Spectral Techniques For Roughness Estimation

Mark Lewis

August 31, 2001

Abstract

Roughness is a relatively untouched field considering its significance to natural scientists. In this thesis mathematical techniques for measuring the roughness of signals are constructed and investigated. Both one dimensional and two dimensional signals are tackled. Applications include geological profiles and biological surfaces. Mathematical techniques include Fourier and Wavelet Transforms.

Acknowledgements

There are many people who helped make this thesis possible

Hugh Murrell for his left-field ideas and always being available to help with the problems they presented

Henda Swart for her continuous financial support and encouragement

Colin Jermy for supplying the geological samples and numerous roughness papers and books

Tally Palmer for supplying the biological samples

Wayne Rasband for developing the image processing shell ImageJ

Thank you to them all.

Preface

Submitted in fulfillment of the academic requirements for the degree of Master of Science (M.Sc.) in the School of Geological and Computer Sciences, University of Natal, Durban campus.

This thesis represents original work by the author and has not been submitted in part or whole to this or any other university. Parts of this thesis have been accepted for publication in the South African Computer Journal (SACJ) [10]. Where use has been made of work of others, it is duly acknowledged in the text.

This thesis was typeset in LaTeX.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer Science post-graduate students are faced with two options: choose a Computer Science topic - probably quite theoretical - to research, or look outside their department and find some scientific application required by a third party. Here at the University of Natal, Durban campus (UND) we have been blessed: the decision to merge departments into schools meant Computer Science was joined with Geology, and a surfeit of problems the geologists needed solving were suggested as research topics. One of the problems proposed was that of measuring the roughness of a profile. The ability to predict how much two surfaces in contact with one another will slip is significant to an engineering geologist. The Geology Department was kind enough to furnish a set of Barton's profiles (from [1]) in digital format, the profiles themselves are used as a reference by geologists when measuring roughness - for more information see [1] or [2].

Rhodes University's Institute of Water Research (IWC) also had an interest in roughness; they use the degradation of may-flies' gills to measure the kaolin content of rivers. The more kaolin in the water, the more the gills will degrade, and the rougher will be their appearance. Thus their requirement for roughness measuring software. The (IWC) provided the $2D$ sample images.

The challenge then is to construct numerical techniques to measure the roughness of both $1D$ geological profiles and $2D$ biological sample signals. The scientists working with these problems currently sort their samples by measuring roughness with eyesight. Hopefully the techniques developed here will manage to classify samples in the same order that they have been ranked to this point.

1.1 A Roughness Definition

Trying to find a standard roughness definition is almost impossible. Russ in [18] defines: "A conventional measurement of surface roughness is the "rms" or *root-mean-square* roughness. For an elevation profile or surface image, this is determined by standard descriptive statistics. The mean value of the elevation values is the average surface elevation. The variance is the sum of the squares of the deviations of individual values from this mean divided by the number of points, and the standard deviation or "rms" value is the square root of the variance." This definition could best be described as a statistical attempt to measure roughness, and has the advantage of working for either profiles or surfaces.

Another commonly used roughness evaluation method (at least by geologists) is the *Joint Roughness Coefficient* (JRC) measure proposed by Barton in [1]. This measure suggests roughness coefficients (JRC values) for a profile in the range 0 to 20. The JRC value is estimated by visual matching with the profiles presented as Barton's Profiles in Appendix A. The method has the advantage of being readily available to geologists in the field, but the disadvantage of only working for profiles. In this thesis Barton's profiles will be used as a set of test profiles.

The third recognized measure for roughness estimation is based on the *fractal dimension* D of either profiles or surfaces. The use of fractals to measure roughness was suggested by Mandelbrot in [11]. For a profile the fractal dimension is a real number in the range $[1, 2]$

whilst for a surface the fractal dimension is in the range $[2, 3]$. The larger the fractal dimension D for a profile or surface, the "rougher" the profile or surface. The *box-counting* algorithm implemented in this thesis is a fractal measure which can be used on both profiles and surfaces.

A scientist looking for information on surface roughness measures will do best to search for books on fractals. Such books seem the most likely place to find any references to roughness, and focus around using fractal measures, like the fractal dimension of a surface. Most popular is the box-counting method. These methods are discussed in, e.g. [13] and [18].

In this thesis the traditional spatial domain techniques currently used by engineering geologists are compared with new methods based on spectral domain techniques from the engineering literature.

Chapter 2

The Simplified Roughness Problem

Before looking at numerical codes it is important to indicate what assumptions are being made about the signals to be analyzed.

2.1 One Dimension Signals

Starting with $1D$ signals, assume that there are 2^n samples which are evenly spaced and lie in the range $[0,\ 255]$ with the minimum being 0. The reason for this last assumption is that roughness must be independent of the mean of the signal. Our roughness calculations will be scaled to produce a measure in the range $[0,\ 2^{n-1}-1]$, for reasons evident after considering two dimensional signals.

2.2 Sampling Two Dimensional Signals

2.2.1 Rose Diagrams

In the case of $2D$ signals again assume evenly spaced samples on a square 2^n by 2^n grid. Again the sample values lie in the range

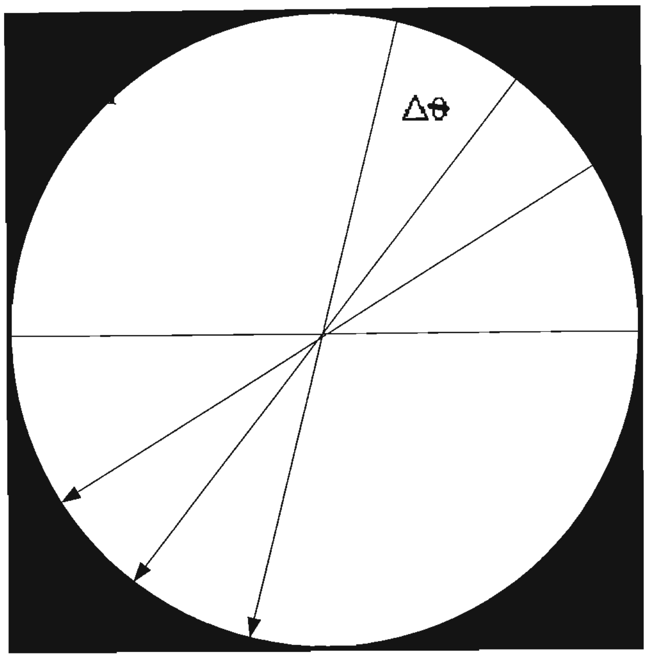


Figure 2.1: Circular sampling

$[0,\ 255]$. To get a measure of roughness for a $2D$ signal take a series of $1D$ slices through the centre of the $2D$ signal and sample these slices at 2^n points generating $1D$ signals. This is illustrated in figure 2.1.

Use the $1D$ techniques to calculate the roughness of each slice. This $1D$ roughness measure can then be used to construct a roughness rose diagram for the $2D$ signal. This rose diagram can be superimposed on top of the $2D$ signal to get a picture of roughness versus direction for the signal. Figure 2.2 shows an example rose diagram. At this point the decision to scale all roughness values to the range of $[0,\ 2^{n-1}-1]$ makes sense: this forces the roughest surface to have a roughness rose-diagram circular in shape, which in extreme cases can just touch the perimeter of the sampling circle. An "overall" roughness according to the area of the rose diagram is then computed. Note that this circular sampling of $2D$ signals will exclude signal detail outside the perimeter of the sampling circle.

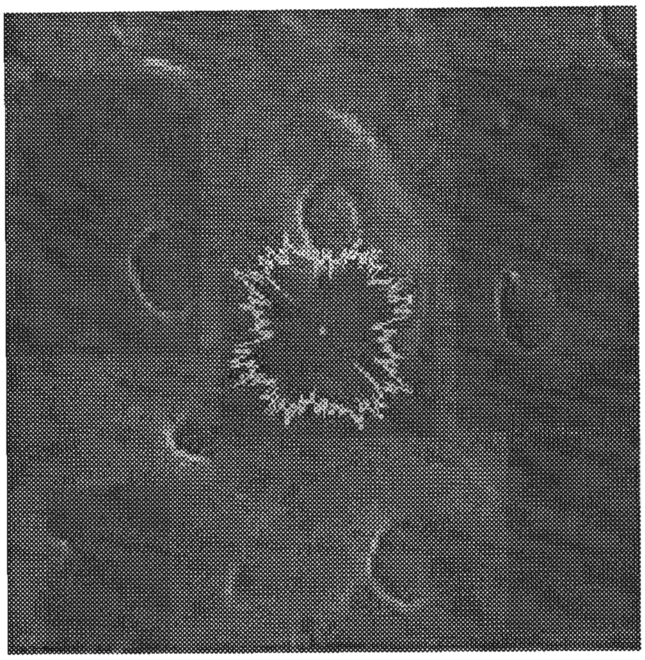


Figure 2.2: An example rose diagram

2.3 Algorithms Overview

Now that the basic rules have been stated, numerical algorithms can be considered. The first distinction to make is the difference between spatial domain, frequency domain, and space/frequency domain analysis techniques.

Spatial domain calculations use the raw signal values with some form of algorithm to ascertain roughness. The output values of these algorithms are then "scaled" to conform to the above-mentioned rules. One of the spatial domain methods used in this project counts how many times the signal value trends change direction: i.e. either hit the peak of a continuous rise in values and start to fall again, or encounter the bottom of a trough in values and start to rise.

Whilst methods like this may seem trivial, they outperformed the transform domain methods with some sample sets.

Frequency domain analysis takes place after the signal values have been transformed to the frequency domain using a Discrete Fourier Transform (DFT). The frequency amplitudes are then used as input to a roughness algorithm.

Finally Discrete Wavelet Transforms are used to transform signals to space/frequency domains, and the resulting coefficients are then used as input to roughness algorithms.

Six key roughness algorithms have been implemented, three spatial domain measures, one frequency domain measure and two space/frequency measures. The methods are: Direction Change Counting (spatial), Height Increment Counting (spatial), Box Counting (spatial), Fast Fourier Transform (frequency), Haar Wavelet Transform (space/frequency) and Daubechies 4 Coefficient Wavelet Transform (space/frequency).

The Direction Change Counting, Height Increment Counting and Box Counting algorithms are covered in chapter 3. Chapter 4 contains an explanation of the Fourier transform, and discrete fast Fourier transform theory and algorithms, whilst chapter 5 contains the algorithm details and results of the discrete Fourier transform. Chapter 6 is an explanation of the wavelet transform, with the discrete wavelet transform included, with chapter 7 having the algorithm details and results (similar to chapter 5). Chapter 8 describes the image processing shell used and has instructions for compiling and running the plugins coded by the author. Chapter 9 is a conclusion which highlights project successes and failures, and indicates which direction future work should be aimed toward.

Unless otherwise mentioned, the algorithms coded have all been developed by the author for the purposes of this thesis.

Chapter 3

Spatial Domain Measures

Two algorithms (Direction Change and Height Increase) used in this chapter have been developed by the author, the Box Counting algorithm is based on ideas suggested in [13].

3.1 Direction Change Measure

One of the more intuitive methods to gauge roughness is to assess how jagged a signal is: the more changes in slope direction (from up to down, or down to up) across the signal, the more rough the signal is. What becomes apparent whilst analyzing the sample data with this Direction Change Count metric is the need to discard any false peaks and troughs. What are false peaks and troughs and how are they caused? Sample data can originate from a variety of sources ranging from scanned images in books to digital photographs of microscopic samples. The input devices used must always perform some sort of sampling and quantization. As a result, some values may be interpolated, omitted, or even assigned maximum or minimum cut offs. Whilst these erroneous values are easily detected and countered for by a human measuring roughness by sight only, a computer program will treat the erroneous values like all the others. The solution is to implement a threshold so that the user can specify how wide a trend change must be in order to be counted as such. The algorithm for a signal $\{s_j\}_{j:\, 0,\ldots\, 2^n-1}$ is as follows:

3.1.1 Algorithm

```
roughness = 0
for $j = 1$ to $2^n - 1$ do
  if $s_j > s_{j-1}$ and downwardTrend = true then
    if trendLength() > threshold then
      roughness++
      downwardTrend = false
    end if
  else if $s_j < s_{j-1}$ and downwardTrend = false then
    if trendLength() > threshold then
      roughness++
      downwardTrend = true
    end if
  end if
end for
return roughness
```

If a user is confident that the signal values are all correct, then a threshold of 0 or 1 will suffice. The less confident the user of her samples, the larger the threshold value should be. The downside to thresholding is that genuine short trend direction changes will be overlooked if the threshold value is greater than the trends are long.

3.1.2 Results

The results for the Direction Change Measure are broken down into the $1D$ profile and $2D$ images, which denote Barton's geological profiles and the gill flap biological samples respectively. The 10 profiles in Barton's roughness set, listed in Appendix A, were analyzed using the $1D$ Direction Change Measure in the hope that the result would be a strictly increasing roughness classification.

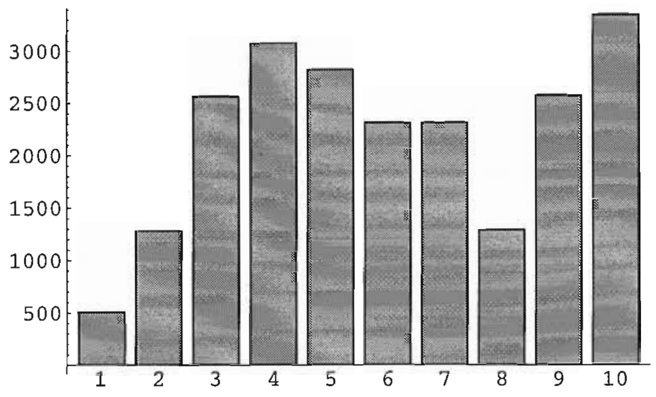


Figure 3.1: Roughness ranking of Barton's profiles by Direction Change Measure

The 7 $2D$ biological sample images, listed in Appendix A, were analyzed using the above algorithm along 256 slices through the image. The number of slices used by the rose-diagram measures always match the pixel width value for the image. the roughness rose diagrams were superimposed on the images and the area of the rose diagram was used as an overall measure of roughness for the image in question. It was hoped that the result would be a strictly increasing roughness classification of the images.

A full set of results for each measure can be found in Appendix B. A summary is given for each measure in the text.

$1D$ Results

The $1D$ results for the Direction Change Measure were not altogether successful, the first four profiles were all ranked correctly, but the fifth to eighth elements were ranked as progressively less rough, with the ninth and tenth elements again resuming the upwards trend. The graph in figure 3.1 shows the unsatisfactory ranking of Barton's profiles by the direction change measure.

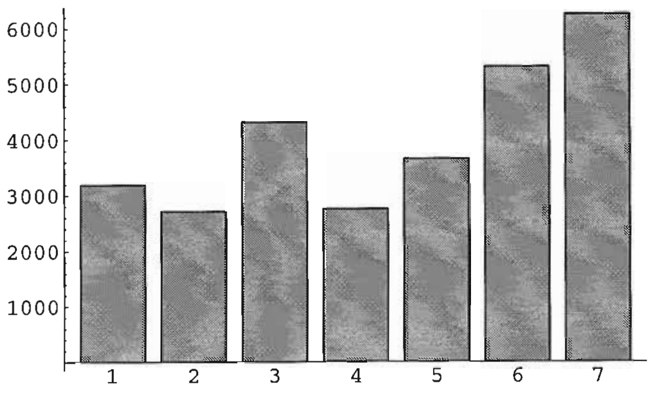


Figure 3.2: Roughness ranking of gill flap samples by Direction Change Measure

$2D$ Results

It must be noted that the third image is contaminated: whoever was responsible for scanning the sample donated their finger or thumb print to the lower left corner; it is clearly discernible!

Of course this sample contamination is the same throughout all the roughness measures, it is interesting to notice how it affects the various measures.

Results for the $2D$ images were considerably better. The ridges of a finger-print make for excellent direction changes, so the roughness measure for the third sample was (somewhat unsurprisingly) quite high considering how relatively plain the image is!

Returning to the results, the first image is marginally more rough than the second and fourth images (we have already mentioned the third sample), but otherwise the images roughness values are returned in the correct order.

Note: the roughness rose diagrams are included along with the results graphs, for all the roughness measures, in Appendix B.

3.2 Height Increase Measure

At first glance the second spatial domain technique seems fairly trivial but turns out to be quite robust in practice. The idea is to compute the total height climbed on each uphill section as one traverses the signal. The algorithm is as follows:

3.2.1 Algorithm

```
roughness = 0
**for** $j = 2$ to $2^n$ **do**
  **if** $s_j > s_{j-1}$ **then**
    roughness $+= (s_j - s_{j-1})$
  **end if**
**end for**
return roughness
```

As this measure can be quite large a user-defined scaling factor is used to scale the totals down. As long as the same scaling value is used for all the samples in a series, scaling will not invalidate the roughness ordering provided.

3.2.2 Results

1*D* Results

After the relative lack of success of the Direction Change Measure, the Height Increase Measure generates excellent results. The measure ranks the profiles in the correct order, all 10 profiles form a steadily increasing curve of roughness values as can be seen in figure 3.3. As this is the simplest roughness measure it will be interesting to see how the more sophisticated measures perform.

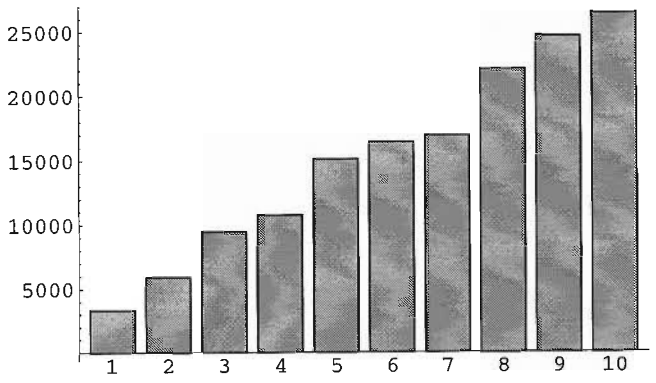


Figure 3.3: Roughness ranking of Barton's profiles by Height Increase Measure

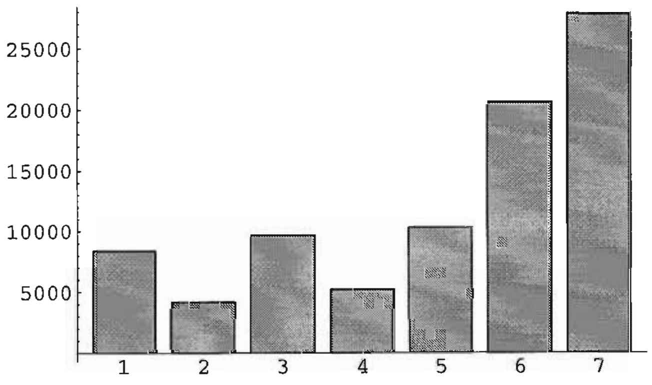


Figure 3.4: Roughness ranking of gill flap samples by Height Increase Measure

$2D$ Results

The Height Increase Measure has very similar $2D$ results to those for the $2D$ Direction Change Measure. The first sample is a little rougher than the second or fourth samples (again omitting the third contaminated sample), but otherwise the samples are all ranked in the correct order.

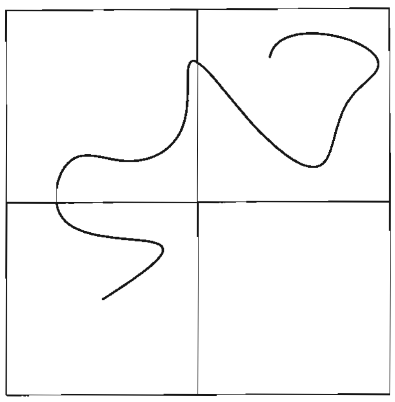


Figure 3.5: Second level

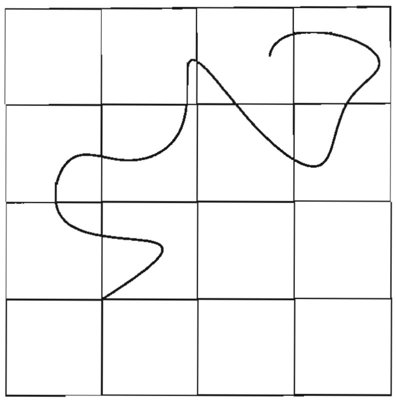


Figure 3.6: Third level

3.3 Box-Counting Roughness Measure

Traditional roughness measuring literature uses fractal methods to measure the roughness of profiles and surfaces. Here the *box-counting* method of calculating a *fractal dimension* value for a profile or surface is used.

Superimpose a grid over the sample image. This grid will necessarily consist of a *mesh* of boxes. The number of boxes in the mesh denotes the mesh size s. For a specific mesh size s, the number of boxes that are occupied (contain some of the sample) is denoted $N(s)$. Now calculate values of $N(s)$ which correlate to different values of s. It is customary to first calculate $N(s)$ for a single box, which should always yield $s = 1$ and $N(s) = 1$. The next iteration is for a quartered image, e.g. figure 3.5, which has $s = 4$, and $N(s) = 3$. Then quarter each quarter and re-calculate, e.g. figure 3.6, which has $s = 16$ and $N(s) = 9$. This quartering process can be repeated *ad infinitum*, or, more usually with computer calculated algorithms, until $s = N^2$, where N^2 is the number of pixels in the sample image.

The values of $\log(N(s))$ vs $\log(s)$ are graphed, and then a best-fit line calculated for the graph's points. The slope of this line is the fractal dimension [13].

Notice that the fractal dimension for a $1D$ profile can only lie between 1.0 and 2.0, but having to try and find a best-fit line can lead to a fractal dimension < 1.0. The algorithm for a signal $\{s_j\}_{j:\,0,\ldots\,2^n-1}$ is as follows:

3.3.1 Algorithm

```
for $j = 0$ to $n$ do
  $p = 2^j$
  for $i = 0$ to $p$ do
    for $x = 0$ to $2^n/p$ do
      if $s_{(i*2^n/p+x)} > max$ then
        $max = s_{(i*2^n/p+x)}$
      else if $s_{(i*2^n/p+x)} < min$ then
        $min = s_{(i*2^n/p+x)}$
      end if
    end for
  end for
  $N(s)$ = boxes from $min$ to $max$
  $s = 2^j$
end for
return $\log(N(s))$ vs $\log(s)$ gradient
```

3.3.2 Results

1*D* Results

The results for Barton's profiles were very good although there were a few profiles which appeared out of sequence. The results graph 3.7 indicates a generally increasing curve.

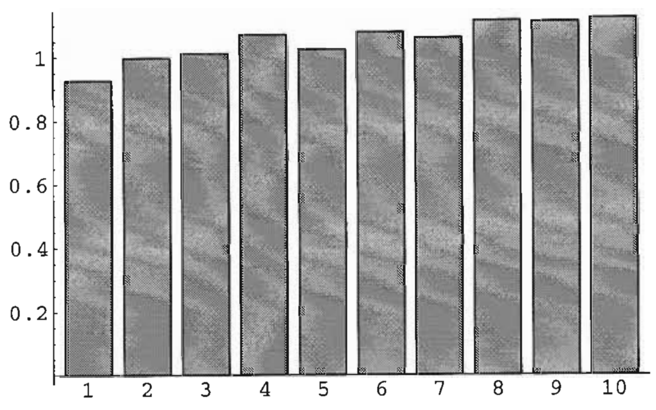


Figure 3.7: Roughness ranking of Barton's profiles by box counting measure

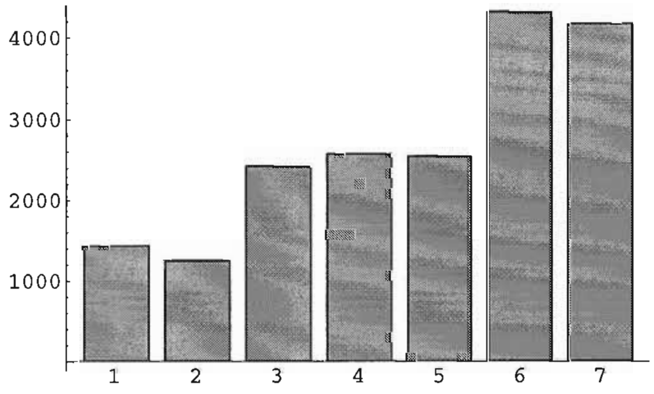


Figure 3.8: Roughness ranking of gill flap samples by box counting measure

$2D$ Results

The gill flap samples also produced satisfying results for the box-count measure, like the other spatial measures the second sample had a slightly lower roughness value than the first, but the measure seemed impervious to the thumb-print in the lower left corner of the third sample, ranking the image correctly, and only the sixth sample appeared out of sequence, being ranked slightly more rough than the seventh sample.

Chapter 4

The Fourier Transform

Much of this material has been adapted from [9], [14] and [15].

4.1 Introduction to the Fourier Transform

The Fourier transform is one of several linear transforms used to solve scientific and engineering spectral problems. Sometimes the Fourier transform is used for computational purposes; for instance to shift a computationally hard problem (in the spatial domain, e.g. convolution or correlation) to the transform domain where it is may be more easily solved; otherwise it can be the transform itself that is of interest, for instance when examining a function's power spectrum.

The Fourier transform has the effect of decomposing a function $f(x)$ into sinusoids of different frequencies, whose respective amplitudes ensure the sinusoids sum to the original function.

A smooth surface will have low frequency components, whilst a rougher surface will have high frequency components. The Fourier transform is a frequency measure, and therefore it seems likely to be an effective roughness measure.

The Fourier transform for a function $f(x)$ is defined by

$$F(u) = \int_{-\infty}^{\infty} f(x)\, e^{i2\pi u x}\, dx \tag{4.1}$$

and the function can be recovered from its Fourier transform according to

$$f(x) = \int_{-\infty}^{\infty} F(u)\, e^{-i2\pi u x}\, du \tag{4.2}$$

provided the integrals exist and any discontinuities are finite. $F(u)$ and $f(x)$ are different representations of the same function, and equations (4.1) and (4.2) are a means of transforming the function between two (different) functional domains. The Fourier transform is often denoted by the transform operator $\mathfrak{F}$.

To entrench the practical uses for the Fourier transform, consider the domain variables x and u. If x was being used to measure time in *seconds*, then the transform units u will be a measure of *cycles per second*, i.e. Hertz. If x were a unit of distance, e.g. *metres*, then the transform units u would be a measure of *cycles per metre*, i.e. inverse wavelength. These relations are summarized in table 4.1.

spatial domain	*frequency domain*
length(metres)	cycles per metre (inverse wavelength)
time(seconds)	cycles per second (Hertz)

Table 4.1: Units of physical functions and their transform equivalents

What properties does the Fourier transform have? There are two important considerations: the first note is that the exponential term $e^{i2\pi u x}$ is a short-hand notation for

$$\cos(2\pi xu) + i\sin(2\pi xu) \tag{4.3}$$

The second consideration is that a function $f(x)$ can always be broken down into even and odd sub-functions as follows

$$f(x) = E(x) + O(x) \tag{4.4}$$

with

$$\begin{aligned} E(x) &= \frac{1}{2}[f(x) + f(-x)] \\ O(x) &= \frac{1}{2}[f(x) - f(-x)] \end{aligned} \tag{4.5}$$

Using this decomposition, the Fourier transform of $f(x)$ becomes

$$F(u) = 2\int_0^\infty E(x)\, \cos(2\pi ux)\, dx + 2\,i\int_0^\infty O(x)\, \sin(2\pi ux)\, dx \tag{4.6}$$

Equation (4.6) reveals that if the function $f(x)$ is real valued then its transform $F(u)$ will be complex valued in general but its real part will be an even function while its imaginary part will be an odd function. These two properties are summarized in table 4.2

spatial domain	*frequency domain*
real and even	real and even
real and odd	imaginary and odd

Table 4.2: Fourier Transform symmetry properties

A complete list of symmetry properties can be found in [15].

The Fourier transform has numerous other properties beyond the symmetry relations described so far.

4.1.1 Spatial Scaling Property

If a is a real, non-zero constant, then

$$\begin{aligned} \mathcal{F}\{f(ax)\} &= \int_{-\infty}^{\infty} f(ax)e^{i2\pi ux}dx \\ &= \frac{1}{|a|}\int_{-\infty}^{\infty} f(\beta)e^{i2\pi\frac{u}{a}\beta}d\beta \\ &= \frac{1}{|a|}F\left(\frac{u}{a}\right) \end{aligned} \tag{4.7}$$

This *spatial scaling property* implies that if a function is compressed in the spatial variable while its amplitude is kept unchanged (i.e. $f(x) \to f(ax), a > 1$), then its Fourier transform will be $\frac{1}{|a|}F\left(\frac{u}{a}\right)$, which is a dilated and scaled down version of the function's "normal" transform $F(u)$.

4.1.2 Frequency Scaling Property

If a is a real, non-zero constant, then

$$\begin{aligned} \mathcal{F}\{\frac{1}{|a|}f\left(\frac{x}{a}\right)\} &= \frac{1}{|a|}\int_{-\infty}^{\infty} f\left(\frac{x}{a}\right)e^{i2\pi ux}dx \\ &= \int_{-\infty}^{\infty} f(\beta)e^{i2\pi au\beta}d\beta \\ &= F(au) \end{aligned} \tag{4.8}$$

which implies that if the period of the function $F(u)$ is shortened to $F(au)$ (i.e. $a > 1$), the reverse transformed function will be $\frac{1}{|a|} f\left(\frac{x}{a}\right)$, i.e. wider and shorter.

The symmetrical nature of the Fourier transform guarantees that the "scaling" of a function in either the spatial or transform domain will lead to some scaling for the function in the opposite domain.

4.1.3 Spatial Translation Property

If x_0 is a real constant then

$$\begin{aligned} \mathfrak{F}\{f(x - x_0)\} &= \int_{-\infty}^{\infty} f(x - x_0) e^{i 2\pi u x} dx \\ &= \int_{-\infty}^{\infty} f(\beta) e^{i 2\pi u (\beta + x_0)} d\beta \\ &= e^{i 2\pi x_0 u} \int_{-\infty}^{\infty} f(\beta) e^{i 2\pi u \beta} d\beta \\ &= F(u) e^{i 2\pi x_0 u} \end{aligned} \tag{4.9}$$

This implies that the Fourier transform of a translated function is the transform of the unshifted function, multiplied by some phase factor.

4.1.4 Frequency Translation Property

$$\begin{aligned}\mathfrak{F}\{f(x)e^{-i2\pi x u_0}\} &= \int_{-\infty}^{\infty} f(x)e^{-i2\pi x u_0}e^{i2\pi x u}dx \\ &= \int_{-\infty}^{\infty} f(x)e^{i2\pi x(u-u_0)}dx \\ &= F(u-u_0)\end{aligned} \tag{4.10}$$

which shows that if $F(u)$ is frequency shifted, it's corresponding inverse transform is no longer real valued.

4.1.5 Convolution

The *convolution* of two functions f and g, denoted $f * g$, is defined by

$$(f * g)(x) \equiv \int_{-\infty}^{\infty} f(x-\tau)g(\tau)d\tau \tag{4.11}$$

Note that $f * g$ is a function in the spatial domain, and that

$$(f * g)(x) = (g * f)(x) \tag{4.12}$$

Now applying definition (4.11), and assuming that $h(x) = f(x) * g(x)$,

$$
\begin{aligned}
\mathfrak{F}\{h(x)\} &= \mathfrak{F}\{f(x) * g(x)\} \\
&= \mathfrak{F}\{\int_{-\infty}^{\infty} f(x-\tau)g(\tau)d\tau\} \\
&= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x-\tau)g(\tau)d\tau\right] e^{i2\pi ux}dx \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\tau)g(\tau)e^{i2\pi ux}d\tau\, dx \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-\tau)g(\tau)e^{i2\pi ux}dx\, d\tau \\
&= \int_{-\infty}^{\infty} g(\tau)\left[\int_{-\infty}^{\infty} f(x-\tau)e^{i2\pi ux}dx\right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau)\left[\int_{-\infty}^{\infty} f(\beta)e^{i2\pi u(\beta+\tau)}d\beta\right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau)e^{i2\pi u\tau}\left[\int_{-\infty}^{\infty} f(\beta)e^{i2\pi u\beta}d\beta\right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau)e^{i2\pi u\tau}d\tau \int_{-\infty}^{\infty} f(\beta)e^{i2\pi u\beta}d\beta \\
&= F(u)\, G(u)
\end{aligned}
$$

This result, known as the Convolution theorem, states that the Fourier transform of a convolution is merely the product of the individual transforms, i.e.

$$\mathfrak{F}\{f(x) * g(x)\} = F(u)G(u) \tag{4.13}$$

This theorem is useful, since calculating the convolution of two functions is computationally expensive, but Fourier transforms can be calculated cheaply.

4.1.6 Correlation

The *correlation* of two functions, denoted $f \otimes g$, is defined as

$$(f \otimes g)(x) = \int_{-\infty}^{\infty} f(\tau + x) g(\tau) d\tau \qquad (4.14)$$

and is often referred to as *lag* [15]. Note that correlation is a function in the spatial domain and

$$(f \otimes g)(x) = (g \otimes f)(-x) \qquad (4.15)$$

Assuming that $h(x) = f(x) \otimes g(x)$,

$$\begin{aligned}
\mathfrak{F}\{h(x)\} &= \mathfrak{F}\{f(x) \otimes g(x)\} \\
&= \mathfrak{F}\{\int_{-\infty}^{\infty} f(x+\tau) g(\tau) d\tau\} \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x+\tau) g(\tau) d\tau \right] e^{i 2\pi u x} dx \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x+\tau) g(\tau) e^{i 2\pi u x} d\tau \, dx \\
&= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x+\tau) g(\tau) e^{i 2\pi u x} dx \, d\tau \\
&= \int_{-\infty}^{\infty} g(\tau) \left[\int_{-\infty}^{\infty} f(x+\tau) e^{i 2\pi u x} dx \right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau) \left[\int_{-\infty}^{\infty} f(\beta) e^{i 2\pi u (\beta-\tau)} d\beta \right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau) e^{-i 2\pi u \tau} \left[\int_{-\infty}^{\infty} f(\beta) e^{i 2\pi u \beta} d\beta \right] d\tau \\
&= \int_{-\infty}^{\infty} g(\tau) e^{-i 2\pi u \tau} d\tau \int_{-\infty}^{\infty} f(\beta) e^{i 2\pi u \beta} d\beta \\
&= F(u) \, \overline{G(u)}
\end{aligned}$$

so the Fourier transform of $f \otimes g$ is the product of F and the complex conjugate of G

$$\mathfrak{F}\{f(x) \otimes g(x)\} = F(u) \, \overline{G(u)} \tag{4.16}$$

Equation (4.16) is known as the Correlation theorem, and is useful when looking for the presence of a reference signal in another signal.

If $f(x)$ and $g(x)$ are different functions, the transform of their correlation is referred to as a *cross-correlation*, otherwise it is known as an *autocorrelation* and is denoted (and calculated) as

$$f(x) \otimes f(x) = |F(u)|^2 \tag{4.17}$$

a result know as the *Weiner-Khinchin* Theorem [15].

4.1.7 Parseval's Theorem

Parseval's theorem is most useful: the power of a signal represented by $f(x)$ in the functional domain and $F(u)$ in the transform domain is the same in both domains

$$\begin{aligned}
\int_{-\infty}^{\infty} |F(u)|^2 du &= \int_{-\infty}^{\infty} F(u)F(u)\, du \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x)e^{i2\pi ux} dx \right] F(u)\, du \\
&= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} F(u)e^{i2\pi ux} du \right] f(x)\, dx \\
&= \int_{-\infty}^{\infty} [f(x)]\, f(x) dx \\
&= \int_{-\infty}^{\infty} |f(x)|^2\, dx
\end{aligned} \tag{4.18}$$

so

$$\text{Total Power} \quad = \int_{-\infty}^{\infty} |f(x)|^2 dx \quad = \int_{-\infty}^{\infty} |F(u)|^2 du \tag{4.19}$$

4.1.8 Power Spectrum Definition

Using Parseval's theorem, the *two-sided power spectral density (PSD)* of the function f can be defined as

$$P_f(u) \equiv |F(u)|^2 + |F(-u)|^2 \tag{4.20}$$

If the function f is real, the PSD simplifies to

$$P_f(u) = 2|F(u)|^2 \tag{4.21}$$

This is another example of a two-sided PSD; most literature will not make reference to the factor of two in equation (4.21), i.e. interchange one and two-sided PSD's.

Often the power between frequencies u and du is required, then it is customary not to consider negative frequencies of u, but rather to use only $0 < u < \infty$, which is a "proper" *one-sided PSD* [15].

Taking a finite section of the function $f(x)$, computing its PSD, and then dividing by the length of the interval gives the *power spectral density per unit length.* Then Parseval's theorem states that the integral of the one-sided PSD per unit length over positive frequency is equal to the mean square amplitude of the signal $f(x)$.

4.2 Discrete Fourier Transform

It is a natural progression to want to use the Fourier transform computationally. It would be ideal to use arrays of evenly spaced sample points of the function $f(x)$ as the input to the Fourier transform algorithm. There is nothing sinister about this suggestion, and yet the impact that the use of sampled points will have on the Fourier transform is substantial.

4.2.1 Sampling Theorem

Assume the function $f(x)$ has been sampled at evenly spaced points in space. Let the specific interval between each sample be denoted by Δ. Then the sample points of $f(x)$ are

$$f_n = f(n\Delta) \qquad \text{n} = \ldots, \text{-}2, \text{-}1, 0, 1, 2, \ldots \tag{4.22}$$

and the reciprocal of the interval Δ is the *sampling rate.* Related to Δ is the *Nyquist critical frequency* f_c, defined by

$$f_c \equiv \frac{1}{2\Delta} \tag{4.23}$$

Nyquist's critical frequency will have both good (the sampling theorem) and bad (aliasing) ramifications.

A function $f(x)$ is *bandwidth limited* if it has no spectral components beyond some frequency β, i.e.

$$F(u) = 0 \qquad \text{for } |u| > 2\pi\beta \tag{4.24}$$

The *sampling theorem* (4.25) is a remarkable result. If a continuous function $f(x)$ is sampled at an interval Δ and is bandwidth limited to frequencies smaller than f_c, i.e. $F(u) = 0$ for all $|u| \geq f_c$, then the function $f(x)$ is completely determined by its samples f_n. In fact $f(x)$ is given explicitly by the formula

$$f(x) = \Delta \sum_{n=-\infty}^{\infty} f_n \frac{\sin[2\pi f_c(x - n\Delta)]}{\pi(x - n\Delta)} \tag{4.25}$$

This result shows that the amount of information carried in a bandwidth limited signal (any low-pass filtered signal) can be described by sampling at a rate Δ^{-1} equal to twice the maximum frequency carried by the signal.

4.2.2 Aliasing

The problem with the sampling theorem is that it assumes the sampled function is bandwidth limited, and this may not be true. Generally speaking, a real-world signal is far more likely to have finite spatial support than be band limited. What happens when the function is not band limited? The answer is quite simple: all the power (PSD) that falls outside the interval $(-f_c, f_c)$ is moved into that range. This occurrence is known as *aliasing*, and once it has happened, there is very little than can be done to remove the aliased power.

How can aliasing be avoided? There are two steps to follow: *i*) either be aware of the natural bandwidth of the signal being sampled or enforce a bandwidth limit by means of a low-pass signal filter; and *ii*) ensure the signal is sampled at a rate which ensures at least two points per cycle of the highest frequency bandwidth occurring.

What if a scientist is given a set of points, and is unsure of whether a natural bandwidth has been observed (or some form of filtering applied), when trying to decide if a signal has been "appropriately" sampled? The answer is logical: when looking at the Fourier transforms of the points, the function should be approaching 0 as the frequency approaches $-f_c$ (from above) and f_c (from below). If the transform is not approaching zero, then there has been a "fold-back" of components outside the range.

4.2.3 Discrete Fourier Transform Derivation

Assuming N consecutive (evenly spaced) sample points, define

$$f_k \equiv f(x_k) \qquad x_k \equiv k\Delta \qquad k = 0, 1, 2, \ldots, N-1 \tag{4.26}$$

For simplicity, assume N even. Assume either that $f(x)$ is only non-zero for the select set of points, or that the selection is "typical" of the function. Using the N values, it will be possible to produce values of the Fourier transform only at N points, specifically at

$$u_n \equiv \frac{n}{N\Delta} \qquad n = -\frac{N}{2}, \ldots, \frac{N}{2} \tag{4.27}$$

which is $N+1$ points! Importantly, because of the periodic nature of aliasing, the two end-values are actually the same value and correspond to the upper and lower limits of the Nyquist critical frequency range, reducing the point count down to N again.

Using this new terminology the derivation is as follows

$$\begin{aligned} F(u_n) &= \int_{-\infty}^{\infty} f(x)\, e^{i 2\pi u_n x}\, dx \\ &\approx \sum_{k=0}^{N-1} f_k\, e^{i 2\pi u_n x_k} \Delta \\ &= \Delta \sum_{k=0}^{N-1} f_k\, e^{i 2\pi k n/N} \end{aligned} \tag{4.28}$$

The final summation is referred to as the *discrete Fourier transform* (DFT) of the N points x_k, denoted F_n

$$F_n \equiv \sum_{k=0}^{N-1} f_k\, e^{i 2\pi k n/N} \tag{4.29}$$

The discrete Fourier transform maps the N numbers (f_k's) into the transform domain (the F_n's). Notice it is independent of Δ. Rewriting the relation between the Fourier transform of the function and the individual point's transforms

$$F(u_n) \approx \Delta F_n \tag{4.30}$$

The index n runs from $-\frac{N}{2}$ to $\frac{N}{2}$. Because F_n is periodic, with period N, this means F_{-n} is equivalent to F_{N-n}, with $n = 1, 2, \ldots$ This leads to n in N varying from 0 to $N-1$. This shifting of indices comes at a price, it is up to the reader to remember that 0 frequency corresponds to $n = 0$, and the positive frequencies $0 < u < f_c$ correspond to $1 < n < N/2 - 1$, and that the negative frequencies $-f_c < u < 0$ correspond to $N/2 + 1 < n < N - 1$. $n = N/2$ corresponds to both $u = f_c$ and $u = -f_c$.

Both the symmetry properties in table 4.2 are true for the discrete form of the Fourier transform. The formula for the inverse Fourier transform which recovers the f_k's from the F_n's is

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n \, e^{-i 2 \pi k n / N} \tag{4.31}$$

This formula is very similar to the forward transform, the only difference being dividing the summation by N and the negative sign in the exponential. This means an algorithm to calculate Fourier transforms can easily be modified to calculate inverse transforms.

4.2.4 Discrete Fourier Transform Theorems

These theorems are stated without proofs, mainly as they are analogous to the continuous proofs (by replacing integrals with summations).

Convolution

A *response function* is a peaked function which tapers from its spike to 0 in both directions. Then the following is a definition of discrete convolution with a response function f of finite duration M [15].

$$(f * g)_j \equiv \sum_{k=-M/2+1}^{M/2} g_{j-k} \, f_k \tag{4.32}$$

If the signal f_j is periodic with period N, it is completely determined by the N values $f_0, \ldots, f_{N-1}$, and the discrete convolution of two real functions f and g transforms according to

$$f * g \Longleftrightarrow FG \tag{4.33}$$

Correlation

Assuming functions f and g are periodic with period N then the discrete correlation of the two functions is

$$(f \otimes g)_j \equiv \sum_{k=0}^{N-1} f_{j+k}\, g_k \tag{4.34}$$

The *discrete correlation theorem* shows this discrete correlation of two real functions f and g transforms according to

$$f \otimes g \Longleftrightarrow F\overline{G} \tag{4.35}$$

[15].

Parseval's Theorem

$$\sum_{k=0}^{N-1} |f_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |F_n|^2 \tag{4.36}$$

[15].

4.2.5 Power Spectrum Definition

The power of the discrete Fourier transform will be crucial to roughness calculations. The N real input points will be transformed into N complex valued output points. Then if $\{F_j\} = \mathrm{DFT}[\{f_k\}]$ represents the Fourier transform of the sample points, the discrete Power Spectrum Density is defined as

$$P_j \equiv \frac{1}{N}\sqrt{\mathfrak{Re}(F_j)^2 + \mathfrak{Im}(F_j)^2} \tag{4.37}$$

4.3 Fast Fourier Transform

The discrete Fourier transform described thus far, whilst workable, is not ideal. To calculate the complexity of the process, consider the Fourier Transform

$$F_n = \sum_{k=0}^{N-1} f_k \, e^{i\,2\pi\,k\,n/N} \tag{4.38}$$

which can be rewritten as

$$F_n = \sum_{k=0}^{N-1} W^{nk} \, f_k \tag{4.39}$$

where

$$W = e^{i\,2\pi/N} \tag{4.40}$$

This means that calculating the complete set of discrete Fourier transformed points amounts to multiplying the set of sample point f_k's (a vector) by a matrix whose $(n,k)^{th}$ element is the constant W to the power nk. Multiplying a vector of length N by an $N \times N$ matrix is an operation of order N^2. Amazingly, the discrete Fourier transform of a set of points can be calculated in a process of order

$N \log_2 N$. Although there are several similar methods, the first *Fast Fourier Transform* algorithm to become well-known was by J.W. Cooley and J.W. Tukey in 1965.

One of the "discoveries" of a Fast Fourier Transform (FFT) was by Danielson and Lanczos in 1942 [4], and their algorithm is given here. Their first breakthrough was that a discrete Fourier transform of length N can be rewritten as the sum of two discrete Fourier transforms, each of length $N/2$. The two sub-transforms are generated from the even and odd-indexed points of the original N-length vector as follows

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} W^{jk} f_j \\ &= \sum_{j=0}^{N/2-1} W^{2jk} f_{2j} + \sum_{j=0}^{N/2-1} W^{(2j+1)k} f_{2j+1} \\ &= \sum_{j=0}^{N/2-1} W^{2jk} f_{2j} + W^k \sum_{j=0}^{N/2-1} W^{2jk} f_{2j+1} \\ &= F_k^e + W^k F_k^o \end{aligned} \tag{4.41}$$

where W^k is as defined before. F_k^e denotes the kth component of the Fourier transform of length $N/2$ formed from the *e*ven components of the original f_j's, while F_k^o corresponds to the *o*dd components. The k in $F_k = F_k^e + W^k F_k^o$ varies from 0 to $N-1$, however the transforms F_k^e and F_k^o are periodic in k, and have length $N/2$.

This algorithm can be used recursively. Now that $F_k = F_k^e + W^k F_k^o$, the same algorithm can be used to find the values of F_k^e, and F_k^o, as follows

$$\begin{aligned} F_k^e &= F_k^{ee} + W^k F_k^{eo} \\ F_k^o &= F_k^{oe} + W^k F_k^{oo} \end{aligned} \tag{4.42}$$

so now F_k is

$$F_k = F_k^{ee} + W^k F_k^{eo} + W^k \left[F_k^{oe} + W^k F_k^{oo}\right] \tag{4.43}$$

Notice the $F_k^{ee}, F_k^{eo}, F_k^{oe}, F_k^{oo}$ transforms will only have $N/4$ data points. Using this method, it is obvious that choosing $N = 2^q$ where $q \in \mathbb{Z}$ will be optimal. In fact it is easier to zero-pad a vector of points up to the nearest power of two than to try and calculate the Fourier transform otherwise. Fairly quickly this method of sub-dividing will reach a stage where the number of e's and o's in the $F_k^{eoeoeoo\ldots oeo}$ superscript is $\log_2 N$, and the vector will have been divided into sub-transforms of a single point. Now all that needs to be done is a copying of an input point f_v (for some v) into the output point. The question is, which point is needed? Reverse the superscripts e's and o's, and then let $e = 0$ and $o = 1$. The binary value these *bit reversed* superscripts yield is that specific v. This is true because the successive subdivisions of the vector points into even and odd sets are tests of successive least-significant bits of v.

The algorithm described is illustrated in the "butterfly" figure (4.1), which shows the process for an 8-point signal. The first three columns show the bit-reversal technique for the original points, the remainder of the diagram shows the algorithm for calculating the Fast Fourier Transformed points (the right hand side of the diagram).

This bit-reversal technique can be used to simplify the FFT algorithm: rearrange all the inputs points into bit-reversed order before beginning to calculate the FFT; so the points are now in the order given by bit-reversing their vector (array) indices. Now the Danielson-Lanczos method of dividing up the data into even and odd transforms becomes almost trivial: the data vector is now a set of one point transforms, and adjacent pairs are combined to form transform pairs, then combine the pairs to get 4-point transforms,

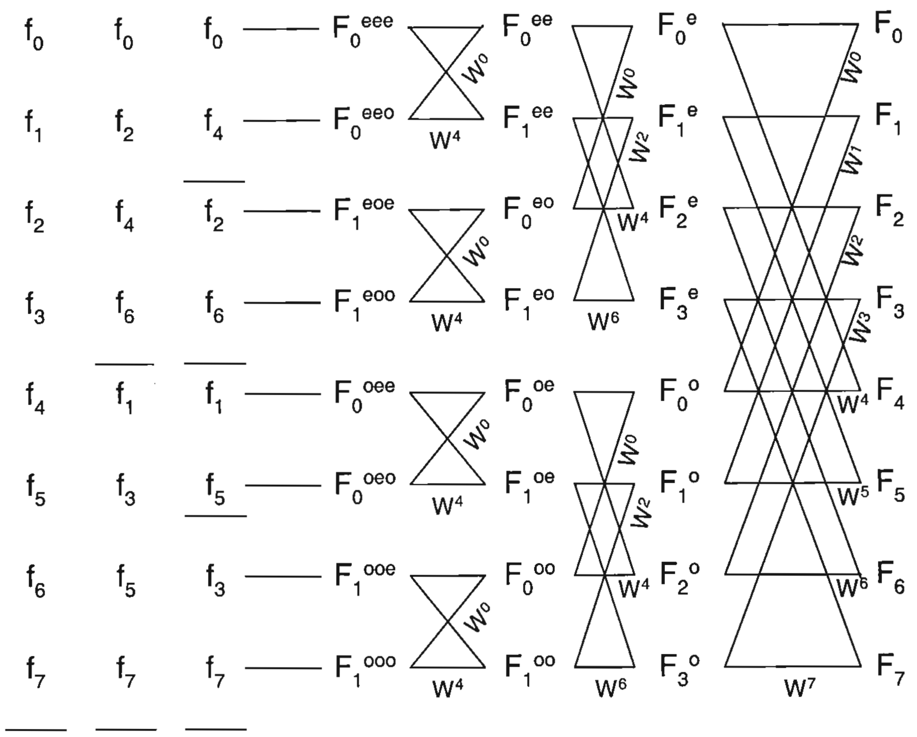


Figure 4.1: 8-point signal FFT calculation

until eventually the first and second halves (the original even and odd halves) are combined to give the final transform. Each column takes N operations, and there are $\log_2 N$ columns, so the entire set of transform points can be calculated in $N \log_2 N$ as claimed.

A Java implementation of the FFT is presented in Appendix C.

Chapter 5

Fourier Roughness Measure

5.1 Fourier Transform Roughness Measure

Here the idea is to use the Fourier spectrum of the signal as input to a roughness algorithm. A standard Discrete Fourier Transform (DFT) algorithm (from [15]) is used to compute 2^n Fourier power coefficients. These are arranged so that the low frequency coefficients appear at the extremities of the transformed signal while the high frequency coefficients appear at the centre. Now since the input signal is real, the Fourier coefficients will be symmetric about the centre and only half of them are required for the roughness calculations. To compute roughness each power coefficient is weighted according to the frequency it represents, i.e. assume that large high frequency coefficients occur in rough signals while large low frequency coefficients occur in smooth signals. The algorithm is then as follows:

5.1.1 Algorithm

```
$\{S_j\} = \mathrm{FFT}[\{s_k\}]$
roughness $\doteq 0$
**for** $j = 1$ to $2^{k-1}$ **do**
    roughness $+= \sqrt{\mathfrak{Re}(S_j)^2 + \mathfrak{Im}(S_j)^2} * (j-1)$
```

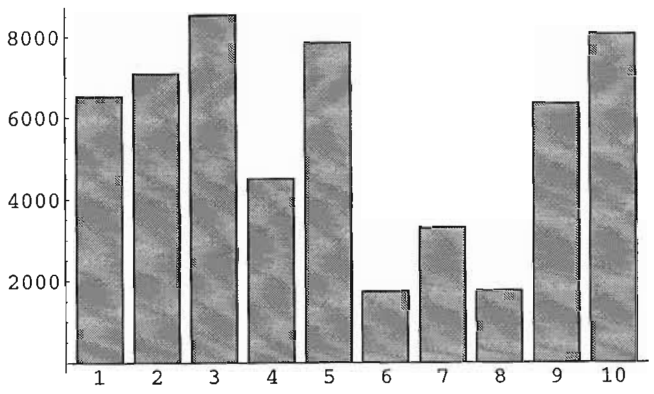


Figure 5.1: Roughness ranking of Barton's profiles by DFT Measure

```
end for
return roughness
```

Note that the coefficient S_0 represents the average value of the signal, therefore it has a weight of zero. Again this algorithm will produce very large output and a scaling factor must be implemented to ensure that the output lies in the range $[0,\ 2^{n-1}-1]$.

5.1.2 Results

1D Results

The DFT measure did not enjoy a high degree of success with the 1D geology profiles. The results were somewhat mixed, with no trend being obvious across the samples (see figure 5.1). This can be attributed to the DFT measure being dependent on continuous basis functions, which are more susceptible to sampling artifacts and non-smooth input data.

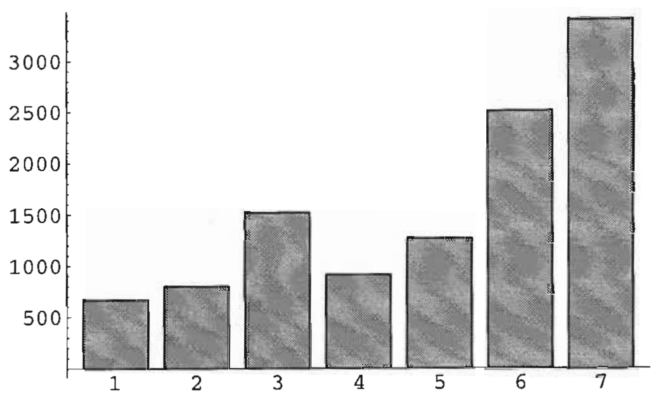


Figure 5.2: Roughness ranking of gill flap samples by DFT Measure

$2D$ Results

By comparison the DFT measure is excellent at ranking the $2D$ biological samples, ordering the images in the correct order, with the notable exception of the third sample, which is disproportionately rough compared with its neighbours because of the errant thumbprint, which serves to prove the sensitivity of the DFT measure to any form of sampling artifact.

Chapter 6

The Wavelet Transform

This chapter closely resembles the expositions in [5], [6], [8], [12], [17], [19] and [20].

The Fourier transform gives an entirely frequency based representation of a signal. The wavelet transform differs from this by giving a representation that is partly spatially based and partly frequency based. A wavelet transform gives the "localized" frequency components of a signal. Wavelet transforms have been successfully employed in problems such as speech recognition [7]. Here they will be employed to measure the local roughness of a signal.

Given a spatial signal of length 2^n the Discrete Wavelet Transform (DWT) algorithm (from [15]) generates 2^n real wavelet coefficients that can be arranged in the space/frequency domain as shown in figure 6.1.

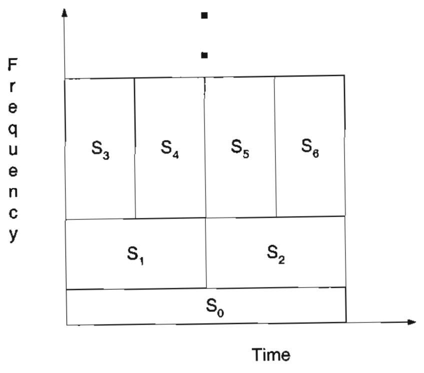


Figure 6.1: The time/frequency relationship.

6.1 Introduction

A single *mother wavelet* $\psi(x)$ is used to generate a wavelet set. The various mother wavelets can have different properties, but all mother wavelets must oscillate and decay, this is expressed by specifying that the mother wavelets integrate to 0

$$\int_{-\infty}^{\infty} \psi(x)\, dx = 0 \tag{6.1}$$

Some wavelets decay to zero outside a finite interval, this compact support leads to good representation of signals which themselves have compact support or sharp spikes.

6.2 Continuous Wavelet Transform

Sets of wavelets are comprised of dilations and translations of their mother wavelet. For the continuous wavelet transform (CWT) any amount of dilation or translation can be applied. The wavelet with dilation s and translation τ is defined as

$$\psi_{s,\tau}(x) = \frac{1}{\sqrt{s}}\psi(sx - \tau) \qquad \text{for } s, \tau \in \mathbb{R} \tag{6.2}$$

which is the mother wavelet dilated by a factor of s and translated to τ/s. For practical purposes, a restricted set of wavelets is generally used, with $s = 2^j$ and $\tau = k$ where $j, k \in \mathbb{Z}$.

6.3 Orthonormality

For carefully chosen mother wavelets the collection $\{\psi_{j,k}(x)\}\ \forall j, k \in \mathbb{Z}$ comprises an orthonormal basis for various function spaces. Orthonormality can be expressed mathematically as

$$\int \psi_{j,k}(x)\,\psi_{m,n}(x)\,dx = \begin{cases} 1 & \text{if } j = m \text{ and } k = n \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

and means the wavelet set can be used to represent functions. A function $f(x)$ can be written as

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} p_{j,k}\,\psi_{j,k}(x) \tag{6.4}$$

The orthonormality property means that the coefficients $p_{j,k}$ can be found using

$$p_{j,k} = \int_{-\infty}^{\infty} f(x)\,\psi_{j,k}\,dx \tag{6.5}$$

The coefficients $p_{j,k}$ required to weight the wavelet basis functions $\psi_{j,k}(x)$ to generate $f(x)$ are referred to as the *Wavelet transform of* *the function $f(x)$.*

6.4 Haar Wavelet

The simplest wavelet is the Haar wavelet. The Haar mother wavelet is defined as

$$\psi(x) = \begin{cases} 1 & \text{for } x \in [0, 1/2) \\ -1 & \text{for } x \in [1/2, 1) \\ 0 & \text{elsewhere} \end{cases} \tag{6.6}$$

Consider the interval $[0, 1)$. At scale $j = 0$, there is only the mother wavelet function ψ on the interval; when $j = 1$, there are two wavelets: $\psi_{1,0}$ and $\psi_{1,1}$. If $j = 2$, there are four wavelets: $\psi_{2,0}$, $\psi_{2,1}$, $\psi_{2,2}$ and $\psi_{2,3}$; at $j = 3$ there are eight wavelets ....

For the case of functions that exist outside the interval $[0, 1)$? join unit intervals together and use translations of the ψ's.

6.4.1 Haar Smoothing Function

At this point there is a stumbling block. Returning to the interval $[0, 1)$, consider any constant function $y = a$. The collection of Haar wavelets is unable to represent it. The solution to the problem is to introduce a *smoothing* function $\phi(x)$. For the Haar transform the smoothing function is defined to be a constant

$$\phi(x) = \begin{cases} 1 & \text{for } x \in [0, 1) \\ 0 & \text{elsewhere} \end{cases} \tag{6.7}$$

Including translations of the smoothing function in the set of basis elements the representation for a function $f(x)$ becomes

$$f(x) = \sum_{k=-\infty}^{\infty} q_k \, \phi_{0,k}(x) + \sum_{j=1}^{\infty} \sum_{k=-\infty}^{\infty} p_{j,k} \, \psi_{j,k}(x) \tag{6.8}$$

which makes use of both the smoothing function and the wavelet function (often referred to as the *scaling* and *detail* functions). Integer translations k of the scaling function ϕ provide an averaged description of the function, whilst the wavelet functions ψ provide detailed features at different scales j. Breaking down functions into specific packets of information as described is the basis for *multi-resolution analysis.*

6.5 The Dilation Equation

Note that the Haar smoothing function $\phi(x)$ is a solution to the dilation equation

$$\begin{aligned} \phi(x) &= c_0 \, \phi(2x) + c_1 \, \phi(2x-1) \\ &= \sum_{k=0}^{1} c_k \, \phi(2x-k) \end{aligned} \tag{6.9}$$

that satisfies

$$c_0^2 + c_1^2 = 2 \qquad \text{and} \qquad c_0 = c_1 \tag{6.10}$$

and that the Haar wavelet function $\psi(x)$ can be generated from the Haar smoothing function according to

$$\begin{aligned}\psi(x) &= c_1\,\phi(2x-1) - c_0\,\phi(2x) \\ &= \sum_{k=0}^{1}(-1)^k\,c_{1-k}\,\phi(2x-k)\end{aligned} \tag{6.11}$$

The conditions (6.10) allow the Haar basis to be generated from solutions to the dilation equation. Daubechies noted that this concept can be generalized to produce smoother orthogonal wavelet bases (in [5]). The coefficients c_k, $k = 0,\ 1,\ 2,\ \ldots$ will, from now on, be referred to as the "basis elements".

6.5.1 Approximation

The basic properties of wavelet approximation are given here, for a more detailed description see [19]. An integer p must characterize the functions ϕ and ψ in the following manner:

1. The polynomials $1, x, \ldots, x^{p-1}$ are combinations of ϕ and its translates.
2. Smooth functions $f(x)$ can be approximated with error $O(h^p)$ by combinations of ϕ at each scale $h = 2^{-j}$.
3. The first p moments of the wavelet ψ are 0

$$\int x^m\,\psi(x)\,dx = 0 \qquad \text{for } m = 0, 1, \ldots p-1 \tag{6.12}$$

6.5.2 Haar Basis Conditions

Strang shows (in [19]) that to satisfy the approximation conditions the two Haar basis elements must satisfy

$$\sum c_{2k} = \sum c_{2k-1} \tag{6.13}$$

and he also shows that orthonormality requires that the basis elements satisfy

$$\sum c_k^2 = 2 \tag{6.14}$$

6.6 Daubechies Wavelet Construction

One way to generate both the Haar and Daubechies' wavelet sets is to use recursion.

Iterate the recursive relation

$$\phi_n(x) = \sum_{k=0}^{m} c_k \, \phi_{n-1}(2x - k) \tag{6.15}$$

starting with the box function as $\phi_0(x)$. Note $m = 1$ for Haar and $m = 3$ for Daubechies. Using basis elements of $[1, 1]$, $[\frac{1}{2}, 1, \frac{1}{2}]$, $[\frac{1}{4}, \frac{3}{4}, \frac{3}{4}, \frac{1}{4}]$ gives first the box function, then the hat, then a B-spline.

However, apart from the Haar basis elements $[1,\ 1]$, these sets do not satisfy orthogonality conditions.

Daubechies wanted smoother orthogonal wavelets with better approximation properties than the Haar wavelet. To this end she derived (in [5]) an additional approximation condition

$$-c_1 + 2c_2 - 3c_3 = 0 \tag{6.16}$$

and an additional orthonormality condition

$$c_2 c_0 + c_3 c_1 = 0 \tag{6.17}$$

which together with equations (6.13) and (6.14) gave her 4 basis elements $[\frac{1+\sqrt{3}}{4}, \frac{3+\sqrt{3}}{4}, \frac{3-\sqrt{3}}{4}, \frac{1-\sqrt{3}}{4}]$. Both (6.17) and (6.16) can be generalized, as Strang has shown in [19].

Iterations of the recursive relation (6.15) with these 4 c_k's give Daubechies' 4 coefficient wavelet, which is denoted $D4$. Daubechies' additional conditions for the coefficients c_k were constructed carefully, so that the wavelet bases they generate range from highly localized basis sets like $D4$ (few coefficients), to highly smooth basis sets (there is no limit to the number of coefficients that can be constructed with Daubechies' conditions) which have many more elements. It is the wavelet user's responsibility to choose an appropriate basis for the problem at hand.

6.7 Wavelet Algorithm

Much of this section follows that in [17].

Assume the function $f(x)$ has been sampled at evenly spaced points in space. Denote this sampled sequence by $\mathbf{x} = (x_i)$.

The wavelet decomposition is derived from sub-band filtering using two filters: (h_k) which is the smoothing or scaling (*low-pass*) filter, and the detail or wavelet (*high-pass*) filter (g_k). These filters have the following property

$$g_j = (-1)^j \, h_{m-j} \tag{6.18}$$

The h_k values equal the c_k basis elements for $0 < k < m$, and are 0 elsewhere.

The wavelet filter (g_k) is often referred to as the *mirror* filter, and it is given by taking the elements of the scaling filter (h_k) in backwards order, with every other element negated.

The filtering of $\mathbf{x}$ is given by

$$\begin{aligned} y_i &= \sum_k h_k \, x_{2i-k} \\ z_i &= \sum_k g_k \, x_{2i-k} \end{aligned} \tag{6.19}$$

i.e. $\mathbf{y} = H\mathbf{x}$ and $\mathbf{z} = G\mathbf{x}$ where in Daubechies case the matrix H is

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 & & \dots & & & & \\ & & h_0 & h_1 & h_2 & h_3 & & & & \\ \vdots & & & & & & \ddots & & & \\ & & & & & & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & & & & & & & h_0 & h_1 \end{pmatrix} \tag{6.20}$$

and the corresponding G is

$$\begin{pmatrix} g_0 & g_1 & g_2 & g_3 & & \dots & & & & \\ & & g_0 & g_1 & g_2 & g_3 & & & & \\ \vdots & & & & & & \ddots & & & \\ & & & & & & g_0 & g_1 & g_2 & g_3 \\ g_2 & g_3 & & & & & & & g_0 & g_1 \end{pmatrix} \tag{6.21}$$

Notice the matrix multiplication down-samples and produces sequences half their original length. Reconstruction involves reversing the filtering function, using transposed matrices, i.e. $\mathbf{x} = H^T\mathbf{y}+G^T\mathbf{z}$

$$x_i = \sum_k \left(h_{i-2k} \, y_k + g_{i-2k} \, z_k\right) \tag{6.22}$$

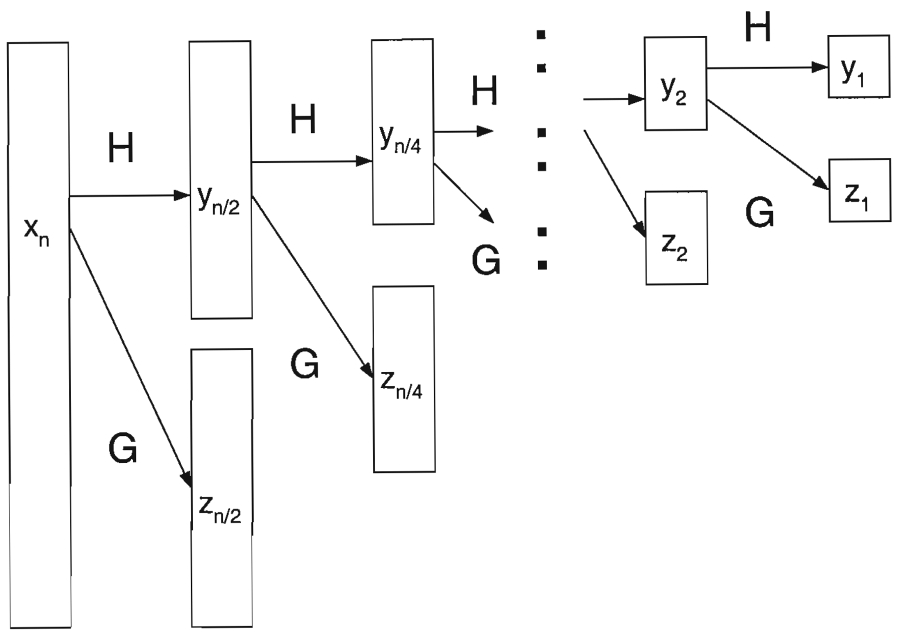


Figure 6.2: Wavelet decomposition of a signal **x**

6.7.1 Decomposition

Assume **x** has length 2^n. A pyramid algorithm is used to find the decomposition. The process takes n steps, with each step halving the size of the sequence, as is shown in figure (6.2). The wavelet coefficients of the data are made up of the detail sequences $z_{n/2}$, $z_{n/4}$, ..., z_2, z_1, as well as the final smoothing sequence y_1. It is customary to stop the pyramid algorithm when z_k and y_k are reached (where k is the number of coefficients g and h), as any further calculations are unnecessary; which means only $(n - \log_2 k)$ pyramid steps are required.

Reconstruction involves the pyramid process being reversed, with sequences z_1 and y_1 used at the beginning, and with transposed matrices H^T and G^T used for all the matrix multiplications until the original sequence **x** is reconstructed.

A Java implementation of the discrete wavelet transform (DWT) is presented in Appendix D.

Chapter 7

Wavelet Roughness Measures

7.1 Haar and Daubechies' 4 Coefficient Measures

These measures are similar to the DFT measure where low frequency coefficients are associated with smooth functions whilst the high frequency coefficients are associated with rough functions. Thus in the wavelet-based calculations, the frequency level of the wavelet coefficient is used as a weight. The squaring of the weights ensures that the total roughness stays within the $[0,\ 2^{n-1}-1]$ range for any signal.

Wavelets come in all manner of shapes and sizes. Two wavelet based roughness measures have been implemented. The fast wavelet code from *Numerical Recipes* [15] was used to compute two wavelet decompositions. One based the roughness measure on the simplest of all wavelets, the Haar wavelet, and the other was based on the famous Daubechies' $D4$ wavelet. The roughness algorithm is as follows:

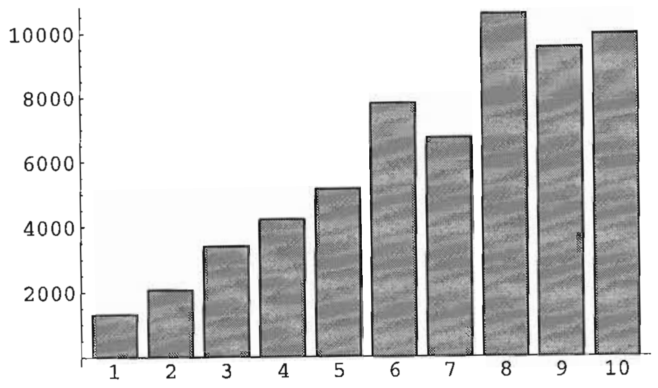


Figure 7.1: Roughness ranking of Barton profiles by Haar Transform Measure

7.1.1 Algorithm

```
$\{S_j\} = \text{FWT}[\{s_k\}]$
roughness $= 0$
**for** $j = 1$ to $2^n$ **do**
  roughness $+ = S_j * \frac{1}{(n - \lfloor \log_2 j \rfloor)^2}$
**end for**
return roughness
```

7.1.2 Results

The results for the two measures will be split into Haar and Daubechies' 4 coefficient sections.

Haar $1D$ Results

The $1D$ samples, the downfall of the DWT measure amongst others, held few problems for the Haar $1D$ transform measure. The profiles were ranked almost perfectly, with only the sixth and eight profiles being ranked out of sequence.

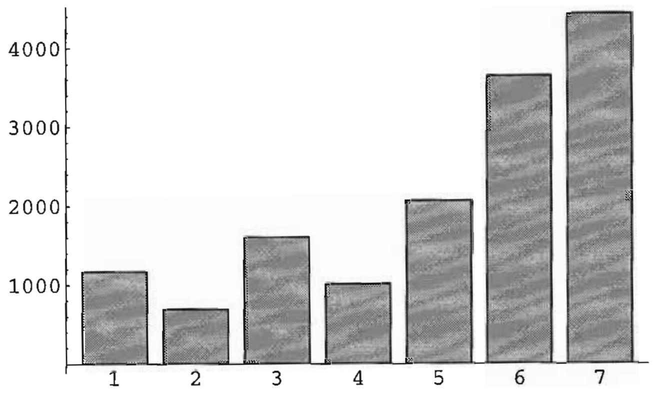


Figure 7.2: Roughness ranking of gill flap samples by Haar Transform Measure

Haar 2*D* Results

The results for the biology samples where somewhat relieving! The first sample was ranked as being slightly more rough than the second (a fault common through most of the measures), but the third sample, the corrupt sample with the thumb-print in the lower-left corner, did not produce as inaccurate a result as it did with, for instance, the DFT (where the third sample was ranked rougher than both the fourth and fifth samples). The remainder of the biology samples were ranked in the correct order.

With only the Daubechies' 4 coefficient measure left to be considered, the Haar transform is the strongest all-round performer across the samples and profiles, and seems less prone to sampling artifact problems.

Daubechies' 4 Coefficient 1*D* Results

The Daubechies' 4 coefficient measure eclipses (marginally) the Haar transform measure when ranking the geology profiles, only having one profile, the fifth profile, out of sequence. Otherwise all the

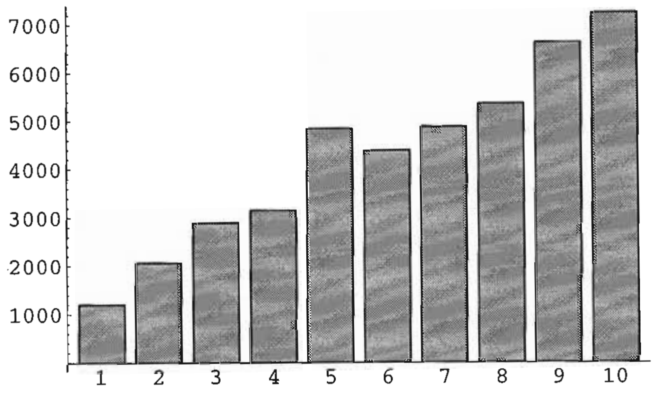


Figure 7.3: Roughness ranking of Barton's profiles by Daubechies Transform Measure

profiles are in the correct order, and their graph (refer to figure 7.3) is almost a straight-line graph.

Daubechies' 4 Coefficient $2D$ Results

The Daubechies' 4 coefficient measure is a slightly better performer than the Haar transform with the biology $2D$ samples, it also ranks the second sample as less rough than the first, but the difference between the two samples is slightly less, also the third sample (with the sampling artifacts) is even closer in roughness to the rest of the sequence than with the Haar transform. The rest of the samples are ranked correctly.

The two wavelet transform measures, being both spatial and frequency measures, are more sophisticated measurement techniques. Whilst neither can claim to have the best results for either the $1D$ or $2D$ results, they both have excellent results when considered across both the profile and sample sets. Included in this is their lesser susceptibility to sampling artifacts. Future work, including true $2D$ measurement techniques, should be based on wavelet transforms.

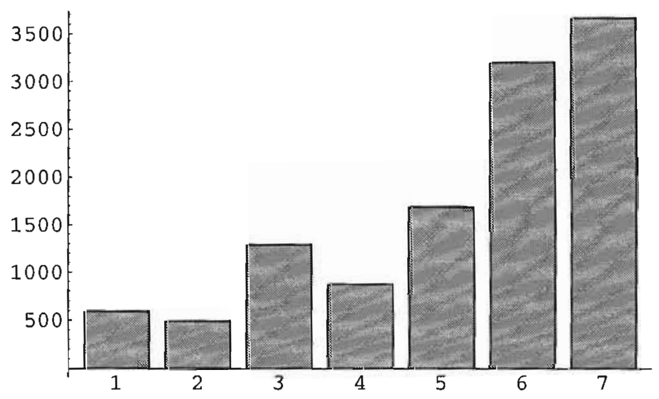


Figure 7.4: Roughness ranking of gill flap samples by Daubechies Transform Measure

Chapter 8

Where And How

The algorithms described in this thesis were written as plug-ins for the image processing shell ImageJ which is developed and maintained by Wayne Rasband of the Research Services Branch of the National Institute of Mental Health located in Bethesda, Maryland. ImageJ is available for free download from *http://rsb.info.nih.gov/ij/* . Also downloadable from the same site and useful for anybody considering developing further plugins for ImageJ is a (fairly sparse) helpfile which documents all the ImageJ classes and their relative positions in the ImageJ class hierarchy. ImageJ ships with detailed setup instructions, which should lead to a successful installation.

After installation of ImageJ, the interested reader can try out the roughness plugins for ImageJ by copying the Java source and class files for the plugins, as well as the sample signals, from the disk included with this thesis, or by downloading them from *http://gannet.cs.und.ac.za/* . Make sure the paths to the Java and ImageJ root folders are included in the environment variable CLASSPATH before running ImageJ.

Chapter 9

Conclusion

The abilities of the measures to order correctly the $1D$ profiles and $2D$ images were mixed. No one measure correctly ranks both sets of samples. All five measures are reasonably successful when it comes to measuring the $2D$ biological sample images: all ranked the third sample as being far more rough than it should be, with the DFT and Direction Change measures being particularly susceptible to the sampling artifacts. When looking at the results for the $1D$ profiles, the Direction Change and FFT measures are both fairly poor performers, with the almost trivial Height Increment Measure ranking the samples correctly, and the two Wavelet transforms also being close to correct.

The FFT will always be over-sensitive to sampling artifacts as its basis functions are continuous. The Height Increment Measure (referred to a by a geologist as the "drag your finger along the surface" technique) had an unfair advantage as Barton's roughness profiles were probably designed with this measure in mind, thus it would appear the wavelet measures are the most accurate measure.

Future work in this field should be centred around developing a series of true $2D$ wavelet measures for $2D$ images, and comparing the resulting roughness coefficients with those generated by $2D$ fractal dimension measures.

Appendix A

Sample Images

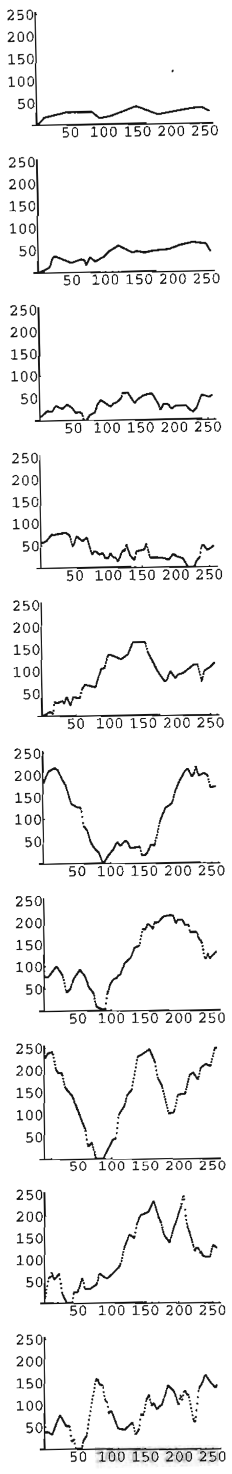


Figure A.1: Barton's Roughness Measure profiles

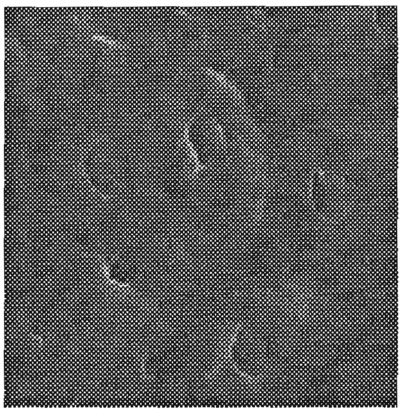


Figure A.2: Gill flap first sample

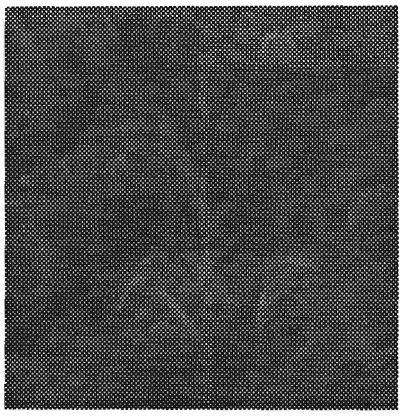


Figure A.3: Gill flap second sample

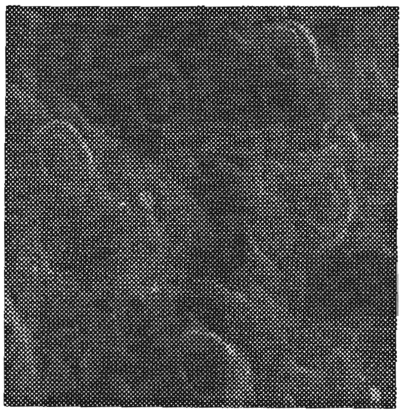


Figure A.4: Gill flap third sample

Figure A.5: Gill flap fourth sample

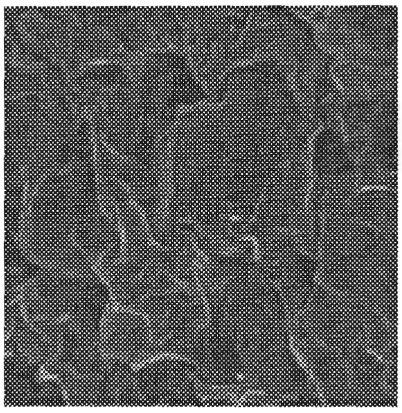


Figure A.6: Gill flap fifth sample

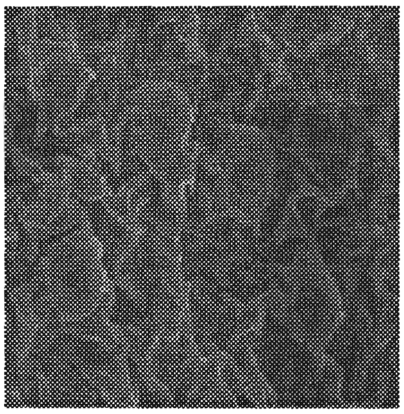


Figure A.7: Gill flap sixth sample

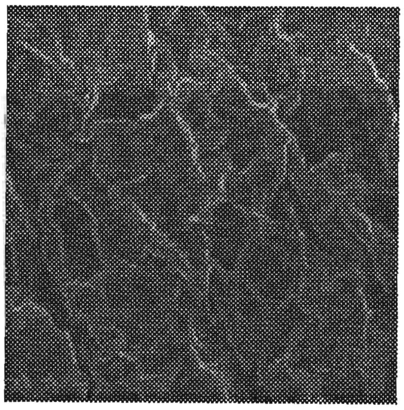


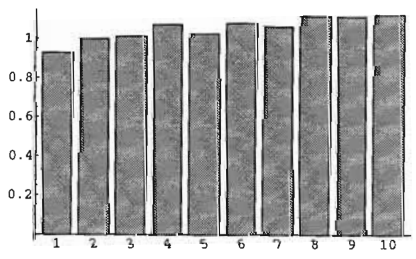
Figure A.8: Gill flap seventh sample

Appendix B

Sample Image Results

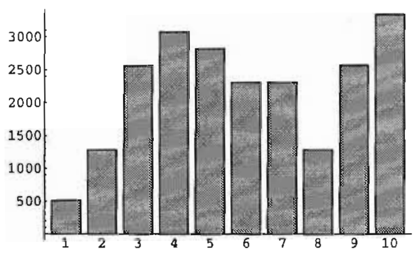
Examination of these results could make for unexpected reading, the resultant images would suggest different pixel areas to those reflected here. The reason is simple, these results have been "normalized" to aid comparison between the various measures, such as comparisons can be made.

B.1 Barton's Samples' Results



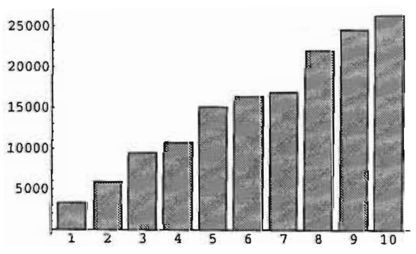
Sample	*Roughness*
1	0.926
2	0.997
3	1.010
4	1.069
5	1.022
6	1.076
7	1.058
8	1.111
9	1.108
10	1.118

Figure B.1: Roughness ranking of Barton's profiles by Box Counting Measure



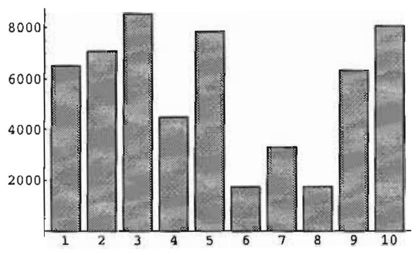
Sample	*Roughness*
1	6.169
2	15.422
3	30.843
4	37.012
5	33.928
6	27.759
7	27.759
8	15.422
9	30.843
10	40.096

Figure B.2: Roughness ranking of Barton's profiles by Direction Change Measure



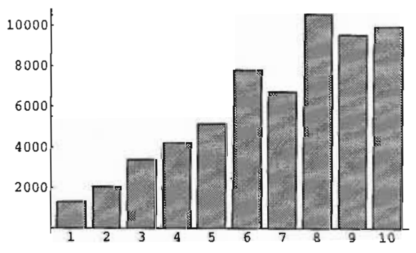
Sample	*Roughness*
1	13.365
2	23.647
3	38.04
4	43.181
5	60.659
6	65.799
7	67.855
8	88.418
9	98.699
10	105.896

Figure B.3: Roughness ranking of Barton's profiles by Height Increase Measure



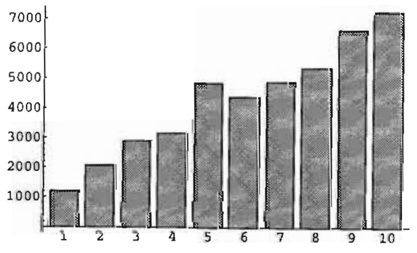
Sample	*Roughness*
1	78.492
2	85.391
3	102.716
4	54.044
5	94.353
6	20.946
7	39.56
8	21.075
9	76.055
10	96.845

Figure B.4: Roughness ranking of Barton's profiles by DFT Measure



Sample	*Roughness*
1	15.675
2	24.839
3	40.925
4	50.795
5	62.022
6	93.776
7	80.807
8	127.025
9	114.517
10	119.491

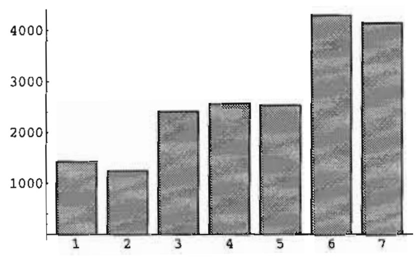
Figure B.5: Roughness ranking of Barton's profiles of Haar Transform Measure



Sample	*Roughness*
1	14.448
2	24.887
3	34.787
4	37.932
5	58.185
6	52.725
7	58.644
8	64.316
9	79.588
10	86.933

Figure B.6: Roughness ranking of Barton's profiles by Daubechies Transform Measure

B.2 Gill Flap Sample Results



Sample	*Roughness*
1	1430.992
2	1247.797
3	2412.374
4	2563.412
5	2530.879
6	4292.611
7	4142.912

Figure B.7: Roughness ranking of gill flap samples by Box Counting Measure

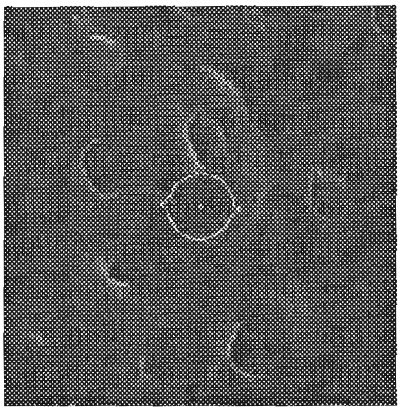


Figure B.8: Box Counting Measure: First sample

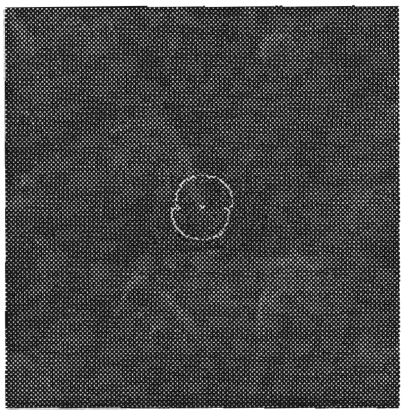


Figure B.9: Box Counting Measure: Second sample

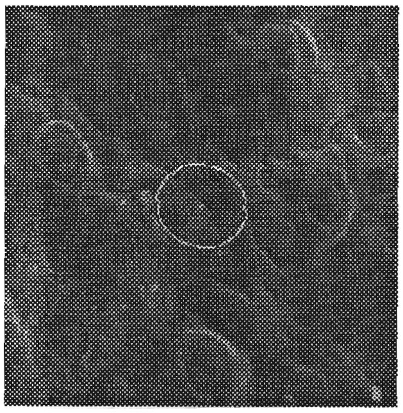


Figure B.10: Box Counting Measure: Third sample

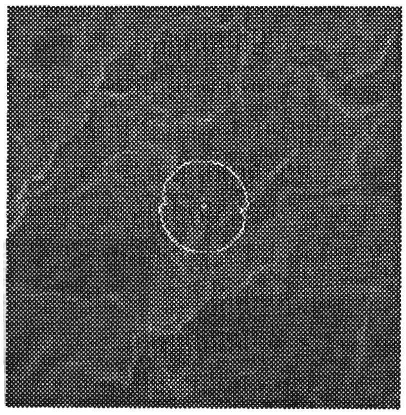


Figure B.11: Box Counting Measure: Fourth sample

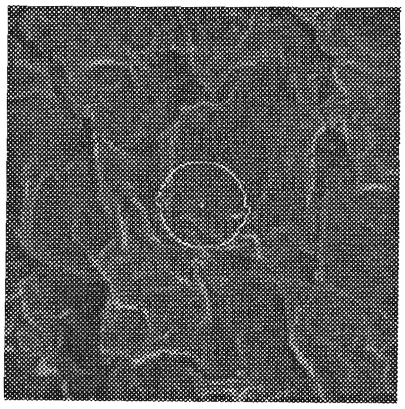


Figure B.12: Box Counting Measure: Fifth sample

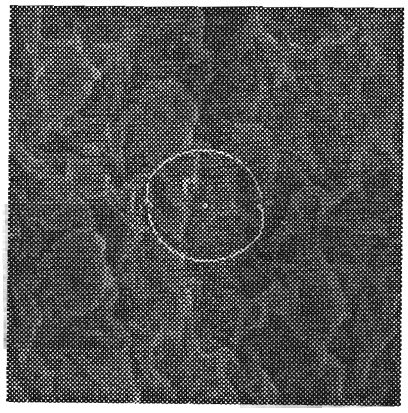


Figure B.13: Box Counting Measure: Sixth sample

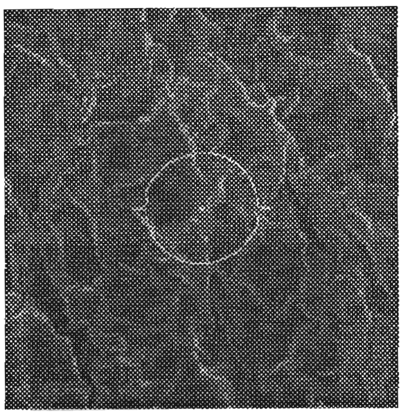
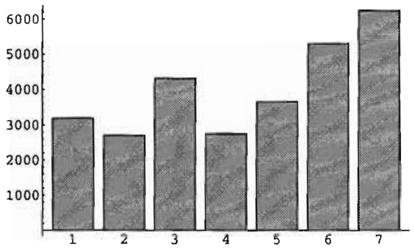


Figure B.14: Box Counting Measure: Seventh sample



Sample	*Roughness*
1	57.067
2	48.381
3	76.955
4	49.073
5	65.172
6	94.437
7	111.19

Figure B.15: Roughness ranking of gill flap samples by Direction Change Measure

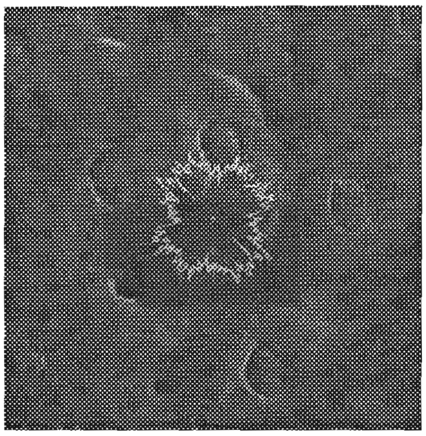


Figure B.16: Direction Change Measure: First sample

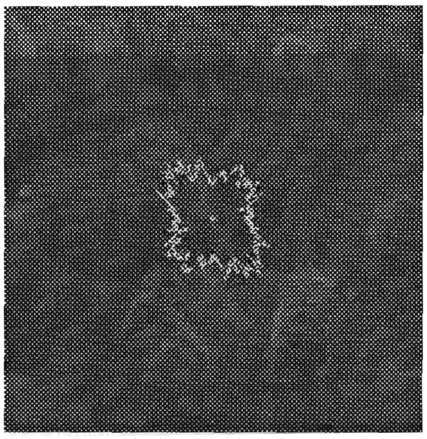


Figure B.17: Direction Change Measure: Second sample

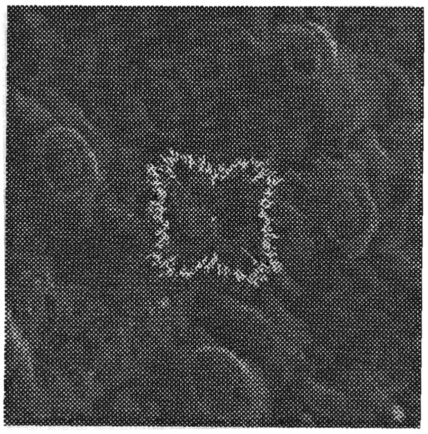


Figure B.18: Direction Change Measure: Third sample

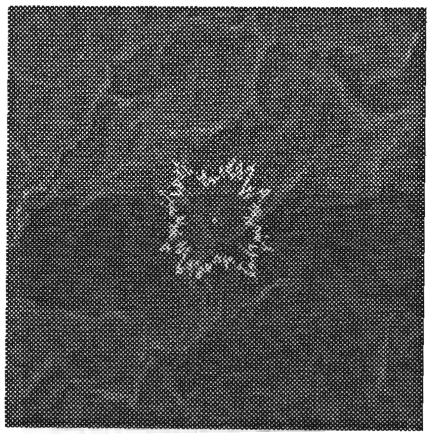


Figure B.19: Direction Change Measure: Fourth sample

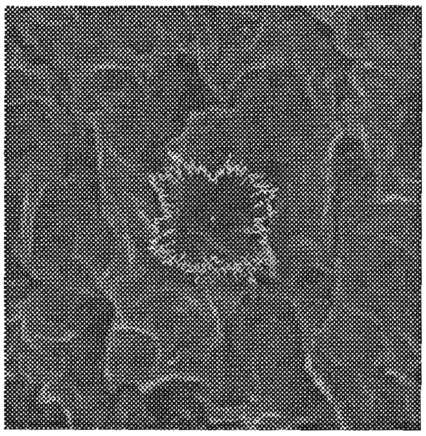


Figure B.20: Direction Change Measure: Fifth sample

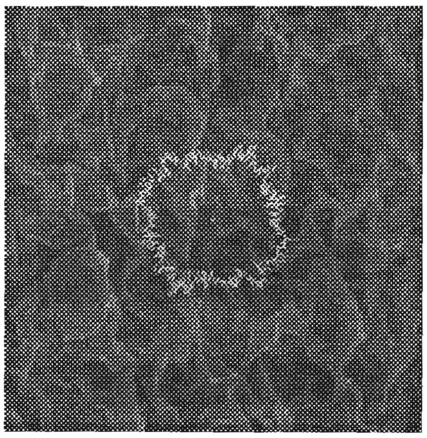


Figure B.21: Direction Change Measure: Sixth sample

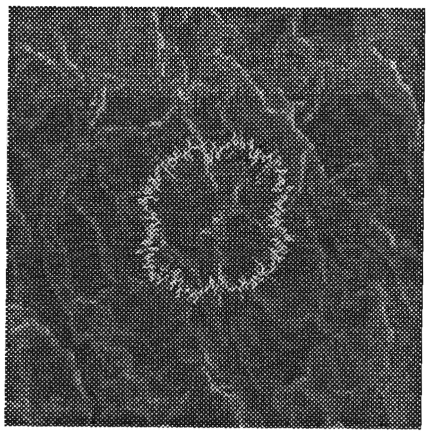
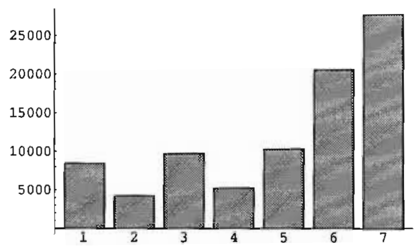


Figure B.22: Direction Change Measure : Seventh sample



Sample	*Roughness*
1	37.525
2	18.656
3	43.166
4	23.198
5	45.881
6	91.582
7	123.833

Figure B.23: Roughness ranking of gill flap samples by Height Increase Measure

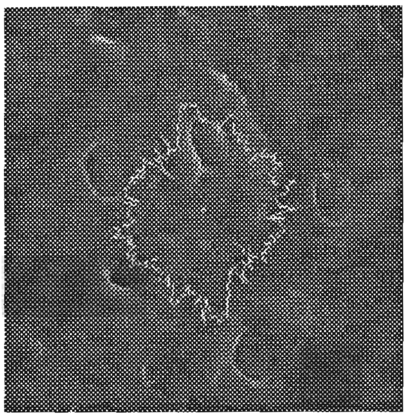


Figure B.24: Height Increase Measure: First sample

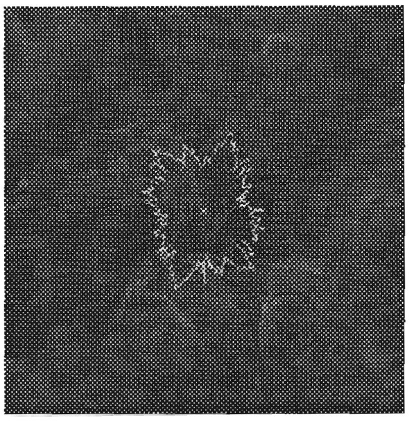


Figure B.25: Height Increase Measure: Second sample

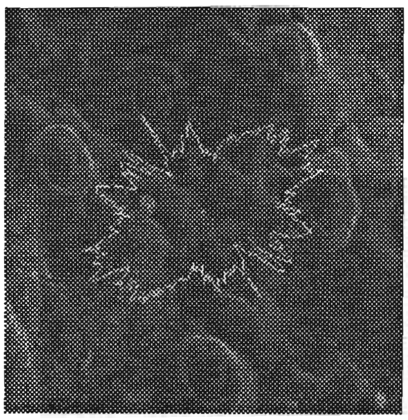


Figure B.26: Height Increase Measure: Third sample

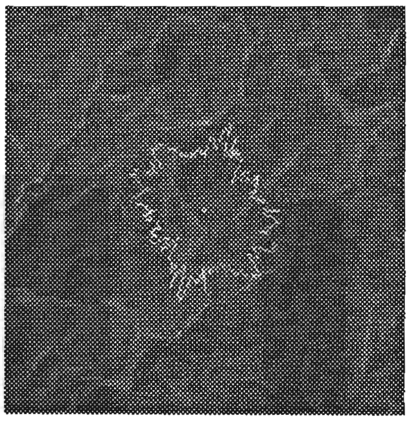


Figure B.27: Height Increase Measure: Fourth sample

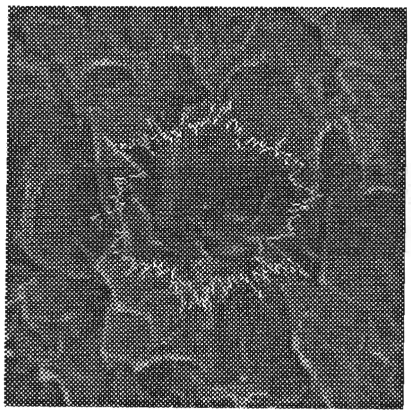


Figure B.28: Height Increase Measure: Fifth sample

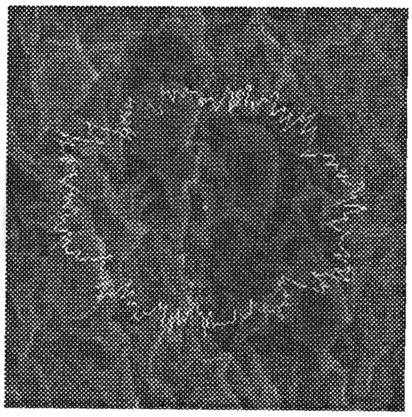


Figure B.29: Height Increase Measure: Sixth sample

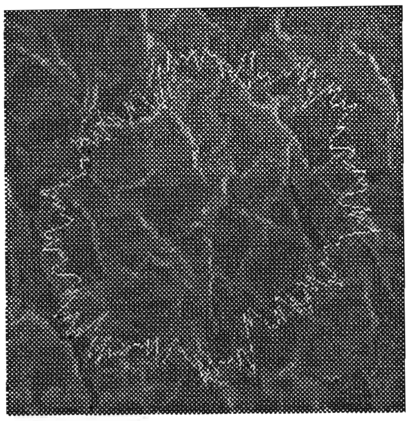
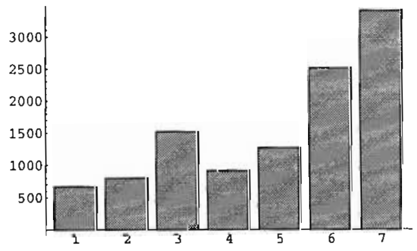


Figure B.30: Height Increase Measure: Seventh sample



Sample	*Roughness*
1	11.915
2	14.241
3	27.113
4	16.254
5	22.579
6	44.754
7	60.697

Figure B.31: Roughness ranking of gill flap samples by DFT Measure

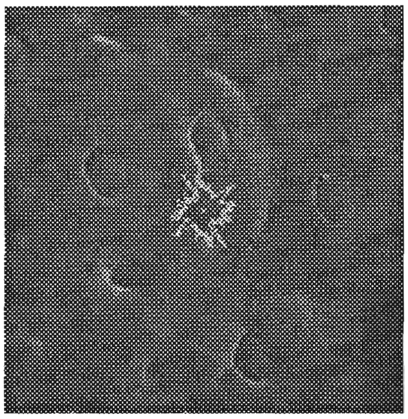


Figure B.32: DFT: First sample

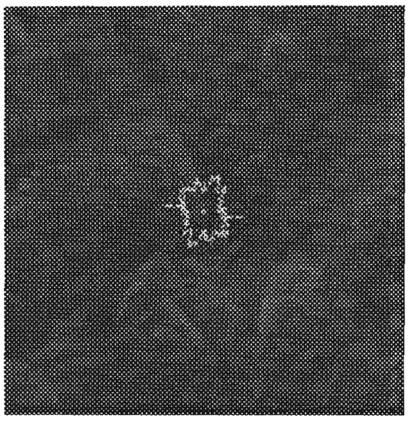


Figure B.33: DFT: Second sample

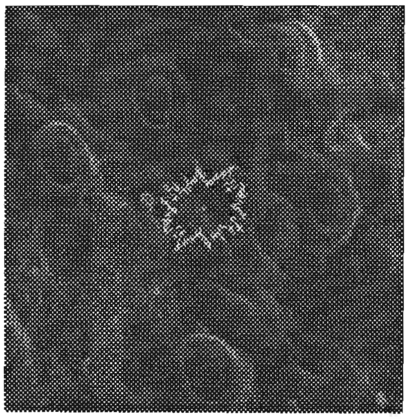


Figure B.34: DFT: Third sample

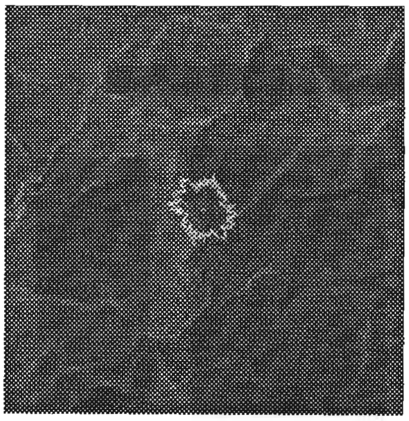


Figure B.35: DFT: Fourth sample

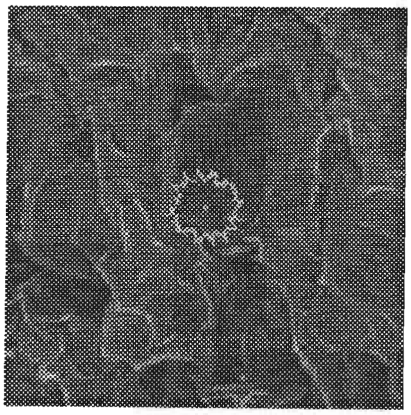


Figure B.36: DFT: Fifth sample

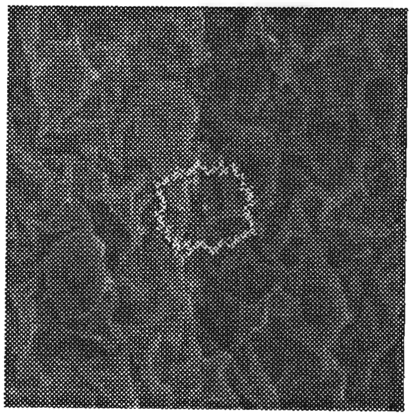


Figure B.37: DFT: Sixth sample

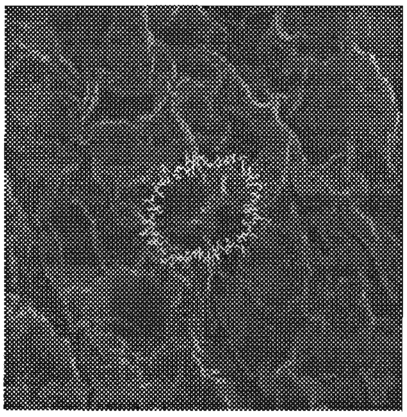
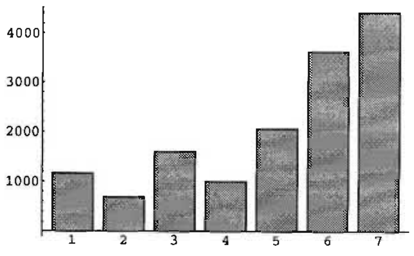


Figure B.38: DFT: Seventh sample



Sample	*Roughness*
1	20.833
2	12.315
3	28.6
4	17.983
5	36.735
6	64.805
7	78.878

Figure B.39: Roughness ranking of gill flap samples by Haar Transform Measure

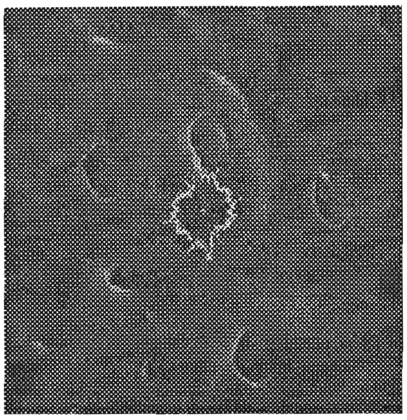


Figure B.40: Haar Wavelet Transform: First sample

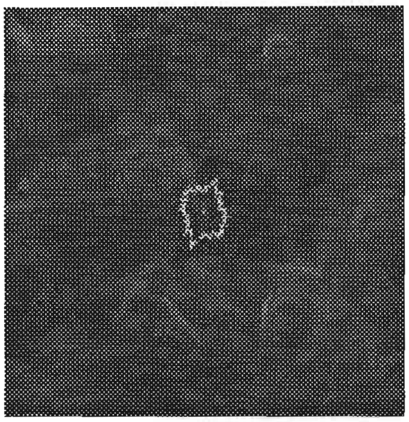


Figure B.41: Haar Wavelet Transform: Second sample

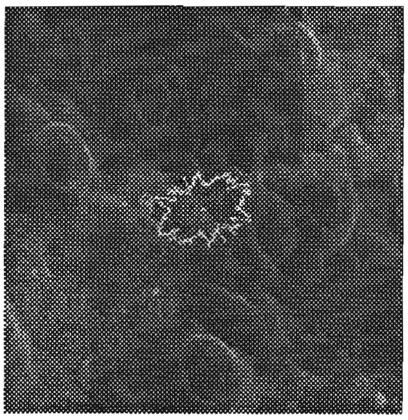


Figure B.42: Haar Wavelet Transform: Third sample

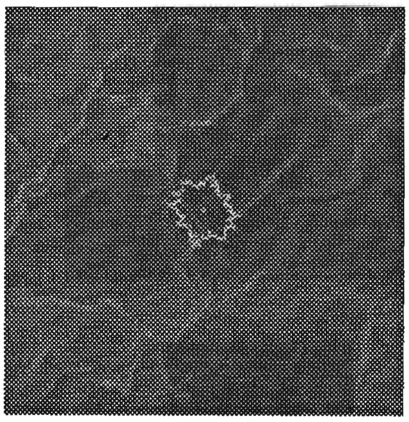


Figure B.43: Haar Wavelet Transform: Fourth sample

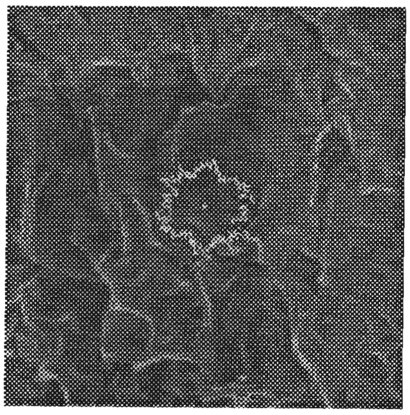


Figure B.44: Haar Wavelet Transform: Fifth sample

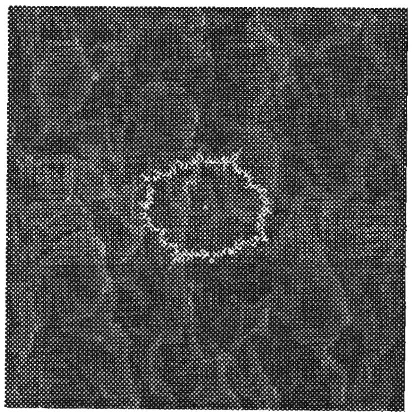


Figure B.45: Haar Wavelet Transform: Sixth sample

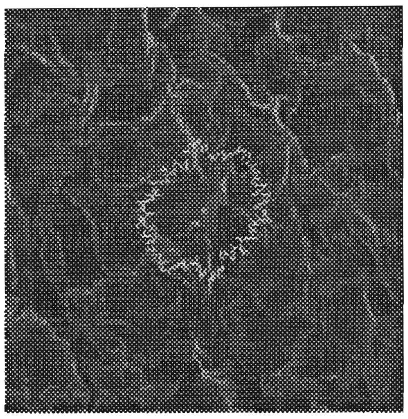
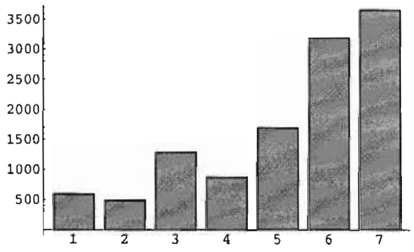


Figure B.46: Haar Wavelet Transform: Seventh sample



Sample	*Roughness*
1	10.514
2	8.668
3	22.92
4	15.526
5	30.055
6	56.866
7	65.022

Figure B.47: Roughness ranking of gill flap samples by Daubechies Transform Measure

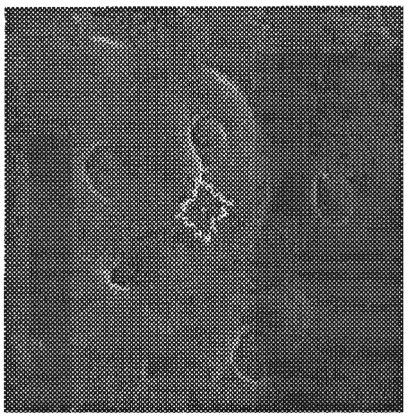


Figure B.48: Daubechies' 4 Coefficient Transform: First sample

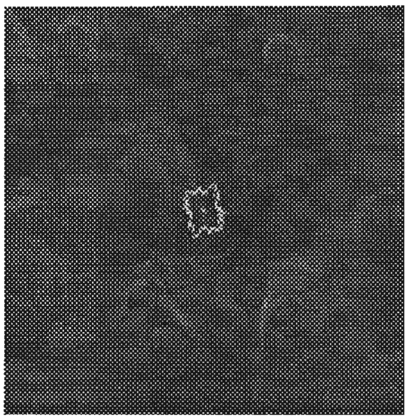


Figure B.49: Daubechies' 4 Coefficient Transform: Second sample

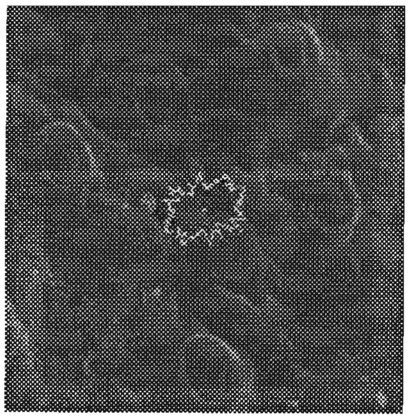


Figure B.50: Daubechies' 4 Coefficient Transform: Third sample

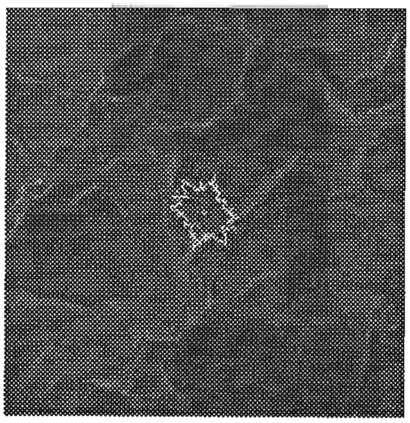


Figure B.51: Daubechies' 4 Coefficient Transform: Fourth sample

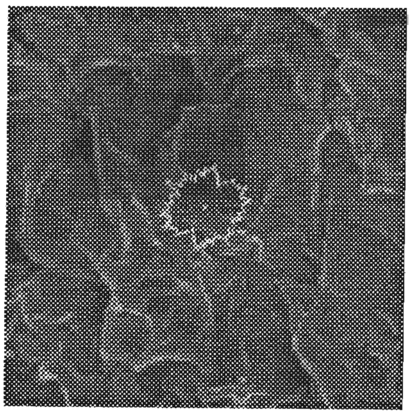


Figure B.52: Daubechies' 4 Coefficient Transform: Fifth sample

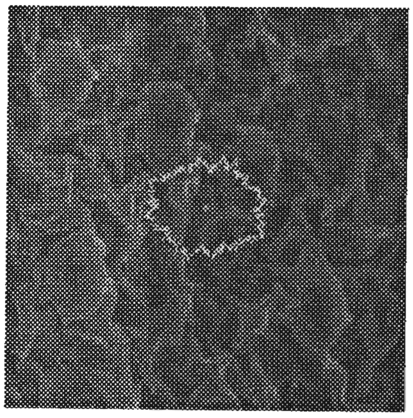


Figure B.53: Daubechies' 4 Coefficient Transform: Sixth sample

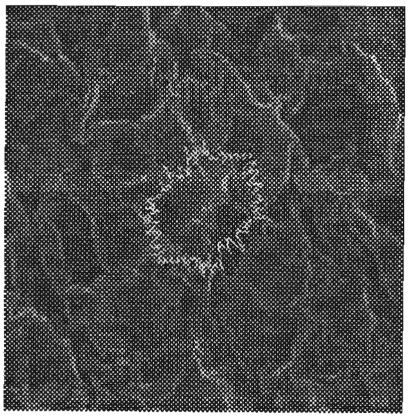


Figure B.54: Daubechies' 4 Coefficient Transform: Seventh sample

Appendix C

Fast Fourier Transform Algorithm

This material is taken almost verbatim from [15], the only changes being those required by the translation from C to Java.

The following is the structure of an FFT algorithm. It has two sections, the first section sorts the data into bit-reversed order; it takes no additional storage, since it involves only swapping pairs of elements (if k_1 is the bit-reverse of k_2, then k_2 is the bit-reverse of k_1). The second section has an outer loop that is executed $\log_2 N$ times and calculates, in turn, transforms of length $2, 4, 8, \ldots, N$. For each state of this process, two nested inner loops range over the sub-transforms already computed and the elements of each transform, implementing the Danielson-Lanczos Lemma [4]. The operation is made more efficient by restricting external calls for trigonometric sines and cosines to the outer loop, where they are made only $\log_2 N$ times. Computation of the sines and cosines of multiple angles is through simple recurrence relations in the inner loops.

The FFT routine given below is based on one originally written by N.M. Brenner [16]. The input quantities are the number of complex data points (nn), the `data` array (`data[1..2*nn]`), and `isign`, which should be set to either ± 1 and is the sign of i in the exponential

$$F_n = \sum_{k=0}^{N-1} f_k \, e^{i\,2\pi\,k\,n/N} \tag{C.1}$$

When isign is set to -1, the routine then calculates the inverse transform

$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n \, e^{-i\,2\pi\,k\,n/N} \tag{C.2}$$

except that it does not multiply by the normalizing factor $1/N$, which must be done by the user.

Notice that the number nn is the number of *complex* data points. The actual length of the array (data[1..2*nn]) is 2*nn, with each complex value occupying two consecutive positions. In other words, data[1] is the real part of f_0, data[2] is the imaginary part of f_0, and so on up to data[2*nn-1], which is the real part of f_{N-1}, and data[2*nn], which is the imaginary part of f_{N-1}. The FFT routine gives back the F_n's packed in exactly the same fashion, as nn complex numbers.

The real and imaginary parts of the zero frequency component F_0 are in data[1] and data[2], the smallest nonzero positive frequency has real and imaginary parts in data[3] and data[4], the smallest (in magnitude) nonzero negative frequency has real and imaginary parts in data[2*nn-1] and data[2*nn]. Positive frequencies increasing in magnitude are stored in the real-imaginary pairs data[3], data[4], ..., up to data[nn-1], data[nn]. Negative frequencies of increasing magnitude are stored in data[2*nn-3], data[2*nn-2] down to data[nn+3], data[nn+4]. Finally the pair data[nn+1] and data[nn+2] contain the real and imaginary parts of the one aliased point that contains the most positive and the most negative frequency. This storage arrangement of complex spectra is the practical standard [15].

```
/* The code for this function was translated directly from
   the C/C++ code given in Numerical Recipes in C, Chapter 12:
   The Fast Fourier Transform. There have been changes made
   with respect to storage and memory references, and variable
   names and types.
   The first half of the algorithm deals with the bit-reversal
   of the array, and the second implements the Danielson-
   Lanczos algorithm.
*/
void four1( float data[], int nn, int isign )
{
  int n, mmax, m, istep, i, j;
  double wtemp, wr, wpr, wpi, wi, theta;
  float tr, ti, mytemp;

  n = nn << 1;
  j = 1;
  for ( i = 1; i < n; i += 2 )
  {
    if ( j > i )
    {
      mytemp = data[ j ];
      data[ j ] = data[ i ];
      data[ i ] = mytemp;
      mytemp = data[ j + 1 ];
      data[ j + 1 ] = data[ i + 1];
      data[ i + 1 ] = mytemp;
    }
    m = n >> 1;
    while ( ( m >= 2 ) && ( j > m ) )
    {
      j -= m;
      m >>= 1;
    }
```

$$\begin{tikzcd}
y_1 & s_1 & s_1 & S_1 & S_1 \\
y_2 & d_1 & s_2 & D_1 & \underline{S_2} \\
y_3 & s_2 & s_3 & S_2 & D_1 \\
y_4 \arrow[r, "DWT"] & d_2 \arrow[r, "permute"] & \underline{s_4} \arrow[r, "DWT"] & D_2 \arrow[r, "permute"] & D_2 \\
y_5 & s_3 & d_1 & d_1 & d_1 \\
y_6 & d_3 & d_2 & d_2 & d_2 \\
y_7 & s_4 & d_3 & d_3 & d_3 \\
y_8 & d_4 & d_4 & d_4 & d_4
\end{tikzcd}$$

Figure D.1: 8-point signal DWT calculation

trated for an 8-point transform in figure D.1. The output of the DWT consists of these remaining components and all the "detail" components that were accumulated along the way.

If the length of the data vector were a higher power of 2, there would be more stages of applying (D.1) and permuting. The endpoint will always be a vector of two smooth elements S and a hierarchy of detail elements d, D etc. Notice that once detail elements are generated, they simply propagate through to all subsequent stages.

A value d_i of any level is termed a "wavelet coefficient" of the original data vector, the S_1, S_2 values should strictly be called the "mother-function" coefficients although the term "wavelet coefficients" is often used loosely for both d's and the final S's. Since the full procedure is a composition of orthogonal linear operations, the whole DWT is itself an orthogonal linear operator.

To invert the DWT, one simply reverses the procedure, starting with the smallest level of the hierarchy and working (in) from right to left. The inverse matrix

$$\begin{pmatrix} c_0 & c_3 & & & \cdots & & & c_2 & c_1 \\ c_1 & -c_2 & & & & & & c_3 & -c_0 \\ c_2 & c_1 & c_0 & c_3 & & & & & \\ c_3 & -c_0 & c_1 & -c_2 & & & & & \\ \vdots & & & & \ddots & & & & \\ & & & & & c_2 & c_1 & c_0 & c_3 \\ & & & & & c_3 & -c_0 & c_1 & -c_1 \end{pmatrix} \tag{D.2}$$

is used instead of (D.1).

The matrices (D.1) and (D.2) embody periodic ("wrap-around") boundary conditions on the data vector. One normally accepts this as a minor inconvenience: the last few wavelet coefficients at each level of the hierarchy are affected by data from both ends of the data vector. By circularly shifting the matrix (D.1) $N/2$ columns to the left, one can symmetrize the wrap-around but this does not eliminate it. It is in fact possible to eliminate the wrap-around completely by altering the coefficients in the first and last N rows of (D.1) giving an orthogonal matrix that is purely band-diagonal. This variant, beyond our scope here, is useful when, e.g. the data varies by many orders of magnitude from one end of the data vector to the other.

Here is a routine, `wt1`, that performs the Daubechies pyramidal algorithm (or its inverse if `isign` is negative) on some data vector `a[1..n]`. Successive applications of the wavelet filter, and accompanying permutations, are done by the routine daub4.

```
/* The code for these two functions was translated directly
   from the C/C++ code given in Numerical Recipes in C,
   Chapter 13: Wavelet Transforms.
*/
```

```
private void wt1( float[] a, int n, int isign )
{
  int nn;

  if ( n < 4 ) return;
  if ( isign >= 0 )
    for ( nn = n; nn >= 4; nn >>= 1 )
      daub2( a, nn, isign );
  else
    for ( nn = 4; nn <= n; nn <<= 1 )
      daub2( a, nn, isign );
}

private void daub4( float[] a, int n, int isign )
{
  double C0 = 0.4829629131445341,
         C1 = 0.8365163037378079,
         C2 = 0.2241438680420134,
         C3 = -0.1294095225512604;

  float[] wksp = new float[n+1];
  int nh, nh1, i, j;

  if ( n < 4 ) return;
  nh1 = ( nh = n >> 1 ) + 1;
  if ( isign >= 0 )
  {
    for ( i = 1, j = 1; j <= n - 3; j += 2, i++ )
    {
      wksp[ i ]    = ( float ) ( C0 * a[j]   + C1 * a[j+1]
                               + C2 * a[j+2] + C3 * a[j+3] );
      wksp[ i + nh ] = ( float ) ( C3 * a[j]   - C2 * a[j+1]
                                 + C1 * a[j+2] - C0 * a[j+3] );
    }
    wksp[i]    = ( float )( C0 * a[n-1] + C1 * a[n]
                          + C2 * a[1]   + C3 * a[2] );
    wksp[i+nh] = ( float )( C3 * a[n-1] - C2 * a[n]
                          + C1 * a[1]   - C0 * a[2] );
  }
  else
  {
    wksp[1] = ( float )( C2 * a[nh] + C1 * a[n]
                       + C0 * a[1]  + C3 * a[nh1] );
    wksp[2] = ( float )( C3 * a[nh] - C0 * a[n]
                       + C1 * a[1]  - C2 * a[nh1] );
    for ( i = 1, j = 3; i < nh; i++ )
    {
      wksp[j++] = ( float )( C2 * a[i]   + C1 * a[i+nh]
                           + C0 * a[i+1] + C3 * a[i+nh1] );
      wksp[j++] = ( float )( C3 * a[i]   - C0 * a[i+nh]
                           + C1 * a[i+1] - C2 * a[i+nh1] );
    }
  }
  for ( i = 1; i <= n; i++ )
    a[i] = wksp[ i ];
}
```

Bibliography

[1] N. Barton. The shear strength of rock and rock joints. *International Journal of Rock Mechanics and Mining Sciences and Geomechanical Abstracts*, pages 255–279, 1976.

[2] F.G. Bell. *Engineering Geology.* Blackwell Science, Cambridge, 1995.

[3] Mac A Cody. The fast wavelet transform. *Dr Dobbs Journal*, pages 16–28, April 1992.

[4] G. Danielson and C. Lanczos. Some improvements in practical fourier analysis and their application to x-ray scattering from liquids, 1942.

[5] Ingrid Daubechies. *Ten Lectures On Wavelets.* SIAM, Philadelphia, 2nd edition, 1992.

[6] Alain Fournier. Wavelets and their applications in computer science. *SIGGRAPH Conference Proceedings*, 1994.

[7] H. Goldstein. *The investigation into an algorithm based on wavelet basis functions for the spatial and frequency decomposition of arbitrary signals.* PhD thesis, University of Natal, Durban campus, Department of Computer Science, January 1994.

[8] Amara Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2), Summer 1995.

[9] Forrest M. Huffman. *An Introduction to Fourier Theory.* http://aurora.phys.utk.edu/~forrest/papers/fourier/.

[10] M.R.M Lewis, H.C. Murrell, C.A. Jermy, and C.G. Palmer. On measuring roughness. *South African Computer Journal*, 26:49–56, August 2001.

[11] B.B. Mandelbrot. Chance and dimension, 1977.

[12] G.P. Nason. Wavelets. *New Electronics*, April 1997.

[13] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Fractals for the Classroom.* Springer-Verlag, New York, 1st edition, 1992.

[14] Robi Polikar. *The Wavelet Tutorial.* Iowa State University, 1996. http://www.public.iastate.edu/~rpolikar/WAVELETS/.

[15] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes In C: The Art of Scientific Computing.* Cambridge University Press, New York, 2nd edition, 1992.

[16] C.M. Rader and N.M. Brenner. A new principle for fast fourier transformation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-24(3), June 1976.

[17] Leena-Maija Reissell. Wavelets and their applications in computer science. *SIGGRAPH Conference Proceedings*, 1994.

[18] J.C. Russ. *Fractal Surfaces.* Plenum Press, New York, 1994.

[19] Gilbert Strang. Wavelet and dilation equations: a brief introduction. *SIAM Review*, 31(4):614–627, December 1989.

[20] C Valens. *A Really Friendly Guide To Wavelets.* mindless.com, 1999. http://perso.wanadoo.fr/polyvalens/clemens/wavelets/-wavelets.html.