ and /r/ in the liquid group and in the semi-vowel group an inversion occurs, /y/ greater than

/w/ at small time scales, reversing with increased time. These results again all support the observation that the Richardson method for calculating a fractal dimension is probably dependent on the gender of the speaker.

6.5 The Minkowski-Bouligand Method

The Minkowski-Bouligand method or disc technique offers the final fractal approach to the characterisation of human speech. The results obtained using this method vary greatly according to length of speech signal as well as by the type of phonetic category under examination.

6.5.1 Minkowski-Bouligand Fricative Elements

Table 6.5.1.1 Minkowski-Bouligand Fricative Elements for Male Speaker 1

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.53	1.60	1.65	1.68	1.71
/θ/	1.61	1.70	1.71	1.73	1.75
/sh/	1.50	1.57	1.62	1.66	1.69
/s/	1.71	1.74	1.75	1.77	1.81
mean	**1.588**	**1.653**	**1.683**	**1.710**	**1.740**
voiced					
/v/	1.60	1.63	1.65	1.66	1.65
/h/	1.45	1.52	1.56	1.59	1.59
/z/	1.63	1.68	1.71	1.73	1.74
/zh/	1.45	1.54	1.60	1.64	1.69
/th/	1.59	1.64	1.66	1.69	1.70
mean	**1.544**	**1.602**	**1.636**	**1.662**	**1.674**

Table 6.5.1.2 Minkowski-Bouligand Fricative Elements for Male Speaker 2

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.53	1.60	1.65	1.68	1.71
/θ/	1.62	1.67	1.71	1.73	1.75
/sh/	1.50	1.58	1.63	1.66	1.69
/s/	1.71	1.74	1.75	1.77	1.77
mean	**1.590**	**1.648**	**1.685**	**1.710**	**1.730**
voiced					
/v/	1.60	1.63	1.65	1.66	1.65
/h/	1.45	1.52	1.56	1.59	1.59
/z/	1.63	1.68	1.71	1.73	1.74

/zh/	1.45	1.54	1.60	1.64	1.67
/th/	1.59	1.64	1.66	1.69	1.70
mean	**1.544**	**1.602**	**1.636**	**1.662**	**1.670**

Table 6.5.1.3 Minkowski-Bouligand Fricative Elements for Female Speaker

			Time Scale		
unvoiced	30ms	40ms	50ms	60ms	70ms
/f/	1.70	1.73	1.75	1.76	1.77
/θ/	1.70	1.73	1.75	1.77	1.78
/sh/	1.67	1.71	1.74	1.76	1.75
/s/	1.73	1.75	1.76	1.77	1.78
mean	**1.700**	**1.730**	**1.750**	**1.765**	**1.768**
voiced					
/v/	1.63	1.67	1.70	1.71	1.72
/h/	1.52	1.56	1.59	1.61	1.63
/z/	1.72	1.75	1.76	1.76	1.76
/zh/	1.68	1.72	1.75	1.77	1.78
/th/	1.72	1.73	1.74	1.75	1.75
mean	**1.654**	**1.686**	**1.708**	**1.720**	**1.728**

For all the plosive experiments, irrespective of speaker, the significant results all occur at the longer time scales where the *fractal gap* between the elements in the voiced and unvoiced groups is more consistent. At the highest time scale the results for Male Speaker one and two are very consistent with each other ranging from at highest 1.81 and 1.77 respectively in the unvoiced sub-group to both 1.59 in the voiced sub-group. The female speaker achieves slightly higher values in both groups but at the expense of a reduced fractal gap.

6.5.2 Minkowski-Bouligand Plosive Elements

Table 6.5.2.1 Minkowski-Bouligand Plosive Elements for Male Speaker 1

			Time Scale		
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.41	1.49	1.53	1.53	1.53
/t/	1.61	1.65	1.68	1.70	1.71
/k/	1.37	1.46	1.52	1.57	1.59
mean	**1.463**	**1.533**	**1.577**	**1.600**	**1.610**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.31	1.35	1.36	1.36	1.37
/g/	1.25	1.30	1.34	1.37	1.40
mean	**1.297**	**1.353**	**1.400**	**1.403**	**1.417**

Table 6.5.2.2 Minkowski-Bouligand Plosive Elements for Male Speaker 2

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.39	1.46	1.49	1.52	1.53
/t/	1.70	1.73	1.75	1.76	1.70
/k/	1.56	1.61	1.64	1.65	1.66
mean	**1.550**	**1.600**	**1.627**	**1.643**	**1.630**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.51	1.52	1.52	1.53	1.55
/g/	1.43	1.49	1.50	1.51	1.50
mean	**1.413**	**1.440**	**1.450**	**1.473**	**1.470**

Table 6.5.2.3 Minkowski-Bouligand Plosive Elements for Female Speaker

	Time Scale				
unvoiced	30ms	40ms	50ms	60ms	70ms
/p/	1.48	1.51	1.52	1.53	1.53
/t/	1.64	1.70	1.72	1.73	1.74
/k/	1.41	1.51	1.57	1.62	1.66
mean	**1.510**	**1.573**	**1.603**	**1.627**	**1.643**
voiced					
/b/	1.46	1.61	1.67	1.65	1.65
/d/	1.39	1.46	1.48	1.50	1.48
/g/	1.51	1.55	1.57	1.55	1.55
mean	**1.420**	**1.460**	**1.477**	**1.487**	**1.477**

The results for the three speakers for the plosive phonetic category at the higher time scales are all very similar in that they demonstrate a consistency with the theoretical expectation and have similar means and ranges of values. In the 70ms time scale the unvoiced values all range between 1.5 and 1.74 while the voiced values range between 1.36 and 1.55

6.5.3 Minkowski-Bouligand Vowels

Table 6.5.3.1 Minkowski-Bouligand Vowels for Male Speaker 1

			Time Scale		
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.38	1.40	1.42	1.43	1.44
/i/	1.36	1.42	1.46	1.49	1.51
/e/	1.28	1.36	1.40	1.44	1.46
/ae/	1.32	1.41	1.47	1.51	1.53
back					
/ah/	1.19	1.26	1.31	1.35	1.38
/aw/	1.11	1.14	1.17	1.20	1.30
/u/	1.18	1.24	1.28	1.33	1.36
/oo/	1.29	1.35	1.40	1.43	1.45
neutral					
/Λ/	1.20	1.28	1.34	1.40	1.41
/uh/	1.32	1.38	1.43	1.46	1.48
/er/	1.30	1.47	1.42	1.45	1.48
diphthong					
/oi/	1.23	1.30	1.35	1.38	1.41
/au/	1.26	1.35	1.40	1.44	1.46
/ei/	1.30	1.36	1.41	1.44	1.46
/ou/	1.25	1.33	1.40	1.42	1.44
/ai/	1.26	1.35	1.41	1.45	1.48
mean	**1.264**	**1.338**	**1.379**	**1.414**	**1.416**

Table 6.5.3.2 Minkowski-Bouligand Vowels for Male Speaker 2

			Time Scale		
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.41	1.46	1.48	1.50	1.52
/i/	1.45	1.54	1.59	1.62	1.64
/e/	1.37	1.45	1.50	1.53	1.54
/ae/	1.30	1.39	1.44	1.48	1.50
back					
/ah/	1.26	1.35	1.41	1.45	1.50
/aw/	1.18	1.21	1.25	1.28	1.31
/u/	1.28	1.36	1.43	1.46	1.50
/oo/	1.33	1.40	1.44	1.48	1.50
neutral					
/Λ/	1.29	1.37	1.42	1.46	1.49
/uh/	1.28	1.36	1.48	1.46	1.49
/er/	1.31	1.39	1.44	1.48	1.49
diphthong					
/oi/	1.22	1.29	1.34	1.37	1.40
/au/	1.26	1.35	1.41	1.45	1.47
/ei/	1.34	1.39	1.44	1.46	1.48
/ou/	1.20	1.27	1.33	1.37	1.40
/ai/	1.27	1.34	1.39	1.43	1.46
mean	**1.297**	**1.370**	**1.424**	**1.455**	**1.481**

Table 6.5.3.3 Minkowski-Bouligand Vowels for Female Speaker

	Time Scale				
front	30ms	40ms	50ms	60ms	70ms
/ee/	1.51	1.54	1.55	1.56	1.57
/i/	1.34	1.39	1.43	1.46	1.48
/e/	1.30	1.33	1.37	1.40	1.43
/ae/	1.32	1.39	1.42	1.44	1.45
back					
/ah/	1.25	1.33	1.39	1.43	1.46
/aw/	1.18	1.24	1.29	1.33	1.36
/u/	1.25	1.32	1.37	1.42	1.46
/oo/	1.16	1.21	1.26	1.30	1.33
neutral					
/Λ/	1.34	1.41	1.47	1.50	1.53
/uh/	1.34	1.40	1.45	1.48	1.50
/er/	1.37	1.44	1.50	1.53	1.55
diphthong					
/oi/	1.33	1.38	1.41	1.44	1.47
/au/	1.33	1.41	1.47	1.51	1.54
/ei/	1.37	1.41	1.44	1.46	1.47
/ou/	1.34	1.40	1.43	1.47	1.49
/ai/	1.28	1.39	1.45	1.50	1.53
mean	**1.313**	**1.374**	**1.419**	**1.452**	**1.476**

The consistency between speakers in the fricative and plosive groups is carried through to the vowel elements. The mean values range from 1.26 to 1.42 for male speaker one, 1.3 to 1.48 for male speaker two and from 1.31 to 1.48 for the female speaker. Elements in the individual results vary enough to coincide with the lower order values in the plosive section. There is no indication that the results are dependant on the gender of the speaker.

6.5.4 Minkowski-Bouligand Nasal, Liquids and Semi Vowels

Table 6.5.4.1 Minkowski-Bouligand Nasal, Liquids and Semi Vowels for Male Speaker 1

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.49	1.56	1.61	1.64	1.66
/m/	1.44	1.48	1.52	1.55	1.57
/ng/	1.54	1.57	1.59	1.60	1.61
/l/	1.31	1.37	1.44	1.47	1.49
/r/	1.12	1.17	1.22	1.25	1.28

/y/	1.41	1.45	1.45	1.46	1.46
/w/	1.04	1.07	1.13	1.17	1.22

Table 6.5.4.2 Minkowski-Bouligand Nasal, Liquids and Semi Vowels for Male Speaker 2

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.56	1.62	1.63	1.67	1.68
/m/	1.38	1.43	1.43	1.41	1.41
/ng/	1.44	1.49	1.53	1.55	1.58
/l/	1.34	1.43	1.49	1.53	1.55
/r/	1.23	1.28	1.31	1.33	1.35
/y/	1.59	1.64	1.64	1.67	1.68
/w/	1.26	1.27	1.27	1.29	1.31

Table 6.5.4.3 Minkowski-Bouligand Nasal, Liquids and Semi Vowels for Female Speaker

	Time Scale				
	30ms	40ms	50ms	60ms	70ms
/n/	1.43	1.47	1.49	1.50	1.51
/m/	1.41	1.46	1.48	1.51	1.52
/ng/	1.39	1.40	1.42	1.44	1.45
/l/	1.54	1.57	1.60	1.61	1.62
/r/	1.33	1.38	1.42	1.45	1.46
/y/	1.65	1.67	1.68	1.67	1.66
/w/	1.42	1.44	1.45	1.48	1.50

In the remaining phonetic categories there is some consistency again using the Minkowski-Bouligand technique with results obtained in the same categories for the previous two techniques. For Male Speaker one at the 70ms time scale in the nasal category the element /n/ has a greater dimension than /ng/ which itself is greater than /m/. This feature is again mirrored by Male Speaker two. The female speaker is not so well defined however with /n/ and /m/ almost identical at all time scales followed very closely by /m/. The liquids and the semi-vowels however are consistent throughout for all three speakers with /l/ greater than /r/ and /y/ greater than /w/.

Chapter 7

Analysis of Experimental Data

7.1 Introduction

Having collected and tabulated the results obtained for the three speakers using the three fractal methods, this chapter is intended to graphically demonstrate and compare the results to give an overall summary of the fractal approach to the characterisation of human speech.

The three techniques will be taken in turn for each speaker comparing the results obtained for the fricative, plosive and vowel elements against each other.

7.2 The Box Counting Graphs

Fig 7.1 Box Counting Technique - Male Speaker 1

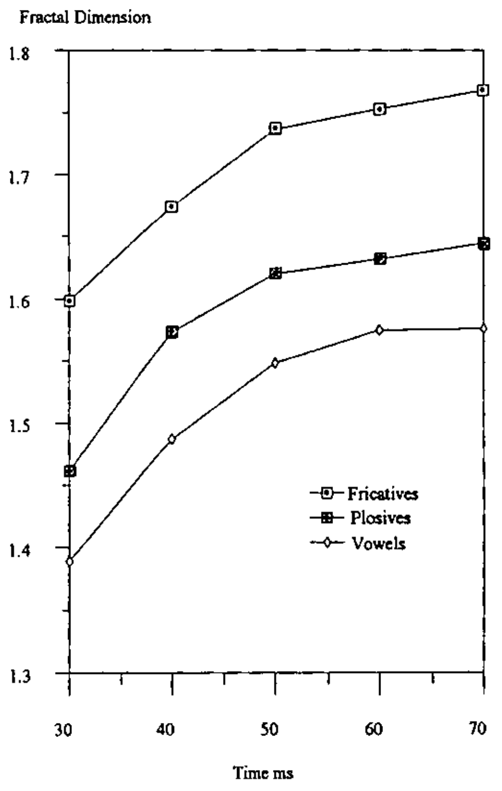


Fig 7.2 Box Counting Technique - Male Speaker 2

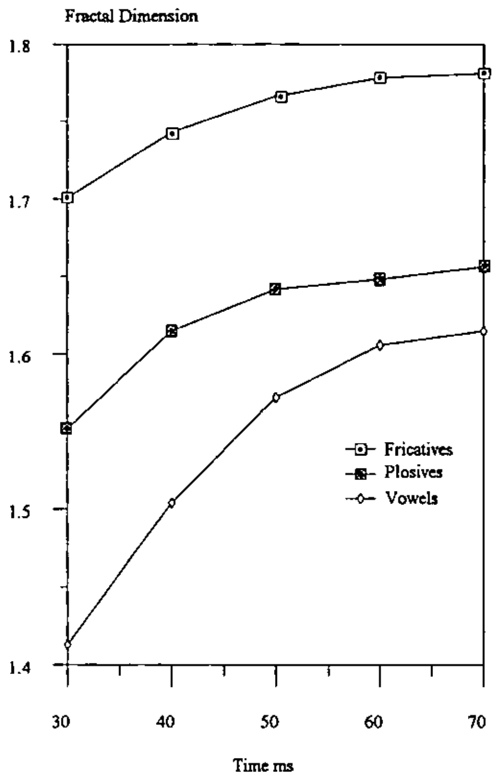
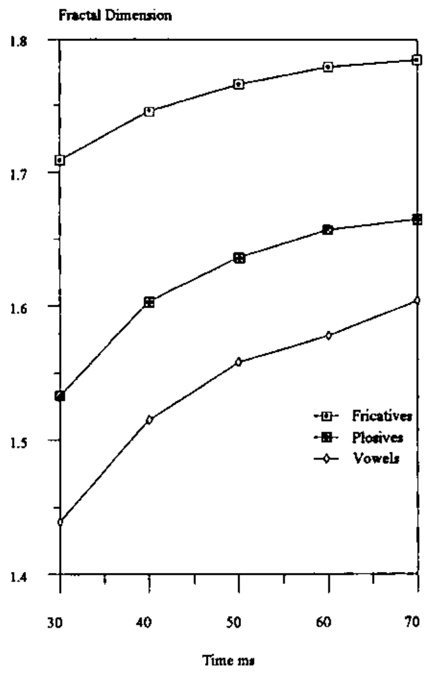


Fig 7.3 Box Counting Technique - Female Speaker



Examining each of the graphs above for the box counting technique, there are unquestionable similarities between each speaker for the three phonetic categories. The technique clearly exhibits speaker independence as the graphs follow identical patterns. Each trace appears to approach a steady value just beyond the 70 ms time slot which suggests that true fractal values could be obtained if a greater proportion of the speech sample was tested. Though this could more readily be achieved with the fricative and vowel elements, generally speaking plosive elements have a duration of no more than 70 ms meaning a direct temporal comparison would be more difficult.

Examining the traces themselves for all three cases, clearly the fricative elements have an average greater dimension, approximately 1.75 at the 70 ms time slot, while the plosive and vowel elements are significantly less but show less distinction between the dimension at the 70 ms time slot, approximately 1.65 and 1.60 respectively.

7.3 The Richardson Graphs

Fig 7.4 Richardson Technique - Male Speaker 1

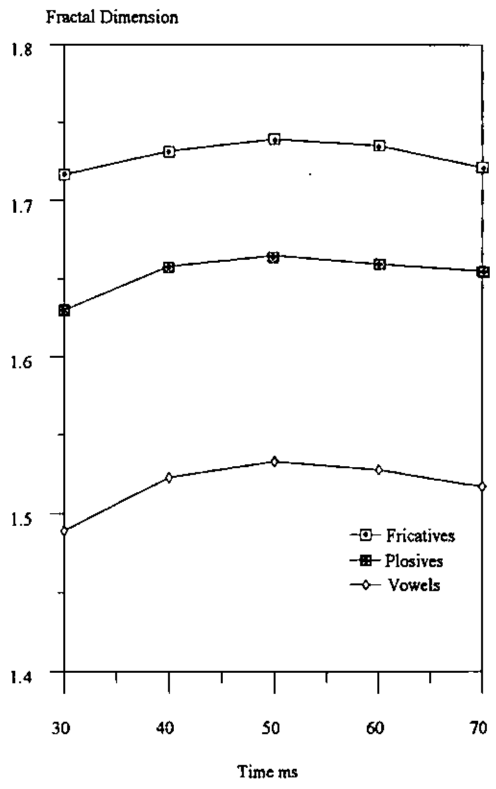


Fig 7.5 Richardson Technique - Male Speaker 2

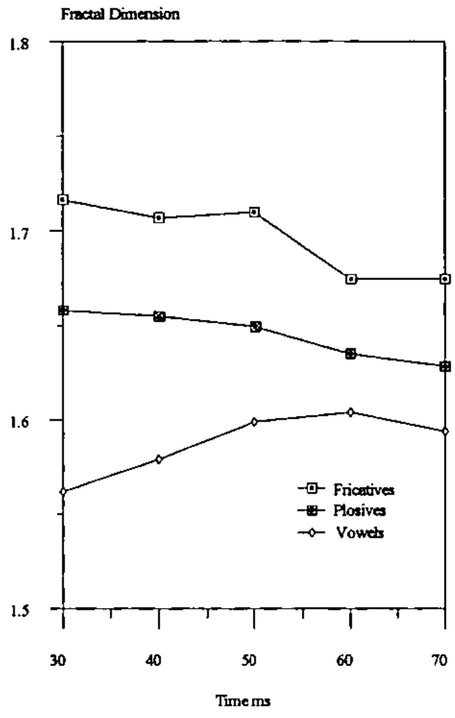
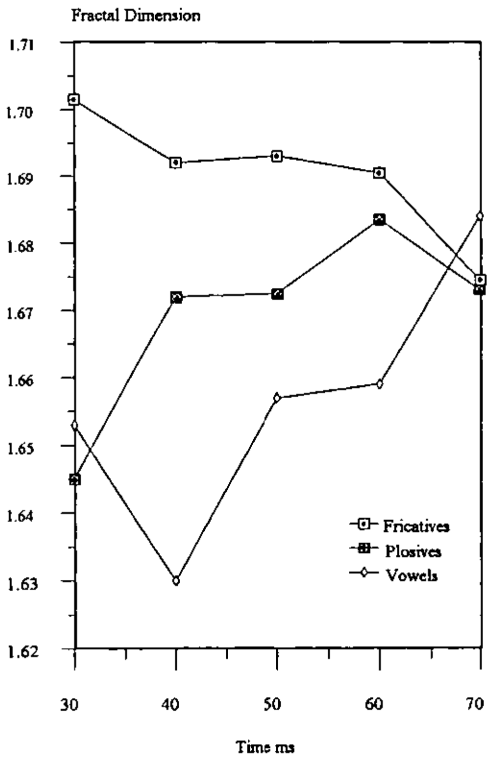


Fig 7.6 Richardson Technique - Female Speaker



The three graphs above each provide some evidence to suggest that the Richardson fractal technique may not be suitable for speech characterisation. The traces for Male Speaker 1 are actually quite similar and appear to follow a pattern in that there appears to be a peak in the dimension at the 50 ms time slot for the three phonetic categories. This may be as a result of the resolution of the graphical waveforms and the size of the 'dividers' selected to obtain a fractal reading. As with the Box Counting technique the fricative average is greater than the plosive average which in turn is greater than the vowel average, the peak values being 1.75, 1.65 and 1.53 respectively.

The situation for the second male speaker and the female speaker is quite different. For Male Speaker 2 though the order of dimension is the same as for Male Speaker 1, ie. fricative elements greater than plosive elements greater than vowel elements, the fricative and plosive traces both show a negative trend with increased time while the vowel trace peaks at the 60 ms time slot and then falls away. Though the traces are different from those of Male Speaker 1 there is still some degree of stability in them.

The Richardson traces for the Female Speaker could well be describes as not only being unstable but even *chaotic* compared to those in the previous graphs. At the 30 ms time slot the fricative trace shows an average fractal dimension of approximately 1.7, the same as the two male speakers. However the plosive and vowel traces are almost inseparable both being around the 1.65 mark which for the plosive trace is comparable to the male graphs but for the vowels is significantly greater. There after both the plosive and vowel traces display a positive trend while the fricative trace goes negative culminating in all three traces finishing with an approximate average fractal dimension of 1.65. At no time slot does there appear to be a period of stability between the traces. This could be due to the technique being more sensitive to the female phonetic elements suggesting perhaps that the technique may be gender dependent. Much more experimentation would of course be necessary to support such a claim.

7.4 The Minkowski-Bouligand Graphs

Fig 7.7 Minkowski Bouligand Technique - Male Speaker 1

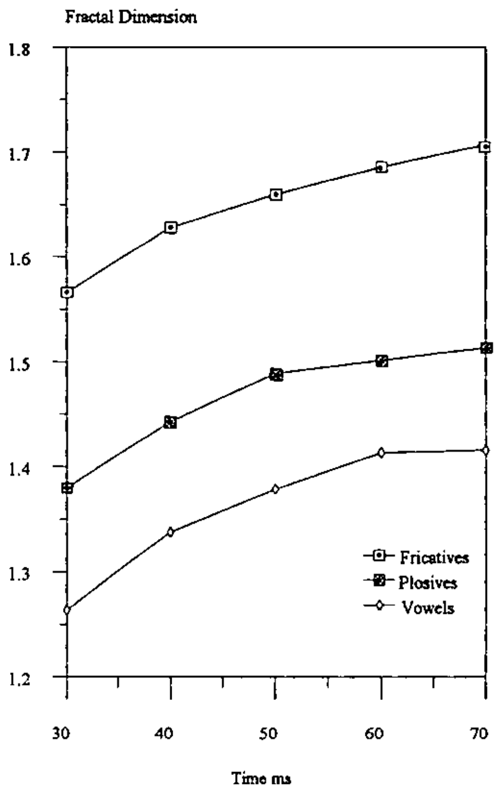


Fig 7.8 Minkowski-Bouligand Technique - Male Speaker 2

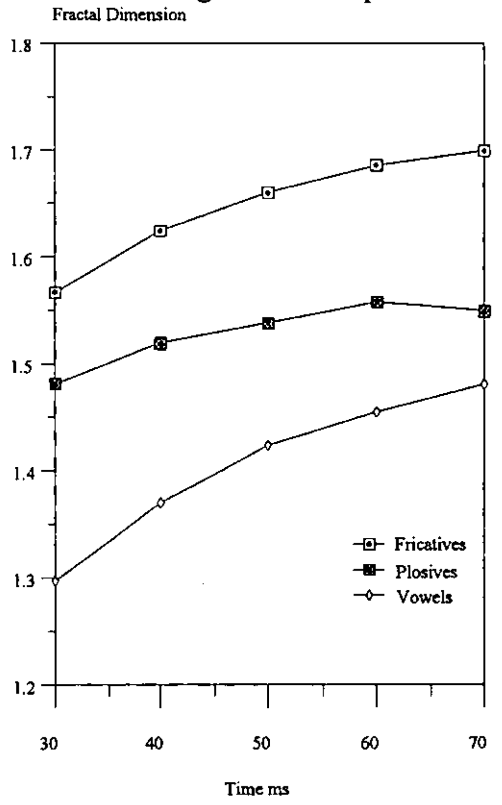
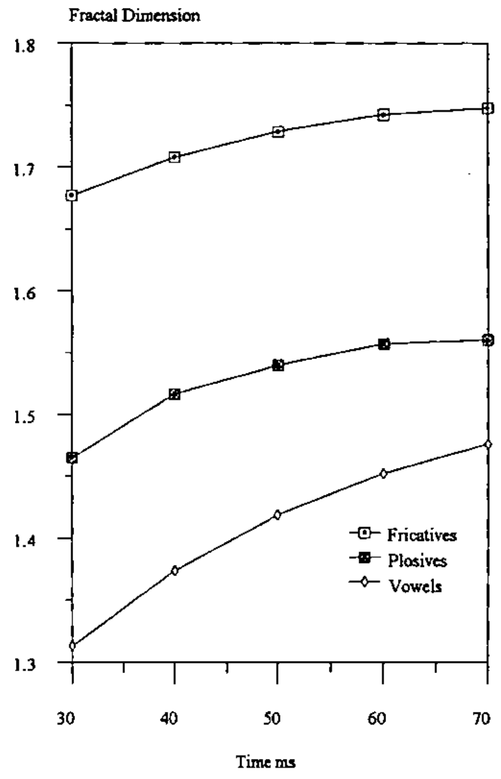


Fig 7.9 Minkowski-Bouligand Technique - Female Speaker



Comparing the above three graphs for the Minkowski-Bouligand technique with those for the Box Counting technique immediate similarities can be seen. The traces are consistent throughout which suggest the technique provides reliable results. The fricative trace gives significantly greater values than the plosive and vowel traces for all three speakers. Likewise all the traces show a positive trend which levels out at the 70 ms time slot culminating in average fractal dimensions of approximately 1.70 for fricative elements, 1.55 for the plosive elements and 1.45 for the vowel elements for all three speakers. This represents a more significant spread of values between the fractal dimension boundaries of the three phonetic categories than the Box Counting technique produced.

Chapter 8

Conclusion

The conclusion of this thesis can be introduced by examining the question of whether or not the aim of the research project, described in chapter 1, was successfully met. To reiterate:

The purpose of this research project was to follow on from the work carried out by Cliff Pickover and Al Khorsani [1] and by Petros Maragos [2] and to investigate whether classes of phonemes or phonetic elements of the English language could be consistently segmented using fractal dimension techniques and whether elements in those classes could themselves be distinguished.

In addressing the problem the production of speech has been discussed in some detail with particular emphasis on the classification of vowels, plosive and fricative consonants.

The concept of Mandelbrot's fractal dimension [3] has been introduced along with three methods by which the fractal dimension of a pattern can be calculated. The Box Counting method [4] which gives a dimension calculated by counting the number of cells a pattern intersects and then repeating the exercise with different cell sizes. The Richardson method [5] which gives a dimension calculated by counting the number of 'divider' lengths taken circumscribe the pattern, repeating the exercise with different lengths. Finally, the Minkowski-Bouligand method [4] which gives a dimension calculated by counting the number of discs taken to cover the perimeter of fractal pattern, repeating the exercise with various radii of disc.

From the results obtained and the subsequent graphical analysis carried out, from a statistical point of view using the Box Counting and the Minkowski-Bouligand techniques, fricative phonetic elements can segmented from plosive elements which in turn can be segmented from vowel elements irrespective of the speaker. The Richardson method however has proved to be unsuitable, even for such broad categorisation.

As also stated in the introduction, the usefulness of fractal geometric mathematics used as a model for characterising speech is inherently limited by the accuracy to which a fractal dimension can be safely calculated, and further by the limited scope over which such dimensions can exist and this has certainly proven to be the limiting factor for segmenting elements within each of the three phonetic speech groups described above. Though some evidence has been found to suggest that fricative and plosive elements can be segmented into their respective voiced and unvoiced subdivisions, none of the three techniques examined provide conclusive enough results to state that a fractal technique can reliably be used for such segmentation. It may be that a different algorithm must be investigated to substantiate such a claim as what is certain is that aperiodic sounds which are associated with unvoiced elements do have a greater fractal dimension than periodic sounds which are associated with voiced elements.

The classification of vowels with reference to the position of the tongue too, has proved to be very difficult to substantiate due to the limited fractal range over which vowels tend to lie in.

The remaining phonetic groups, nasal, liquids and semi-vowels could not be distinguished at all from the three main categories. The interesting features that did

appear were that for the male speakers using the Box counting and the Minkowski-Bouligand methods the elements within the phonetic groups listed above could actually be reasonably segmented though this was not the case for the female speaker.
Before these conclusions can be categorically confirmed more work needs to be carried out using more fractal methods with many more speakers of various gender. If the results found in this research project are found to be consistent with future work then there is no question that a new use for Mandelbrot's ingenious geometric mathematics will have been found.

References

[1] Pickover C., Khorsani A. 1986 'Fractal Characterization of Speech Waveform Graphs' Computer & Graphics Vol.10 No. 1, pp 51-61

[2] Maragos P. 1991 'Fractal Aspects of Speech Signals : Dimension and Interpolation' Proceedings IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP) Vol. 1 pp 417-424

[3] Mandelbrot B.B. 1977 'Fractals Form, Chance and Dimension' W.H. Freeman and Company San Francisco

[4] Richardson, L.F. 1961 'The Problem of contiguity: an appendix of statistics of deadly quarrels.' General Systems Yearbook 6, 139-187

[5] Feder J. 1988 'Fractals' Penum Press New York and London

Appendix A

Software Design

A.1 Introduction

The purpose of this chapter is to discuss the software techniques that were used to create the various programs that could display speech data and calculate fractal dimensions.

All the software was developed using Turbo Pascal 6.0 on a 386 33 MHz PC running under DOS 6.0. The speech recordings were made on a Speech Workstation developed by Loughborough Sound Images Ltd.

Various algorithms were experimented with before satisfactory results for the three fractal techniques, as discussed in chapter 4, were obtained. Each algorithm will be outlined along with a discussion about it's relative use ability, i.e. speed of process, accuracy of results etc.

A.2 Program Structures

Each of the programs were written in a modularised fashion to facilitate the divide-and-conquer approach to design. This generally leads to more reliable, functional software which is easier to understand and test.

The basic program structure was identified as outlined below :

1. Initialisation

2. Data read in
3. Data transformation
4. Graphical waveform display
5. Fractal processes
6. Calculations
7. Result display and store

A.2.1 Initialisation

The initialisation process consists of setting up the system to operate in a graphics mode. The setting up of all paths and file names that the program uses is carried out in an auxiliary file called GLOBALS.PAS. This file also contains information used by the data transforming procedures, in particular the scaling figure (discussed later), the number of iterations carried out by the fractal process and the time scale of the file examined (e.g. 30 ms, 40 ms, full file). By using an auxiliary file it was possible to write batch programs that could test many recordings consecutively thus saving a considerable amount time.

A.2.2 Data Read In

The format of the digitised sound produced by the Speech Workstation is a binary file containing 16 bit samples, the first 8 bits of each sample being the most significant byte. The data is read in by the program in a hexadecimal format and is immediately converted into a format so that the signal is centred on the X axis, i.e. that the d.c. component of any waveform of any signal appears as a line running horizontally across the middle of the screen.

A.2.3 Data Transformation

Due to possible differences in gain at the time of sampling it was decided that each of the files read in would be amplitude normalised before the fractal process was applied to it.

This was achieved by calculating the standard deviation of the recording and dividing each of the samples by this figure. The samples were than scaled up and written to a temporary file so that the displayed waveform would fit just within the limits of the screen parameters, i.e. 479 by 639 pixel display. The procedure also use the parameter, 'file_extent', in the auxiliary file to normalise the time scale that 'file_extent' corresponds to. For example if 'file_extent' is set to 1200 and the sampling frequency was 40 kHz then only the mid 30 ms of that file would be normalised with respect to itself and written to the temporary location.

A.2.4 Graphical Waveform Display

The graphical waveform display procedure uses the data in the temporary file created by the normalisation procedure to display the waveform. As the number of samples to be displayed is often much greater than the X-axis screen resolution, the procedure effectively 'compresses' the number of samples passed to it so that the waveform can be displayed in its entirety. This ensures that the fractal process operates on the same 'area' of data regardless of the time scale of speech data examined.

A.2.5 Fractal Processes

The principle and origins behind each of the fractal methods is explained in detail in chapter 4.

The software procedures written to carry out the three fractal processes each use the graphical information in a slightly differently manner. The Box Counting method procedure relies solely on the waveform image that is produced on the screen to calculate a fractal dimension whereas the Richardson method needs to know the XY location of the first data sample and the Minkowski Bouligand procedure requires the XY location of every sample step by step in order to calculate the fractal dimension. This will be explained in more depth in the sections below.

A.2.5.1 The Richardson Method

The Richardson method is perhaps the most well known technique for calculating the fractal dimension of fractal patterns. It is based on a technique similar to stepping a set of dividers of a set length round a fractal pattern, such as a coastline, to estimate the perimeter length. The fractal dimension is calculated by repeating the process using dividers of greater or smaller length to establish a range of estimates for the perimeter length.

The initial programs that were written to simulate this procedure required the XY location sequentially of every sample of a waveform file in order to superimpose a set of lines of set pixel length through the waveform, thus imitating the action of dividers. The resulting log log graphs were distinctly non linear indicating that this method was incorrect for calculating a fractal dimension as a straight line could not be reasonably fitted to the points obtained.

For that reason the algorithm was altered so that the lines approximating the length of the waveform only measured the perimeter of the pattern rather than the absolute or 'stretched out' length of the pattern which the earlier algorithms were trying to measure. This had the effect of producing log log graphs which were linear enough for a line of regression to be fitted much closer to the points obtained.

The procedure which carries out this later algorithm requires only the XY location of the first sample point in order to establish a starting position. The waveform is first drawn in white and the procedure then simulates an 'out swing' measurement of the pattern. This is best thought of as analogous to one using a pair of dividers in order to measure a coastline, but only allowing the dividers to swing in one direction, either outward or inward.

Using a polar co-ordinate method, the procedure examines the pixel colour at vertical 0 degrees right round to vertical 180 degrees at the set divider length until a white pixel (corresponding to the waveform) is found. A line is then drawn to this point which then becomes the starting point for the next line to be drawn. The number of lines or 'divider steps' to approximate the waveform is then counted and recorded and the process is then iterated with different line sizes to establish a range of measurements which are then stored in an array.

The speed of the of the process depends very much on the number and size of dividers used to obtain a result. Using large divider sizes, although quick, can produce very non linear results. The rule of thumb used is that the largest divider length should not be more than half the length of the fractal pattern at its widest or highest point.

A.2.5.2 The Minkowski-Bouligand Method

The Minkowski-Bouligand dimension is based on calculating the area of a set of circular discs that cover the fractal pattern at different locations which are eroded or dilated producing various results of area measurement of a given fractal pattern.

The procedure which carries out this operation relies on the same file that the waveform display procedure uses to produce the speech image on the screen. The file is completely re-read by the fractal procedure and at the XY location of every sample, a disc of set pixel radius is drawn. The procedure actually goes further than this because for small radii, the resulting pattern using this method is a series of discrete spots on the screen which only covers the waveform at its extremities rather than a complete covering of discs. Though this is actually another method for producing a variation of the Minkowski-Bouligand dimension, known as the Packing dimension, experimentation has shown that this method does not produce reliable results when used on speech waveforms in such a manner.

For that reason the procedure used not only covers the waveform with discs at all the XY locations of the file samples, but actually covers as many points as necessary between each sample in order to produce a 'blanket covering' of discs over the whole waveform.

The waveform is initially drawn in white and the discs laid over it drawn in green. The procedure then interrogates the colour of every pixel on the screen (306081 locations) and counts every green pixel that it then finds which is effectively a measure of area. The process then repeats with dilated or eroded discs which produces a different area of measurement, all results then being stored in an array.

The process is inherently slow because of the re-reading of the file and further calculations necessary to produce the 'blanket covering' and because of the fact that every pixel must be examined after each iteration regardless of the diameter of disc used.

A.2.5.3 The Box Counting Method

The Box Counting method is by far the most efficient algorithm at obtaining the necessary figures to calculate a fractal dimension for a speech waveform. The technique used to calculate it is very simple and is particularly suited to computer processing.

A grid pattern of set cell size is first drawn on the screen so that when the speech waveform is laid over it, the image intersects the grid pattern at various locations. These intersects produced are then detected and the number of cells affected is counted. The program memorises the top left XY location of each cell and draws the grid pattern in red and the waveform in white. Each pixel of every cell is then interrogated until the top left XY location is reached or a white pixel is detected, indicating that the cell has been intersected by the waveform. This process is then repeated on the next cell and so until all the cells of the grid pattern have been examined. The number of cells intersected for that particular grid size is then stored in an array. The process is then repeated with different cell sizes of smaller resolution which produces a different result.

The process is extremely quick as the algorithm needs only count the pixels that form the grid and as soon as an intersection is found, the next cell is examined.

A.2.6 Calculations

The calculation of a fractal dimension for all three methods is based upon fitting a line of simple linear regression, $y = a + bx$, to a graph of the log log plot of measuring device size against number of devices counted to represent the waveform.

The observed number of steps, area or cells intersected for each iteration carried out by the fractal process are stored in one array and the equivalent line, disc or cell size are stored in another array. This makes carrying out the necessary logarithmic calculations straightforward.

For n pairs of observations (x_i, y_i)

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

For the purpose of finding a fractal dimension the calculation procedure first finds the mean values of the logarithmically transformed arrays which is achieved using a simple sum and divide with a FOR DO loop sub procedure.

The denominator of the equation is calculated next in a similar manner with the exception of the inclusion of the square factor in another sub procedure.

The summations of the numerator are then carried out in the main calculation procedure again using a simple FOR DO loop. The Fractal Dimension is then just the result of dividing the numerator by the denominator except for Minkowski Bouligand method where this result must be subtracted from 2.

A.2.7 Result Display and Store

The fractal dimension is immediately out putted to the screen for the user to read. The result of calculation along with the file name, the measurements taken and the size of measuring devices used are then stored on disk in two file formats. The first is in a text file which can be updated to continually keep track of batch programs which carry out multiple tests. The second is in a binary format which can then subsequently be read by a graph package to check the validity and linearity of the result.

Summary

The software techniques used to achieve the calculation of a fractal dimension of speech have been discussed and the structure of the individual programs broken down and examined. The relative performance of the three fractal programs used has also been discussed in brief.

Appendix B

Code Listings

Initialisation file

```
UNIT Globals;
interface

const
        iterations = 6; {sets number of iterations for fractal process}
        datapath = 'h:\project\data\07_04_93\'; {sets path}
        dosfile = 'p_1.wks';  {sets file}
        File_Extent = 2800; {sets extnt of file to be examined (40 = 1 ms) }
        scale_factor = 5000; { sets scaling factor after normalisation }

implementation

begin
end.
```

Box Counting Program

```
Program box_counting_method;

uses dos,crt,graph,globals;

type
   square_sizes = array[1..iterations] of integer;
   box_count_calculator = array[1..iterations] of real;
   sizes_string = array[1..iterations] of string;
var
  fractal_dimension : real;
  total_boxes : box_count_calculator;
  box_counter : longint;
  square_size : square_sizes;
  scale_factor : integer;
  MaxYLimit : Integer;
  MaxXLimit : Integer;
  MinYLimit : Integer;
  MinXLimit : Integer;
  steps : integer;
  stan_dev : real;
  Pixels : Integer;
  File_Extent : word;
  deltaX : Real;
  dosfile : string;
  speechfile : file of word;
  temp : file of real;
  conver : sizes_string;
  boxes : sizes_string;

procedure fd_initialise;
var auxcount : integer;
begin
   fd_Set_Drivers;
   assign (SpeechFile,DataPath+dosfile);
   reset (speechfile);
   if FileSize (SpeechFile) < 2800 then
     file_extent := FileSize (SpeechFile);
   stan_dev := 0;

   for auxcount := 1 to iterations do
      square_size[auxcount]  := auxcount+3;   { sets box sizes }
   {square_size[1]:= 2;
   square_size[2]:= 4;
   square_size[3]:= 8;
   square_size[4]:= 16;
   square_size[5]:= 32;}
end;



procedure FD_Sizes (sq_size : integer);    { sets relative sizes of screen dimension }
var bestxfit,bestyfit : integer;
   clipon : boolean;
begin
   {    Graph view port sizes     }
   bestxfit := trunc (getmaxx/sq_size);
   bestyfit := trunc (getmaxy/sq_size);
   MaxYLimit := trunc(getMaxY - (getmaxy-(bestyfit*sq_size))/2);
   MaxXLimit := trunc(getMaxX - (getmaxx-(bestxfit*sq_size))/2);
   MinYLimit := maxylimit-sq_size*bestyfit;
   MinXLimit := maxxlimit-sq_size*bestxfit;
   Pixels := bestxfit*sq_size;
   rectangle(minxlimit,minylimit,maxxlimit,maxylimit);
end;

procedure FD_Exit;   { Graph clean up procedure }
begin
```

```
   closegraph;
   restorecrtmode;
   window (1,1,80,25);
   close(SpeechFile);
end;

procedure FD_Set_Drivers;  { Initailises all grahic fonts and drivers }
var
  graphdriver,
  graphmode,
  error : integer;
begin
   graphdriver := detect;                { autodetect the hardware }
   initgraph (graphdriver,graphmode, ''); { activate graphics }
   if graphresult <> grOk then           { any errors? }
   begin
      writeln('graphics init error: ', GraphErrorMsg(GraphDriver));
      halt(1);
   end;
   cleardevice;
end;

function scale_value (var plotvalue : real) : real;
begin                                              { scales values relative }
   plotValue:= (plotValue+32768)/(65535/(479));      { to screen dimnsions }
   scale_value := plotvalue;
end;




procedure read_sample (var plotvalue : real);      { reads actual file }
var sample : word;
     x,sampleint : integer;
begin
   read (SpeechFile,sample);
   sampleInt := sample;
   plotValue:= round (sampleint);
end;

procedure convert_file;  { used to change the format of files if necessary }
var sample,x : byte;
   plotvalue : real;
   speechfile_2 : file of byte;
   speechfile_3 : file of byte;
   speechfile_4 : file of word;
   sample_word : word;
begin
```

```
   assign(speechfile_3,'d:\project\07_04_93\s_1.wks');
   reset(speechfile_3);
   assign(speechfile_2,'moretemp.raw');
   rewrite(speechfile_2);
   repeat
       read (SpeechFile_3,Sample);
       x := sample;
       read(speechfile_3,sample);
       write(speechfile_2,sample);
       write(speechfile_2,x);
   until eof(speechfile_3);
   close(speechfile_2);
   close(speechfile_3);

   assign(speechfile_4,'moretemp.raw');
   reset(speechfile_4);
   rewrite(speechfile);
   repeat
       read (SpeechFile_4,Sample_word);
       write(speechfile,sample_word);
   until eof (speechfile_4);
   close(speechfile);
   close(speechfile_4);

   reset(speechfile);
   repeat
       read_sample (plotvalue);
   until eof (speechfile);
end;



procedure fd_read_draw;   { draws the waveform on screen }
var plotvalue : real;
   X_wor : integer;
   c : char;
   clipon : boolean;
   xpos : real;
begin
   reset(temp);
   setcolor (white);
   deltaX := (pixels / file_extent);
   xpos := 0;
   setviewport(minxlimit,minylimit,maxxlimit,maxylimit,clipon);
   read (temp,plotvalue);
   moveto(0,round(plotvalue));
   repeat
       xpos := xpos + deltaX;
       read (temp,plotvalue);
```

```
        lineto(round(xpos),round(plotvalue));
    until eof(temp);
    setviewport(0,0,getmaxX,getmaxY,clipon);
end;

procedure write_to_temp;
var z : integer;
   plotvalue : real;
begin
    assign (temp,'update.tmp');
    rewrite (temp);
    for z := 1 to file_extent do
       begin
          read_sample (plotvalue);
          plotvalue := (plotvalue / stan_dev) * scale_factor;
          plotvalue := scale_value (plotvalue);
          write (temp,plotvalue);
       end;
    close (temp);
    reset(temp);
end;

{procedure fd_normalise; } {  carries out amplitude normalisation of data }
{var
  plotvalue,mean,mean_sum,x : real;
  int_value : integer;
begin
    reset(speechfile);
    mean_sum := 0;x := 0;
    repeat
        read_sample (plotvalue);
        int_value := round(plotvalue);
        mean_sum := mean_sum + (plotvalue);
    until eof (speechfile);
    mean := mean_sum / filesize (speechfile);
    reset(speechfile);
    repeat
        read_sample (plotvalue);
        x := x + sqr(plotvalue - mean);
    until eof (speechfile);
    stan_dev := sqrt(x/ filesize (speechfile));
    reset (speechfile);
    write_to_temp;
end;}

procedure fd_normalise;
var
   plotvalue,mean : real;
   int_value,local_scale,total_scale : integer;
   mean_sum : real;
   x : real;
begin
    mean_sum := 0; x := 0;
    total_scale := filesize (speechfile);
    local_scale :=  file_extent;
    if total_scale < local_scale then
      local_scale := total_scale
    else
       seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
    repeat
        read_sample (plotvalue);
        int_value := round(plotvalue);
        mean_sum := mean_sum + (plotvalue);
    until filepos(speechfile) = (trunc((total_scale / 2) - local_scale / 2)+file_extent);
    mean := mean_sum / file_extent;
    reset(speechfile);
    seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
    repeat
        read_sample (plotvalue);
        x := x + sqr(plotvalue - mean);
    until filepos(speechfile) = (trunc((total_scale / 2) - local_scale / 2)+file_extent);
    stan_dev := sqrt(x/ file_extent);
    reset (speechfile);
    seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
    write_to_temp;
end;

procedure calc_mean (variables : box_count_calculator;
                     var mean : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
       summation := summation + variables[aux_count];
    mean := summation / iterations;
end;

procedure calc_numerator (variables : box_count_calculator;
                          mean : real;
                          var ans : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
       summation := summation + (variables[aux_count] - mean);
    ans := summation;
```

```
end;

procedure calc_denominator (variables : box_count_calculator;
                    mean : real;
                    var ans : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
       summation := summation + sqr(variables[aux_count] - mean);
    ans := summation;
end;

procedure fd_calculate;        { primary fractal dimension procedure }
var log_total_boxes, log_square_size : box_count_calculator;
   auxcount : integer;
   mean_log_square_size, mean_log_total_boxes : real;
   X,Y,var_x, summation : real;
   fd_string : string;
   ch : char;
begin
    for auxcount := 1 to iterations do
       begin
          log_total_boxes[auxcount] := (ln(total_boxes[auxcount]));
          log_square_size[auxcount] := (ln(square_size[auxcount]));
       end;
    calc_mean (log_total_boxes,mean_log_total_boxes);
    calc_mean (log_square_size,mean_log_square_size);
    calc_denominator(log_square_size,mean_log_square_size,var_X);
    summation := 0;
    for auxcount := 1 to iterations do
       summation := summation + ((log_total_boxes[auxcount] -
mean_log_total_boxes)
                *(log_square_size[auxcount] - mean_log_square_size));
    fractal_dimension := ((summation / var_X))*-1;
    str(fractal_dimension:3:3,fd_string);
    gotoxy(minxlimit,minylimit);
    outtext('fractal dimension = '); outtext (fd_string);
end;

procedure line_walk (var x,y : integer;
                var match : boolean);
var colour : word;
begin
    colour := getpixel(x,y);
    if colour = white then
      begin
          inc(box_counter);
          match := true;
          exit;
       end;
end;


procedure boundary_walk (x,y : integer;
                    var match : boolean;
                    sq_size : integer);
var auxcount,
   x1,y1: integer;
begin
   x1 := x; y1 := y;
   match := false;
   moveto(x1,y1);
   if sq_size <> 1 then
   begin
       for auxcount := 1 to sq_size do
          begin
              line_walk (x1,y1,match);
              if match then exit;
                 putpixel(x1,y1,yellow);
              inc(x1);
          end;
       for auxcount := 1 to sq_size do
          begin
              line_walk (x1,y1,match);
              if match then exit;
                 putpixel(x1,y1,yellow);
              inc(y1);
          end;
       for auxcount := 1 to sq_size do
          begin
              line_walk (x1,y1,match);
              if match then exit;
                 putpixel(x1,y1,yellow);
              dec(x1);
          end;
       for auxcount := 1 to sq_size do
          begin
              line_walk (x1,y1,match);
              if match then exit;
                 putpixel(x1,y1,yellow);
              dec(y1);
          end;
       end
    else
       begin
           line_walk (x1,y1,match);
```

```
        putpixel(x1,y1,yellow);
      end
end;

procedure fd_count_squares(var sq_size:integer); { primary box count procedure }
var inner_count,outer_count,x,y,x_lines,y_lines,total_squares : integer;
   match : boolean;
begin
   box_counter := 0;
   moveto(minxlimit,minylimit);
   x := minxlimit;
   y := minylimit;
   x_lines := trunc((maxxlimit-minxlimit) div sq_size);
   y_lines := trunc((maxylimit-minylimit) div sq_size);
   total_squares := x_lines * y_lines;
   for outer_count := 1 to y_lines do
   begin
      x := minxlimit;
      for inner_count := 1 to x_lines do
         begin
            boundary_walk(x,y,match,sq_size);
            x:= x + sq_size;
         end;
      y := y + sq_size;
   end;
end;

procedure fd_draw_grid (square_size : integer);  { draws grid over waveform }
var
 x1, y1, x2, y2,auxcount,reps,x_lines,y_lines : Integer;
begin
   setcolor (lightred);
   x1 := (maxxlimit); y1 := (maxylimit); x2 := (minxlimit); y2 := (minylimit);
   x_lines := trunc((maxxlimit-minxlimit) div square_size);
   y_lines := trunc((maxylimit-minylimit) div square_size);
   for auxcount := 1 to x_lines + 1 do
      begin
         moveto(x2,minylimit);
         lineto(x2,maxylimit);
         x2 := x2 + square_size;
      end;
   for auxcount := 1 to y_lines + 1 do
      begin
         moveto(maxxlimit,y1);
         lineto(minxlimit,y1);
         y1 := y1 - square_size;
      end;
end;
```

```
procedure write_to_file; { save results in a pre named file }
var outfile : text;
   saveerror, auxcounter : integer;
   fd_string,space : string;
   boxes_str, sizes_str : string[10];
   totalstring : string;
begin
    assign(outfile,'result17.rec'); { file name }
    {$I-}
    append(outfile);
    saveerror := IOresult;
    if saveerror <> 0 then
      rewrite(outfile);
    str(fractal_dimension:3:3,fd_string);

    for auxcounter := 1 to iterations do
       begin
           str(square_size[auxcounter],conver[auxcounter]);
           while length(conver[auxcounter]) <> 7 do
               insert(' ',conver[auxcounter],length(conver[auxcounter]) +1);
           str(trunc(total_boxes[auxcounter]),boxes[auxcounter]);
           while length(boxes[auxcounter]) <> 7 do
               insert(' ',boxes[auxcounter],length(boxes[auxcounter]) +1);
       end;
    space := '      ';
    writeln(outfile);
    writeln(outfile,dosfile,space,fd_string);
    write(outfile,space);
    for auxcounter := 1 to iterations do
       write(outfile,conver[auxcounter]);
    writeln(outfile);
    write(outfile,space);
    for auxcounter := 1 to iterations do
       write(outfile,boxes[auxcounter]);
    writeln(outfile);
    close(outfile);
    {$I+}
end;

procedure write_for_graph;   { save results into file for use with graph package }
var outfile : text;
   count : integer;
   name : string;
begin
    count:= length(dosfile);
    name := copy(dosfile,1,count-3);
    assign(outfile,'d:\project\box_coun\box_data\'+name+'rec');
    rewrite(outfile);
    for count := 1 to iterations do
```

```
        writeln(outfile,square_size[count]);
     writeln(outfile);
     for count := 1 to count do
        writeln(outfile,total_boxes[count]);
     close(outfile);
end;

procedure fd_Display;          { control procedure }
var auxcount,
    sq_size : integer;
begin

    { convert_file;}
     fd_normalise;
     for auxcount := 1 to iterations do
     begin
         sq_size := square_size[auxcount];
         fd_sizes(sq_size);
         fd_draw_grid(sq_size);
         fd_Read_Draw;
         fd_count_squares(sq_size);
         total_boxes[auxcount] := box_counter;
         cleardevice;
     end;
     fd_Calculate;
    { write_to_file;}
     write_for_graph;
end;

       { MAIN PROGRAM }
begin
     fd_Initialise;
     fd_Display;
     close(temp);
     fd_exit;
end.
```

Minkowski Bouligand Program

```
Program Minkowski_Bouligand_Method;

uses dos,crt,graph,globals;

type
    radii = array[1..iterations] of integer;
    area_calculator = array[1..iterations] of longint;
    fract_calculator =array[1..iterations] of real;
var
```

```
  x1,y1,x2,y2,x2_ref,y2_ref : real;
  box_counter : longint;
  radius : radii;
  areas : area_calculator;
  scale_factor : integer;
  MaxYLimit : Integer;
  MaxXLimit : Integer;
  MinYLimit : Integer;
  MinXLimit : Integer;
  Start_Size : integer;
  fall_rate : integer;
  steps : integer;
  stan_dev,fractal_dimension : real;
  colourset : integer;
  Pixels : Integer;
  File_Extent : LongInt;
  deltaX,hyp : Real;
  Y_Axis_Scale : LongInt;
  SpeechFile : File of Word;
  temp : file of real;
  {box_result : box_store;}

procedure FD_Set_Drivers;  { Initailises all grahic fonts and drivers }
var
  GraphDriver, GraphMode, Error : integer;
Begin
   GraphDriver := Detect;              { autodetect the hardware }
   InitGraph(GraphDriver, GraphMode, ''); { activate graphics }
   if GraphResult <> grOk then         { any errors? }
   begin
      Writeln('Graphics init error: ', GraphErrorMsg(GraphDriver));
      Halt(1);
   end;
   ClearDevice;
End;

procedure fd_initialise;
var aux_count : integer;
begin
   fd_set_drivers;
   assign (SpeechFile,DataPath+{Soundbase.DosFile}dosfile);
   reset (speechfile);
   Y_Axis_Scale := 65535;
   File_Extent := FileSize (SpeechFile);

   for aux_count := 1 to iterations do
      radius[aux_count] := aux_count  ;
   stan_dev := 0;
   scale_factor := 5000;
   pixels := getmaxx;
end;

procedure FD_Exit;   { Graph clean up procedure }
Begin
   closegraph;
   restorecrtmode;
   Window (1,1,80,25);
   Close(SpeechFile);
End;

function Scale_value (var plotvalue : real) : real;
begin
   PlotValue:= (PlotValue+32768)/(65535/(479));
   scale_value := Plotvalue;
end;

procedure read_sample (var plotvalue : real);
var sample : word;
  sampleint : integer;
begin
   read (SpeechFile,Sample);
   SampleInt := Sample;
   PlotValue:= Sampleint;
end;

procedure fd_read_draw; { Reads selected file and draws all data on graph }
var plotValue : Real;
  X_Word : Integer;
  C : Char;
  ClipOn : Boolean;
  xpos : real;
Begin
   reset(temp);
   SetColor (White);
   deltaX := (Pixels / file_Extent); {function of read in size}
   XPos := 0;
   read (temp,plotvalue);
   MoveTo(0,Round(PlotValue));
   repeat
       XPos := XPos + deltaX;
       read (temp,plotvalue);
       LineTo(Round(XPos),Round(PlotValue));
       seek(temp,FilePos(temp)); {funtion of readin size}
   until FilePos(temp) >= file_extent;
End;

procedure fd_normalise;   {standard deviation method}
var
```

```
   plotvalue,mean : real;
   int_value,z : integer;
   mean_sum : real;
   x: real;
begin
   mean_sum := 0;
   x := 0;
   repeat
       read_sample (plotvalue);
       int_value := round(plotvalue);
       mean_sum := mean_sum + (plotvalue);
   until eof (speechfile);
   mean := mean_sum / filesize (speechfile);
   reset(speechfile);
   repeat
       read_sample (plotvalue);
       x := x + sqr(plotvalue - mean);
   until eof (speechfile);
   stan_dev := sqrt(x/ filesize (speechfile));
   reset (speechfile);
   assign (temp,'update.tmp');
   rewrite (temp);
   for z := 1 to file_extent do
   begin
      read_sample (plotvalue);
      plotvalue := (plotvalue / stan_dev) * scale_factor;
      plotvalue := scale_value (plotvalue);
      write (temp,plotvalue);
   end;
   close (temp);
   reset(temp);
end;

procedure calc_mean (variables :fract_calculator;var mean : real);
var summation : real;
   aux_count : integer;
begin
   summation := 0;
   for aux_count := 1 to iterations do
      summation := summation + variables[aux_count];
   mean := summation / iterations;
end;

procedure calc_numerator (variables : fract_calculator;mean : real;var ans : real);
var summation : real;
   aux_count : integer;
begin
   summation := 0;
   for aux_count := 1 to iterations do
```

```
      summation := summation + (variables[aux_count] - mean);
    ans := summation;
end;

procedure calc_denominator (variables : fract_calculator;mean : real;var ans : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
       summation := summation + sqr(variables[aux_count] - mean);
    ans := summation;
end;


procedure FD_Calculate;
var log_areas,log_radii : fract_calculator;
   auxcount : integer;
   mean_log_radii, mean_log_areas : real;
   X,Y,var_x,summation : real;
   fd_string : string;
   ch : char;
begin
    for auxcount := 1 to iterations do
       begin
          log_areas[auxcount] := (ln(areas[auxcount])) / 2.302585093;
          log_radii[auxcount] := (ln(radius[auxcount])) / 2.302585093;
       end;
    calc_mean (log_areas,mean_log_areas);
    calc_mean (log_radii,mean_log_radii);
    calc_denominator(log_radii,mean_log_radii,var_X);
    summation := 0;
    for auxcount := 1 to iterations do
       summation := summation + ((log_areas[auxcount] - mean_log_areas)
                  *(log_radii[auxcount] - mean_log_radii));
    fractal_dimension := ((summation / var_X));

    str(fractal_dimension:3:3,fd_string);
    moveto(30,30);
    outtext('fractal dimension = '); outtext (fd_string);
    ch := readkey;
end;

procedure fd_count_area(rad : word);
var colour : word;
   pixel_counter : longint;
   x,y : integer;
begin
    pixel_counter := 0;
    for y := 0 to getmaxy do
       for x := 0 to getmaxx do
          begin
             colour := getpixel(x,y);
             if colour = lightgreen then
               inc(pixel_counter);
             putpixel(x,y,yellow);
          end;
    areas[rad {div 2}] := pixel_counter;
end;

procedure check_centres;
begin
    hyp := sqrt(sqr(x2-x1) + sqr(y2-y1));
end;

procedure cover_line(rad : word;var No_of_circles :integer; var Circle_counter :
longint);
var x_divisions, y_divisions : real;
   aux_x,aux_y : real;
   counter : integer;
begin
    x_divisions := (x2-x1)/No_of_circles;
    y_divisions := (y2-y1)/No_of_circles;
    x1 := x1 + X_divisions/2;
    y1 := y1 + Y_divisions/2;
    pieslice(Round(X1),Round(y1),0,360,rad);
    inc(circle_counter);
    for counter := 1 to no_of_circles-1 do
       begin
           x1 := x1 + x_divisions;
           y1 := y1 + y_divisions;
           pieslice(Round(X1),Round(y1),0,360,rad);
           inc(circle_counter);
       end;
    x1 := x2;
    y1 := y2;
end;

procedure overlay_circles (rad : word);
var PlotValue: real;
   Circle_Counter : longInt;
   no_of_circles  : integer;
   C : Char;
   ClipOn : Boolean;
   xpos : real;
Begin
    circle_counter :=0;
    reset(temp);
   setfillstyle(1,lightgreen);
   SetColor (lightgreen);
   deltaX := (Pixels / file_Extent); {function of read in size}
   XPos := 0;
   read (temp,plotvalue);
   y1 := plotvalue;
   x1 := xpos;
   repeat
       read (temp,plotvalue);
       XPos := XPos + deltaX;
       y2 := plotvalue;
       x2 := xpos;
       check_centres;
       No_of_circles := round((hyp)/(2*rad));
       if No_of_circles = 0 then
         No_of_circles :=1;
       cover_line(rad,No_of_circles,circle_counter);
   until eof(temp);
   {areas[rad div 3] := trunc(pi*(sqr (rad)) * circle_counter);}
end;


procedure FD_Display;          { Main Menu choice procedure }
var auxcount,sq_size : integer;
   rad : word;

begin
   fd_Initialise;
   fd_normalise;
   for auxcount := 1 to iterations do
   begin
      fd_Read_Draw;
      rad := radius[auxcount];
      overlay_circles(rad);
      fd_count_area(rad);
      cleardevice;
   end;
   fd_Calculate;
End;


{ MAIN PROGRAM }

Begin
   fd_Display;
   close(temp);
   {fd_exit;}
End.
```

Richardson Method Program

```
Program Richardsons_method;

uses graph,dos,crt,globals;

type
   Step_store = array[1..iterations] of real;
var
  scale_factor : integer;
  step_size : step_store;
  steps : integer;
  stan_dev : real;
  fractal_dimension : real;
  Pixels : Integer;
  dosfile : string;
  G_Extent : LongInt;
  deltaX : Real;
  SpeechFile : File of Word;
  temp : file of real;
  Dividers : real;
  step_result : step_store;
  file_extent : word;

procedure FD_Set_Drivers;  { Initailises all grahic fonts and drivers }
var
  GraphDriver, GraphMode, Error : integer;
Begin
   GraphDriver := Detect;                { autodetect the hardware }
   InitGraph(GraphDriver, GraphMode, ''); { activate graphics }
   if GraphResult <> grOk then           { any errors? }
   begin
      Writeln('Graphics init error: ', GraphErrorMsg(GraphDriver));
      Halt(1);
   end;
   ClearDevice;
End;

procedure fd_initialise;
begin
   FD_Set_Drivers;
   assign (SpeechFile,DataPath+dosfile);
   reset (speechfile);
   file_extent := 400;  {10 ms}
   G_Extent := FileSize (SpeechFile);
   stan_dev := 0;
   Pixels := getmaxx;
   scale_factor := 5000;
```

```
end;

procedure FD_Params;
var count : integer;
begin
   {for count := 1 to iterations do
      step_size[count] := count+1;}
   step_size[1] := 2;
   step_size[2] := 4;
   step_size[3] := 8;
   step_size[4] := 16;
   step_size[5] := 32;
   step_size[6] := 64;
end;

procedure FD_Exit;   { Graph clean up procedure }
Begin
   closegraph;
   restorecrtmode;
   Close(SpeechFile);
End;

function Scale_value (var plotvalue : real) : real;
begin
   PlotValue:= (PlotValue+32768)/(65535/(479));
   scale_value := Plotvalue;
end;

procedure read_sample (var plotvalue : real);
var sample : word;
   sampleint : integer;
begin
    read (SpeechFile,Sample);
   SampleInt := Sample;
   PlotValue:= round (Sampleint);
end;

procedure FD_Read_Draw; { Reads selected file and draws all data on graph }
var PlotValue : Real;
   Counter,X_Word : Integer;
   C : Char;
   ClipOn : Boolean;
   xpos : real;
Begin
    reset(temp);
    SetColor (White);
    deltaX := (Pixels / file_extent); {g_extent}
    XPos := 0;
    read (temp,plotvalue);
   MoveTo(0,Round(PlotValue));
   Counter := 0;
   REPEAT
      XPos := XPos + deltaX;
      read (temp,plotvalue);
      LineTo(Round(XPos),Round(PlotValue));
      Inc (Counter);
   UNTIL eof(temp);
End;

{procedure fd_normalise;
var
   plotvalue,mean : real;
   int_value,z : integer;
   mean_sum : real;
   x: real;
begin
   mean_sum := 0;
   x := 0;
   repeat
      read_sample (plotvalue);
      int_value := round(plotvalue);
      mean_sum := mean_sum + (plotvalue);
   until eof (speechfile);
   mean := mean_sum / filesize (speechfile);
   reset(speechfile);
   repeat
      read_sample (plotvalue);
      x := x + sqr(plotvalue - mean);
   until eof (speechfile);
   stan_dev := sqrt(x/ filesize (speechfile));
   reset (speechfile);
   assign (temp,'update.tmp');
   rewrite (temp);
   for z := 1 to g_extent do
   begin
      read_sample (plotvalue);
      plotvalue := (plotvalue / stan_dev) * scale_factor;
      plotvalue := scale_value (plotvalue);
      write (temp,plotvalue);
   end;
   close (temp);
   reset(temp);
end;}

procedure fd_normalise;
var
   plotvalue,mean : real;
   int_value,z,local_scale,total_scale : integer;
```

```
  mean_sum : real;
  x : real;
begin
  reset(speechfile);
  mean_sum := 0; x := 0;
  total_scale := filesize (speechfile);
  local_scale :=  file_extent;

  if total_scale <= local_scale then
    begin
      local_scale := total_scale;
      file_extent := local_scale;
    end
  else
    seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
  repeat
     read_sample (plotvalue);
     int_value := round(plotvalue);
     mean_sum := mean_sum + (plotvalue);
  until filepos(speechfile) = (trunc((total_scale / 2) - local_scale / 2)+local_scale);
  mean := mean_sum / local_scale;
  reset(speechfile);
  seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
  repeat
     read_sample (plotvalue);
     x := x + sqr(plotvalue - mean);
  until filepos(speechfile) = (trunc((total_scale / 2) - local_scale / 2)+local_scale);
  stan_dev := sqrt(x/ local_scale);
  reset (speechfile);
  seek(speechfile,trunc((total_scale / 2) - local_scale / 2));
  assign (temp,'update.tmp');
  rewrite (temp);
  for z := 1 to local_scale do
  begin
     read_sample (plotvalue);
     {plotvalue := (plotvalue / stan_dev) * scale_factor;}
     plotvalue := scale_value (plotvalue);
     write (temp,plotvalue);
  end;
  close (temp);
  reset(temp);
end;


procedure top_draw (var x1,y1 : real; var actual_steps : real;var last_angle :integer);
var angle : integer;
  x2,y2 : real;
  colorfound : real;
  triggered : boolean;
```

```
begin
   if x1 <= pixels then
   begin
      for angle := 180 downto 0 do
         begin
            triggered := false;
            x2 := x1 + dividers *(sin(angle*(pi/180)));
            y2 := y1 + dividers *(cos(angle*(pi/180)));
            colorfound := getpixel(trunc(x2),trunc(y2));
            if (colorfound = white) or (colorfound = lightred) then
              if (last_angle = 0) and (angle = 180) then
              else
                 begin
                    line(trunc(x1),trunc(y1),trunc(x2),trunc(y2));
                    x1 := x2; y1 := y2;
                    last_angle := angle;
                    angle := 0;
                    actual_steps := actual_steps + 1;
                    triggered := true;
                 end;
         end;
      if not triggered then
        begin
           x2 := x1 + dividers *(sin(45*(pi/180)));
           y2 := y1 + dividers *(cos(45*(pi/180)));
           line(trunc(x1),trunc(y1),trunc(x2),trunc(y2));
           x1 := x2; y1 := y2;
           actual_steps := actual_steps + 1;
           last_angle := 45;
        end;
   end;
end;

procedure bottom_draw (var x1,y1 : real; var actual_steps : real;var last_angle :
integer);
var angle: integer;
   x2,y2 : real;
   colorfound : word;
   triggered : boolean;
begin
   if x1 <= pixels then
   begin
      for angle := 0 to 180 do
         begin
            triggered := false;
            x2 := x1 + dividers *(sin(angle*(pi/180)));
            y2 := y1 + dividers *(cos(angle*(pi/180)));
            colorfound := getpixel(trunc(x2),trunc(y2));
            if (colorfound = white) or (colorfound = lightred) then
```

```
              if (last_angle = 180) and (angle = 0) then
              else
                 begin
                    line(trunc(x1),trunc(y1),trunc(x2),trunc(y2));
                    x1 := x2; y1 := y2;
                    last_angle := angle;
                    angle := 180;
                    actual_steps := actual_steps + 1;
                    triggered := true;
                 end;
         end;
      if not triggered then
        begin
           x2 := x1 + dividers *(sin(135*(pi/180)));
           y2 := y1 + dividers *(cos(135*(pi/180)));
           line(trunc(x1),trunc(y1),trunc(x2),trunc(y2));
           x1 := x2; y1 := y2;
           actual_steps := actual_steps + 1;
           last_angle := 135;
        end;
   end;
end;


procedure fd_fract (dividers : real; var actual_steps : real);
var plotvalue : real;
   x1_pos,y1_pos,x1_neg,y1_neg : real;
   pos_last_angle,neg_last_angle : integer;
begin
    pos_last_angle := 0;
    neg_last_angle := 0;
    setcolor(lightred);
    actual_steps := 0;
    reset(temp);
    read (temp,plotvalue);
    y1_pos := plotvalue; y1_neg := plotvalue;
    x1_pos := 0; x1_neg := 0;
    repeat
         top_draw (x1_pos,y1_pos,actual_steps,pos_last_angle);
         bottom_draw (x1_neg,y1_neg,actual_steps,neg_last_angle);
    until (x1_pos >= pixels-2) and (x1_neg >= pixels-2);
end;

procedure calc_mean (variables : step_store;var mean : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
```

```
        summation := summation + variables[aux_count];
    mean := summation / iterations;
end;

procedure calc_numerator (variables : step_store;mean : real;var ans : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
        summation := summation + (variables[aux_count] - mean);
    ans := summation;
end;

procedure calc_denominator (variables : step_store;mean : real;var ans : real);
var summation : real;
   aux_count : integer;
begin
    summation := 0;
    for aux_count := 1 to iterations do
        summation := summation + sqr(variables[aux_count] - mean);
    ans := summation;
end;


procedure FD_Calculate;
var log_size, length, log_length : step_store;
   auxcount : integer;
   mean_log_size, mean_log_length : real;
   X,Y,var_x,summation : real;
   fd_string : string;
  ch : char;
 begin
    for auxcount := 1 to iterations do
        begin
            length[auxcount] := step_result[auxcount] * step_size[auxcount];
            step_result[auxcount] := round (step_result[auxcount]);
            log_length[auxcount] := (ln(length[auxcount]) / 2.3026);
            log_size[auxcount] := (ln(step_size[auxcount])) / 2.3026;
        end;
    calc_mean (log_length,mean_log_length);
    calc_mean (log_size,mean_log_size);
    calc_denominator(log_size,mean_log_size,var_X);
    summation := 0;
    for auxcount := 1 to iterations do
        summation := summation + ((log_length[auxcount] - mean_log_length)
                  *(log_size[auxcount] - mean_log_size));
    fractal_dimension := ((summation / var_X) -1)*-1;
    str(fractal_dimension:3:3,fd_string);
    moveto(30,30);
    setcolor(white);
    outtext('fractal dimension = '); outtext (fd_string);
    {ch := readkey;}
end;

procedure write_to_file;
type sizes_string = array[1..iterations] of string;
var outfile : text;
   saveerror, auxcounter : integer;
   fd_string,space : string;
   conver : sizes_string;
   boxes : sizes_string;
   boxes_str, sizes_str : string[10];
   totalstring : string;

begin
    assign(outfile,'result25.rec');
    {$I-}
    append(outfile);
    saveerror := IOresult;
    if saveerror <> 0 then
      rewrite(outfile);
    str(fractal_dimension:3:3,fd_string);

    for auxcounter := 1 to iterations do
       begin
           str(trunc(step_size[auxcounter]),conver[auxcounter]);
           while length(conver[auxcounter]) <> 7 do
               insert(' ',conver[auxcounter],length(conver[auxcounter]) +1);
           str(trunc(step_result[auxcounter]),boxes[auxcounter]);
           while length(boxes[auxcounter]) <> 7 do
               insert(' ',boxes[auxcounter],length(boxes[auxcounter]) +1);
       end;
    space := '      ';
    writeln(outfile);
    writeln(outfile,dosfile,space,fd_string);
    write(outfile,space);
    for auxcounter := 1 to iterations do
       write(outfile,conver[auxcounter]);
    writeln(outfile);
    write(outfile,space);
    for auxcounter := 1 to iterations do
       write(outfile,boxes[auxcounter]);
    writeln(outfile);
    close(outfile);
    {$I+}
end;
```

```
procedure write_for_graph;
var outfile : text;
   count : integer;
   name : string;
begin
    count:= length(dosfile);
    name := copy(dosfile,1,count-3);
    assign(outfile,'d:\project\richmeth\ric_data\'+name+'rec');
    rewrite(outfile);
    for count := 1 to iterations do
       writeln(outfile,step_size[count]);
    writeln(outfile);
    for count := 1 to count do
       writeln(outfile,step_result[count]);
    close(outfile);
end;


procedure FD_Display;          { Main Menu choice procedure }
var auxcount:integer;
   steps_counted : real;
Begin
    ClearDevice;
    fd_normalise;
    fd_params;
    for auxcount := 1 to iterations do
       begin
           fd_read_draw;
           dividers := step_size[auxcount];
           FD_Fract(dividers,steps_counted);
           step_result[auxcount] := steps_counted;
           cleardevice;
       end;
    FD_Calculate;
    {write_to_file;}
    {write_for_graph;}
End;

{ MAIN PROGRAM }

Begin
    fd_Initialise;
    FD_Display;
    close(temp);
End.
```