UNIVERSITEIT·STELLENBOSCH·UNIVERSITY
jou kennisvennoot • your knowledge partner

Automated stratigraphic classification and feature detection from images of borehole cores

by

Stéfan Johann van der Walt

Thesis presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leaders:

Professor J.H. Cloete   Professor B.M. Herbst

April 2005

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: ....
S.J. van der Walt

Date: ....

Abstract

Automated stratigraphic classification and feature detection from images of borehole cores

S.J. van der Walt

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (E&E with CS)

April 2005

This thesis describes techniques proposed for analysing images of borehole cores. We address two problems: firstly, the automated stratigraphic classification of cores based on texture and secondly, the location of thin chromitite layers hidden in pyroxenite cores.

Texture features of different rock types are extracted using wavelets, the theory of which provides an intuitive and powerful tool for this purpose. A Bayesian classifier is trained and used to discriminate between different samples.

Thin, planar chromitite layers are located using a shortest path algorithm. In order to estimate the physical orientation of any layer found, a sinusoidal curve is fitted.

The proposed algorithms were implemented and tested on samples taken from photographed cores. A high success rate was obtained in rock classification, and thin planar layers were located and characterised.

Samevatting

Geoutomatiseerde stratigrafiese klassifisering en kernmerkonttrekking vanuit beelde van boorgatkerne

S.J. van der Walt

Departement van Elektriese en Elektroniese Ingenieurswese
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (E&E met RW)

April 2005

Verskeie tegnieke word aangebied vir die analise van boorgatkernfotos. Twee probleme word ondersoek: eerstens, die geoutomatiseerde stratigrafiese klassifikasie van kerne, gebaseer op tekstuur en tweedens, die uitkenning van dun kromitietlagies verskuil in piroksenietkerne.

Verskillende rotstipes se tekstuurkenmerke word onttrek deur gebruik te maak van golfies. Golfieteorie bied ’n intuïtiewe en kragtige stuk gereedskap vir hierdie doel. ’n Bayese klassifiseerder word opgelei en gebruik om verskillende rotsteksture van mekaar te onderskei.

Vlakke van dun kromitietlagies sny soms deur die boorgatkerne. ’n Kortstepad-algoritme word gebruik om sulke lagies te vind. Om die oriëntasie van so ’n vlak te bepaal, word ’n sinuskurwe gepas.

Die algoritmes is geïmplementeer en getoets met monsters, geneem vanuit kernfotos. ’n Hoë sukseskoers is verkry by die uitkenning van rotstipes en dun vlaklagies is gevind en hul eienskappe bepaal.

Acknowledgements

I would like to thank the following people and institutions for their contributions to this project:

- My academic advisors, Professors J.H. Cloete and B.M. Herbst, for generously donating their time to this project.
- DefenceTek, GeoMole and the National Research Foundation for funding this project.
- The University of Stellenbosch for usage of laboratory facilities.
- My wife, Michèle, for supporting me with so much love and enthusiasm.
- My family, P.W, Elaine and Madelé, for their motivation and support.
- My in-laws, Michael, Brigitte and Tristan Janse van Rensburg, who always show much interest in what I do.
- Josef Ekkerd, Geologist, De Beers Finsch Mine, for his assistance in photographing dolomite cores.
- Karen Hunter for her insights on wavelet theory.
- My colleagues from the laboratory, Tim Sindle and Lötter Kock, for humorous discourse and a pleasant working environment.
- Wessel Croukamp and Ulrich Buttner for designing and manufacturing the "Klipspringer" camera suspension arm.
- Paul Kienzle and David Bateman, for welcoming me into the Octave Forge community.
- My university friends, who had an uncanny understanding of what I was going through. You cheered up many a gloomy night with interesting conversation!

Contents

List of Figures

List of Tables

Notation

The notation corresponds to that used in [Mal99] and [SN97].

Mathematical

$\mathbb{Z}$	Integers.
$\mathbb{R}$	Real numbers.
$\langle f, g \rangle$	Inner product, $\int_{-\infty}^{+\infty} f(t)\, g(t)\, dt$.
x^*	Complex conjugate of x, i.e. if $x = Ae^{j\theta}$ then $x^* = Ae^{-j\theta}$.
$\mathbf{L}^2(\mathbb{R})$	The function space of measurable, one-dimensional, finite-energy functions $f(x)$, so that $\int_{-\infty}^{+\infty} \lvert f(x) \rvert^2 \, dx < \infty$.
$l^2(\mathbb{Z})$	The vector space of all sequences $c = \{c_k : k \in \mathbb{Z}\}$, so that $\sum_{k=-\infty}^{+\infty} \lvert c_k \rvert^2 < \infty$.

Signals and filters

$f(t)$	Continuous-time signal.
$f(n)$	Discrete-time signal.
$\psi_{j,k}(t)$	Wavelet at scale j with translation k.
$\phi_{j,k}(t)$	Scaling function at level j with translation k.
$h(n)$	Coefficients of a discrete-time, finite impulse response filter.

Probability

X	Random variable.
$\mathsf{E}[\,\cdot\,]$	Expected value.
$\tilde{\mathsf{E}}[\,\cdot\,]$	Estimated expected value based on a set of samples.
$\bar{X}$	Mean or expected value of X.
$\eta(\mu, \sigma)$	Normal distribution with mean μ and standard deviation σ.
$P(x)$	Probability mass function with $P(x) \geq 0$ and $\sum_x P(x) = 1$.

$p(x)$	Probability density function with $p(x) \geq 0$ and $\int_{-\infty}^{+\infty} p(x)dx = 1$.

Vectors and Matrices

a	Scalar
$\mathbf{a}$	Vector
$\bar{\mathbf{a}}$	Complex conjugate of $\mathbf{a}$ (every element $a + ib$ becomes $a - ib$).
$\mathbf{a}^T$	Transpose of $\mathbf{a}$.
$\mathbf{a}^H$	Conjugate transpose or Hermitian of $\mathbf{a}$ ($\mathbf{a}^H = \bar{\mathbf{a}}^T$)
I	Identity matrix

Transforms

$F(\omega)$	Fourier transform of $f(t)$.
$Sf(u, s)$	Windowed or short-time Fourier transform of $f(t)$.
$Wf(u, s)$	Wavelet transform of $f(t)$.
$\check{f}_{L,E}(n)$	Discrete wavelet transform at level $K - L$, with extension type E (values: P (periodic), S (symmetric) and Z (zero-padding)). The signal to be decomposed is defined at level K.
$X(z)$	The z-transform of $x(n)$: $X(z) = \sum_{n=-\infty}^{+\infty} x(n)z^{-n}$.

Chapter 1

Introduction

Mining companies use stratigraphic and geotechnical models to estimate where profitable ore may be safely extracted. In order to construct such models, boreholes of small diameter are drilled [Hei94]. The cores extracted from inside these boreholes are carefully examined, and the models adjusted according to the observed properties of the core.

Individually examining each piece of core can be a laborious and time-consuming task. Such boreholes can reach lengths exceeding 2 km and, in 2003 alone, Anglo-Platinum produced 667 km of core [Pla]. Figures 1.0.1 and 1.0.2 show a core yard and give an indication of the enormous magnitude of the task.

A geologist must classify the rock type according to depth and also note any strange anomalies present. This information is often entered into a database, from which logs similar to the one shown in Appendix C.1 can be generated.

Core scanning is becoming more commonplace, allowing permanent, high-quality digital stratigraphic records to be stored [Pla]. With so much raw, digital data available, one inevitably wonders whether a computer can assist in classifying cores and in finding geological features [HPGM96].

In this thesis, we discuss image processing techniques that address these problems. Two outcomes are specified: firstly, the stratigraphic classification of borehole cores to a given accuracy and secondly, the location and identification of certain thin layers present. Specifically, we are interested in chromitite stringers — thin, brittle layers that may result in the collapse of hanging walls (roofs) of stopes and tunnels.

We want to perform stratigraphic classification based on the texture of core images. Much research has been done on the analysis of texture [AM03, CJ83, HSD73, HSL92, CK93, PS95, Skl78, Uns95, vdW98], but very little aimed specifically at natural texture [CDE96] and rock texture [HPGM96]. This thesis

Figure 1.0.1: Kilometres of core stored at the De Beers/Finsch core yard.

investigates the use of more modern texture features and uses them to classify dolomite rock samples successfully.

Locating thin layers is similar to fracture detection in oil wells. Sinusoidal fitting [HTE94] (sometimes found using the Hough transform [THK97]) is a popular method for addressing the problem. It works well for images where a high contrast is observed between the fracture and the rock wall, but this is not the case for pyroxenite cores, as is shown in Figure 1.0.3. We propose isolating the pixels of thin layers by using a shortest path algorithm, as suggested by Sun and Pallottino [SP03]. Sinusoidal curves can then be fitted to these pixels in order to obtain the orientation of the plane cutting through the borehole.

We now provide a short overview of each chapter.

Wavelet analysis is a relatively recent addition to the arsenal of signal processing tools. It is the culmination of fruitful collaboration between many different scientific disciplines and provides a method of analysing signals, not only in the frequency domain, but also in the time (or spatial) domain[1]. CHAPTER 2 explores this localisation in the time-frequency domain and explains why this is a desirable property. The tight coupling between discrete wavelets

[1] We use terms "time domain" and "spatial domain" interchangeably.

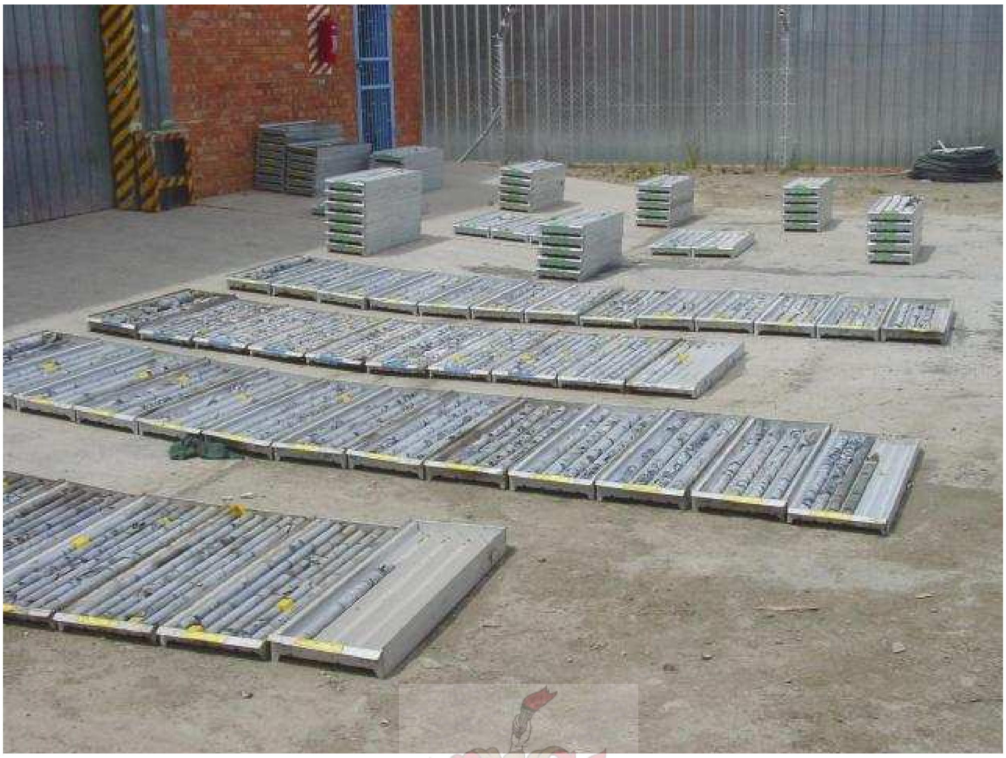


Figure 1.0.2: Cores laid out in the De Beers, Finsch core yard after classification.

and filter banks is explained and illustrated.

The classification of rock types is based on texture analysis and recognition. Classical approaches to texture analysis are re-iterated in CHAPTER 3. Texture descriptors are typically formed from the statistical properties of textured images. This "time-domain" approach is sometimes complemented by spectral descriptors, which analyse the frequency content of textures. Wavelet methods are neither "time-domain" nor "frequency-domain", but a combination of both. It allows the representation of different texture frequencies, as well as where they occur spatially. Wavelet families have different attributes (associated with different filters). A short discussion is given on the appropriate choice of wavelet family. The suitability of the wavelet packet transform for texture recognition is evaluated.

A statistical classifier is used to discriminate between different texture types. While this thesis focuses mainly on the types of texture features involved, these features need to be evaluated experimentally. As such, the Bayesian classifier was chosen under the assumption that the class data is normally distributed. Of the simpler classifiers, it has been shown to give good results, provided that

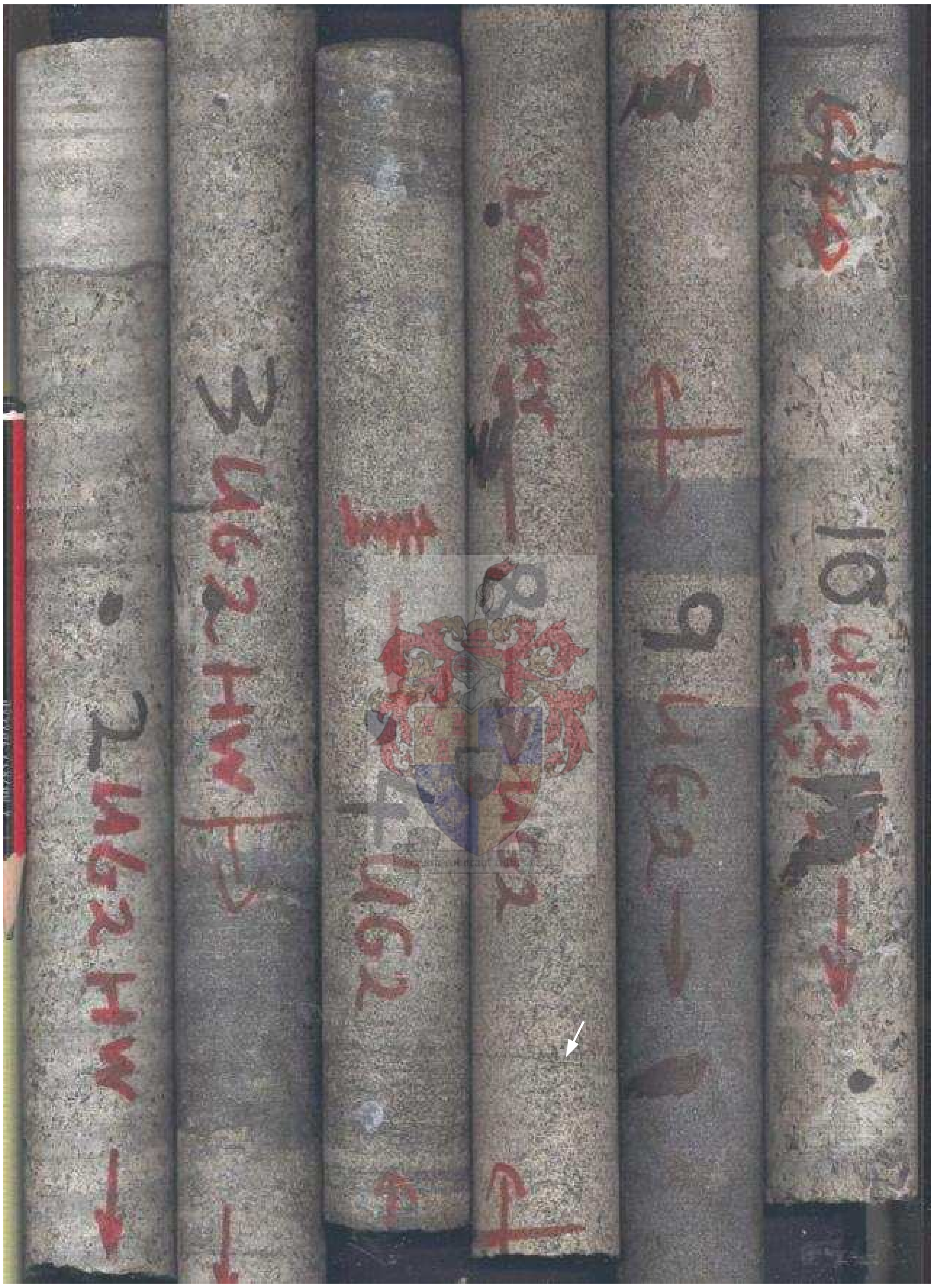


Figure 1.0.3: A chromitite stringer visible in a pyroxenite core (fourth core from the left, bottom).

the normality assumption is accurate.

While training the classifier, data scarcity proved to be a problem. The feature vectors were too sparse in the feature space to provide accurate estimates of the parameters of the normal-model. Principal component analysis and multiple discriminant analysis were evaluated as ways of reducing the dimensionality of the feature data. The feature vectors were thereby "compressed" in the feature space, allowing more accurate training of the classifier.

CHAPTER 4 deals with the location of thin layers. The layers are often dark and hidden in a fairly "noisy" texture background. Localised methods fail to isolate the layers. Consequently, algorithms that analyse *complete* layers/paths have to be taken into consideration. The shortest paths algorithm has proved successful in isolating pixels of chromitite stringers, found in the UG2 system of the Bushveld Igneous Complex (see the geological log in C.1).

A thin, planar layer cutting through a borehole is observed as a sinusoidal pattern in the core photograph. A (non-linear) sinusoidal model is fitted to the pixels obtained. The dip and azimuth of the plane can be calculated from the resulting phase and amplitude.

Experimental results for the texture classification system are presented in CHAPTER 5. The experiments are based on dolomite images, acquired from the De Beers, Finsch diamond mine. The configuration and lighting requirements for photographing cores are described. After a brief overview of the software used, we present the success rates obtained by the texture classifier. The trade-off between accuracy and success, depending on the number of rejections made, is illustrated.

We refer the reader to APPENDIX A for mathematical results used throughout the text.

Chapter 2

An Overview of Wavelet Analysis

2.1 Introduction

> “Wavelets are not based on a ’bright new idea’, but on concepts that already existed under various forms in many different fields. The formalization and emergence of this ’wavelet theory’ is the result of a multidisciplinary effort that brought together mathematicians, physicists and engineers, who recognized that they were independently developing similar ideas. For signal processing, this connection has created a flow of ideas that goes well beyond the construction of new bases or transforms.”
>
> — Stéphane Mallat [Mal99]

Petroleum geologists often use seismic waves to investigate underground structures. One such an engineer, Jean Morlet, found a special way of analysing seismic reflection signals. He extracted signal components localised in space and called them *wavelets* [M^{+}01]. Little was he to know that, years later, the development of a mathematical foundation describing these waves would bring together many different disciplines of science. The culmination of the effort is what is known today as *wavelet theory*.

The Fourier transform is an excellent way of examining the characteristics of linear time-invariant systems, but in many applications the transient properties of a signal are just as, if not more, important than the Fourier properties. In the case of textures, not only is it important what frequencies the samples contain, but also where these frequency components occur spatially.

The Fourier transform cannot provide this information, since it considers the signal as a whole, being defined as

$$F(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-j\omega t}dt,$$

where ω is the radial frequency. The original signal can be perfectly reconstructed over continuous regions, using the inverse Fourier transform:

$$f(t) \;=\; \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{j\omega t}d\omega.$$

2.1.1 The short-time Fourier transform

The Fourier transform analyses the frequency content of a signal, but cannot show *where* those frequencies occur. The short time Fourier transform, introduced by Gabor, addresses this problem. Here, the input signal is multiplied by a real and symmetric shifting window of finite support[1] in order to localise the transform, which then becomes

$$Sf(u,\xi) = \int_{-\infty}^{+\infty} f(t)\, g(t-u)e^{-j\xi t}dt, \tag{2.1.1}$$

where $g(t)$ is the window function, ξ the radial frequency and u the translation in time. This transform provides information about both the time and the frequency content of an image. Equation (2.1.1) shows the transform in the time domain, where it is localised around u, allowing us to narrow down the position of transient effects in time. The window function is not an impulse and is spread in time, and the energy of the transform is localised over an interval of size Δt.

Mallat [Mal99, p. 3] also examines the window in the frequency domain, showing a similar localisation here. The transform is obtained by correlating the signal with a modulated window or *atom*, $g_{u,\xi}(t)$, translated in time and frequency and defined as

$$g_{u,\xi}(t) = g(t-u)e^{j\xi t}. \tag{2.1.2}$$

The short time Fourier transform is then

$$Sf(u,\xi) = \int_{-\infty}^{+\infty} f(t)\, g^*_{u,\xi}(t)dt = \int_{-\infty}^{+\infty} f(t)\, g(t-u)\, e^{-j\xi t}dt. \tag{2.1.3}$$

[1] Support: the interval centred around u where $f(t-u)$ is non-negligible.

Parseval's theorem [PZP01, A. 3] states that

$$\int_{-\infty}^{+\infty} f_1(t)\, f_2^*(t)\, dt = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F_1(\omega) F_2^*(\omega)\, d\omega,$$

which allows (2.1.3) to be written as

$$Sf(u,\xi) \quad = \quad \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)\, G_{\mu,\xi}^*(\omega)\, d\omega. \tag{2.1.4}$$

Here, the Fourier transform of the atom is

$$\begin{aligned} G_{u,\xi}(\omega) &= \int_{-\infty}^{+\infty} g(t-u) e^{j\xi t} e^{-j\omega t} dt \\ &= \int_{-\infty}^{+\infty} g(t-u) e^{-j(\omega-\xi)t} dt \\ &= \int_{-\infty}^{+\infty} g(v) e^{-j(\omega-\xi)(v+u)} dv \\ &= G(\omega-\xi) e^{-j(\omega-\xi)u} \end{aligned}$$

and the complex conjugate becomes

$$G_{u,\xi}^*(\omega) = G(\omega-\xi) e^{j(w-\xi)u}.$$

The atom is centred in the frequency domain around ξ. The atom is not an impulse in the frequency domain, but is spread, so that the transform energy is localised over an interval of size Δf. According to Gabor's interpretation of the uncertainty principle [Gab46, p. 432],

$$\Delta t \Delta f \geq \frac{1}{2}.$$

The product $\Delta t \Delta f$ is *minimum when the window $g(t)$ is Gaussian*, in which case $g_{u,\xi}$ is known as a *Gabor atom.*

Now, rewrite the transform (2.1.4) in terms of frequency as

$$Sf(u,\xi) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)\, G(\omega-\xi)\, e^{ju(\omega-\xi)}\, d\omega.$$

The transform energy is localised over an interval of size Δf in the frequency domain and, as shown earlier, over an interval of size Δt in the time domain.

The short time Fourier transform thus has a fixed, finite time-support Δt centred around u, and a fixed, finite frequency spread Δf centred around ξ (see a representation of the atom energy spread in Figure 2.1.1). This enables

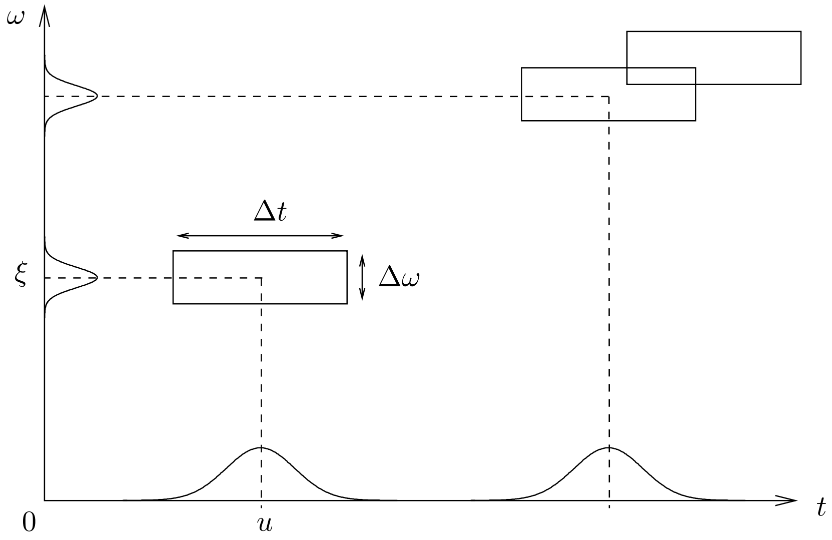


Figure 2.1.1: Energy spread of a Gabor atom in the time-frequency plane.

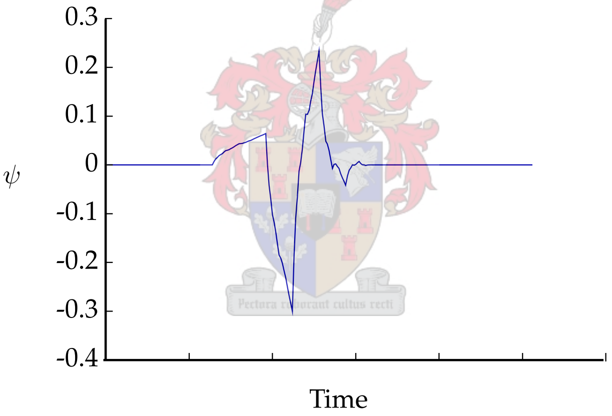


Figure 2.1.2: The Daubechies wavelet with 4 vanishing moments.

the short-time Fourier transform to point out transient effects in both the time and frequency domains. The transform is not perfect, however. It would make more sense to have atoms with varying time and frequency support, depending on their positions in the time-frequency plane. An atom at a low frequency, for example, should have longer time support (and then necessarily has smaller frequency spread). The wavelet transform achieves this.

2.1.2 Wavelets in the time-frequency plane

Figure 2.1.2 shows a typical wavelet, ψ. An (orthogonal) wavelet is a signal whose dilations and translations form an orthogonal basis of $\mathbf{L}^2(\mathbb{R})$, as explained later. For now, simply note that the wavelet is localised in time, with a zero average, i.e.

$$\begin{aligned} \psi(t) &= 0 \quad |t| > T, \\ \int_{-\infty}^{+\infty} \psi(t)dt &= 0. \end{aligned}$$

This wavelet is scaled by a scaling factor s and translated in time by u, to form the atom

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right). \tag{2.1.5}$$

The wavelet atom's Fourier transform is calculated in a way similar to that used with the Gabor atom. The atom in the frequency domain is

$$\Psi_{u,s}(\omega) = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) e^{-j\omega t}dt,$$

which can be simplified using the substitution $v = \frac{t-u}{s}$, so that it becomes

$$\begin{aligned} \Psi_{u,s}(\omega) &= \int_{-\infty}^{+\infty} \sqrt{s}\psi\left(v\right) e^{-j\omega(sv+u)}dv \\ &= e^{-j\omega u}\sqrt{s}\int_{-\infty}^{+\infty} \psi(v)e^{-j(\omega s)v}dv \\ &= e^{-j\omega u}\sqrt{s}\Psi(\omega s). \end{aligned}$$

The wavelet transform is obtained by correlating a function with the wavelet atom, $\psi_{u,s}(t)$, and is

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t)\,\psi_{u,s}^*(t)\,dt = \int_{-\infty}^{+\infty} f(t)\,\frac{1}{\sqrt{s}}\psi^*\left(\frac{t-u}{s}\right)\,dt, \tag{2.1.6}$$

which can be written in terms of frequency, using Parseval's theorem, as

$$Wf(u,s) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)\,\Psi_{u,s}^*(\omega)\,d\omega = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)\,e^{-j\omega u}\sqrt{s}\Psi(\omega s)\,d\omega. \tag{2.1.7}$$

From (2.1.6) and (2.1.7) it can be seen that the energy of the wavelet transform is, like in the case of the short time Fourier transform, more or less localised in time and frequency. The time-support Δt increases in proportion to the scale s, while the frequency spread is inversely proportional to s. This makes intuitive sense: time support should be long for low frequencies and short for high frequencies.

Orthogonal wavelet bases

Both the short time Fourier and the wavelet transforms discussed in Section 2.1.2 are highly redundant. This redundancy stems from the correlation of the signal with an atom, as the atom is shifted over all time. The set of all atoms, shifted over all time and scaled by all factors,

$$\left\{\psi_{u,s}(t)\right\}_{u,s\in\mathbb{R}^2},$$

does not form an orthogonal basis of the function space $\mathbf{L}^2(\mathbb{R})$. We define a new set of atoms (or a family of wavelets), translated by integer values of n, as

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n)\in\mathbb{Z}^2}.$$

If $\psi(t)$ is chosen carefully, this family does form an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$ [Mal99, p. 220]. A signal can then be expressed as a linear combination of the orthogonal wavelets — an idea central to the discrete wavelet transform.

2.2 Fundamental theory

2.2.1 An example filter bank

A filter bank is a set of parallel, discrete-time filters through which a signal is fed. The wavelet decomposition is implemented by means of such filter banks, as is shown in Section 2.2.2. Here, we give a short overview of filters and demonstrate filter banks by means of the Haar wavelet.

Examine the filter with the Haar coefficients

$$h_0(n) = \left[\begin{array}{cc} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{array}\right].$$

The result of filtering a discrete signal x, using h_0, is

$$y(n) = \sum_{k=-\infty}^{+\infty} h_0(k)x(n-k), \tag{2.2.1}$$

or, expressed as a convolution[2],

$$y(n) = (h_0 * x)(n).$$

If both $x(n)$ and $h_0(n)$ are zero for $n < 0$, the filter response, $y(n)$, is too. Any realisable filter is *causal*: in other words, it cannot produce a result before receiving an input. Since the filter has a limited number of coefficients, T, the filter response to an input of length N is also zero for $n > N + T - 1$ — it is a *finite impulse response* or *FIR filter*. This means that, for a limited number of input values, the output will have a limited number of non-zero values as well.

It follows from (2.2.1) that the discrete Fourier transform of a convolution is equivalent to

$$\begin{aligned} Y(\omega) = \sum_{n=-\infty}^{\infty} y(n)e^{-j\omega n} &= \sum_{n=-\infty}^{\infty} \sum_{k=-\infty}^{+\infty} h_0(k)x(n-k)e^{-j\omega n} \\ &= \sum_{k=-\infty}^{\infty} \sum_{u=-\infty}^{+\infty} h_0(k)x(u)e^{-j\omega(u+k)} \\ &= H_0(\omega)X(\omega). \end{aligned}$$

The *frequency response* $H_0(\omega)$ of the filter is the discrete Fourier transform of $h_0(n)$,

$$\begin{aligned} H_0(\omega) &= \sum_{n=-\infty}^{+\infty} h_0(n)e^{-jn\omega} \\ &= h_0(0) + h_0(1)e^{-j\omega} \\ &= \frac{1}{\sqrt{2}}(1 + e^{-j\omega}). \end{aligned} \tag{2.2.2}$$

This can be rewritten as

$$\begin{aligned} H_0(\omega) &= \frac{1}{\sqrt{2}}(e^{j\omega/2} + e^{-j\omega/2})e^{-j\omega/2} \\ &= \sqrt{2}\cos\left(\frac{\omega}{2}\right) e^{-j\omega/2}. \end{aligned}$$

The filter frequency response is that of a low-pass filter (over $0 \le \omega \le \pi$), with a magnitude of zero at $\omega = \pi$, $\sqrt{2}$ at $\omega = 0$ and with a linear phase response of

[2] A convolution between $a(n)$ and $b(n)$ is defined as $y(n) = (a*b)(n) = \sum_{n=-\infty}^{+\infty} a(k)b(n-k)$.

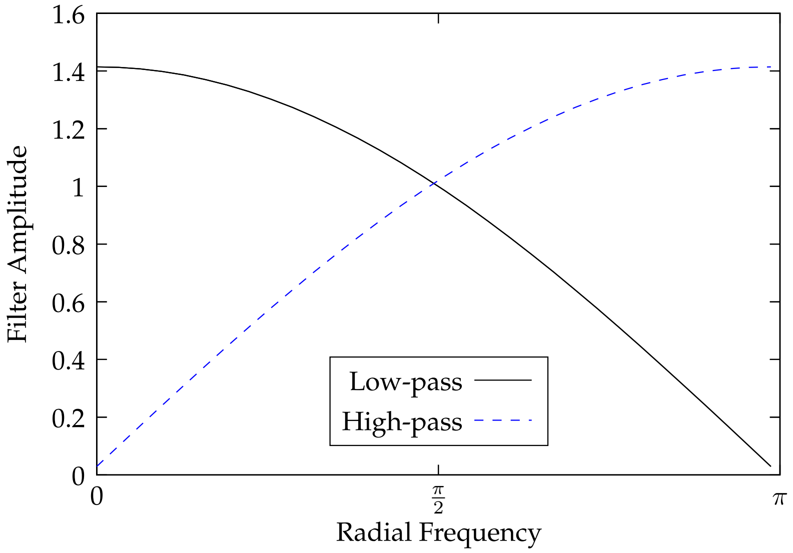


Figure 2.2.1: Frequency response of the Haar low- and high-pass filters.

$-\omega/2$. It can be shown in the same way that the filter

$$h_1(n) = \left[\begin{array}{cc} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{array} \right]$$

is a high-pass filter: the response is a mirrored image of $H_0(\omega)$ around $\frac{\pi}{2}$, as shown in Figure 2.2.1.

Filtering a signal by h_0, we obtain a low-pass version of that signal — an approximation of the signal, with higher frequencies (or detail) removed. The opposite is true when using h_1: here we remove low frequencies (or slowly changing parts of the signal) in order to obtain the details removed by the low-pass filter. As we will show later, low-pass filters produce *scaling functions*, while high-pass filters produce *wavelets*.

Define the input vector

$$x = \left[\begin{array}{cccccccc} 0 & 0 & 1 & 1 & 0.75 & 0.5 & 0.25 & 0.0 \end{array} \right]^T.$$

The responses of x when filtered by h_0 and h_1 are

$$x * h_0 = \left[\begin{array}{ccccccccc} 0 & 0 & 0.707 & 1.414 & 1.237 & 0.884 & 0.53 & 0.177 & 0 \end{array} \right]^T \quad (2.2.3)$$

$$x * h_1 = \left[\begin{array}{ccccccccc} 0 & 0 & 0.707 & 0 & -0.177 & -0.177 & -0.177 & -0.177 & 0 \end{array} \right]^T \quad (2.2.4)$$

Notice how the detail coefficients, $x * h_1$, are generally smaller than the low-pass approximation coefficients, $x * h_0$, since they only describe *changes* in the signal. Because the elements have smaller values, they can be represented using fewer bits, which is useful for signal compression. Also, the signal has been split into an approximation and a detail sub-signal, which is useful for analysis. If the signal is to be represented this way, there remains one problem: $x(n)$ consists of 8 elements, while the resulting combination of coefficients comprise 18 elements, more than twice the original number.

For each result, examine the vector consisting of the last 8 elements. *Downsample* this vector (denoted by $(\downarrow 2)$) by discarding every second element. The coefficients then become

$$\begin{aligned} (\downarrow 2)\, x * h_0 &= \begin{bmatrix} 0 & 1.414 & 0.884 & 0.177 \end{bmatrix}^T \\ (\downarrow 2)\, x * h_1 &= \begin{bmatrix} 0 & 0 & -0.177 & -0.177 \end{bmatrix}^T. \end{aligned}$$

Now there are 8 elements in total, the same number as in original input. The first level wavelet decomposition is

$$\breve{x}_{1,P} = \begin{bmatrix} 0.0 & 1.414 & 0.884 & 0.177 & 0.0 & 0.0 & -0.177 & -0.177 \end{bmatrix}^T,$$

which can also be written in the matrix form[3]

$$\breve{x}_{1,P} = Ax = \begin{bmatrix} H_0 \\ H_1 \end{bmatrix} x = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & & & & & & \\ & & 1 & 1 & & & & \\ & & & & 1 & 1 & & \\ & & & & & & 1 & 1 \\ -1 & 1 & & & & & & \\ & & -1 & 1 & & & & \\ & & & & -1 & 1 & & \\ & & & & & & -1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.75 \\ 0.5 \\ 0.25 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1.414 \\ 0.884 \\ 0.177 \\ 0 \\ 0 \\ -0.177 \\ -0.177 \end{bmatrix}.$$

The transform would be of little use if the original signal x can not be recovered from these coefficients. Since A can be shown to be orthonormal ($A^{-1} = A^T$),

[3] Notation: all elements with a value of zero are left blank.

we note that

$$A^T\check{x}_{1,P} = A^TAx = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & & & & -1 & & & \\ 1 & & & & 1 & & & \\ & 1 & & & & -1 & & \\ & 1 & & & & 1 & & \\ & & 1 & & & & -1 & \\ & & 1 & & & & 1 & \\ & & & 1 & & & & -1 \\ & & & 1 & & & & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1.414 \\ 0.884 \\ 0.177 \\ 0 \\ 0 \\ -0.177 \\ -0.177 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.75 \\ 0.5 \\ 0.25 \\ 0 \end{bmatrix} = x.$$

Perfect reconstruction! The multiplication by A^T has the same effect as upsampling[4] (2.2.3) and (2.2.4), filtering by the synthesis filters g_0 and g_1 respectively and then adding the results. Thus,

$$\begin{aligned} (\uparrow 2)(\downarrow 2)\, x * h_0 &= \begin{bmatrix} 0 & 0 & 1.414 & 0 & 0.884 & 0 & 0.177 \end{bmatrix}^T \\ (\uparrow 2)(\downarrow 2)\, x * h_1 &= \begin{bmatrix} 0 & 0 & 0 & 0 & -0.177 & 0 & -0.177 \end{bmatrix}^T, \end{aligned}$$

is filtered by

$$\begin{aligned} g_0 &= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \\ g_1 &= \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \end{aligned}$$

and the resulting vectors added.

This simple example shows that it is possible to decompose a signal to obtain an orthogonal wavelet representation of the same length. It is also possible to reconstruct the original signal from such a wavelet representation. How are these filters related to wavelets? The next section proceeds to answer this question, after which Section 2.2.3 describes the filters involved in more detail.

2.2.2 Multi-resolution analysis

The discrete wavelet transform is intuitively described in terms of multi-resolution analysis. Mallat outlines this approach to wavelet theory in [Mal89], describing the very important interaction between multi-resolution signal approximation and that of conjugate digital filters (also see Section 2.2.3).

The resolution at which an image is viewed determines the attributes that

[4] Upsampling by 2, denoted by $(\uparrow 2)$, means inserting a zero after every element except the last.

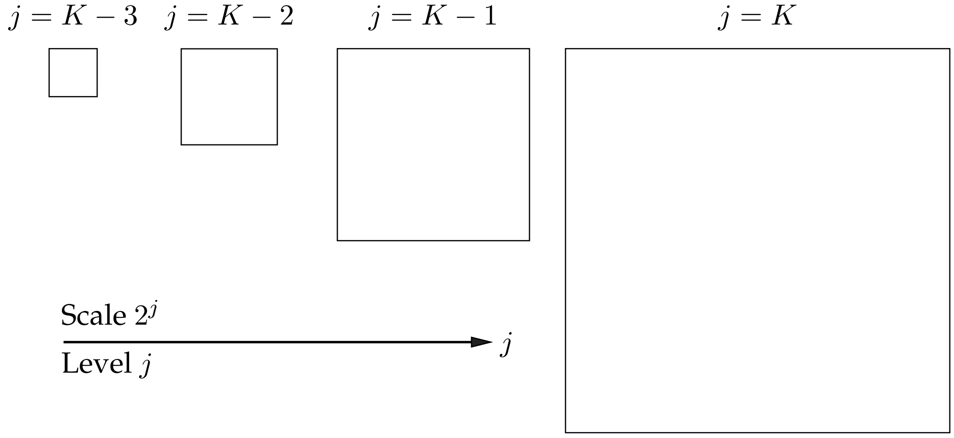


Figure 2.2.2: Levels and scales.

become visible. Examining a coarse version of an image emphasises large objects and slowly changing frequencies, whereas the finer detail becomes apparent only at higher resolutions. There are two main reasons why images are decomposed into different resolutions: to obtain invariance of distance, or to discriminate between objects of different sizes. The first, for example, occurs when an object needs to be recognised from images taken at different distances. The second, when trying to extract differently sized attributes from a fixed point-of-view.

In either scenario, it is necessary to artificially scale an image by projecting it to lower resolutions. A resolution step of 2 is chosen for ease of computation. Thus, an image at *scale* 2^K (also *level* K, see Figure 2.2.2) is projected to the scale 2^{K-1} , such that the image resolution is halved.

When viewing (continuous) signals at multiple resolutions, it is useful to think of a function space $\mathbf{V}_j$ that contains all signals at a certain level j. Consider a signal defined in the higher resolution space $\mathbf{V}_{j+1}$. A low resolution approximation of this signal at level j exists in the function space $\mathbf{V}_j$. But the low resolution approximated signal also exists in all higher resolution spaces, $\mathbf{V}_k$ where $k \geq j$. The spaces are nested, i.e.

$$\mathbf{V}_j \subset \mathbf{V}_{j+1}.$$

Bases must be found for these spaces in order to express signals at different levels. While not a requirement, we are interested in orthonormal bases which simplify the theory.

The difference between a signal approximation at level j and $j + 1$ is contained in a wavelet space, $\mathbf{W}_j$ (see Figure 2.2.3). Again, for simplicity, we

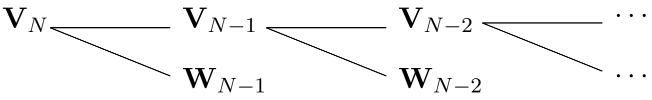


Figure 2.2.3: The decomposition of multi-resolution spaces.

Table 2.1: Properties of multi-resolution subspaces [Mal99, p. 221].

For all $j \in \mathbb{Z}$ and $k \in \mathbb{Z}$:

1	$f(x) \in \mathbf{V}_j \Leftrightarrow \ f(x - 2^{-j}k) \in \mathbf{V}_j$
2	$\mathbf{V}_j \subset \mathbf{V}_{j+1}$
3	$f(x) \in \mathbf{V}_j \Leftrightarrow f\,(2x) \in \mathbf{V}_{j+1}$
4	$\lim_{j \to -\infty} \mathbf{V}_j = \bigcap_{j=-\infty}^{+\infty} \mathbf{V}_j = \{0\}$
5	$\lim_{j \to +\infty} \mathbf{V}_j = \text{Closure}\left(\bigcup_{j=-\infty}^{+\infty} \mathbf{V}_j\right) = \mathbf{L}^2(\mathbb{R})$

There exists a scaling function $\phi(t)$ such that $\{\phi(t-n)\}_{n \in \mathbb{Z}}$ is an orthonormal basis for $\mathbf{V}_0$.

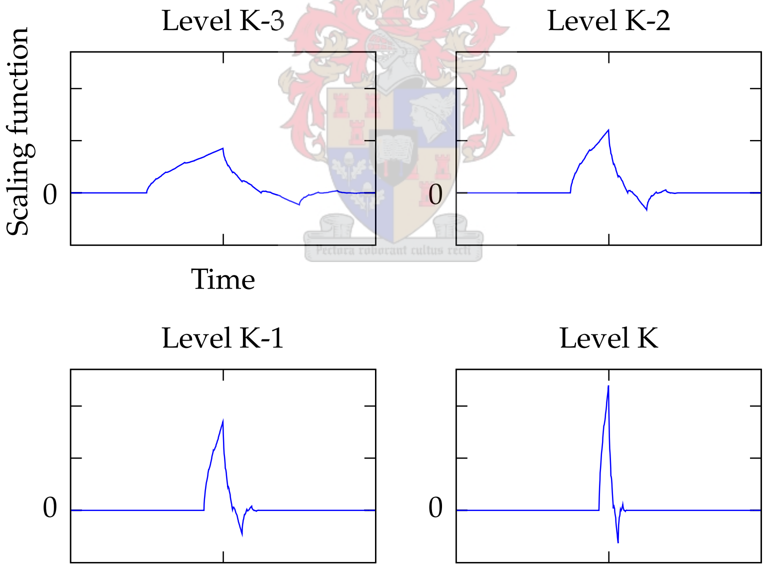


Figure 2.2.4: Dilation of the Daubechies 4 scaling function at different levels.

would like $\mathbf{W}_j$ to be the orthogonal complement of $\mathbf{V}_j$ in $\mathbf{V}_{j+1}$. This is useful in feature extraction, where wavelet coefficients of different levels should not carry the same information.

A complete list of properties required of $\mathbf{V}_j$ to be a multi-resolution space is given in Table 2.1.

We base the rest of the discussion on the following principles and assumptions:

- A multi-resolution space $\mathbf{V}_0$ exists, with orthonormal basis functions $\{\phi(t-n)\}_{n\in\mathbb{Z}}$ (see Table 2.1). As such, any function in $\mathbf{V}_0$ can be expressed as
$$f_0(t) = \sum_n a_0(n)\phi(t-n).$$
The dilated scaling function , $\phi(2t)$, exists in $\mathbf{V}_1$ and its shifts
$$\{\sqrt{2}\phi(2t-n)\}_{n\in\mathbb{Z}}$$
form an orthonormal basis for that space (see Table 2.1, properties 1 and 3 and Figure 2.2.4).

- The wavelet space $\mathbf{W}_0$ is the orthogonal complement of $\mathbf{V}_0$ in $\mathbf{V}_1$, thus $\mathbf{V}_0 \cap \mathbf{W}_0 = \{\varnothing\}$. The shifted wavelets $\psi(t-n)$ form a basis of $\mathbf{W}_0$, so that any function in the wavelet space can be expressed as
$$f_{W0}(t) = \sum_n b_0(n)\psi(t-n).$$

- Any function in $\mathbf{V}_1$ can be expressed as a linear combination of wavelet and scaling functions. We denote this with the notation $\mathbf{V}_1 = \mathbf{V}_0 \oplus \mathbf{W}_0$ (see Fig 2.2.5). Therefore,
$$f_1(t) = \sum_n a_0(n)\phi(t-n) + \sum_n b_0(n)\psi(t-n).$$

We first examine the relationship between the scaling functions at different levels. Any function that exists in $\mathbf{V}_0$ (including the scaling function $\phi(t)$), also exists in $\mathbf{V}_1$. Given coefficients $c(n)$, the scaling function can be expressed in terms of the basis functions of $\mathbf{V}_1$ (shifted, dilated scaling functions) as stated by the *dilation equation*

$$\phi(t) = \sqrt{2}\sum_n c(n)\phi(2t-n). \tag{2.2.5}$$

A shifted scaling function becomes

$$\begin{aligned}\phi(t-k) &= \sqrt{2}\sum_n c(n)\phi(2t-2k-n)\\ &= \sqrt{2}\sum_l c(l-2k)\phi(2t-l). \end{aligned}\tag{2.2.6}$$

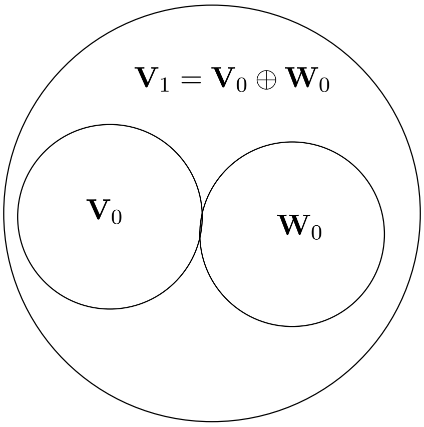


Figure 2.2.5: The relationship between scaling and wavelet spaces in multiresolution analysis.

This shows that any function in $\mathbf{V}_0$ can also be expressed in $\mathbf{V}_1$ in terms of the basis functions

$$\{\phi(2t-l)\}_{l\in\mathbb{Z}}.$$

The scaling function $\phi(t)$ is characterised by the coefficients $c(n)$, and can be found by applying the dilation equation recursively. It only converges for certain $c(n)$, but we will not discuss the conditions for convergence here (see [SN97, p. 240]). The wavelet can be found directly from the scaling function (not recursively[5]), using the coefficients $d(n)$. The wavelet is given by

$$\psi(t) = \sqrt{2}\sum_n d(n)\phi(2t-n),$$

where $d(n)$ is the alternating flip of $c(n)$,

$$(-1)^n c(N-n).$$

It can be shown that this choice of $d(n)$ ensures that $\mathbf{W}_j \perp \mathbf{V}_j$.

The coefficients $c(n)$ have a valuable property: they are orthogonal over double shifts [SN97, p. 182]. Multiply the dilation equation (2.2.5) by $\sqrt{2}\phi(2t-m)$ on both sides and integrate to obtain $c(m)$:

$$\begin{aligned}\int_{-\infty}^{+\infty}\phi(t)\phi(2t-m)dt &= \int_{-\infty}^{+\infty}\sum c(n)\phi(2t-n)\phi(2t-m)dt \\ &= c(m).\end{aligned}$$

[5] While the scaling function spaces are nested, the wavelet spaces are not. A wavelet in $\mathbf{W}_0$ is not present in $\mathbf{W}_1$, but rather exists in $\mathbf{V}_1$.

The product of $c(n)$ with any double shift $c(n-2m)$ is then

$$\sum_n c(n)c(n-2m) = \int_{-\infty}^{+\infty} \phi(t)\phi(2t-n)dt \int_{-\infty}^{+\infty} \phi(t)\phi(2t-n+2m)dt = \delta(m),$$

since the integrals both evaluate to one only when $m = 0$. This property proves to be useful later, when these coefficients are used in an orthogonal filter bank.

The part of a signal $f(t)$ that resides in $\mathbf{V}_j$ is expressed in terms of coefficients $a_j(n)$ as

$$f_j(t) = 2^{j/2} \sum_n a_j(n)\phi(2^j t - n).$$

The coefficients form the *discrete approximation* of f in $\mathbf{V}_j$. What is the connection between the coefficients $a_{j+1}(n)$ and $a_j(n)$? Consider $a_0(n)$, the coefficients which describe the orthogonal projection of $f(t)$ in $\mathbf{V}_0$. Here

$$\begin{aligned} a_0(n) &= \langle f(t), \phi(t-n)\rangle \\ &= \int_{-\infty}^{+\infty} f(t)\phi(t-n)dt, \end{aligned}$$

which, using (2.2.6), can be written as

$$\begin{aligned} a_0(n) &= \int_{-\infty}^{+\infty} f(t)\sqrt{2}\sum_l c(l-2n)\phi(2t-l)dt \\ &= \sqrt{2}\sum_l c(l-2n)\int_{-\infty}^{+\infty} f(t)\phi(2t-l)dt \\ &= \sqrt{2}\sum_l c(l-2n)\left\langle f(t), \phi(2t-l)\right\rangle . \end{aligned}$$

Since the inner product $\langle f(t), \phi(2t-l)\rangle$ gives the coefficients $a_1(l)$,

$$a_0(n) = \sum_l \sqrt{2}c(l-2n)a_1(l).$$

This is a very important result. It shows that the scaling function coefficients at level $j-1$ are related to those at level j by the coefficients $c(l)$. In fact, the double shifts of $c(l)$ are used, which were shown earlier to be orthogonal. If we define a filter $h_0(n)$ as

$$h_0(n) = \sqrt{2}c(N-n),$$

(where N is the length of c) the coefficients $a_0(n)$ can be written as the convolution

$$a_0(n) = (a_1 * h_0)(2n) = \sum_{n=-\infty}^{+\infty} a_1(l)h_0(2n-l).$$

The coefficients are obtained by filtering $a_1(l)$ with $h_0(l)$ and then downsampling the resulting vector by 2. Once a continuous signal has been written in terms of coefficients $a_K(n)$ at level K, the coefficients for any lower level can be calculated by filtering operations.

In practice, we often work with sampled signals. The continuous signal $f(t)$ is never seen, but the coefficients $f(nT)$ (where T is the sampling interval) can be used as $a_K(n)$. Strictly speaking, this approach is incorrect, since it assumes an underlying function

$$f(t) = 2^{K/2} \sum_n f(nT)\phi(2^K t - n).$$

Strang discusses this "wavelet crime" in [SN97, p. 232] and recommends that samples $f(nT)$ be pre-filtered before being used as coefficients. Still, this is seldom done in practice — it is much easier to simply use the sampled values directly as coefficients.

In the same way that $a_0(n)$ is related to $a_1(n)$, the wavelet coefficients $b_0(n)$ are related to $b_1(n)$ by

$$b_0(n) = (a_1 * h_1)(2n) = \sum_{n=-\infty}^{+\infty} b_1(l)h_1(2n-l)$$

where h_1 is a filter of length L and is the alternating flip of h_0,

$$h_1(l) = (-1)^l h_0(L-l).$$

The discrete signal $a_1(n)$ can now be reconstructed from the scaling and wavelet coefficients at level 0. It is

$$a_1(l) = \sum_n a_0(n)c(l-2n) + \sum_n b_0(n)d(l-2n).$$

This is clear from the reconstruction of $f_1(t)$:

$$\begin{aligned} f_1(t) &= \sum_n a_0(n)\phi(t-n) + \sum_n b_0(n)\psi(t-n) \\ &= \sum_n a_0(n)\sum_l \sqrt{2}c(l)\phi(2t-l) + \sum_n b_0(n)\sum_l \sqrt{2}d(l)\phi(2t-l) \\ &= \sqrt{2}\sum_l \left[\sum_n a_0(n)c(l-2n) + \sum_n a_0(n)d(l-2n)\right]\phi(2t-l) \\ &= \sqrt{2}\sum_l a_1(l)\phi(2t-l). \end{aligned}$$

Summary

We discussed the coefficients c and d which link the function spaces $\mathbf{V}_0$ and $\mathbf{V}_1$, or, in general, $\mathbf{V}_j$ and $\mathbf{V}_{j+1}$. It was shown that $a_1(n)$ can be decomposed into the scaling and wavelet coefficients $a_0(n)$ and $b_0(n)$, and that these coefficients can be combined to reconstruct $a_1(n)$. Generally, a signal has a discrete approximation $a_K(n)$ at level K. The approximation $a_{K-1}(n)$ at a lower level $K-1$ can be obtained by simply filtering these coefficients. The wavelet coefficients $b_j(n)$ can be found in a similar manner, with a different filter. The coefficients $a_K(n)$ at level K can be reconstructed from $a_{K-1}(n)$ and $b_{K-1}(n)$.

2.2.3 Sub-band coding and filter banks

As shown in Section 2.2.2, the wavelet decomposition can be implemented by means of filter banks. These filter banks implement a technique known as *sub-band coding*, whereby a signal is split into different frequency bands before being coded. Normally data signals have more energy in the lower frequency bands (or channels). Coding the channels separately allows for more accurate quantisation in these regions [PM96]. Further, the bands can be examined individually to extract features.

The signal spectrum can be split in different ways, some more convenient than others. Any such division of the spectrum must be complete: the recombination of all channels must be able to yield the original signal. Typically, the spectrum is split equally into a low- and high-pass band. The splitting process is then repeated recursively for the low-pass band, creating channels of rapidly decreasing bandwidth. This scheme is shown in Figure 2.2.6 for an ideal filter. In reality, there is always overlap between the low and high-pass filters, as shown in Figure 2.2.7. Since the bandwidth is halved with every split, it is possible to downsample the resulting signals by 2 without losing any informa-

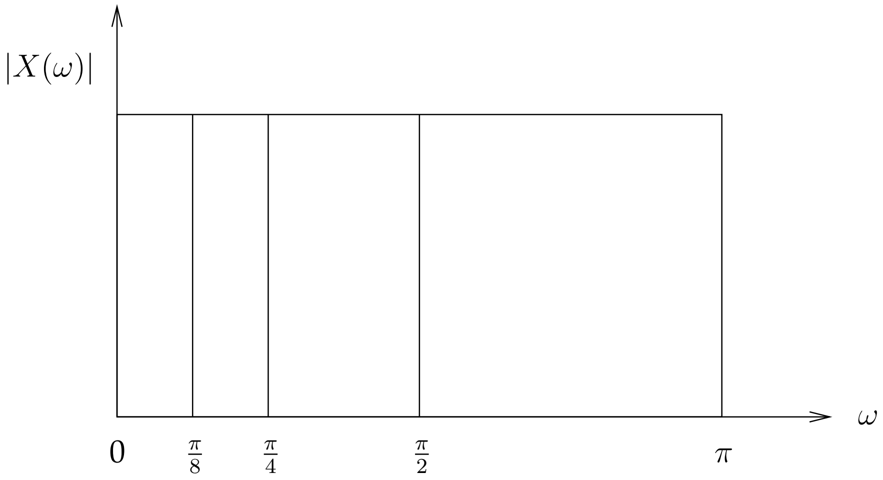


Figure 2.2.6: An example of an ideal spectrum partitioning.

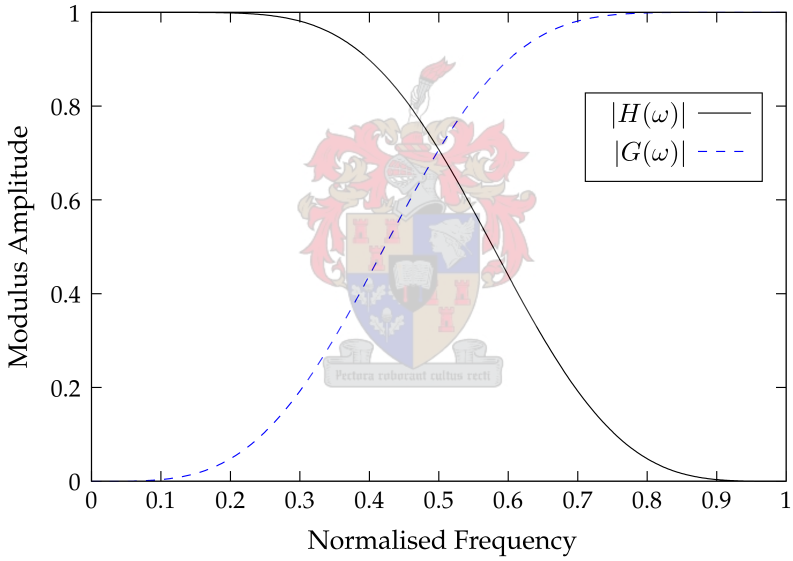


Figure 2.2.7: Frequency response: Daubechies 8-tap low and high-pass filters.

tion. After downsampling, the combined length of the filter output signals is the same as that of the original signal — the signal is still *critically sampled*.

Figure 2.2.9 illustrates the concept underlying filter banks [SN97]. The filter $h_0(n)$ is a low-pass and $h_1(n)$ a high-pass filter. A perfect reconstruction of x can be obtained if the filter coefficients h_0 and h_1 are carefully chosen. Next, we examine the signal as it moves through the filter bank.

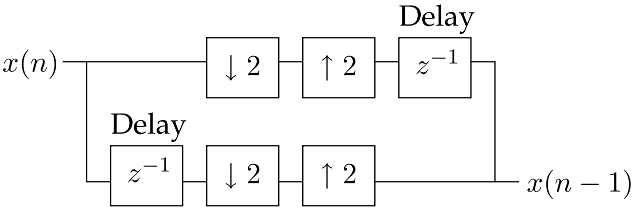


Figure 2.2.8: Reconstruction from downsampled coefficients [SN97, p. 95].

Downsampling and upsampling in the frequency domain

The downsampling operator selects all even elements from a sequence $x(n)$. The upsampling operator inserts a 0 after each element, except the last. Downsampling and then upsampling is equivalent to a point-wise multiplication by the Dirac-comb,

$$\delta_p = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \end{bmatrix}$$

or equivalently

$$\delta_p(n) = \frac{1}{2}\left(1 + (-1)^n\right).$$

The discrete Fourier transform of $u(n) = (\uparrow 2)(\downarrow 2)x(n) = \delta_p(n)x(n)$ is

$$\begin{aligned} U(\omega) &= \sum_n \frac{1}{2}\left(1 + (-1)^n\right) x(n)e^{-jn\omega} \\ &= \frac{1}{2}\left[\sum_n x(n)e^{-jn\omega} + \sum_n x(n)e^{-jn(w+\pi)}\right] \\ &= \frac{1}{2}\left[X(\omega) + X(\omega + \pi)\right]. \end{aligned} \tag{2.2.7}$$

When $X(\omega)$ and $X(\omega+\pi)$ overlap, *aliasing* takes place. By downsampling, we are effectively halving the sampling rate, which destroys the original signal. It is still sometimes possible to combine two such downsampled signals to obtain the original, as shown in Figure 2.2.8.

The z-transform is defined as

$$X(z) = \sum_{n=-\infty}^{+\infty} x(n)z^{-n}.$$

The equivalent of down- and upsampling in the z-domain (derived directly from (2.2.7) and Table 2.2) is

$$U(z) = \frac{1}{2}\left(X(z) + X(-z)\right),$$

Table 2.2: Properties of the z-transform [PM96].

Property	**Time Domain**	**z-Domain**
Notation	$x[n]$	$X(z)$
Time shifting	$x[n-k]$	$z^{-k}X(z)$
Scaling in the z-domain	$a^n x[n]$	$X(a^{-1}z)$
Time reversal	$x[-n]$	$X(z^{-1})$

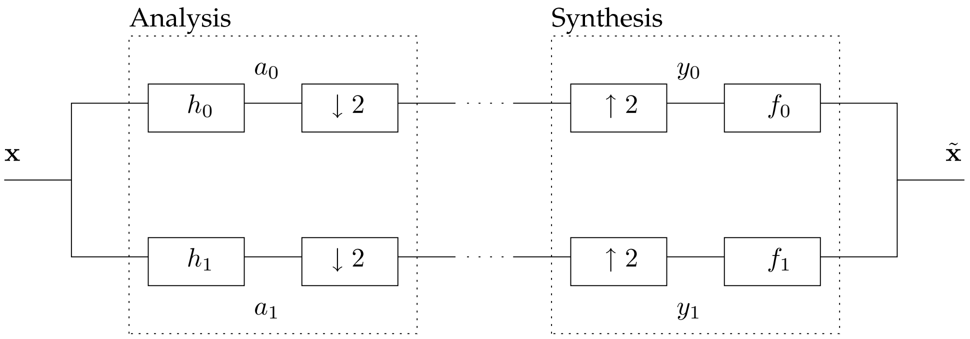


Figure 2.2.9: Filter bank analysis and synthesis.

where $X(-z)$ is the aliasing term.

The road to perfect reconstruction

During the analysis step (see Figure 2.2.9), the signal is filtered by h_0 and h_1 to obtain

$$A_0(z) = H_0(z)X(z) \quad \text{and} \quad A_1(z) = H_1(z)X(z).$$

The last step in analysis is downsampling, followed by the first step of synthesis: upsampling. The results are

$$\begin{aligned} Y_0(z) &= \frac{1}{2}(H_0(z)X(z) + H_0(-z)X(-z)) \\ Y_1(z) &= \frac{1}{2}(H_1(z)X(z) + H_1(-z)X(-z)). \end{aligned}$$

Finally, y_0 and y_1 are filtered by f_0 and f_1. The results are combined in an attempt to reconstruct x as

$$\begin{aligned}\tilde{X}(z) &= F_0(z)Y_0(z) + F_1(z)Y_1(z) \\ &= \frac{1}{2}[F_0(z)H_0(z) + F_1(z)H_1(z)]X(z) + \\ &\quad \frac{1}{2}[F_0(z)H_0(-z) + F_1(z)H_1(-z)]X(-z).\end{aligned}$$

There are two conditions for perfect reconstruction. The first is that the *aliasing term* must be zero,

$$F_0(z)H_0(-z) + F_1(z)H_1(-z) = 0.$$

The second is that the filter bank must introduce *no distortion* — the output must be a delayed version of the input [VH92], in other words,

$$\tilde{X}(z) = z^{-l}X(z),$$

which implies that

$$F_0(z)H_0(z) + F_1(z)H_1(z) = 2z^{-l}.$$

Designing filters for perfect reconstruction

It is easy to take care of aliasing by choosing

$$\begin{aligned}F_0(z) &= H_1(-z) \\ F_1(z) &= -H_0(-z).\end{aligned}$$

The previous section also hinted at the fact that some high-pass filters can be derived from their low-pass counterparts. A common choice is the alternating flip

$$H_1(z) = -z^{-N}H_0(-z^{-1}),$$

or (see also Table 2.2)

$$h_1(k) = (-1)^k h_0(N-k).$$

It can then be shown that perfect reconstruction is possible when the product filter $P(z) = H_0(z^{-1})H_0(z)$ is a "halfband filter" [SN97, p. 107], in other words

$$P(z) + P(-z) = 2.$$

It is easy to show that, if h_0 is a low-pass filter, then h_1 is a high-pass filter. Examine the amplitude of the frequency response of h_1, which is

$$\begin{aligned}
|H_1(\omega)| &= \left|\sum_k (-1)^k h_0(N-k)e^{-jk\omega}\right| \\
&= \left|\sum_k h_0(N-k)e^{-jk(\omega+\pi)}\right| \\
&= \left|\sum_n h_0(n)e^{-j(N-n)(\omega+\pi)}\right| \\
&= |H_0(-\omega-\pi)| = |H_0(\omega+\pi)|\,.
\end{aligned}$$

This shows that the response of h_1 is simply a mirrored version of that of h_0. If h_0 is a low-pass filter, then h_1 is a high-pass filter.

Further techniques involved in filter design are described in [SN97].

2.2.4 Fitting the parts together

There are many paths to understanding wavelets and while this makes it accessible to many disciplines, it also makes it easy to get lost. The previous sections attempted to provide an answer to a basic question: how do multi-resolution analysis and filter banks tie together to form a foundation for discrete wavelet analysis?

The low-pass filter h_0 is chosen to have certain properties and to satisfy certain conditions, described in the literature mentioned. When satisfying these conditions, it can be used to generate scaling functions. From h_0 we derive the high-pass filter h_1, which generates the wavelet functions. These functions are the basic building blocks of function spaces in multi-resolution analysis. The principles by which the filters are chosen are determined by the theory of conjugate mirror filters; the filter banks that combine them lead to perfect reconstruction.

Wavelets in two dimensions

The signals used in this thesis are mostly gray-level images: two-dimensional signals of light intensity. In order to apply the wavelet transform to these signals, we must know how the one-dimensional wavelet transform is extended to two dimensions. For a transform in two dimensions, two-dimensional wavelets and scaling functions are needed.

For simplicity, it would be ideal to rewrite the two-dimensional transform in terms of the one dimensional transform. An example of such a *separable transform* is the two-dimensional fast Fourier transform (FFT), which is calculated by simply applying the one dimensional FFT to the rows and then to the columns of an image. It turns out that the same can be done for the wavelet transform, given that the scaling functions and wavelets form an orthonormal basis of $\mathbf{L}^2(\mathbb{R})$.

Given such a scaling function $\phi(t)$ and a wavelet $\psi(t)$, the separable, two-dimensional scaling function is

$$\phi(x, y) = \phi(x)\phi(y)$$

and the three wavelets are the tensor products [Mal99, A.5]

$$\begin{aligned} \psi_V(x, y) &= \psi(x)\phi(y) \\ \psi_H(x, y) &= \phi(x)\psi(y) \\ \psi_D(x, y) &= \psi(x)\psi(y). \end{aligned}$$

The subscripts H, V and D refer to the horizontal, vertical and diagonal directions along which these wavelets measure variation [GW01, p. 386]. This can be seen by examining the wavelets, shown in Figure 2.2.10. Notice how each wavelet has a peak and a dip in the specified direction. When used as a filter, this combination is essentially a subtraction and has a high-pass effect. The scaling function, on the other hand, does not have a dip —- it leads to smoothing in the horizontal and vertical directions. The Fourier transform of 2-dimensional Daubechies wavelets are shown in Figure 2.2.11. Note that the low-frequency area of the frequency-plane is covered by the scaling function. Also note the orientation of the areas covered by the respective wavelets.

The discrete (orthogonal) wavelet transform is implemented by the filter bank shown in Figure 2.2.12. The input image at level j is decomposed into its approximation at level $j-1$ ($\mathbf{A}_{j-1}$), and its horizontal, vertical and diagonal wavelet components, $\mathbf{D}^H_{j-1}$, $\mathbf{D}^V_{j-1}$ and $\mathbf{D}^D_{j-1}$.

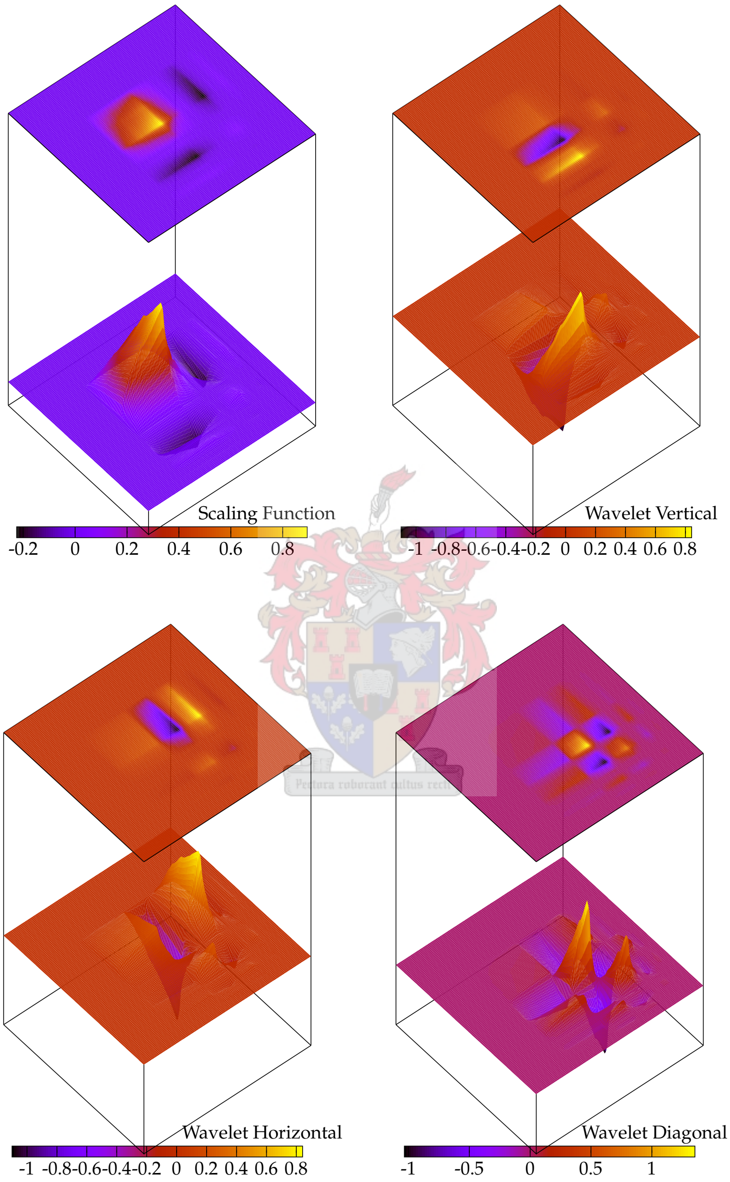
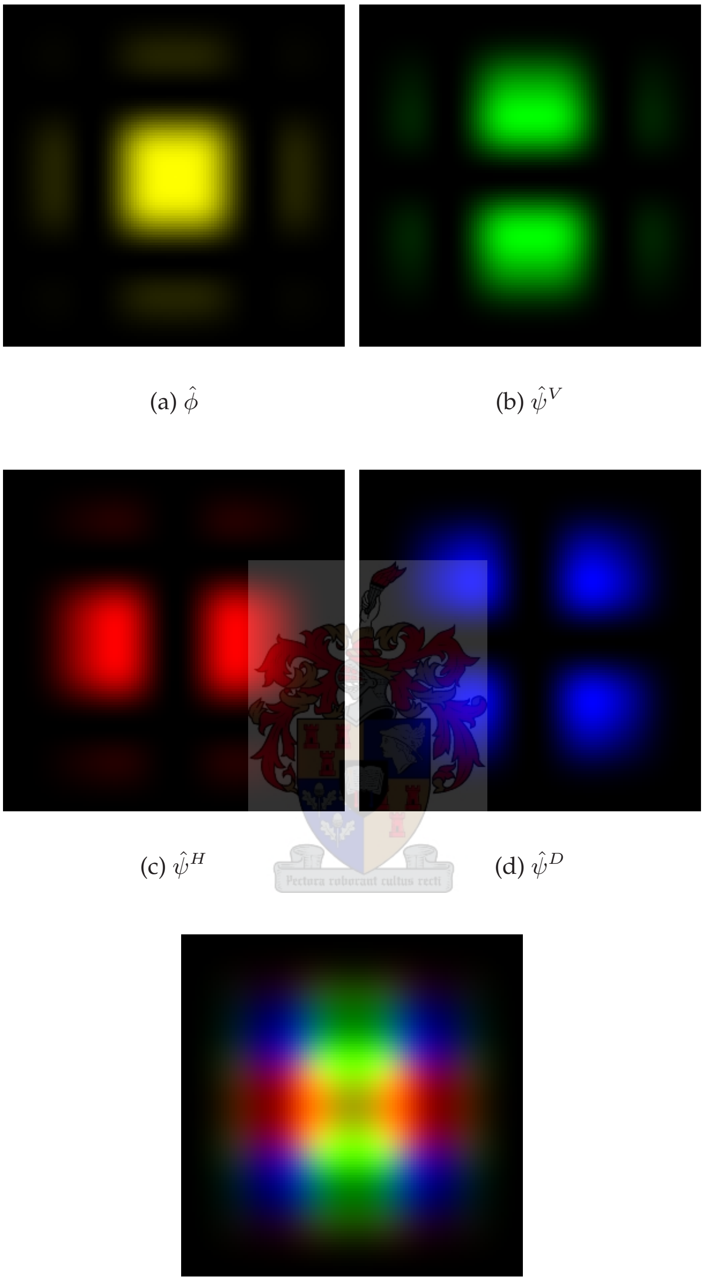


Figure 2.2.10: 2-D Daubechies (4 coefficient) scaling function and wavelets.



(a) $\hat{\phi}$

(b) $\hat{\psi}^V$

(c) $\hat{\psi}^H$

(d) $\hat{\psi}^D$

(e) Combined

Figure 2.2.11: The 2D Daubechies (4 coefficient) scaling function and wavelets in the frequency domain.

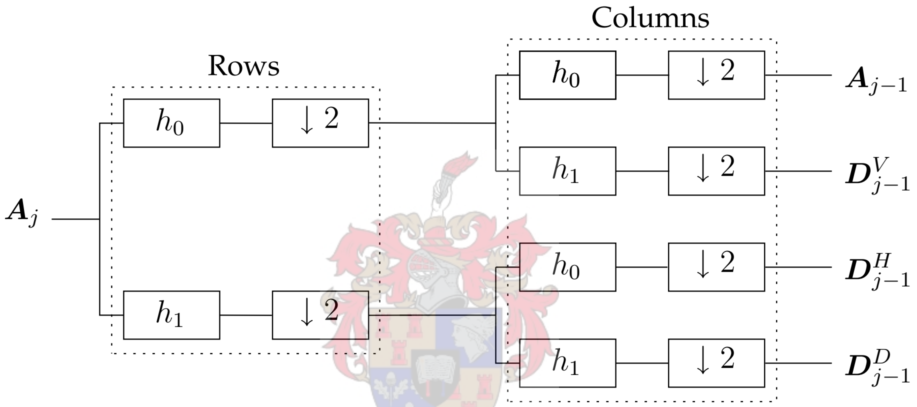


Figure 2.2.12: Filter bank of the separable 2D wavelet transform.

Note that the filters are applied either across rows or columns, as indicated. The high-pass filter h_1 emphasises edges in the direction applied, hence the “vertical”, “horizontal” and “diagonal” wavelet coefficients.

Chapter 3

Texture Analysis

3.1 Introduction

While texture is an intuitive concept, it is not easy to provide a formal definition[1]. The human concept of texture is closely linked to the sensations of touch and sight. Touching an object provides a sense of its physical smoothness, while the patterns and colours on the surface can only be observed visually. When processing photographs, there is no direct information available about the smoothness of the photographed object (although it sometimes can be related to or deduced from visible attributes) and the observed patterns must be used to describe the texture.

Methods for characterising texture fall into three categories: statistical (spatial), structural and spectral methods [GW01, p. 665]. Structural methods are especially useful for texture synthesis but require a model of the underlying structure. For rock types, no simple structural models are available, or can be built from the photo, which renders this class of methods less useful. Statistical and spectral methods are later described in more detail (see Section 3.2), as well as the wavelet transform (Section 3.3) which belongs to both the spatial and the spectral categories.

In order to classify an unknown texture, we must have prior knowledge of known texture classes. This information is used to train a *classifier*, which is then able to assign textures, with a certain accuracy, to known classes. A simple and effective classifier is the Bayesian classifier, which is described in Section 3.4.2.

[1] Attempts have been made. “An image region has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying or approximately periodic. One may view a textured object as a structure constructed by a machine, or an algorithm, whose major parameters remain approximately constant or vary slowly throughout the process of construction.” [Skl78]

3.2 Texture representation

3.2.1 Statistical

The variance of image intensity values can be linked intuitively to the underlying texture. It indicates the spread of data around the mean — a high variance (many values differing notably from the mean) suggests a coarse texture. Variance is but one of a range of moments used to describe texture. The n^{th} central moment for an image Z with gray-levels $0 \ldots L-1$ is defined as

$$\mu_n = \mathsf{E}\left[(Z-\bar{Z})^n\right] = \sum_{l=0}^{L-1}(l-\bar{Z})^n f_Z(l)$$

where

$$\bar{Z} = \sum_{l=0}^{L-1} l\, f_Z(l)$$

and $f_Z(l)$ is the probability of gray level l occurring in Z [PZP01]. For discrete signals, $f_Z(z)$ is a normalised histogram. The n^{th} moment can be estimated in terms of the image values, $Z(x,y)$, as

$$\mu_n = \frac{1}{M \times N}\sum_{x=0}^{M}\sum_{y=0}^{N}\left[Z(x,y)-\bar{Z}\right]^n$$

where

$$\bar{Z} = \frac{1}{M \times N}\sum_{x=0}^{M}\sum_{y=0}^{N} Z(x,y)$$

and M and N are the number of rows and columns respectively.

Table 3.1 shows other often used statistical texture descriptors [GW01].

Gray level co-occurrence matrices

Another popular method of characterising texture examines the statistical properties of the gray-level co-occurrence matrix. This 4-dimensional histogram, $P = f(i,j,d,\theta)$, describes the number of times that the gray level i occurs at a distance d and at an angle θ in relation to the gray level j. Different statistics of the co-occurrence matrix can be calculated as discussed in [NS93].

Due to computational complexity, this method is ineffective when examining large images. For any histogram P of D independent dimensions, the

Table 3.1: Statistical texture descriptors.

Moment about origin	$m_n = \sum_{x=0}^{L-1} x^n f_X(x)$
Central moment	$\mu_n = \sum_{x=0}^{L-1} (x - \bar{X})^n f_X(x)$
Mean	$\bar{X} = m_1$
Variance	$\sigma^2 = \mu_2$
Roughness	$R = 1 - \frac{1}{1+\sigma^2}$
Uniformity	$U = \sum_{x=0}^{L-1} f_X^2(x)$
Entropy [bits]	$e = -\sum_{x=0}^{L-1} f_X(x) \log_2 f_X(x)$

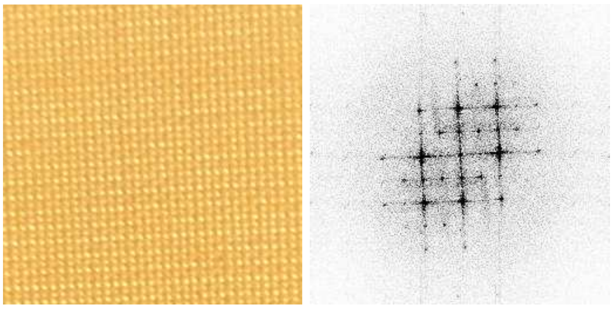
minimum number of calculations required to produce P is

$$S_{min} = \sum P = \sum_{d_1} \sum_{d_2} \cdots \sum_{d_D} f(d_1, d_2, \ldots, d_D),$$

which is the number of elements represented — each of which must have been examined at least once. If θ is limited to K directions so that $\theta \in \{n\pi/K \mid n = 0 \ldots K-1\}$, d is limited to D pixels and the gray levels are quantised to L levels, the size of the co-occurrence matrix is L^2DK. For each pixel in the image, the surrounding area of radius D is examined at the angles in θ, leading, for an image containing N pixels, to $S_{min} \approx NDK$ calculations. If D is much smaller than N, the computational complexity of the algorithm is $O(N)$, whereas if D is almost as large as N, it is $O(N^2)$.

If D is limited to a small number, the co-occurrence matrix describes local texture — a concession that unfortunately has to be made in order to be able to calculate the histogram efficiently.

Statistics of the co-occurrence matrix: Features are generated by calculating different statistical properties of the co-occurrence matrix [GW01, p. 669]. Haralick et al. [HSD73] proposed 14 such features, but it has been noted [HSL92] and experienced that not all of these statistics are appropriate for natural texture discrimination. It seems that, in the presence of noise, the co-occurrence



(a) Curtain (b) Fourier transform

Figure 3.2.1: Fourier transform (inverted) of a quasi-periodic texture.

matrix is, in fact, a very unreliable source.

3.2.2 Spectral

Spectral representations describe the frequency contents of a texture image. This is especially useful when the texture is semi-periodic, i.e. has repetitive surface patterns. Localised spatial methods struggle to find these spread patterns which deliver such strong responses in the frequency domain (see Figure 3.2.1).

The two-dimensional Fourier transform of an $M \times N$ image, $f(x, y)$, is

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)}.$$

The two-dimensional frequency plane is shown in Figure 3.2.2 along with two paths, C_θ and C_R. When C_θ is followed, the frequency remains constant. An annulus placed around this path would describe a certain frequency band, and a pie-slice placed around C_R describes frequency components that lie in a certain range of directions.

In some applications, for example, rotation invariance is required. A classic technique, then, is to divide the frequency plane into several annuli shaped around C_θ at different offsets. The frequency bands contained within encompass all angles and statistics calculated on the area are rotation invariant. Note that by using these annuli the image is effectively filtered using an ideal bandpass filter in an attempt to focus on certain frequency bands — a process which resembles a filter bank, only with the different bands spaced linearly and with

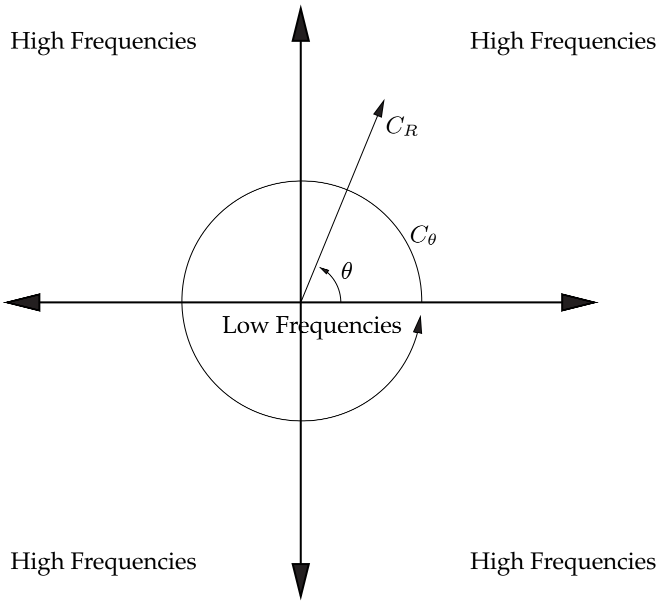


Figure 3.2.2: The 2-dimensional frequency plane.

over-simplified filters[2]. There is a trade-off here: if the annuli are constant in area, their bandwidths differ, while if their bandwidths are constant, the number of coefficients brought into calculation differs.

Figure 3.2.3 shows the outline of the largest annulus that fits inside a square image. Roughly 21% of the image remains unused, which can become a problem when working with small images, where every pixel is valuable. It is possible to work around the problem by using all the coefficients outside of the outer annulus as an extra frequency band, but this solution is less than optimal, as the frequency range in this band would be larger than those covered by the other annuli.

3.3 Wavelet methods

3.3.1 Why wavelets?

When examining an image purely in the frequency domain, all localisation in the time domain is lost. There is no way to determine *where* certain frequencies occur in an image. For regularly spaced (synthetic) textures this does not pose a problem, since the frequency content is constant over the whole image. That is certainly not true for natural texture images!

[2] A filter with a very sharp cutoff in the frequency domain causes severe rippling in the time domain.

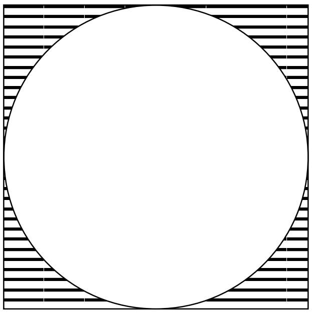


Figure 3.2.3: Unused area in rotation invariant frequency description.

Section 3.2.2 shows how the frequency information in an image can be quantified, by examining annuli that cover different frequency bands. This task becomes much simpler and more flexible in the wavelet domain. Here, the frequency bands are available at the different decomposition levels[3], without any need of artificial separation.

Each level of the decomposition contains unique information (since $\mathbf{W}_j$ is the orthogonal complement of $\mathbf{V}_j$ in $\mathbf{V}_{j+1}$), aiding in the creation of uncorrelated feature vectors. Because of the compact support of the wavelets, they are well adapted to describing finite signals — like images.

All in all, wavelet analysis fits this specific application like a glove.

Why not Gabor filters?

Gabor filters are finely adjustable. In order to apply these filters, detailed knowledge of the underlying problem is needed, so that the optimal filter parameters can be calculated. In fact, certain parameters cause the Gabor filters to become a wavelet family. Unfortunately, there is no easy way to determine the best parameters for a specific problem. It is also not easy to intuitively see how small changes in the problem should influence the choice of parameters [PS95].

The filters are also highly orientation specific. This is a useful attribute when discriminating between synthetic textures, but natural textures are not as sensitive to direction. Indeed, some authors claim that Gabor filters are better than wavelets for the purpose of texture discrimination [AM03], but these trials compare the dyadic wavelet transform to a frame-like Gabor wavelet approach, which also has a much higher computational cost. The outputs of the Gabor filter banks are not mutually orthogonal, which may result in correla-

[3] This analogy is not strictly accurate, due to the overlap of the low- and band-pass filters.

tion between texture features [Uns95].

The dyadic wavelet transform severely restricts the choice of filter parameters. Rather than adjusting these parameters, emphasis is placed on the choice of wavelet family, which should correspond to the underlying problem.

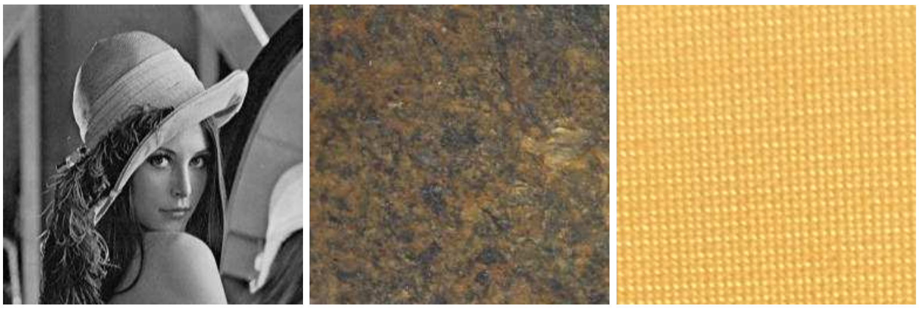
3.3.2 Choosing the right basis

A signal can be seen as being constructed of fundamental building blocks: short, finitely supported signals. The correct combination of the right building blocks assures a compact signal representation. The "best" blocks differ for each and every signal and their exact shapes are unknown. We want to find a wavelet family that closely resembles these elementary units.

The wavelet family that leads to the smallest fine-scale wavelet coefficients is considered the best representation, having concentrated the signal energy in as few levels possible. The regularity of the signal influences this, as does the number of vanishing moments and length of support of the wavelet [Mal99, p. 254].

Jawerth and Sweldens [JS94] list the following important wavelet properties to consider:

- **Orthogonality** leads to much simplified algorithms.
- **Compact support** wavelets are associated with finite impulse response filters.
- **Dyadic coefficients** lead to very fast hardware implementations.
- **Symmetry** allows for linear phase filters.
- The **smoothness** of the wavelet is important when compressing images, or when the signal is highly regular. Smooth functions lead to better frequency localisation.
- The **number of vanishing moments** dictate the degree of polynomials that can be approximated at any level.
- **Analytic expressions** of the scaling function and wavelets are useful in theoretic and simulated experiments.
- If the wavelets **interpolate** data on a grid, none of the original data points are lost.



(a) Lena (b) rock (c) Curtain

Figure 3.3.1: Test Images

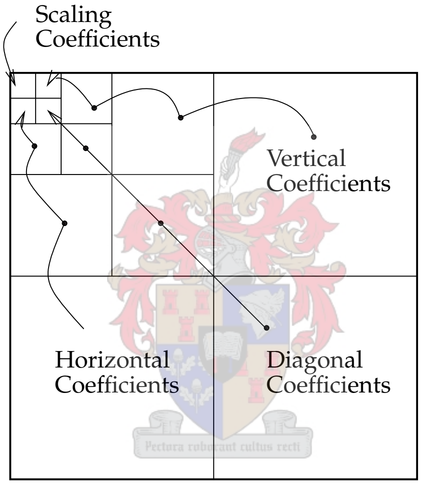


Figure 3.3.2: Pyramid-structured partitioning

All wavelets have different properties — the goal is to find a family with a set of properties best suited to the application. Villasenor, Belzer and Liao [VBL95] develop several filters and present ways to evaluate them. After considering the criteria above and the analysis given by Villasenor et al., we choose their "Filter Number 3" for texture analysis.

3.3.3 Wavelet packet bases

The standard pyramid-structured wavelet transform decomposes a signal into a series of frequency channels (see Figure 3.3.2). The channels associated with lower frequencies have smaller bandwidths, allowing higher frequency resolution. This is only an effective decomposition when the bulk of the signal en-

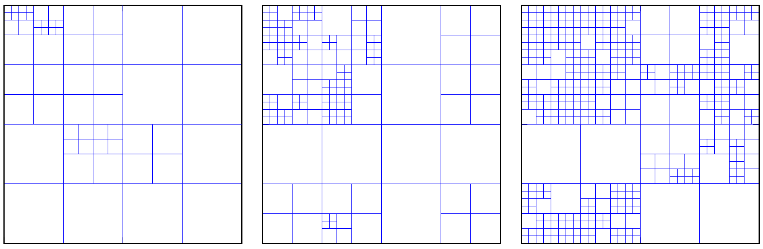


Figure 3.3.3: Adaptive Wavelet Packet Partitioning for Lena, Rock and Curtain

ergy is concentrated in the lowest frequency bands, which is indeed the case with most images. Texture images, however, tend to be quasi-periodic with the information concentrated in the middle-to-low frequency bands [CK93].

Coifman, Meyer and Wickerhauser [RBC+92] proposed a more flexible decomposition where partitioning is done according to the significance of a channel. This decomposition, known as the wavelet packet transform, allows for the iterative filtering of the detail coefficients and creates new partitions similar to those shown in Figure 3.3.3 [GW01, p. 396]. The most intuitive way to determine the significance of a channel is to calculate the l^2-norm signal[4] energy, i.e.

$$e(\mathbf{x}) = \frac{1}{N}\|\mathbf{x}\|^2 = \frac{1}{N}\sum_{i=1}^{N}|x_i|^2,$$

where N is the length of $\mathbf{x}$. If a channel contains much less energy than others in the same level, it does not need to be decomposed any further.

The wavelet packet transform comes at a slight cost — for a signal of length N it has an order complexity of $O(N \log N)$ compared to the $O(N)$ for the fast wavelet transform.

Figure 3.3.1 shows three test images (256×256 pixels in size): the first represents a typical photo (*Lena*), the second a rocky texture (*Rock*) and the third a quasi-periodic texture (*Curtain*). Is the rocky texture also periodic to some extent? The answer is hidden in the standard wavelet decomposition. Examine the wavelet coefficients shown in Figure 3.3.4 — the inside bands of *Curtain* respond strongly, indicating the presence of middle frequencies. Whether this also occurs in *Rock*, however, is still hard to see since there are no strong edges for the eye to follow.

To further investigate how much information the middle-level coefficients carry, the image is reconstructed after setting all other levels to 0 as indicated

[4] If we are only interested in the element-values of an $M \times N$ image, and not in their positions, we also refer to the image as a signal $f(n)$, $0 < n < MN - 1$.

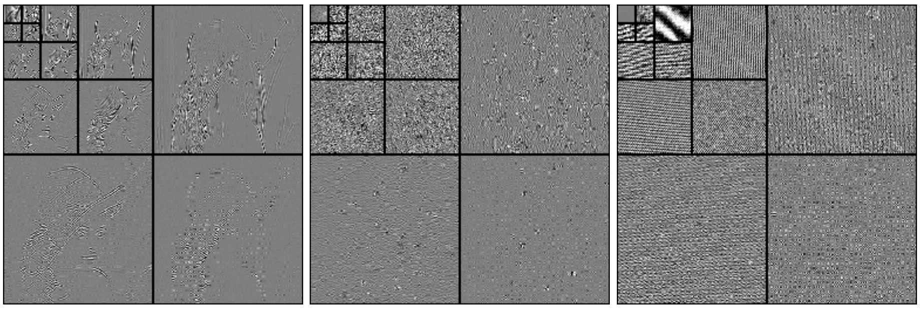


Figure 3.3.4: 4-Level wavelet decompositions of *Lena*, *Rock* and *Curtain*.

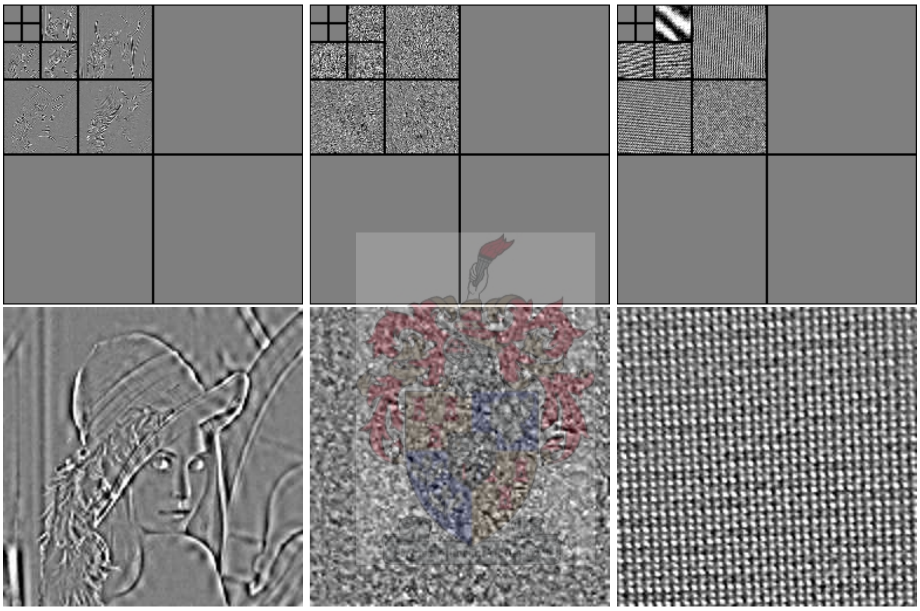


Figure 3.3.5: Inverse wavelet transform after muting channels

in Figure 3.3.5. *Curtain* clearly resembles the original closely with *Lena* looking badly out of shape. The quality of *Roc*k is hard to judge, but appears to be somewhere between that of *Lena* and *Curtain*.

In an effort to explain this behaviour, a cumulative energy graph is shown in Figure 3.3.6 (note that the starting energy percentages differ — we are interested only in the *shape* of the energy curve). This graph shows how the energies of the different levels add to form the complete signal. The curves suggest a different distribution in each image. *Lena* has a very strong lower frequency response, while *Curtain* has more energy distributed through the middle frequencies. *Rock*, on the other hand, has energy almost linearly distributed in both bands.

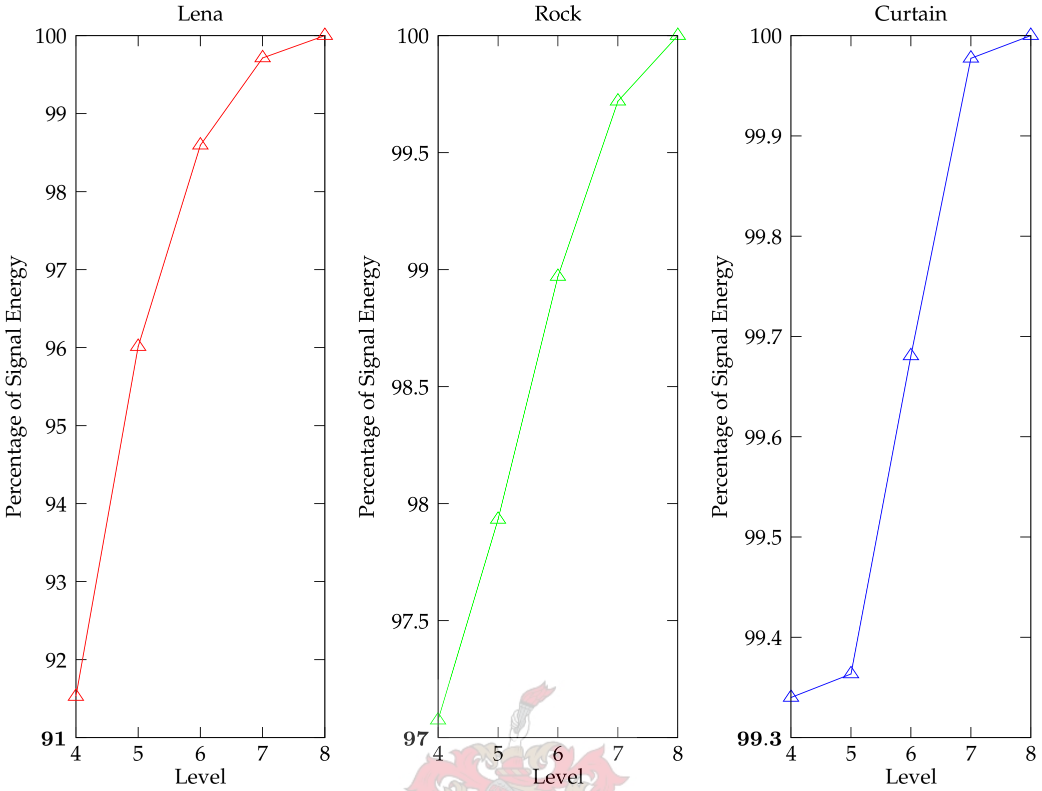


Figure 3.3.6: Cumulative level energies of three wavelet transforms.

It should be noted that high levels of signal energy do not necessarily imply the presence of useful information. In *Curtain*, the highest energies occur in the lowest level (more than 99% of the total signal energy), but this level (with its low-pass coefficients) contains almost *no* descriptive information about the texture itself!

Examine the partitioning of the *adaptive wavelet packet* decompositions (Symmlet 8), as shown in Figure 3.3.3. Densely partitioned portions indicate pockets of concentrated energy and intuitively confirms the frequency distributions discussed. *Lena* has an almost pyramidal decomposition, while *Curtain* is nearly fully decomposed.

[CK93] rightly claims that the average synthetic texture contains much information in the middle bands, visible as regularly spaced patterns as is often caused by synthetic fabrication. These *quasi-periodic* signals are therefore not perfectly suited to the standard pyramidal wavelet transform. It is then interesting to see that the typical rocky texture investigated here is not quasi-periodic, but contains a lot of information in the middle *as well as* the lower frequency bands. It is therefore well suited to the standard wavelet partitioning.

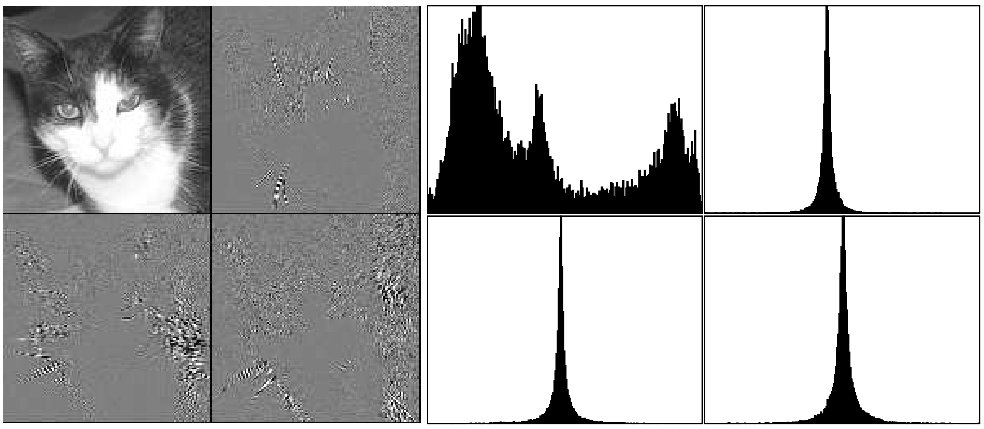


Figure 3.3.7: Wavelet transform and level histograms of a typical image.

The histograms have been scaled — their shapes and not the exact values they represent are important here. The peaks of the wavelet coefficients visible in the histograms are centred near zero.

3.3.4 Features from wavelet coefficients

A signal is constructed out of fundamental building blocks, or wavelets. The wavelet coefficients form the building plans, showing the amplitude and position of each wavelet. We would like to characterise texture in an image, by examining these wavelet coefficients.

A wavelet decomposition of a typical image (having a foreground object and a background) is shown in Figure 3.3.7. Keeping this in mind while examining the wavelet decomposition, note the multi-modal shape of the scaling function coefficients. The wavelet decomposition represents the original image as a smoother, low-resolution image augmented by sharp wavelets, which fill in the missing detail. In the image ofT the cat, the low resolution is a fairly good representation of the cat, but large wavelet corrections need to be made to render the fur, the beard, and so forth.

For texture, on the other hand, more corrections of much smaller amplitude need to be made, since the textured image is like a noisy surface superimposed on a smooth background (see Figure 3.3.8). The histogram of coefficients is more widely spread than for the picture of the cat.

Histograms of wavelet coefficients also differ for different textures. While smoother textures, consisting of many small coefficients, have histograms with wide main modes, scratchy surfaces are constructed from a mixture of very large and very small coefficients and have sharper main modes. The "sharpness" of the histogram is therefore strongly associated with the properties of the texture involved.

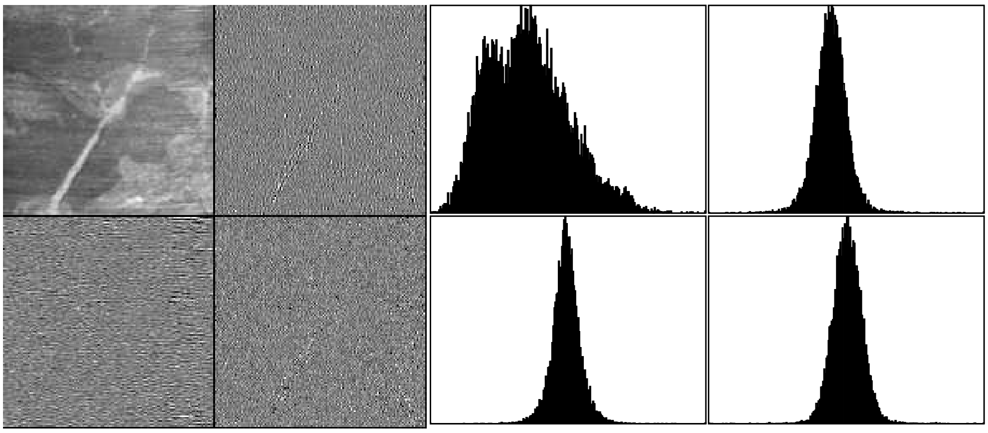


Figure 3.3.8: Wavelet transform and level histograms of a textured image.

Mallat [Mal89] modeled texture histograms with the function

$$h(u) = Ke^{-(|u|/\alpha)^{B}}.$$

This model perfectly fits the coefficients of the image of the cat, but performs less well on the smoother histogram of textures in general. The main advantage of the model is that the histogram can be characterised by two parameters, the *wavelet signature* [vdW98] α and β, but these are difficult to calculate.

By experimentation, it has been found that the simple Gaussian distribution fits the histogram as well as or better (in the case of natural texture) than the wavelet signature. This distribution

$$h(u) = Ke^{-(x-\mu)^2/2\sigma^2}$$

also has two parameters, the mean, μ, and the variance, σ, which characterise the distribution. Wavelet signatures are only applicable to centred histograms, as illustrated in Figure 3.3.9. The signature is especially well suited to very sharp histograms.

The two parameters of the normal distribution are much easier to estimate than the wavelet signature and provides similar results. To characterise a texture, we then do an N-level standard wavelet decomposition of the image. Each level is separated into horizontal, vertical and diagonal coefficients, which are then used to calculate sets of the parameters (μ, σ). For an N-level decomposition, $2 \times 3 \times N$ parameters are available, which are used in the texture feature vector.

In Section 3.3.3 the cumulative energy graph is shown (Figure 3.3.6). The

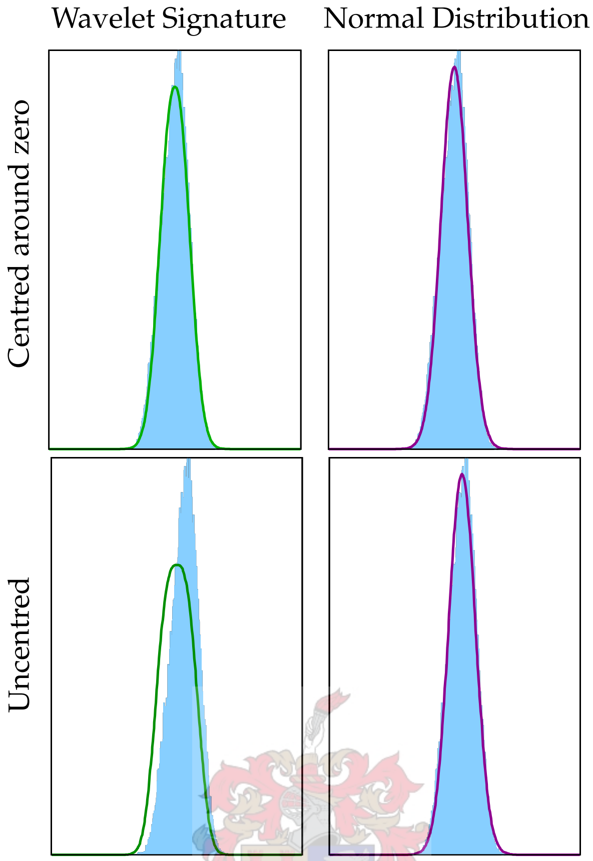


Figure 3.3.9: Signature comparison

shape of the energy curve also provides some information as to the underlying texture. A third degree polynomial is fitted to the curve and all the coefficients, except for the constant offset, are included in the texture feature vector. We call this the *wavelet energy signature*.

3.4 Pattern recognition

3.4.1 Introduction

Given an unknown sample, a classifier's task is to assign it to some class. It discriminates between these different classes of data, using previously known or *a priori* evidence. The process of gathering this evidence is known as *feature extraction*, with the features being stored in a feature vector $\mathbf{f} \in \mathbb{R}^d$ (d being the number of features measured). Section 3.3.4 discusses the features used in representing textures specifically.

By analyzing the features extracted from a *training set* of data, the classifier is provided with an estimated statistical model of the data to be analysed. If the statistical model is a good description of the underlying data, it should be possible to classify unknown samples with high accuracy — given, of course, that discrimination is possible based on the available features.

If enough data is available, the classifier can be used to classify samples from a *validation set* after which the classifier parameters are modified to produce the best results. The final step in evaluating the classifier and the statistical model is to classify data from a *test set*. These results are then an indication of how well the classifier would perform in practice.

The classifier has the freedom to reject samples which do not seem to belong to any known class. The *success rate* is the percentage of all samples correctly identified. The *accuracy*, in contrast, is the percentage of correctly classified samples of those not rejected. The accuracy is therefore greater or equal to the success rate.

If many different features are extracted from a small data set, the features space $\mathbb{R}^d$ is sparsely populated. This makes it difficult to accurately estimate the underlying statistical model parameters. Also, these features increase the number of calculations needed for training and classification. A way is needed to reduce the dimensionality of the feature vectors, so that the space $\mathbb{R}^k,\ k < d$ is more densely populated.

This can be achieved in three ways [DHS01, p. 113]: redesign the feature extractor, make a selection from the available features or combine the features in some way. While we carefully choose the features to extract, knowing which would be best for discrimination is not always easy. We therefore rather extract too many features than too few. This makes the last option all the more attractive: combine all the features in some way (linearly, if possible) and thereby densely populate a feature space of lower dimension. Two methods of doing this is discussed in Section 3.4.3.

We make the assumption that, for each class, feature vectors are random variables, drawn from some Gaussian distribution. This assumption may not be strictly accurate, but what we lose in the accuracy of the model, we gain in the simplicity of the classifier implementation (see Section 3.4.2). The Gaussian distribution is appropriate for describing samples disturbed by a large number of random processes. Imagine any class as being represented by an ideal feature vector, and that any measured feature is a randomly corrupted version of it.

We base our classifier on Bayesian decision theory, as explained in the next section.

3.4.2 The Bayesian classifier

Bayesian decision theory

As an example, imagine that we would like to design a classifier to distinguish between two classes of human beings: male and female. The classes are denoted by the symbols ω_1 and ω_2 respectively. We know beforehand the *a priori probabilities* of the two classes: that the world population is 50.25% male and 49.75% female (i.e., $P(\omega_1) = 0.5025$). If asked to tell the sex of the next human being we were about to meet, the best guess would therefore be "male".

A seamstress now provides us with measurements of people's circumferences and heights, asking us to classify the clients accordingly. These measurements form two-dimensional *feature vectors*

$$\mathbf{x} = \left[\begin{array}{cc} \text{circumference} & \text{height} \end{array} \right].$$

Building a classifier now becomes slightly more involved, since much more information is available. Not only do we have measurements, but we can also make an educated guess as to the statistical distribution of these measurements. It is often shown in biology textbooks [ST03, p. 189] that student lengths are *normally distributed*. The mean and variance of the distribution is different for male and female students, but the underlying model is the same. We assume that this also holds for circumference, so that a bivariate normal probability density function describes each class.

The probability of a measurement belonging to class ω_i and having the feature vector $\mathbf{x}$ is

$$p(\omega_i, \mathbf{x}) = P(\omega_i \mid \mathbf{x})p(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i)$$

which leads to the *Bayes formula* [DHS01, p. 24]

$$P(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i)P(\omega_i)}{p(\mathbf{x})}.$$

This means that, given a feature vector $\mathbf{x}$, we can calculate the *posterior probability* that the vector belongs to class ω_i. All we need to know is, for each class, the statistical distribution $p(\mathbf{x} \mid \omega_i)$ and the prior probability $P(\omega_i)$. The *evidence* $p(\mathbf{x})$ merely scales the posterior probability, and is given by

$$p(\mathbf{x}) = \sum_i p(\mathbf{x} \mid \omega_i)P(\omega_i).$$

We can now classify the tailor's clients by deciding on *male* (ω_1) whenever

$$p(\mathbf{x} \mid \omega_1)P(\omega_1) > p(\mathbf{x} \mid \omega_2)P(\omega_2).$$

In general, decide on ω_i whenever

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \forall j \neq i \tag{3.4.1}$$

where

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid \omega_i)P(\omega_i).$$

Discriminant functions for multivariate Gaussian distributions

Any monotonically increasing function can be applied to the decision criterion without changing the outcome. In other words,

$$\ln g_i(\mathbf{x}) > \ln g_j(\mathbf{x}) \quad \forall j \neq i$$

is equivalent to (3.4.1). We assumed earlier that, for each class, the underlying statistical model is the N-dimensional multivariate Gaussian distribution, i.e.

$$p(\mathbf{x} \mid \omega_i) = \frac{1}{(2\pi)^{N/2} \left|\boldsymbol{\Sigma}_i\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right]$$

where $\boldsymbol{\Sigma}_i$ is the covariance matrix and $\boldsymbol{\mu}_i$ the mean for the distribution of class i. Applying the log transform, our discrimination function becomes

$$\ln g_i(\mathbf{x}) = \ln P(\omega_i) - \frac{N}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| - \frac{1}{2}\left[(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right]. \tag{3.4.2}$$

Since the second term is independent of i, we drop it to obtain the final discrimination function

$$d_i(\mathbf{x}) = \ln P(\omega_i) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| - \frac{1}{2}\left[(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right]. \tag{3.4.3}$$

If the prior probabilities are equal, the first term can also be neglected.

Given an unknown feature vector $\mathbf{x}$, classification is done by evaluating $d_i(\mathbf{x})$ for all classes, and picking the class associated with the largest value. Discrimination based on (3.4.3) requires fewer calculations, whereas (3.4.2) has the advantage that it allows *rejection* of certain samples when their posterior probabilities are less than a given threshold.

3.4.3 Feature reduction

Given a large set of data with a certain statistical distribution, it is often not difficult to estimate the underlying parameters of the distribution. The accuracy of the estimates depends heavily on the number of data points available[5]. For example, examine the random variable X from the normal or Gaussian distribution

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma_X^2}} e^{-(x-\mu_X)^2/2\sigma_X^2}, \tag{3.4.4}$$

with mean μ_X and variance σ_X^2. Since $f_X(x)$ is a probability density function (PDF), the value of its probability distribution function is 1 at infinity ([Pap65], also see Appendix A.1),

$$\int_{-\infty}^{+\infty} f_X(x)dx = 1. \tag{3.4.5}$$

For mean $\mu_X = 0$ and variance $\sigma_X^2 = \frac{1}{2}$, (3.4.4) becomes

$$f_X(x) = \frac{1}{\sqrt{\pi}} e^{-x^2}.$$

Equation (3.4.5) then shows that

$$\int_{-\infty}^{+\infty} e^{-x^2} dx = \sqrt{\pi}. \tag{3.4.6}$$

The mean value of a random variable X is given by

$$\mathsf{E}[X] = \int_{-\infty}^{+\infty} x f_X(x)dx \tag{3.4.7}$$

and is, in the case of the normal distribution,

$$\mathsf{E}[X] = \frac{1}{\sqrt{2\pi\sigma_X^2}} \int_{-\infty}^{+\infty} x e^{-(x-\mu_X)^2/2\sigma_X^2} dx.$$

[5] Earlier, we discussed the Bayesian classifier based on the Gaussian distribution. Such classifiers need to estimate covariance matrices based on the samples in the different classes. An accurate estimation can only be made if enough samples are available, otherwise different methods should be used to improve the estimate [TGF04].

Making the substitution $v = \frac{x-\mu_X}{\sqrt{2}\sigma_X}$, this becomes

$$\begin{aligned}\mathsf{E}[X] &= \frac{1}{\sqrt{\pi}}\int_{-\infty}^{+\infty}(\sqrt{2}\sigma_X v + \mu_X)e^{-v^2}dv \\ &= \sigma_X\sqrt{\frac{2}{\pi}}\int_{-\infty}^{+\infty} ve^{-v^2}dv + \frac{1}{\sqrt{\pi}}\int_{-\infty}^{+\infty}\mu_X e^{-v^2}dv.\end{aligned}$$

Because v and e^{-v^2} are odd and even functions respectively, the first term of the sum is zero. Using (3.4.6), the expected value then is

$$\mathsf{E}[X] = \bar{X} = \mu_X.$$

In a similar fashion, the variance can be calculated as

$$\mathsf{E}[(X-\bar{X})^2] = \sigma_X^2.$$

In practice, we often work with a set of N samples, x_i (where $i = 1, 2, \ldots, N$), generated from an unknown underlying probability density function. If we assume that the underlying PDF is Gaussian, we can easily estimate the sample mean and variance.

First, we generate a histogram $p(z)$ of the samples x_i. Here, z denotes the K distinct values of x and $p(z)$ is the number of occurrences of value z in x_i. The sample mean is then analogous to (3.4.7). Assuming that there are K different values of x, it is

$$\begin{aligned}\tilde{\mathsf{E}}[\mathsf{X}] = \tilde{\bar{X}}_N &= \sum_{i=0}^{K-1} z_i p(z_i) \\ &= \frac{1}{N}\sum_{n=0}^{N-1} x_n.\end{aligned}$$

Similarly the (biased) variance can be estimated by

$$\begin{aligned}\tilde{\mathsf{E}}[(X-\bar{X})^2] &= \sum_{i=0}^{K-1}(z_i - \tilde{\bar{X}}_N)^2 p(z_i) \\ &= \frac{1}{N}\sum_{n=1}^{N-1}(x_n - \tilde{\bar{X}}_N)^2.\end{aligned}$$

The estimated parameters of the underlying model converge to the actual parameters as $N \to \infty$ [PZP01, p. 165]. On the other hand, as N becomes smaller, the accuracy of the estimation decreases.

In practice, it often happens that only a small set of data is available. It is therefore impossible to accurately estimate the parameters of the underlying statistical model. We handle this problem by projecting the data onto a space of lower dimensionality. In this space, the data is less sparse and the model parameters can be estimated with greater confidence.

Principal component analysis (PCA)

An n-dimensional vector can be expressed as a linear combination of n orthonormal basis vectors, so that

$$\mathbf{x} = y_1\mathbf{u}_1 + y_2\mathbf{u_2} + \ldots + y_n\mathbf{u}_n.$$

The vector can then be approximated in a lower dimension as

$$\begin{aligned}\hat{\mathbf{x}} &= y_1\mathbf{u}_1 + y_2\mathbf{u}_2 + \ldots + y_m\mathbf{u}_m \\ &= \left(\mathbf{x}^T\mathbf{u}_1\right)\mathbf{u}_1 + \left(\mathbf{x}^T\mathbf{u}_2\right)\mathbf{u}_2 + \ldots + \left(\mathbf{x}^T\mathbf{u}_m\right)\mathbf{u}_m, \qquad (3.4.8)\end{aligned}$$

where $m < n$. The aim of principal component analysis is to choose such a set of basis vectors, $\{\mathbf{u}_1 \ldots \mathbf{u}_m\}$, that minimises the least square error between the approximated and the original vector. For any single vector $\mathbf{x}$, the error is

$$\mathbf{x} - \hat{\mathbf{x}} = \sum_{j=n+1}^{m} y_j\mathbf{u}_j$$

and we aim to minimise $|\epsilon|^2 = |\mathbf{x} - \hat{\mathbf{x}}|^2$ over all vectors in a data set.

From (3.4.8) it follows [Cas98] that

$$\begin{aligned}\mathsf{E}[\epsilon] &= \mathsf{E}\left[|\mathbf{x} - \hat{\mathbf{x}}|^2\right] \\ &= \mathsf{E}\left[(\mathbf{x} - \hat{\mathbf{x}})^T(\mathbf{x} - \hat{\mathbf{x}})\right] \\ &= \sum_{j=m+1}^{n} \mathsf{E}\left[\left(\mathbf{x}^T\mathbf{u}_j\right)^T\left(\mathbf{x}^T\mathbf{u}_j\right)\right] \\ &= \sum_{j=m+1}^{n} \mathsf{E}\left[\mathbf{u}_j^T\mathbf{x}\mathbf{x}^T\mathbf{u}_j\right] \\ &= \sum_{j=m+1}^{n} \mathbf{u}_j^T\mathbf{A}\mathbf{u}_j \qquad (3.4.9)\end{aligned}$$

where $\mathbf{A}$ is the correlation matrix $\mathsf{E}\left[\mathbf{x}\mathbf{x}^T\right]$. Seeking to minimise $\mathsf{E}[\epsilon]$ conditional to the constraint $\mathbf{u}_j^T\mathbf{u}_j - 1 = 0$, we construct the Langrange multiplier function

(see Appendix A.3)

$$f(\mathbf{u}_{m+1}, \mathbf{u}_{m+2}, \ldots, \mathbf{u}_n) = \sum_{j=m+1}^{n} \left(\mathbf{u}_j^T \mathbf{A} \mathbf{u}_j - \lambda_j \mathbf{u}_j^T \mathbf{u}_j \right),$$

find the gradient (see Appendix A.4) and set it to zero (a necessary condition for $\mathsf{E}[\epsilon]$ to be a minimum):

$$\frac{\partial f}{\partial \mathbf{u}_j} = 2 \left(\mathbf{A} \mathbf{u}_j - \lambda_j \mathbf{u}_j \right) = 0, \quad j = m+1, m+2, \ldots, n.$$

We recognise the eigenvalue problem, where the different $\mathbf{u}_j$ are eigenvectors of the matrix $\mathbf{A}$, corresponding to the eigenvalues λ_j. If the data is centred, these eigenvectors correspond to those of the scatter matrix $\mathbf{S}$.

The scatter matrix of a set of n_i feature vectors X_i, belonging to class i with a sample average of $\mathbf{m}_i$, is defined as

$$\mathbf{S}_i = \sum_{\mathbf{x} \in X_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T,$$

which is proportional to the covariance matrix. We see that

$$\begin{aligned} \mathbf{S} &= \mathsf{E}\left[(\mathbf{x} - \mathbf{m}) (\mathbf{x} - \mathbf{m})^T \right] \\ &= \mathsf{E}\left[\mathbf{x}\mathbf{x}^T \right] - \mathbf{m}\mathbf{m}^T \\ &= \mathbf{A} - \mathbf{m}\mathbf{m}^T, \end{aligned}$$

where $\mathbf{m}$ is the mean or expected value of $\mathbf{x}$ (over all data).

Using (3.4.9), the error term $\mathsf{E}[\epsilon]$ can be expressed in terms of the eigenvalues of $\mathbf{A}$ as

$$\mathsf{E}[\epsilon] = \sum_{j=m+1}^{n} \lambda_j,$$

since

$$\mathbf{u}_j \mathbf{A} \mathbf{u}_j = \lambda_j.$$

This expression is minimised by choosing $\{\lambda_{m+1} \ldots \lambda_n\}$ as the set of smallest eigenvalues. The basis vectors $\{\mathbf{u}_{m+1} \ldots \mathbf{u}_n\}$ are therefore the eigenvectors associated with the smallest eigenvectors of the scatter matrix $\mathbf{S}$, provided that the data is centred. Because $\mathbf{S}$ is symmetric, these eigenvectors are orthonormal and form a basis [Str93, p. 273].

Principal component analysis projects an n-dimensional dataset onto the eigenvectors associated with the m largest eigenvalues of the scatter matrix.

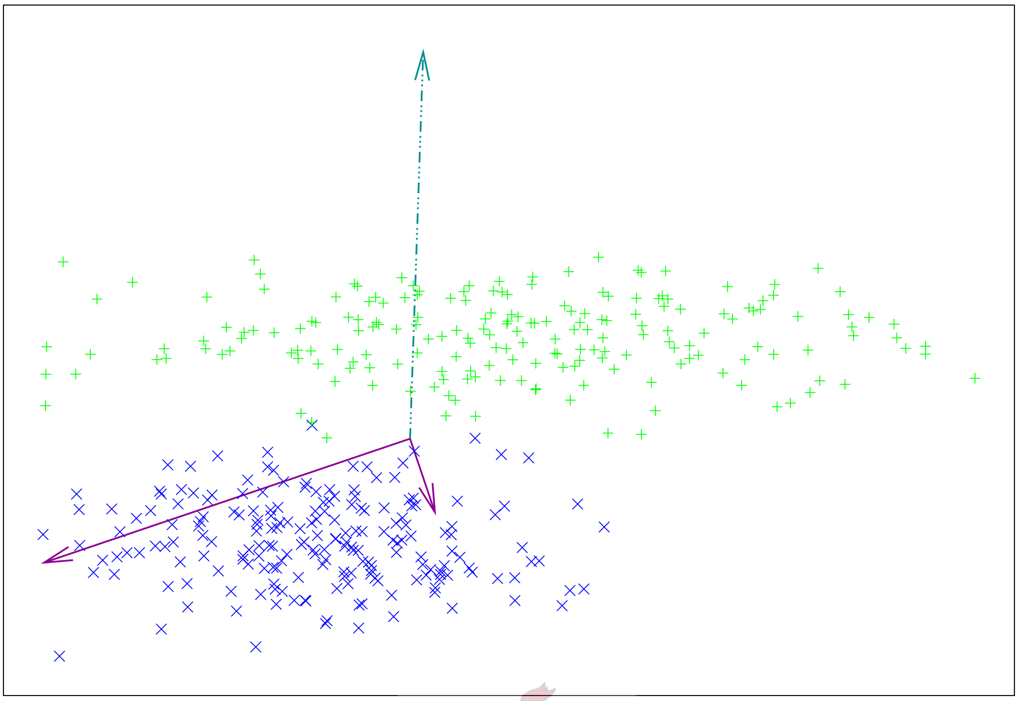


Figure 3.4.1: Axes of projection for PCA (solid) and MDA (dashed).

This projection minimises the least-squares error between the original data in n dimensions and projected data in m dimensions.

Multiple discriminant analysis (MDA)

Principal component analysis projects a set of data onto the axes of maximum variation, reducing the number of dimensions if required. When presenting data for visual inspection, this is ideal, but it is not necessarily the best space for discrimination [DHS01, p.Ĩ17]. Figure 3.4.1 shows a set of data and the axes calculated for PCA. Projecting the data onto the main axis of variation is clearly not a good choice for discrimination — the projected data will sometimes overlap. A better approach to calculating the projection is multiple discriminant analysis, a generalisation of Fisher's linear discriminant functions, described below.

When considering a dataset comprising C different classes, we recall our earlier definition of the scatter matrix and define the within-class scatter matrix as

$$\mathbf{S}_W = \sum_{i=1}^{C} \mathbf{S}_i$$

and the between-class scatter matrix is

$$\mathbf{S}_B = \sum_{i=1}^{C} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m}),$$

with $\mathbf{m}_i$ and $\mathbf{m}$ being the sample averages of samples in the i-th class and of the pooled data respectively.

A linear transform, $\mathbf{W}$, is applied to the feature vectors, mapping the space $\mathbb{R}^d \to \mathbb{R}^k$ and thereby reducing the dimensionality from d to k. The resulting within-class and between-class scatter matrices are $\mathbf{W}^T\mathbf{S}_W\mathbf{W}$ and $\mathbf{W}^T\mathbf{S}_B\mathbf{W}$ respectively. To improve discrimination between classes, the ratio of the determinants of the between-class to the within-class scatter matrices,

$$J(\mathbf{W}) = \frac{|\mathbf{W}^T\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^T\mathbf{S}_W\mathbf{W}|}, \tag{3.4.10}$$

can be maximised (here, the determinants are used as a scalar measure of scatter). Intuitively, this makes sense: the data should ideally be widely spread in the feature space, yet each class needs to be concentrated. We are therefore interested in finding the argument $\mathbf{W}$ that maximises $J(\mathbf{W})$. Examine the simpler vector problem

$$\arg\max_{\mathbf{w}} J(\mathbf{w}) = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}},$$

which has the solution

$$J(\mathbf{w})\mathbf{S}_W\mathbf{w} = \mathbf{S}_B\mathbf{w}. \tag{3.4.11}$$

This is the generalised eigenvalue problem,

$$\mathbf{S}_B\mathbf{w} = \lambda\mathbf{S}_W\mathbf{w}, \tag{3.4.12}$$

which can be solved using the QZ algorithm [MS73, PTVF03, GvL83]. The vector $\mathbf{w}$ that maximises $J(\mathbf{w})$ is the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}_W^{-1}\mathbf{S}_B$ (see Appendix A.2). The solution to (3.4.10) is similar and is given [DHS01] by

$$\mathbf{W} = \left[\begin{array}{cccc} \mathbf{w}_{\lambda 1} & \mathbf{w}_{\lambda 2} & \ldots & \mathbf{w}_{\lambda k} \end{array}\right],$$

where the columns of $\mathbf{W}$ are the eigenvectors in (3.4.11) corresponding to the largest eigenvalues. The feature vectors can now be transformed, using $\mathbf{W}$, so that

$$\mathbf{y} = \mathbf{W}^T\mathbf{x}.$$

For the resulting data set, the ratio of the determinants of the between-class to the within-class scatter matrices is maximised. The feature vectors are now in a space ideal for discrimination.

Chapter 4

Locating Thin Layers

4.1 Introduction

Thin layers are of interest to geologists because they often point to special stratigraphic circumstances. Chromitite layers, for example, are very brittle and their presence raises warning signs. Mining underneath such layers can cause the hanging walls (roofs) of stopes and tunnels to collapse, endangering the lives of workers. Can these layers, which are difficult enough to find by eye, be located automatically, given a photograph (see Figure 4.1.1) of a diamond-drill borehole core (or, equivalently, of the inside of a borehole)? [Hei94] We present a method that shows promising results.

Two algorithms are used: *shortest-path extraction* and *non-linear least squares fitting*. Examine, for example, Figure 4.1.1 and note the thin chromitite layer visible near the right-hand side of the photo. To locate this layer, a window is shifted over the image in overlapping steps. At each step, the shortest-path algorithm is applied to extract possible chromitite pixels.

The shortest-path algorithm always delivers a path — whether or not chromitite is present. When does such a path run along a chromitite stringer? Any thin planar layer will project a sinusoidal shape, because of the way a plane cuts a cylinder (see Section 4.2). If the path conforms to this shape, it is likely

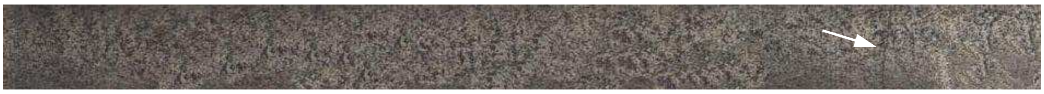


Figure 4.1.1: A pyroxenite borehole core from the UG2 system of the Bushveld Igneous Complex, containing a well hidden chromitite layer (see the geological log in C.1).

to be representative of such a layer. One period of a sinusoid (with variable phase and amplitude) is fitted to the pixels on the path. The mean-square error between the pixels and the sinusoid, as well as the path cost, is then a measure of fit, and of whether the path points to a chromitite layer. If it does, the dip and azimuth can be established from the amplitude and phase of the fitted sinusoid.

4.2 Curves and parameters

Thin layers can be difficult to see, since they are often dark and hidden in noise-like surface texture. Localised methods, like edge detection, are therefore not effective in isolating these artifacts. Methods that produce a measure of likelihood based on a complete possible path are less prone to be influenced by noise and are more successful in this context.

The Hough transform is an example of such a method. Define a curve C in two dimensions, with N variable parameters, as

$$C(x) = f(x \mid p_1, p_2, \ldots, p_N).$$

Given an image $Z(x, y)$ containing a curve visually similar to C (call it C'), our goal is to identify the parameters $\{p_1 \ldots p_N\}$ that will allow C to approximate C' as closely as possible. First, the pixel positions of C' are isolated, using thresholding or more sophisticated methods (noisy images, for example, cannot be thresholded easily). The parameters $\{p_1 \ldots p_N\}$ are varied to modify the trajectory of C. Depending on how well C approximates C', the parameters are rated; in the case of the Hough transform, according to the number of overlapping pixels between C and C'. We are looking for a set of parameters that maximises a certain measure of fit.

In the case of a a straight line, the curve is

$$C(x) = mx + d,$$

with parameters m (gradient) and d (offset). For near-vertical lines, the gradient m grows very large and therefore the curve is often rather expressed in the normal representation (see Figure. 4.2.1)

$$x \sin\theta + y \cos\theta \quad = \quad \rho,$$

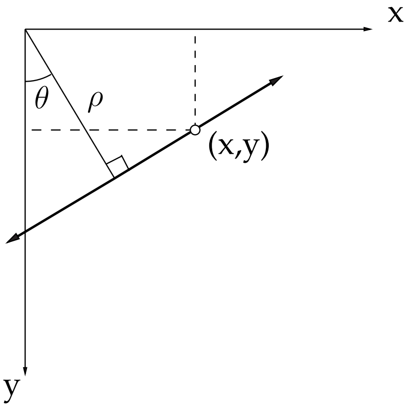


Figure 4.2.1: Normal line representation.

or

$$C(x) = \frac{\rho}{\cos\theta} - x\tan\theta,$$

which causes the parameters to fall in a more limited range. To determine which parameters achieve the best fit, all sets in the parameter space must be evaluated. In other words, for every combination $\{\theta, \rho\}$, a line C must be calculated and compared to C'. We can do slightly better by examining the whole range of angles and, for every non-zero pixel in the image, deriving the value of ρ. For $\theta = \{-90^\circ \ldots 90^\circ\}$, the total number of calculations are then $\leq 180 \times M \times N$. The resulting Hough transform is shown in Figure 4.2.2 (the curve was "isolated" by simple thresholding). The maxima of the Hough transform point to the parameter sets $\{\theta, \rho\}$ that produce the best fitting lines.

While the Hough transform works well enough for straight lines, it is computationally intensive. The curves of importance here are those generated when a rock layer cuts through a borehole (see Figure 4.2.3), which can be described as

$$C(x) = d + A\sin\left(\frac{2\pi x}{P} + \theta\right),$$

where P is the number of pixels in a picture matrix row. We assume that the thin layer is only one pixel thick. An approach similar to the straight line Hough transform now becomes very expensive, since there are 3 different parameters to consider: d, A and θ. Instead, we opt for a *non-linear least squares* fitting of $C'(x)$ to the model $C(x)$ (Section 4.2.2). The pixels belonging to the curve $C'(x)$ are identified using *shortest paths*, as shown in Section 4.2.1. The mean-squared error between the model and the observed pixels is a measure of the goodness-of-fit, and an indication of whether the pixels belong to a chromitite layer (Section 4.3).

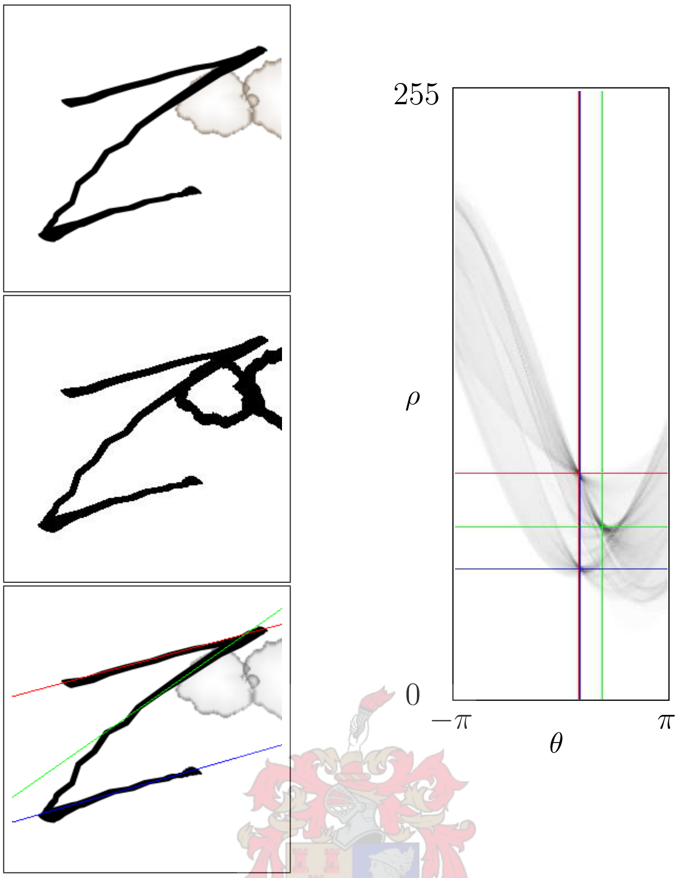


Figure 4.2.2: The Hough transform applied.

(Left top) The original image. (Left middle) Thresholded image. (Left bottom) The lines found superimposed on the original. (Right) Hough transform. The location of the parameters corresponding to the lines found are indicated.

4.2.1 Shortest paths

A path is an ordered series of connected nodes or elements. Every connection has an associated cost. The cost or length of a path is the sum of the costs of all its connections. If the type of cost is chosen correctly, the *shortest path* will travel along the thin, dark layers we are looking for. Figure 4.2.4 shows a typical path in a regular grid or matrix.

One of the simplest measures of cost is the difference between the values of two connected nodes, A and B,

$$C_{AB} = |B - A| \, .$$

This measure is the absolute value of the gradient of the path between the two nodes. Such a path of minimum length is therefore close to a contour path.

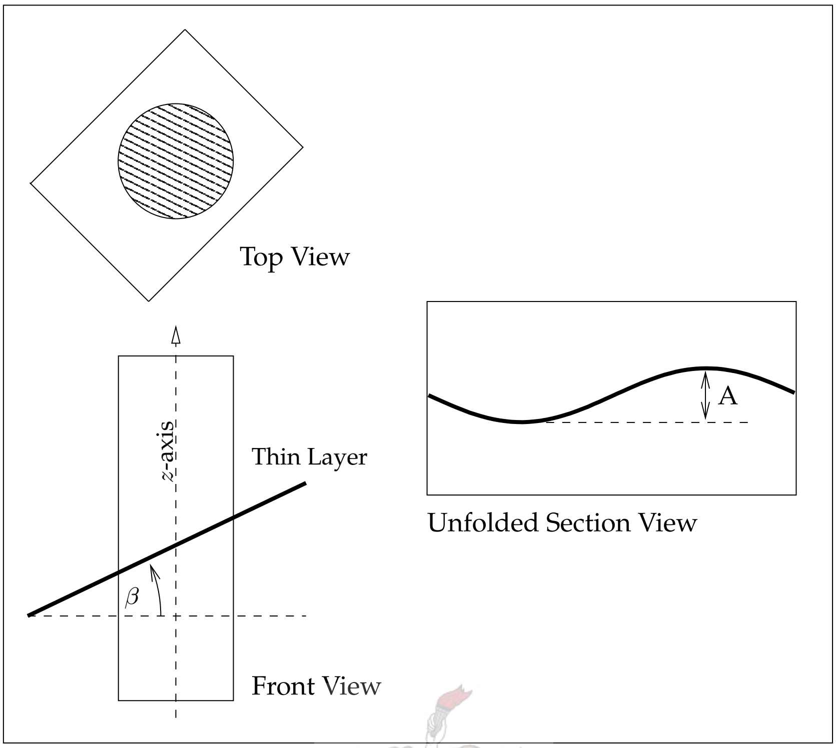


Figure 4.2.3: A cylinder sliced by a plane.

When working with images, the nodes are pixels, and the node values are pixel intensities or grey-levels.

Provided with such an image, $I(i,j)$, the shortest path algorithm (see Algorithm 4.2.1) provides cost and path matrices, $C(i,j)$ and $P(i,j)$, while the backtracking algorithm (see Algorithm 4.2.2) delivers the shortest path, $s(j)$. The form of the algorithm listed here is applicable to matrices and produce left-to-right paths.

Chromitite layers are always dark in colour, a property we incorporate into the cost-measure:

$$C_{AB} = w_d\,|B - A| + w_i B.$$

The cost is now a weighted sum of the magnitude of the path intensity-gradient and of the path intensity itself. This measure favours dark contour paths. As previously shown in Figure 4.2.3, a plane cutting through a cylinder like a borehole projects a sinusoidal shape onto a photograph. This sinusoid has exactly one period, and starts at the same image row at which it ends. The shortest paths we seek should also have this property, and we restrict the search to such *circular shortest paths*.

The backtracking algorithm can easily be modified to return only paths

Algorithm 4.2.1: Shortest path algorithm [BY97].

Given an image, $I(x, y)$, calculate the cost and path matrices.

1. Create an $M \times N$ cost matrix, C and an $M \times N$ path matrix, P. Here, $C(i, j)$ is the cost of the shortest path to pixel $I(i, j)$. The element that came before $I(i, j)$ in such a path is stored in $P(i, j)$.
2. Set the elements of the first column of C and the first column of P to 0.
3. Move on to the next column ($j = j+1$). For each element i of the column, examine the connections to each of the p closest elements in the previous column ($j - 1$). Pick the connection and element, $I(k, j - 1)$, associated with the smallest cost.
4. Update the cost matrix: $C(i, j) = C(k, j - 1) + C_{ij} = C(k, j - 1) + |I(i, j) - I(k, j - 1)|$.
5. Update the path matrix: $P(i, j) = k$.
6. Repeat steps 2 through 5 until the last column has been processed ($j = N + 1$).

Algorithm 4.2.2: Backtracking algorithm [BY97].

Given an $M \times N$ cost matrix C and an $M \times N$ path matrix P, determine the shortest path $s(j)$.

1. Start with the last column of the cost matrix ($j = N$).
2. Pick the smallest cost $C(k, N)$ and set the last path element, $s(N) = k$.
3. Move one column to the front ($j = j - 1$).
4. Update the path: $s(j) = P(s(j + 1), j)$.
5. Repeat steps 3 through 4 until s is complete ($j = 0$).

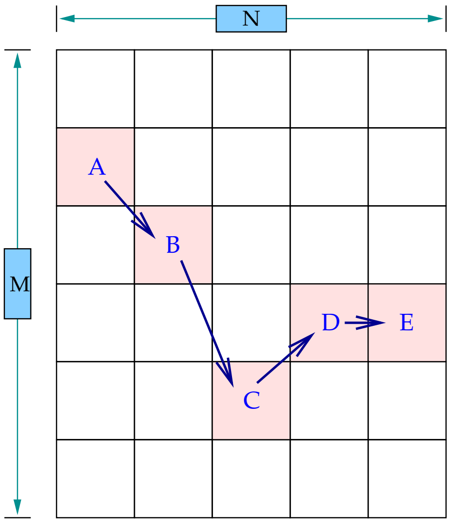


Figure 4.2.4: A left-to-right path through an $M \times N$ matrix.

conforming to this criterion. During execution, this unfortunately requires backtracking every element of the last column of the path matrix. A computationally more effective method [SP03] is to patch the image before applying the algorithm (without limiting the search to circular paths). Figure 4.2.5 shows how the image is patched by appending the first columns to the last and by prepending the last columns to the first. The shortest path algorithm is then likely to favour circular paths (in the original and not in the patched image), even though the path obtained is not guaranteed to be circular.

The measure of cost determines the shape of the shortest paths obtained, as well as the route followed. We chose the cost so the paths would be dark contours, similar to thin chromitite layers. The parameter p to the shortest path algorithm (Algorithm 4.2.1) determines how many rows the path may jump at any connection. To follow a path with a high gradient, this value needs to be set high — but this also means that very thin paths are less likely to be found. As a compromise, we chose $p = 5$, which allow the path to veer two pixels up- or downwards.

Figure 4.2.6 shows two experimental results. In these examples, a borehole core was photographed from the top. That means that only $180°$ of the core is visible and therefore the patching algorithm could not be applied. A human finds it difficult to distinguish the chromitite layer in the second image, but the computer has a keen eye for contours! This is all the more apparent in the third example, shown in Figure 4.2.7.

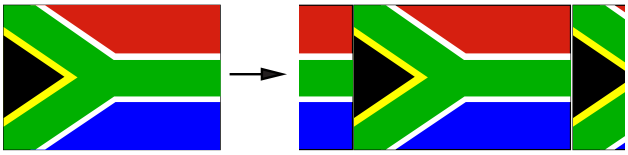


Figure 4.2.5: Patching an image for circular shortest paths.

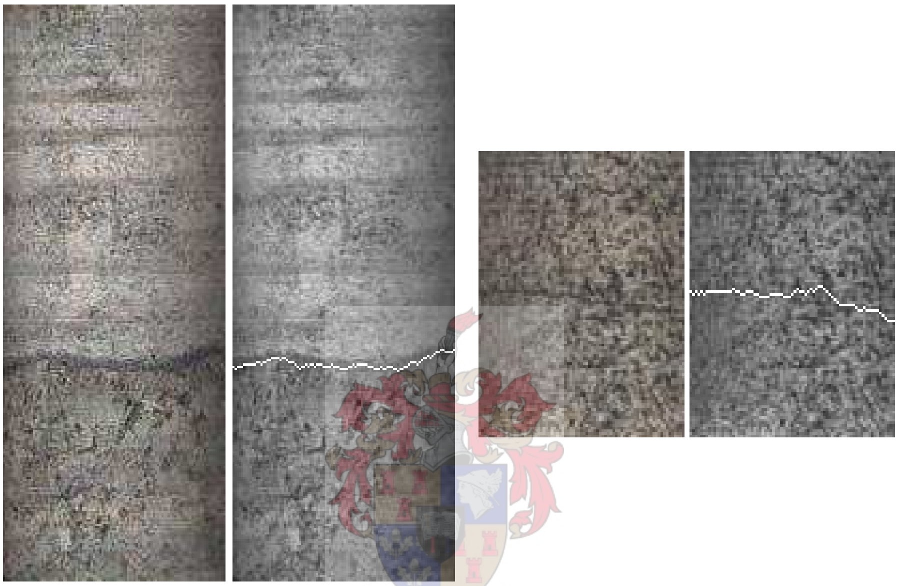


Figure 4.2.6: Chromitite layers found using shortest paths.

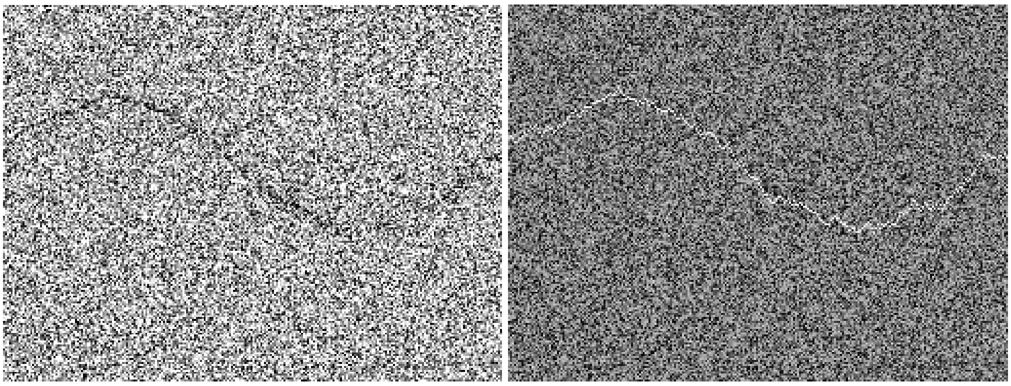


Figure 4.2.7: An artificial layer in a noisy background (left) is found to be the shortest path (right).

4.2.2 Non-linear least squares fitting

A shortest path can be seen as a collection of coordinates,

$$(x_i, y_i) \quad \text{for } 1 \leq i \leq N.$$

We model the path as

$$y(x_i) = d + A \sin\left(\frac{2\pi x_i}{P} + \theta\right)$$

and need to find the parameters A, d and θ for which y best approximates y_i. A linear least-squares solution is not applicable, since the function cannot be written as a linear combination of its basis functions,

$$y(x) = \sum_{i=1}^{M} a_i f_i(x).$$

Note that a linear least-squares solution would still have been appropriate for the case in which one or more of the basis functions $f_i(x)$ were non-linear.

Let us examine a more general case of this non-linear problem, where we have an M-parameter model

$$y = y(x_i \mid p_1 \dots p_M).$$

We will assume that every measurement y_i is a randomly corrupted version of its model value $y(x_i)$ and that the error, $y_i - y(x_i)$, is normally distributed[1] around 0. The expected value of a measurement y_i is $y(x_i)$ and therefore the likelihood of it occurring is

$$p(y_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-(y_i - y(x_i))^2/2\sigma_i^2}.$$

If the measurement errors are statistically independent, the likelihood of a complete path, $y_1 \dots y_N$, occurring is

$$P = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-(y_i - y(x_i))^2/2\sigma_i^2}$$

[1] We make this assumption based on the central limit theorem, which states that the probability distribution function of the sum of a large number of random variables approaches a Gaussian distribution [PZP01, p. 125]. The topic is discussed in detail in [PTVF03, p. 664], where the authors point out that errors not conforming to the normal distribution (i.e. outlier points) can severely skew a least-squares fit.

or, in log form,

$$\ln P = \sum_{i=1}^{N} \left[\ln \left(\frac{1}{\sqrt{2\pi}} \right) + \ln \sigma_i^{-2} - \frac{1}{2} \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2 \right].$$

We wish to maximise P (or $\ln P$), which is equivalent to minimising the *figure-of-merit function*

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i)}{\sigma_i} \right]^2,$$

or, explicitly stating the dependence on the parameters $\mathbf{p} = p_1 \ldots p_N$,

$$\chi^2(\mathbf{p}) = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i \mid \mathbf{p})}{\sigma_i} \right]^2.$$

We seek the parameter vector $\mathbf{p}$ that minimises the *Chi-square* function.

Sufficiently close to its minimum, $g = \chi^2$ can be approximated in quadratic form. First, expand the function as a Taylor series around the point $\mathbf{r}$ [PTVF03, p. 418], to give

$$g(\mathbf{p}) = g(\mathbf{r}) + \sum_{i=1}^{N} \left. \frac{\partial g}{\partial p_i} \right|_{\mathbf{r}} (p_i - r_i) + \frac{1}{2} \sum_{i,j} \left. \frac{\partial^2 g}{\partial p_i \partial p_j} \right|_{\mathbf{r}} (p_i - r_i)(p_j - r_j) + \cdots$$

and then drop higher order terms to obtain the quadratic approximation

$$g(\mathbf{p}) \approx g(\mathbf{r}) + \nabla g(\mathbf{r}) \cdot (\mathbf{p} - \mathbf{r}) + \frac{1}{2} (\mathbf{p} - \mathbf{r})^T \mathbf{A} (\mathbf{p} - \mathbf{r}),$$

where

$$A_{ij} = \left. \frac{\partial^2 g}{\partial p_i \partial p_j} \right|_{\mathbf{r}}.$$

The gradient of the quadratic function (see Appendix A.4) is

$$\nabla g(\mathbf{p}) = \nabla g(\mathbf{r}) + \mathbf{A}(\mathbf{p} - \mathbf{r}).$$

The function g has a minimum where this gradient is zero. Thus, in the region of the current point in the parameter space $\mathbf{p}_{cur}$, we have

$$\mathbf{A}(\mathbf{p}_{min} - \mathbf{p}_{cur}) = -\nabla g(\mathbf{p}_{cur})$$

Since g is known, so is the matrix $\mathbf{A}$ (the second partial derivative or *Hessian* matrix) and the gradient. A *direct estimate* of the parameters minimising the function can be made:

$$\mathbf{p}_{min} = \mathbf{p}_{cur} - \mathbf{A}^{-1}\nabla g(\mathbf{p}_{cur}).$$

This estimate is accurate only if the quadratic approximation is good at $\mathbf{p}_{cur}$. If this is not the case, the above method cannot be used and we have to step down the gradient in order to move closer to the minimum. The method of *steepest gradient descent* can be expressed as

$$\mathbf{p}_{next} = \mathbf{p}_{cur} - K \cdot \nabla g(\mathbf{p}_{cur}).$$

The Hessian matrix

In order to use either of the above methods, we need to calculate the gradient and the Hessian matrix. In general form, the function is

$$\chi^2(\mathbf{p}) = \sum_{i=1}^{N} \left[\frac{y_i - y(x_i \mid \mathbf{p})}{\sigma_i} \right]^2,$$

of which the gradient is found, using the chain and product rules, to be

$$\frac{\partial \chi^2}{\partial p_k} = -2 \sum_{i=1}^{N} \left[\frac{y_i - y(x_i \mid \mathbf{p})}{\sigma_i^2} \right] \frac{\partial y(x_i \mid \mathbf{p})}{\partial p_k}, \qquad k = 1, 2, \ldots, M.$$

The second partial derivative matrix becomes

$$\frac{\partial^2 \chi^2}{\partial p_k \partial p_l} = 2 \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i \mid \mathbf{p})}{\partial p_k} \frac{\partial y(x_i \mid \mathbf{p})}{\partial p_l} - [y_i - y(x_i \mid \mathbf{p})] \frac{\partial^2 y(x_i \mid \mathbf{p})}{\partial p_k \partial p_l} \right].$$

Assuming that the function is relatively smooth, the second derivative can be neglected. For a discussion as to why this is a good idea, refer to [PTVF03, p. 688]. The term multiplying the second derivative, $y_i - y(x_i \mid \mathbf{p})$, should roughly sum to zero over all i near the minimum and is therefore likely to be much smaller than the first term. In general, modifying the Hessian slightly still leads to convergence, even if it may take slightly longer. For our model,

$$y = d + A \sin(x + \theta),$$

we have

$$\mathbf{p} = \begin{bmatrix} d \\ A \\ \theta \end{bmatrix}.$$

The gradient is

$$\nabla\chi^2(\mathbf{p}) = -2\sum_{i=1}^{N} \frac{y_i - y(x_i \mid \mathbf{p})}{\sigma_i^2} \left[\begin{array}{ccc} 1 & \sin(x_i+\theta) & A\cos(x_i+\theta) \end{array} \right]^T,$$

and the Hessian matrix is

$$\mathbf{A}|_{\mathbf{p}} = \sum_{i=1}^{N} \frac{2}{\sigma_i^2} \left[\begin{array}{ccc} 1 & A_{21} & A_{31} \\ \sin(x_i+\theta) & \sin^2(x_i+\theta) & A_{32} \\ A\cos(x_i+\theta) & \frac{1}{2}\sin(2x_i+2\theta) & A^2\cos^2(x_i+\theta) \end{array} \right].$$

The values of the variances σ_i are unknown and set to 1 in the implementation.

Levenberg-Marquardt method

Two methods were listed in the previous section for finding the parameter set that minimises χ^2, namely

$$\text{Direct estimate : } \quad \mathbf{p}_{next} = \mathbf{p}_{cur} - \mathbf{A}^{-1}\nabla\chi^2(\mathbf{p}_{cur}) \tag{4.2.1}$$

or

$$\text{Gradient descent : } \quad \mathbf{p}_{next} = \mathbf{p}_{cur} - K \cdot \nabla\chi^2(\mathbf{p}_{cur}). \tag{4.2.2}$$

The steepest descent should be taken when we are far away from the function minimum, while a direct estimate should be made once we are close. The Levenberg-Marquardt algorithm makes a smooth transition between these two approaches. The mathematical detail of the algorithm is discussed in [Mar63] and two methods of implementation are detailed in [HZ03, p. 600] and [PTVF03, p. 688]. We discuss the simpler of the two methods to illustrate the concept.

The idea behind the algorithm is to merge the two methods in (4.2.1) and (4.2.2), which is accomplished by

$$(\mathbf{A} + \lambda\mathbf{I})\boldsymbol{\delta}_{\mathbf{p}} = -\nabla\chi^2(\mathbf{p}_{cur}) \tag{4.2.3}$$

where

$$\boldsymbol{\delta}_{\mathbf{p}} = (\mathbf{p}_{next} - \mathbf{p}_{cur}).$$

When λ is very small, (4.2.3) becomes

$$\mathbf{A}\boldsymbol{\delta}_{\mathbf{p}} = -\nabla\chi^2(\mathbf{p}_{cur})$$

Algorithm 4.2.3: The Levenberg-Marquardt algorithm [PTVF03, p. 689].

1. Choose λ small, i.e. $\lambda = 0.001$ and make an initial guess of the parameter vector, $\mathbf{p}$.
2. Compute $\chi^2(\mathbf{p})$.
3. Calculate $\boldsymbol{\delta}_{\mathbf{p}}$ using (4.2.3) and compute $\chi^2(\mathbf{p} + \boldsymbol{\delta}_{\mathbf{p}})$.
4. If $\chi^2(\mathbf{p} + \boldsymbol{\delta}_{\mathbf{p}}) \geq \chi^2(\mathbf{p})$, multiply λ by 10 and repeat from step 2.
5. If $\chi^2(\mathbf{p}+\boldsymbol{\delta}_{\mathbf{p}}) < \chi^2(\mathbf{p})$, divide λ by 10, update $\mathbf{p}$ to $\mathbf{p}+\boldsymbol{\delta}_{\mathbf{p}}$ and repeat from step 2.

The algorithm is iterated until $\chi^2(\mathbf{p})$ decreases by less than a certain small factor, say 0.01.

which is the direct estimate. On the other hand, when λ is very large, (4.2.3) becomes roughly

$$\boldsymbol{\delta}_{\mathbf{p}} = -\frac{\nabla\chi^2(\mathbf{p}_{cur})}{\lambda}.$$

The factor λ therefore provides a smooth transition between the gradient descent approach, which always guarantees movement closer to the function minimum, and the direct estimate, which is only applicable near the function minimum.

Algorithm 4.2.3 outlines the Levenberg-Marquardt method.

A sinusoid, fitted to a shortest path as obtained in the previous section, is shown in Figure 4.2.8. Are we guaranteed that this is the best possible fit? Unfortunately not, as the parameters might correspond to a local minimum in χ^2. It is therefore worthwhile making a good initial parameter estimate[2].

Given the shortest path pixels belonging to a chromitite stringer, the parameters d, A and θ of the fitted sinusoid are related to the location and orientation of the thin planar layer. It cuts through the cylindrical borehole core with main axis z. The parameter d gives the offset along this axis, while the amplitude A describes the angle of the plane with the horizon. The phase θ determines its rotation around z.

[2] A method like simulated annealing can be used to improve the chances of finding a minimum. Here, the parameter vector $\mathbf{p}$ is disturbed to see if the algorithm converges to the same minimum. If it does, it is likely that a global minimum has been found.

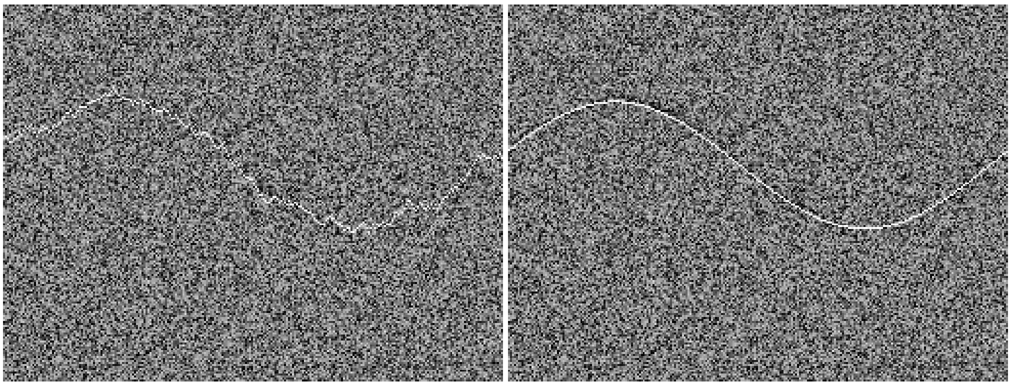


Figure 4.2.8: A shortest path (left) and the fitted sinusoid (right).

4.3 Finding a chromitite layer hidden in a pyroxenite core

Given a photograph of a pyroxenite core, a way is sought to locate thin chromitite layers. The only predetermined knowledge is that they are dark in colour and roughly sinusoidal in shape. They are often hidden in very "noisy" surface textures. Isolating such paths then becomes harder: any localised method (like the Sobel operator) immediately fails and thresholding is not applicable. The only option left is to consider *the whole path* and not only parts of it. Two intuitive approaches to the problem are now evaluated.

The previous section outlined a way of fitting a sinusoidal model to the pixels of a thin layer, obtained using the shortest paths algorithm. This algorithm returns a shortest path for *any* section of the borehole photograph. It is not expected for such a path to be sinusoidal unless a thin layer is present. The root-mean-square (RMS) error between the path pixels and the model,

$$\epsilon = \sqrt{\chi^2(\mathbf{p})} = \sum_{i=1}^{N} \left[y_i - y(x_i \mid \mathbf{p})\right]^2 ,$$

is a measure of the goodness-of-fit of the sinusoidal model. If this error is high, it is unlikely that a path is present. Yet, it was found in practice that, when no layers were present, the shortest path often turned out to be a straight line, very well represented by the sinusoidal model with zero amplitude. The RMS error cannot be trusted to be low only when a chromitite layer is present.

The second approach is to examine the cost of the shortest path. For a dark contour to exist, the cost must be low. Shift a window over the borehole photograph, each window overlapping the previous by half. The window must

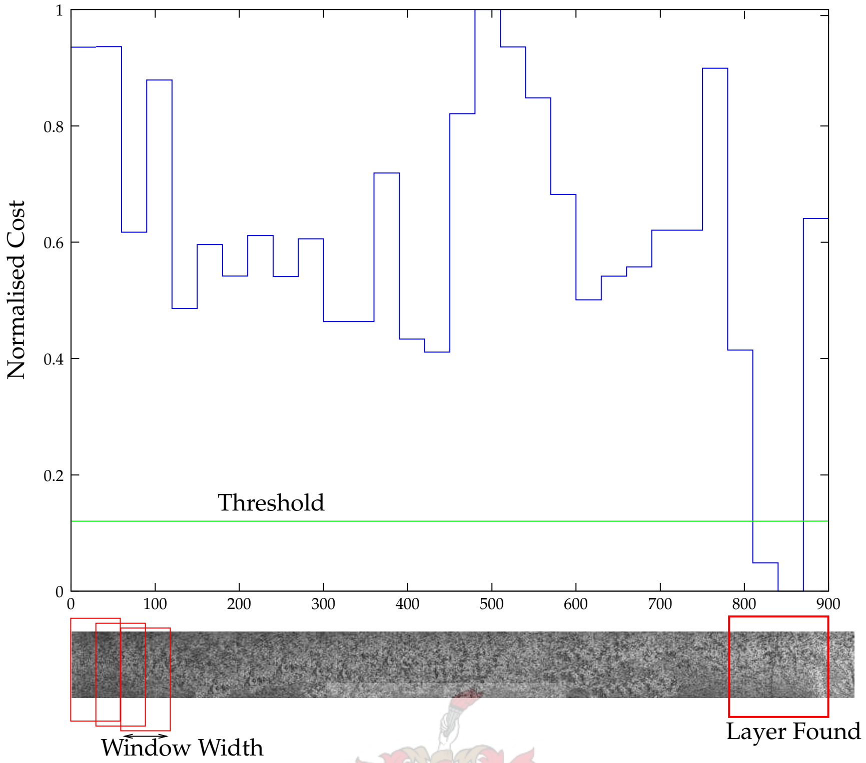


Figure 4.3.1: Finding thin layers in a borehole photograph.

Finding thin layers in a borehole photograph. A rectangular window moves across the image from left to right, as indicated by the three red boxes. At each step, the shortest path and associated cost (shown as a blue line) is calculated. When the cost is lower than a certain threshold (green line), a chromitite layer might be present.

be large enough to easily cover any chromitite layer. At each step, the shortest path and its associated cost, c_i, is obtained. Set a threshold to a factor of the average cost,

$$T = \lambda \mathsf{E}[c],$$

where the factor is low (typically 0.2). Wherever the error ϵ_i is lower than this threshold, a chromitite layer is likely to be present in window i. The factor λ can be varied: a high value leads to increased sensitivity, but also to decreased accuracy. Figure 4.3.1 illustrates the process and a typical result.

Varying intensity

The procedure outlined above makes the implicit assumption that the image intensity remains nearly constant over the borehole photograph. If it does not, the path cost will be influenced, with the threshold method possibly failing. To prevent this problem, some pre-processing of the cost vector $\mathbf{c}$ is done. The mean intensity value of each block i is subtracted from the cost c_i as a simple corrective measure.

Chapter 5

Texture Classification of Borehole Cores

5.1 Introduction

A borehole televiewer is an instrument used to photograph the inside of a borehole. During the evaluation of the algorithms presented, we did not have access to such equipment, or to any in-hole photographic data. Fortunately, we were able to photograph core trays (containing dolomite [Beu78]) at the De Beers Finsch Mine near Kimberley [Bar98]. The classifier performance discussed below is based on results obtained using this data.

One never has perfect control over measurements. In photographing the cores, obtaining a constant, equally illuminating yet diffuse light source proved difficult. While this probably influenced the experimental results, no corrections were made or preprocessing (other than selection) done on the data. If the algorithms perform well under these circumstances, we can safely infer that the results will be at least as good on higher quality data.

5.2 Photographing cores

5.2.1 Configuration of equipment

A camera is positioned directly over a core tray as shown in Figure 5.2.1. The distance to the lens is h, the camera's field of view is 2θ and the width of the picture taken is W. The camera maps the physical length W to D discrete pixels.

The Nyquist sampling theorem states that a band-limited continuous-time signal of bandwidth B can be uniquely recovered from its samples, provided

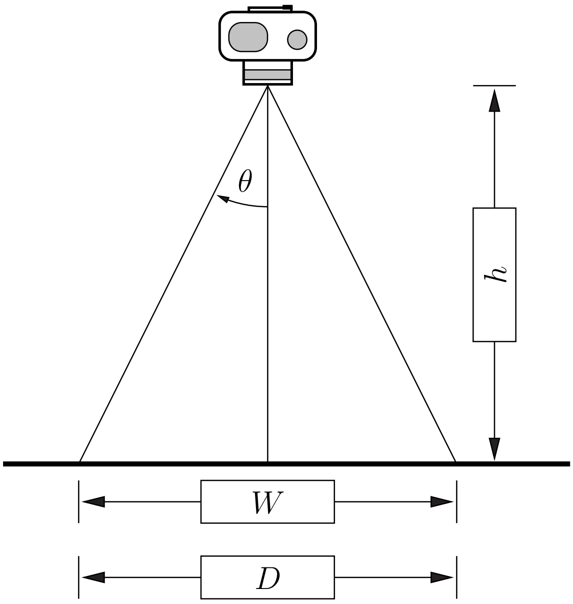


Figure 5.2.1: Camera setup for photographing cores.

that the sampling rate $F_s \geq 2B$ samples per second. The bandwidth of a core photograph depends on the texture involved and is considered to be the highest frequency found over all rock samples. The higher frequencies are visible in attributes such as the "spottiness" or the number of sharp ridges. As long as these features are clearly discernible, the sampling frequency is sufficiently high. According to observation, a pixel density of 130 dots per inch (or roughly one pixel every 0.2mm) is adequate.

A typical digital camera has a field of view of around $2\theta = 50^\circ$ and can take pictures at resolutions of more than 3 megapixels. A 3-megapixel photo has 1536×2048 pixels. At a height of 1.2 metres the photograph width is

$$W = 2h\tan(\theta) \approx 1 \text{ m}.$$

This is mapped onto 2048 pixels, which produces a density of roughly 50 dpi. With a zoom lens, the photograph width can be decreased to 30 cm, which results in a satisfactory density of 150 dpi.

5.2.2 Light sources

Core samples have dull, rough surfaces, caused by grinding of the diamond borehole drill bit against the stone. In order to hide the rough outer layer and to emphasise the underlying rock texture, the cores need to be wetted.

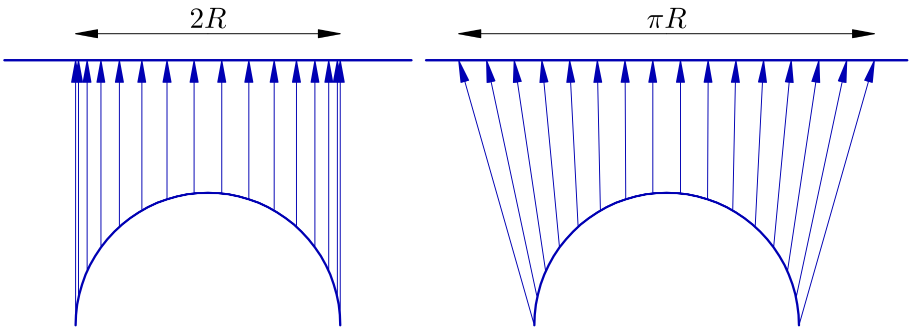


Figure 5.2.2: Geometrical distortion is observed in a cylinder, photographed from above (left). Using a different projection (right), the photograph can be transformed to remove such distortion.

Unfortunately, the water applied causes specular reflection and special care must be taken to use a diffuse light source for photography.

The samples were placed in a room, lit brightly by the sun shining onto closed blinds. A shadow was then cast onto the core samples (assuring that no direct light reached their surfaces) and the photographs taken, using a slightly extended exposure time ($\frac{1}{6}$s) to compensate for the shadow. The film sensitivity was set to a minimum to reduce noise levels.

5.2.3 Geometrical correction

The cylindrical cores are photographed from above, causing geometrical distortion of the surface texture in the photograph (see Figure 5.2.2). The distortion can be corrected by a simple trigonometric transformation, yet such a correction is highly sensitive to the exact core radius and position of the central axis. If only the central strip of each core photograph is used, the deformation does not present a problem to feature extracting algorithms (this is especially true for the way in which the wavelet coefficients are characterised: not individually, but by the statistical parameters of their distribution). The levels of geometrical distortion involved do not have a notable effect on the error rate obtained by the classifier.

5.3 Software overview

5.3.1 Numerical Simulations

The algorithms described in this thesis were implemented in **GNU Octave**, a high-level language for numerical computations, developed by John W. Eaton (University of Wisconsin-Madison) and the Octave community. Octave is free software (free as in *free speech)* and can be distributed or modified under the terms of the GNU General Public License as published by the Free Software Foundation.

Octave is extended by **Octave Forge** — a large collection of algorithms provided by the Octave community and released either under GPL-compatible licenses or in the public domain.

WaveLab was developed at Stanford University and implements the wavelet transform and other wavelet-related algorithms.

Extending Octave in C++

Octave is an interpreter and, under certain circumstances, lacks the execution speed required by image processing algorithms. To address this issue, modules written in C++ can be hooked into the interpreter. Algorithms can be rapidly developed in the vector and matrix oriented Octave language and translated to C++ if necessary.

Octave provides a C++ library for the handling of structures like matrices and vectors. As a trivial example, examine the module or *dynamically linked function* for creating the matrix

$$A(i,j) = i + j.$$

Listing 5.1: An example Octave extension.

```
#include <octave/oct.h>

DEFUN_DLD(example, args, , "A simple demonstration") {

    Matrix m(3,3);

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            m(i, j) = i + j;
        }
    }

    return octave_value(m);
}
```

This function exhibits the following behaviour in Octave:

```
octave:1> help example
example is the dynamically-linked function from the file
/home/stefan/thesis/example.oct

A simple demonstration
octave:2> a = example()
a =


  0  1  2
  1  2  3
  2  3  4
```

The library overrides certain operators, like the indexing braces "( )", which allow the C++ code to be similar in form to the scripted code.

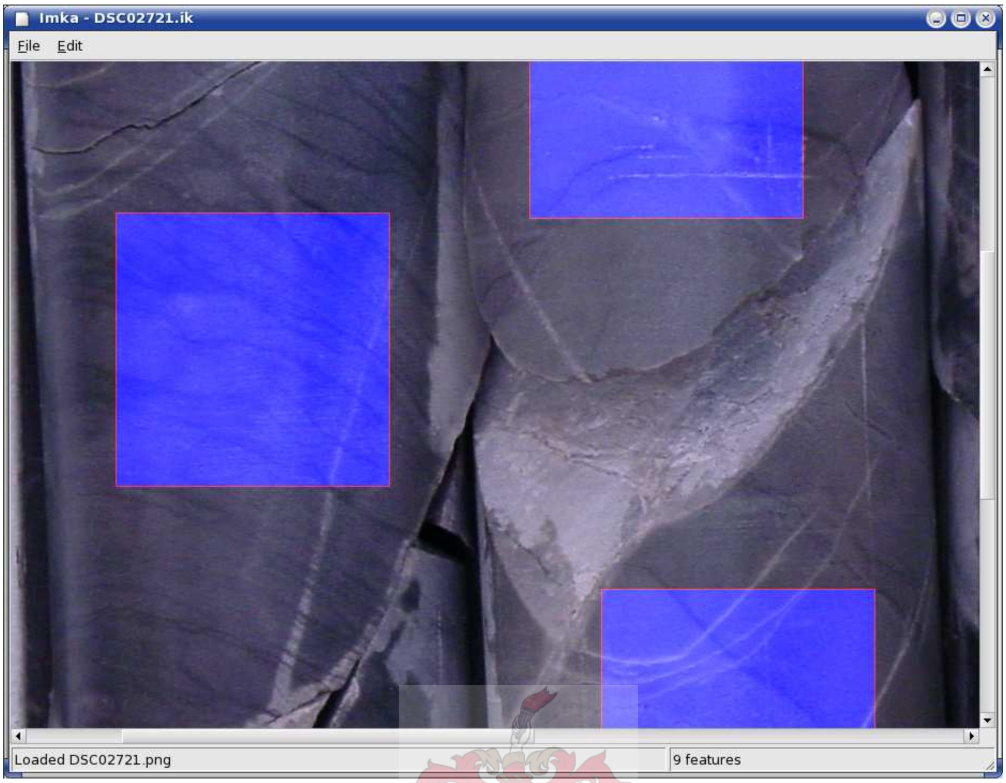
Contributed functions

Please see Appendix B for a listing of functions contributed to the Octave Forge.

5.3.2 Feature Selection

As discussed in Section 5.2, only the central section of each core is used for feature extraction. Unfortunately, the cores produced by the mining process are not always intact. Some means was needed to extract the usable central, intact parts from photographs for further processing. A screenshot of the program written to achieve this goal is shown below (Figure 5.3.1).

The program was written in Python, with graphical user interface and image loading routines from the wxPython library. A core photograph is loaded, after which translucent blue rectangles (256 × 256 pixels) are placed and moved across the picture to mark suitable pieces of rock. The marked blocks can then be exported to separate images, ready for further processing. The positions of the blocks can also be saved in a text file for usage in Octave, or for later reloading and manipulation in the program itself.



Imka - DSC02721.ik

Loaded DSC02721.png

9 features

Figure 5.3.1: Screenshot of the image cutting software.

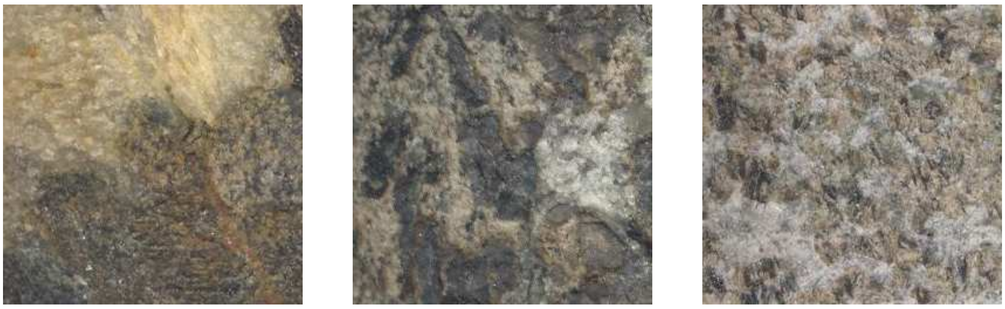
5.3.3 Download locations

Package	URL
GNU Octave	http://www.octave.org
Octave Forge	http://octave.sourceforge.net
WaveLab	http://www-stat.stanford.edu/~wavelab
Python	http://www.python.org
wxPython	http://www.wxpython.org

5.4 Texture Analysis

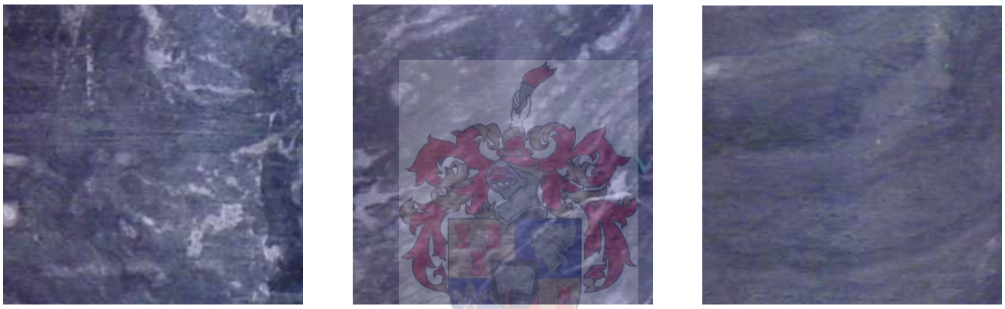
Two sets of data are used to set up the classifier. The first set, of which three samples are shown in Figure 5.4.1, is used to establish the ideal combination of feature vectors for good texture recognition. The second, shown in Figure 5.4.2, is used to test the classifier performance and to obtain an estimated error rate.

The two datasets (*platinum* and *diamond*) are broken up into two groups of samples each; the first, consisting of two thirds of the samples, is used for



(a) Type A (b) Type B (c) Type C

Figure 5.4.1: Three core samples from the Bushveld Igneous Complex [VvG86] (the *platinum* dataset).



(a) Stromatolitic (b) Ripple-laminated (c) Siliceous

Figure 5.4.2: Three types of dolomite from the Ghaap group [Beu78, Bar98, Ekk04] (the *diamond* dataset).

training while the second, containing the remainder of the samples, is used in evaluating (or testing) the classifier. It is to be noted that the *platinum* dataset (Figure 5.4.1) is much smaller than the *diamond* dataset; it also consists of three classes, but these were chosen to differ significantly in their texture properties. The *diamond* data set, on the other hand, is used as a whole — no samples were rejected beforehand to benefit discrimination.

The *platinum* set (with its *training* and *testing* samples) is firstly used to train and evaluate the classifier. Based on the outcome, adjustments are made to the system; most often, this involves changing the combination of the different types of features used. The tuning procedure is repeated, until the classifier performance is satisfactory. The *system parameters are then fixed*, not to be modified for the rest of the evaluation.

The *diamond* set is now used to train and evaluate the system, based on the choice of features obtained from trials under the *platinum* dataset. The error rate thus obtained is then an indication of the overall ability of the classifier to discriminate between different rock types.

Training

The texture features used are described in detail in Section 3.3.4. A three-level wavelet decomposition is done on each sample. At each of the three levels, the variance and mean of the horizontal, vertical and diagonal coefficients are calculated as features. A full wavelet decomposition is done, after which the wavelet energy signature is calculated as another feature. Together, the features form a feature vector — there is one vector per sample.

A Bayesian classifier (see Section 3.4.2) is trained using features from the *training set*. Based on this training, the classifier is asked to classify each feature vector from the *testing set*. It is also allowed to reject samples, based on the likelihood of their occurrance in all of the classes. The number of correct, incorrect and unclassified samples are noted. Two important quantities are calculated: the accuracy and the success rate. The accuracy is the number of successes as a percentage of all samples classified (and not rejected). The success rate is calculated the same way, but includes rejected samples. The error rate is the number of failures as a percentage of all samples.

Classification

We can view this experiment as a Bernoulli trial, assuming that no rejections take place (which is easy to achieve, by setting the rejection threshold to a low value). Each sample classification has one of two outcomes: success or failure. We repeat the procedure M times, once for each sample in the dataset. The binomial distribution models such situations.

The classification experiment produces the results shown in Table 5.1.

The actual classifier error rate τ differs from the estimated error rate ϵ. We would like to be able to predict the error rate of the classifier for future trials — it might be slightly higher, or lower, than the current estimate. To do this, a 95% *confidence interval* (τ_a, τ_b) is calculated, so that

$$P(\tau_a < \tau < \tau_b \mid \epsilon) = 0.95.$$

Table 5.1: Classifier results.

Class of Dolomite	Correct	Incorrect	Unclassified
Stromatolitic	17	8	0
Ripple-laminated	26	0	0
Siliceous	26	1	0

Number of samples	$M = 78$
Correct classifications	$w = 69$
Misclassifications	$m = 9$
Rejection threshold	$T = 40\%$ (certainty)
Estimated classifier error rate	$\epsilon = 11.54\%$
Estimated classifier success rate	$\mathbf{S = 88.46\%}$

The binomial probability density function (PDF) for N trials is defined as

$$f_X(x) = \sum_{k=0}^{N} \binom{N}{k} p^k (1-p)^{N-k} \delta(x-k)$$

where p is the probability of success (per trial), $(1-p)$ the probability of failure and where

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}$$

is the binomial coefficient. Accordingly, the probability of exactly m misclassifications taking place in M trials, given the true error rate τ is

$$P(m \mid \tau, M) = \frac{M!}{m!(M-m)!} \tau^m (1-\tau)^{M-m}.$$

For a large number of trials, the binomial PDF converges to the normal PDF. It is common practice to make use of this fact when calculating a confidence interval. The rule of thumb by Hodges and Lehman [HL70],

$$M\epsilon(1-\epsilon) \geq 10,$$

gives an indication of the size of M required to validate this assumption. In our case,

$$M\epsilon(1-\epsilon) = 78 \times 0.1154 \times (1 - 0.1154) = 7.9625 \leq 10,$$

which indicates that such an approximation is indeed invalid. Rather, we make use of the Beta distribution and the approximation formulas given in [MH96]. The 95% confidence limits are given as

$$\tau_a = C - H \quad \tau_b = C + H$$

where

$$C = (1 - 2\epsilon)\frac{1.96\sqrt{0.5}}{M+3} + \epsilon$$

and

$$H = \begin{cases} C & \text{if } m \leq 1 \\ 1.96\sqrt{\frac{\epsilon(1-\epsilon)}{M+2.5}} & \text{otherwise} \end{cases}.$$

For our experiment, we therefore have the 95% confidence interval on the true classification error as

$$5.88\% < \tau < 19.83\%.$$

In other words, we can say with 95% confidence that the texture classifier will not obtain a worse success rate than 80% under similar circumstances.

The accuracy of the classifier can be adjusted by setting the rejection threshold. A high threshold allows the classifier to reject samples more easily, increasing the accuracy but decreasing the overall success rate, as shown in Figure 5.4.3.

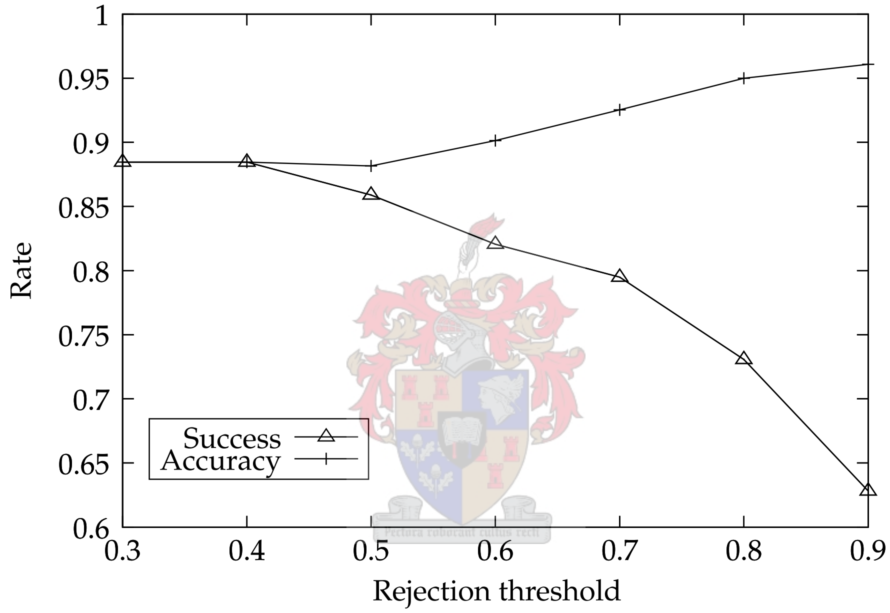


Figure 5.4.3: Accuracy and success rates as a function of the rejection threshold.

Chapter 6

Conclusion

This thesis described the foundation of wavelet theory, of shortest path techniques, of texture analysis and of Bayesian pattern classification.

Wavelet theory was presented in terms of multi-resolution analysis and the link to digital filters prominently emphasised. By choosing filters carefully, wavelets with desirable properties can be generated. Although these wavelets may never be calculated, their shapes dictate the applications of the wavelet transform.

Wavelet theory is truly a magnificent fusion of ideas originating in vastly different scientific disciplines. It has proved to be especially well suited to the problem of texture analysis. Different classical approaches to texture classification were explained, after which a tour of more modern wavelet techniques was given. A Bayesian classifier, chosen for its simplicity and effectiveness, was trained using wavelet features. These features were calculated from different levels of wavelet coefficients, and consisted of wavelet signatures (modified to describe normal distributions) and our cumulative energy signatures. The classifier was able to distinguish between different rocky textures (all very much alike to the untrained human eye). The high success rate allows us to say with confidence that the techniques (the combination of a Bayesian classifier and the described wavelet features) are likely to perform well on similar rock-texture classification problems.

In addition, we searched for thin chromitite stringers in "noisy" pyroxenite rock surfaces. An algorithm was designed to locate such layers in long borehole core photographs. The layers were isolated using shortest path techniques, after which a sinusoidal curve was fitted, using a non-linear least squares fitting. The parameters of these curves were used to determine the orientation of the layers in space. Whole cores containing these stringers are scarce (chromitite layers are brittle, causing the core to break), thus the algorithms were mainly evaluated using simulated data and few field samples. Yet, these experiments were successful, suggesting that this novel combination of techniques has merit.

In the future, we hope that more data will become available to support the results obtained thus far. As for wavelets: they already seem to have carved a niche for themselves in the exciting world of image processing.

6.1 Further research

Multi-scale methods are not limited to wavelets and their applications. One example of a related field of research is that of *beamlets*, *curvlets* and other line-integral representations, which tie in closely with graph theory. In the beamlet domain, an image is represented as a collection of coefficients obtained by calculating integrals along predefined line segments. These segments are described in a dyadically organised *beamlet dictionary*, containing lines at different orientations, scales and locations [DH00].

Beamlets have been successfully used to identify paths in signals with very low signal to noise ratios. This has direct application to the location of thin layers in borehole cores. A search for connected line segments conforming to a predefined set of requirements can produce border descriptions for arbitrary objects — chromitite intrusions, in the case of pyroxenite cores.

The field of wavelets is growing exponentially and this thesis does not cover all the possible approaches to the widely defined "texture classification" problem. The unique attributes specific to natural textures — like the interesting distribution of energy at different decomposition levels discussed in Section 3.3.3 — need to be fully explored and understood. It has been suggested that Markov random field processes over wavelet coefficients would be a good approach to texture characterisation [Mal99, p. 160]. This method should deliver results similar to those obtained using co-occurrence statistics [CJ83]. There are a number of fields in wavelet analysis to be further explored, like the packet transform, multi-wavelets and wavelet frames. The edge effects caused by the periodic padding, ignored in our experiments for the sake of simplicity, can be removed by using symmetric signal extension, or specific boundary wavelets.

In his extensive treatment of the topic, Van de Wouwer [vdW98] suggests the use of non-separable wavelets for texture analysis. The separable wavelets, while easy to implement, are highly orientation specific (in the $0°$, $45°$ and $90°$ directions). Using colour textures and features are mentioned.

The algorithms in this thesis were evaluated without performing preprocessing on any input data. This allowed us to evaluate the robustness of the algorithms. Also, in-borehole cameras are becoming more commonplace, negating the need for geometric correction (but not for illumination equilisation) in photo samples. These corrections should, however, normally be applied. Sklansky [Skl78] states that "preprocessing, such as linear filtering or histogram transformation, has significant effects on the performance of segmenters and feature extractors."

For texture discrimination, we assumed that each class is distributed normally. This may or may not be true — with more data available, this aspect can be explored and distributions like Gaussian mixture models may be considered.

Appendix A

Useful Mathematical Results

A.1 The error function at infinity

The error function [Pap65] is defined as

$$\operatorname{erf}(x) = F(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-y^2/2} dy.$$

We wish to calculate the value of the error function at infinity. Start by squaring $F(\infty)$ [Her04].

$$\begin{aligned} F^2(\infty) &= \frac{1}{2\pi} \int_0^\infty e^{-y^2/2} dy \int_0^\infty e^{-v^2/2}\, dv \\ &= \frac{1}{2\pi} \int_0^\infty \int_0^\infty e^{-(y^2+v^2)/2}\, dy\, dv. \end{aligned}$$

Now write this double integral in polar coordinates as

$$\begin{aligned} F^2(\infty) &= \frac{1}{2\pi} \int_0^{\pi/2} \int_0^\infty e^{-r^2/2}\, r\, dr\, d\theta \\ &= \frac{1}{4} \int_0^\infty r e^{-r^2/2}\, dr. \end{aligned}$$

Making the substitution $u = r^2/2$, this becomes

$$\begin{aligned} F^2(\infty) &= \frac{1}{4} \int_0^\infty e^{-u} du \\ &= \frac{1}{4} \left(-e^{-u}\right)\Big|_0^\infty \\ &= \frac{1}{4}. \end{aligned}$$

Therefore,

$$F(\infty) = \frac{1}{2}.$$

A.2 Maximising the generalised Rayleigh quotient

We wish to show that a function of the form

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}$$

can be written as the Rayleigh quotient when $\mathbf{A}$ and $\mathbf{B}$ are symmetric matrices and $\mathbf{B}$ is a positive definite[1] matrix. Symmetric positive definite matrices can be factorised as $\mathbf{B} = \mathbf{P}^T\mathbf{P}$ and therefore

$$\begin{aligned} J(\mathbf{w}) &= \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{P}^T\mathbf{P}\mathbf{w}} \\ &= \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{(\mathbf{P}\mathbf{w})^T(\mathbf{P}\mathbf{w})}. \end{aligned}$$

Let $\mathbf{y} = \mathbf{P}\mathbf{w}$ (which implies that $\mathrm{w} = \mathbf{P}^{-1}\mathrm{y}$), then

$$\begin{aligned} J(\mathbf{y}) &= \frac{\mathbf{y}^T\mathbf{P}^{-T}\mathbf{A}\mathbf{P}^{-1}\mathbf{y}}{\mathbf{y}^T\mathbf{y}} \\ &= \frac{\mathbf{y}^T\mathbf{R}\mathbf{y}}{\mathbf{y}^T\mathbf{y}}. \end{aligned} \tag{A.2.1}$$

Equation (A.2.1) is in the form of a Rayleigh quotient. Note that $\mathbf{R}$ is symmetric, since

$$\begin{aligned} \mathbf{R} &= \mathbf{P}^{-T}\mathbf{A}\mathbf{P}^{-1} \\ &= \mathbf{P}^{-T}\mathbf{S}^T\mathbf{S}\mathbf{P}^{-1} \\ &= \left(\mathbf{S}\mathbf{P}^{-1}\right)^T\left(\mathbf{S}\mathbf{P}^{-1}\right) \\ &= \mathbf{Q}^T\mathbf{Q} \end{aligned}$$

and

$$\mathbf{R}^T = \mathbf{R}.$$

The N eigenvectors of $\mathbf{R}$, $\{\mathbf{q}_1 \ldots \mathbf{q}_\mathbf{N}\}$, therefore form an orthonormal basis [Str93, p. 275] in which $\mathbf{y}$ can be expressed as

$$\mathbf{y} = c_1\mathbf{q}_1 + c_2\mathbf{q}_2 + \ldots + c_N\mathbf{q}_N.$$

[1] A matrix $\mathbf{B}$ is positive definite if $\mathbf{x}^T\mathbf{B}\mathbf{x} > 0$ for every non-zero vector $\mathbf{x}$.

The Rayleigh quotient of (A.2.1) now becomes

$$\begin{aligned} J(\mathbf{y}) &= \frac{\left(c_1\mathbf{q_1} + c_2\mathbf{q}_2 + \ldots + c_N\mathbf{q}_N\right)^T \left(c_1\lambda_1\mathbf{q_1} + c_2\lambda_2\mathbf{q}_2 + \ldots + c_N\lambda_N\mathbf{q}_N\right)}{\left(c_1\mathbf{q_1} + c_2\mathbf{q}_2 + \ldots + c_N\mathbf{q}_N\right)^T \left(c_1\mathbf{q_1} + c_2\mathbf{q}_2 + \ldots + c_N\mathbf{q}_N\right)} \\ &= \frac{c_1^2\lambda_1 + c_2^2\lambda_2 + \ldots + c_N^2\lambda_N}{c_1^2 + c_2^2 + \ldots + c_N^2}. \end{aligned}$$

The maximum value of this expression is the value of the largest eigenvalue, λ_{max}, since

$$\begin{aligned} J(\mathbf{y}) &= \frac{c_1^2(\lambda_{max} - \Delta_1) + c_2^2(\lambda_{max} - \Delta_2) + \ldots + c_N^2(\lambda_{max} - \Delta_N)}{c_1^2 + c_2^2 + \ldots + c_N^2} \\ &= \lambda_{max} - \frac{c_1\Delta_1 + c_2\Delta_2 + \ldots + c_N\Delta_N}{c_1^2 + c_2^2 + \ldots + c_N^2} \\ &\leq \lambda_{max}. \end{aligned}$$

The expression is at its maximum when $\mathbf{y}$ is the eigenvector of $\mathbf{R}$ associated with λ_{max}.

How are the eigenvalues of $\mathbf{R}$ related to $\mathbf{A}$ and $\mathbf{B}$? Recall that

$$\mathbf{R} = \mathbf{P}^{-T}\mathbf{A}\mathbf{P}^{-1}$$

and that

$$\mathbf{B} = \mathbf{P}^T\mathbf{P}.$$

If we assume that $\mathbf{B}$ is non-singular, it then follows that

$$\begin{aligned} \mathbf{R} &= \mathbf{P}\mathbf{B}^{-1}\mathbf{A}\mathbf{P}^{-1} \\ &= \mathbf{P}\mathbf{M}\mathbf{P}^{-1} \end{aligned}$$

and that $\mathbf{R}$ is similar to $\mathbf{B}^{-1}\mathbf{A} = \mathbf{M}$. Similar matrices share the same eigenvalues and their eigenvectors are related. If $\mathbf{x}$ is an eigenvector of $\mathbf{M}$, then $\mathbf{P}\mathbf{x}$ is an eigenvector of $\mathbf{R}$.

Let $\mathbf{v}_{max}(\mathbf{R})$ denote the eigenvector of matrix $\mathbf{R}$ associated with the largest eigenvalue. Since the value of $\mathbf{y}$ that maximises $J(\mathbf{y})$ is

$$\mathbf{y} = \mathbf{v}_{max}(\mathbf{R}) = \mathbf{P}\mathbf{v}_{max}(\mathbf{B}^{-1}\mathbf{A}),$$

we can now find $\mathbf{w}$. We know $\mathbf{w} = \mathbf{P}^{-1}\mathbf{y}$ and therefore

$$\mathbf{w} = \mathbf{P}^{-1}\mathbf{y} = \mathbf{v}_{max}(\mathbf{B}^{-1}\mathbf{A}).$$

We have therefore shown that, for the case where $\mathbf{A}$ and $\mathbf{B}$ are symmetric and $\mathbf{B}$ is positive-definite and non-singular, the expression

$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{A}\mathbf{w}}{\mathbf{w}^T\mathbf{B}\mathbf{w}}$$

is at a maximum when $\mathbf{w}$ is the eigenvector associated with the largest eigenvalue of $\mathbf{B}^{-1}\mathbf{A}$. The case in which $\mathbf{B}$ is singular is discussed in [TK99, p. 163].

A.3 Lagrangian multipliers

Given a function

$$f(x_1, x_2 \dots x_3) = f(\mathbf{x})$$

we seek the extremum of that function, subject to the constraint $g(\mathbf{x}) = 0$. At the extremum it is true [FT94] that

$$\nabla f(\mathbf{x}) = \lambda \nabla g(\mathbf{x}) \tag{A.3.1}$$

$$\text{and}$$

$$g(\mathbf{x}) = 0. \tag{A.3.2}$$

Define the Lagrangian function [Kle] as

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda g(\mathbf{x})$$

then

$$\nabla L(\mathbf{x}, \lambda) = 0 \tag{A.3.3}$$

which satisfies (A.3.1) and (A.3.2), since

$$\begin{aligned} \frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} &= \nabla f(\mathbf{x}) - \lambda \nabla g(\mathbf{x}) \\ &\text{and} \\ \frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} &= -g(\mathbf{x}). \end{aligned}$$

The solution of (A.3.3) is the constrained value of $\mathbf{x}$ at an extremum [DHS01, p. 610].

A.4 The gradient of $\mathbf{x}^T\mathbf{A}\mathbf{x}$

We calculate the gradient

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

where

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$$

and $\mathbf{A}$ is a symmetrical matrix. We can express $f(\mathbf{x})$ as the sum

$$f(\mathbf{x}) = \sum_i \sum_j A_{ij} x_i x_j,$$

for which

$$\frac{\partial f(\mathbf{x})}{\partial x_s} = \frac{\partial}{\partial x_s}\left[\sum_i \sum_j A_{ij} x_i x_j\right]$$

holds. This is equivalent to

$$\begin{aligned} \frac{\partial f(\mathbf{x})}{\partial x_s} &= \frac{\partial}{\partial x_s}\left[\sum_i \sum_j A_{ij} x_i x_j\right] \\ &= \sum_i \sum_j \frac{\partial}{\partial x_s}(A_{ij} x_i) x_j + \sum_i \sum_j A_{ij} x_i \frac{\partial}{\partial x_s}(x_j) \\ &= \sum_j A_{sj} x_j + \sum_i A_{is} x_i \end{aligned}$$

and because $\mathbf{A}$ is symmetrical, this reduces to

$$\frac{\partial f(\mathbf{x})}{\partial x_s} = 2\sum_j A_{sj} x_j$$

which leads to the solution

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x}.$$

Appendix B

Code Contributions

The following three algorithms were contributed to and accepted into the Octave Forge.

Listing B.1: Two dimensional convolution using the fast Fourier transform.

```
## Copyright (C) 2004 Stefan van der Walt <stefan@sun.ac.za>
##
## This program is free software; redistribution and use in source and
## binary forms, with or without modification, are permitted provided that
## the following conditions are met:
##
## 1. Redistributions of source code must retain the above copyright
##    notice, this list of conditions and the following disclaimer.
## 2. Redistributions in binary form must reproduce the above copyright
##    notice, this list of conditions and the following disclaimer in the
##    documentation and/or other materials provided with the distribution.
##
## THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ‘‘AS IS’’ AND
## ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
## IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
## ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
## FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
## DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
## OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
## HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
## LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
## OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
## SUCH DAMAGE.

## FFTCONV2 Convolve 2 dimensional signals using the FFT.
##
## usage: fftconv2(a, b[, shape])
##        fftconv2(v1, v2, a, shape)
##
```

```
## This method is faster but less accurate for large a,b.  It
## also uses more memory. A small complex component will be
## introduced even if both a and b are real.
##
## see also: conv2

## Author: Stefan van der Walt <stefan@sun.ac.za>
## Date: 2004

function X = fftconv2(varargin)
    if (nargin < 2)
        usage("fftconv2(a,b[,shape]) or fftconv2(v1, v2, a, shape)")
    endif

    shape = "full";
    rowcolumn = 0;

    if ((nargin > 2) && ismatrix(varargin{3}))
        ## usage: fftconv2(v1, v2, a[, shape])

        rowcolumn = 1;
        v1 = varargin{1}(:)’;
        v2 = varargin{2}(:);
        orig_a = varargin{3};

        if (nargin == 4) shape = varargin{4}; endif
    else
        ## usage: fftconv2(a, b[, shape])

        a = varargin{1};
        b = varargin{2};
        if (nargin == 3) shape = varargin{3}; endif

    endif

    if (rowcolumn)
        a = fftconv2(orig_a, v2);
        b = v1;
    endif

    ra = rows(a);
    ca = columns(a);
    rb = rows(b);
    cb = columns(b);

    A = fft2(impad(a, [0 cb-1], [0 rb-1]));
    B = fft2(impad(b, [0 ca-1], [0 ra-1]));
    X = ifft2(A.*B);

    if (rowcolumn)
        rb = rows(v2);
        ra = rows(orig_a);
        cb = columns(v1);
        ca = columns(orig_a);
    endif

    if strcmp(shape,"same")
        r_top = ceil((rb + 1) / 2);
        c_top = ceil((cb + 1) / 2);
        X = X(r_top:r_top + ra - 1, c_top:c_top + ca - 1);
    elseif strcmp(shape, "valid")
        X = X(rb:ra, cb:ca);
    endif
endfunction

%!# usage: fftconv2(a,b,[, shape])
%!shared a,b
%! a = repmat(1:10, 5);
%! b = repmat(10:-1:3, 7);
%!assert(norm(fftconv2(a,b)-conv2(a,b)), 0, 1e6*eps)
%!assert(norm(fftconv2(b,a)-conv2(b,a)), 0, 1e6*eps)
%!assert(norm(fftconv2(a,b,’full’)-conv2(a,b,’full’)), 0, 1e6*eps)
%!assert(norm(fftconv2(b,a,’full’)-conv2(b,a,’full’)), 0, 1e6*eps)
%!assert(norm(fftconv2(a,b,’same’)-conv2(a,b,’same’)), 0, 1e6*eps)
%!assert(norm(fftconv2(b,a,’same’)-conv2(b,a,’same’)), 0, 1e6*eps)
%!assert(isempty(fftconv2(a,b,’valid’)));
%!assert(norm(fftconv2(b,a,’valid’)-conv2(b,a,’valid’)), 0, 1e6*eps)

%!# usage: fftconv2(v1, v2, a[, shape])
%!shared x,y,a
%! x = 1:4; y = 4:-1:1; a = repmat(1:10, 5);
%!assert(norm(fftconv2(x,y,a)-conv2(x,y,a)), 0, 1e6*eps)
%!assert(norm(fftconv2(x,y,a,’full’)-conv2(x,y,a,’full’)), 0, 1e6*eps)
%!assert(norm(fftconv2(x,y,a,’same’)-conv2(x,y,a,’same’)), 0, 1e6*eps)
%!assert(norm(fftconv2(x,y,a,’valid’)-conv2(x,y,a,’valid’)), 0, 1e6*eps)

%!demo
%! ## Draw a cross
%! N = 100;
%! [x,y] = meshgrid(-N:N, -N:N);
%! z = 0*x;
%! z(N,1:2*N+1) = 1; z(1:2*N+1, N) = 1;
%! imshow(z);
```

```
%!
%! ## Draw a sinc blob
%! n = floor(N/10);
%! [x,y] = meshgrid(-n:n, -n:n);
%! b = x.^2 + y.^2; b = max(b(:)) - b; b = b / max(b(:));
%! imshow(b);
%!
%! ## Convolve the cross with the blob
%! imshow(real(fftconv2(z, b, ’same’)*N))
```

Listing B.2: The straight line Hough transform.

```
/* Copyright (C) 2004 Stefan van der Walt <stefan@sun.ac.za>

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
   met:

   1. Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
   2. Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.

  THIS SOFTWARE IS PROVIDED BY THE AUTHOR ‘‘AS IS’’ AND ANY EXPRESS OR
  IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
  DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
  GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
  IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
  IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. */

#include <octave/oct.h>

DEFUN_DLD( houghtf, args, , "\
\
usage: [H, R]  = houghtf(I[, angles])\n\
\n\
  Calculate the straight line Hough transform of an image.\n\
\n\
  The image, I, should be a binary image in [0,1].  The angles are given\n\
  in degrees and defaults to -90..90.\n\
\n\
  H is the resulting transform, R the radial distances.\n\
\n\
  See also: Digital Image Processing by Gonzales & Woods (2nd ed., p. 587)\n") {

    octave_value_list retval;
    bool DEF_THETA = false;

    if (args.length() == 1) {
        DEF_THETA = true;
    } else if (args.length() != 2) {
        print_usage("houghtf");
        return retval;
    }

    Matrix I = args(0).matrix_value();

    ColumnVector thetas = ColumnVector();
    if (!DEF_THETA) {
        thetas = ColumnVector(args(1).vector_value());
    } else {
        thetas = ColumnVector(Range(-90,90).matrix_value());
    }

    if (error_state) {
        print_usage("houghtf");
        return retval;
    }

    thetas = thetas / 180 * M_PI;

    int r = I.rows();
    int c = I.columns();

    Matrix xMesh = Matrix(r, c);
    Matrix yMesh = Matrix(r, c);
    for (int m = 0; m < r; m++) {
        for (int n = 0; n < c; n++) {
            xMesh(m, n) = n+1;
            yMesh(m, n) = m+1;
        }
    }

    Matrix size = Matrix(1, 2);
    size(0) = r; size(1) = c;
    double diag_length = sqrt( size.sumsq()(0) );
    int nr_bins = 2 * (int)ceil(diag_length) - 1;
    RowVector bins = RowVector( Range(1, nr_bins).matrix_value() ) - (int)ceil(
        nr_bins/2);
    Matrix J = Matrix(bins.length(), 0);

    for (int i = 0; i < thetas.length(); i++) {
        double theta = thetas(i);
        ColumnVector rho_count = ColumnVector(bins.length(), 0);

        double cT = cos(theta); double sT = sin(theta);
        for (int x = 0; x < r; x++) {
            for (int y = 0; y < c; y++) {
                if ( I(x, y) == 1 ) {
                    int rho = (int)round( cT*x + sT*y );
                    int bin = (int)(rho - bins(0));
                    if ( (bin > 0) && (bin < bins.length()) ) {
                        rho_count( bin )++;
                    }
                }
            }
        }

        J = J.append( rho_count );
    }

    retval.append(J);
    retval.append(bins);
    return retval;

}

/*
%!test
%! I = zeros(100, 100);
%! I(1,1) = 1; I(100,100) = 1; I(1,100) = 1; I(100, 1) = 1; I(50,50) = 1;
%! [J, R] = houghtf(I); J = J / max(J(:));
%! assert(size(J) == [length(R) 181]);
%!

%!demo
%! I = zeros(100, 150);
%! I(30,:) = 1; I(:, 65) = 1; I(35:45, 35:50) = 1;
%! for i = 1:90, I(i,i) = 1;endfor
%! I = imnoise(I, ’salt & pepper’);
%! imshow(I);
%! J = houghtf(I); J = J / max(J(:));
%! imshow(J, bone(128), ’truesize’);
*/
```

Listing B.3: Gray level co-occurrence matrices.

```
/* Copyright (C) 2004 Stefan van der Walt <stefan@sun.ac.za>

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions are
   met:

   1. Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
   2. Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.

  THIS SOFTWARE IS PROVIDED BY THE AUTHOR ‘‘AS IS’’ AND ANY EXPRESS OR
  IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
  WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY
  DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
  GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
  INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER
  IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
  OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
  IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. */

#include <octave/oct.h>

DEFUN_DLD( graycomatrix, args, , "\
\
usage: P = graycomatrix(I, levels, distances, angles)\n\
\n\
  Calculate the gray-level co-occurrence matrix P = f(i,j,d,theta)\n\
  of a gray-level image.\n\
\n\
  P is a 4-dimensional matrix (histogram). The value P(i,j,d,theta)\n\
  is the number of times that gray-level j occurs at a distance ’d’ and\n\
  at an angle ’theta’ from gray-level j.\n\
\n\
  ’I’ is the input image which should contain integers in [0, levels-1],\n\
  where ’levels’ indicate the number of gray-levels counted (typically\n\
  256 for an 8-bit image).  ’distances’ and ’angles’ are vectors of\n\
  the different distances and angles to use.\n" ) {

    // 4-dimensional histogram
    // P = f(i, j, d, theta) where i and j are gray levels
    // See Pattern Recognition Engineering (Morton Nadler & Eric P. Smith)
    octave_value_list retval;

    if (args.length() != 4) {
        print_usage("graycomatrix");
        // 'I' must be integer values [0, nr_of_levels-1]

        return retval;
    }

    // Input arguments
    Matrix I = args(0).matrix_value();
    int L = args(1).int_value();
    ColumnVector d = ColumnVector(args(2).vector_value());
    ColumnVector th = ColumnVector(args(3).vector_value());

    if (error_state) {
        print_usage("graycomatrix");
        return retval;
    }

    // Create output NDArray, P
    dim_vector dim = dim_vector();
    dim.resize(4);
    dim(0) = L; dim(1) = L; dim(2) = d.length(); dim(3) = th.length();
    NDArray P = NDArray(dim, 0);

    // Run through image
    int d_max = (int)ceil(d.max());
    int cnt = 0;

    for (int r = 0; r < I.rows(); r++) {
        for (int c = 0; c < I.columns(); c++) {
            int i = (int)I(r,c);

            for (int d_idx = 0; d_idx < d.length(); d_idx++) {
                int d_val = (int)d(d_idx);
                for (int th_idx = 0; th_idx < th.length(); th_idx++) {

                    double angle = th(th_idx);

                    int row = r + (int)round(cos(angle) * d_val);
                    int col = c - (int)round(sin(angle) * d_val);

                    if ( ( row >= 0 ) && ( row < I.rows() ) &&
                         ( col >= 0 ) && ( col < I.cols() ) ) {

                        int j = (int)I(row, col);
                        if (i >= 0 && i < L && j >= 0 && j < L) {
                            Array<int> coord = Array<int> (4);
                            coord(0) = i;
                            coord(1) = j;
                            coord(2) = d_idx;
                            coord(3) = th_idx;

                            P(coord)++;
                        } else {
                            warning("Image contains invalid gray-level! (%d, %d)
                                ", i, j);
                        }
                    }

                }
            }

        }
    }

    retval.append(P);
    return retval;

}

/*

%!shared a
%!test
%!  a = [0 0 0 1 2;
%!       1 1 0 1 1;
%!       2 2 1 0 0;
%!       1 1 0 2 0;
%!       0 0 1 0 1];
%!  squeeze(graycomatrix(a, 3, 1, -pi/4)) == [4 2 0;
%!                                     2 3 2;
%!                                     1 2 0];
%!
%!assert(size(graycomatrix(a, 3, 1:5, [0:3]*-pi/4)), [3, 3, 5, 4])

%!demo
%!
%!  # Pattern Recognition Engineering (Nadler & Smith)
%!  # Digital Image Processing (Gonzales & Woods), p. 668
%!
%!  a = [0 0 0 1 2;
%!      1 1 0 1 1;
%!      2 2 1 0 0;
%!      1 1 0 2 0;
%!      0 0 1 0 1];
%!
%!  graycomatrix(a, 3, 1, [0 1]*-pi/4)
%!
*/
```

Appendix C

Geological Logs

C.1 A section of the UG2 system of the Bushveld Igneous Complex

Please see the logs on the following pages.

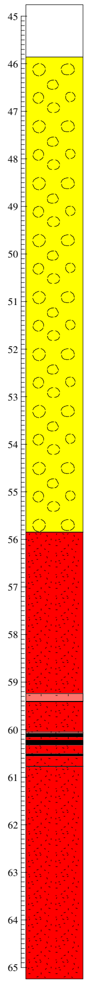
svsWAL_SAMPLING

GEOLOGICAL LOG OF: Rustenburg Section - Waterval

SECTION ID: *WV171* **Geologist:** *JM* A

Date Started: *2003/07/24* **Date Plotted:** *2003/08/21* *Scale 1:100*

Sampling *Dip* *Description*



10

45.860 - 55.850 *Poikilitic Anorthosite* Grainsize: Coarse Texture: Poikiltic Fabric: Inequigranular **UG2 Anorthosite**

12

12 chromite

55.848 - 55.850 *Chromitite* Grainsize: fine Texture: Stringers Fabric: equigranular

55.850 - 65.440 *Feldspathic pyroxenite* Grainsize: medium Texture: Cumulate Fabric: Inequigranular Comments: Augite phenocrysts below triplets **UG2 Pyroxenite (below triplets)**

59.240 - 59.405 *Pegmatoidal Feldspathic pyroxenite* Grainsize: Coarse Texture: PEG Fabric: Inequigranular

59.405 - 59.410 *Chromitite* Grainsize: fine Texture: Stringers Fabric: equigranular

10 chromite
10
5 chromite
9 chromite
3 chromite
5 chromite
3

60.050 - 60.080 *Pegmatoidal Feldspathic pyroxenite* Grainsize: Coarse Texture: PEG Fabric: Inequigranular

60.080 - 60.150 *Chromitite* Grainsize: fine Texture: Cumulate Fabric: equigranular **UG2 Triplet 3**

60.225 - 60.320 *Chromitite* Grainsize: fine Texture: Cumulate Fabric: equigranular Comments: Stringers at base **UG2 Triplet 2**

60.510 - 60.550 *Chromitite* Grainsize: fine Texture: Stringers Fabric: equigranular **UG2 Triplet 1**

60.770 - 60.772 *Chromitite* Grainsize: fine Texture: Stringers Fabric: equigranular svsWAL_SAMPLING

GEOLOGICAL LOG OF: Rustenburg Section - Waterval

SECTION ID: *WV171*

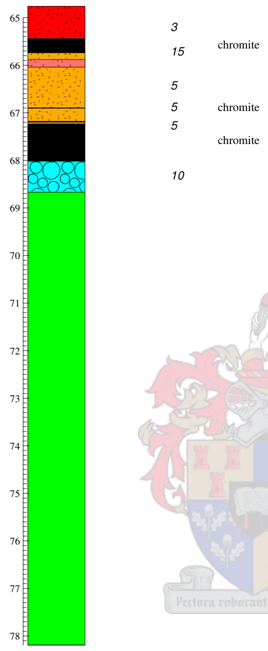
Geologist: *JM*

A

Date Started: *2003/07/24*

Date Plotted: *2003/08/21* *Scale 1:100*

Sampling ***Dip*** ***Description***



65.440 - 65.730 *Chromitite* Grainsize: fine Texture: Cumulate Fabric: equigranular **UG2 main leader**

65.730 - 65.740 *Anorthosite with chrome stringers* Grainsize: Fine to medium Texture: Stringers Fabric: Inequigranular Comments: Chromite stringers

65.730 - 67.180 *Feldspathic pyroxenite* Grainsize: Fine to medium Texture: Porphyritic Fabric: Inequigranular Comments: Augite phenocrysts **UG2 Pyroxenite (below leader)**

65.880 - 66.040 *Pegmatoidal Feldspathic pyroxenite* Grainsize: Coarse Texture: PEG Fabric: Inequigranular

66.900 - 66.901 *Chromitite* Grainsize: fine Texture: Stringers Fabric: equigranular **CR Stringer above UG2 & below Leader**

67.180 - 68.020 *Chromitite* Grainsize: fine Texture: Cumulate Fabric: equigranular **UG2 chromitite**

67.200 - 67.230 *Feldspathic pyroxenite* Grainsize: Fine to medium Texture: Cumulate Fabric: Inequigranular **UG2 Main Seam Parting**

68.020 - 68.670 *Pegmatoidal Feldspathic pyroxenite* Grainsize: Coarse Texture: PEG Fabric: Inequigranular **Pegmatoidal Feldspathic Pyroxenite (UG2 Footwall)**

68.670 - 78.190 *Melanorite* Grainsize: medium Texture: Cumulate Fabric: Inequigranular **UG2FW**

78.190 - 78.190 *End of hole*

Bibliography

[AM03] A. Ahmadian and A. Mostafa. An efficient texture classification algorithm using Gabor wavelets. In *Annu Int Conf IEEE Eng Med Biol Proc*, pages 930–933. Institute of Electrical and Electronics Engineers Inc., 2003. 1, 37

[Bar98] Wayne Peter Barnett. The structural and engineering geology of the country rock at Finsch mine. Master's thesis, University of Cape Town, 1998. ix, 72, 78

[Beu78] Nicolas Johannes Beukes. *Die Karbonaatgesteentes en Ysterformasies van die Ghaap-groep van die Transvaal-supergroep in Noord-Kaapland*. PhD tesis, Randse Afrikaanse Universiteit, 1978. ix, 72, 78

[BY97] M. Buckley and J. Yang. Regularised shortest-path extraction. *Pattern Recognition Letters*, 18(7):621–629, 1997. 61

[Cas98] Michael Anthony Casey. *Auditory Group Theory with Applications to Statistical Basis Methods for Structured Audio*. PhD thesis, Massachusetts Institute of Technology, 1998. 51

[CDE96] Boaz Cohen, Its'hak Dinstein, and Moshe Eyal. Computerized classification of color textured perthite images. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 2, pages 601–605, 1996. 1

[CJ83] George R. Cross and Anil K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):44–58, January 1983. 1, 84

[CK93] Tianhorng Chang and C.-C. Jay Kuo. Texture Analysis and Classification with Tree-Structured Wavelet Transform. *IEEE Transactions on Image Processing*, 2(4):429–441, October 1993. 1, 40, 42

[DH00] David L. Donoho and Xiaoming Huo. Beamlets and multiscale image analysis. Technical report, Stanford University and Georgia Institute of Technology, 2000. 84

[DHS01] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, New York, 2001. 46, 47, 53, 54, A–4

[Ekk04] Josef Ekkerd. The dolomites at Finsch mine. Private communications, October 2004. ix, 78

[FT94] Ross L. Finney and George B. Thomas. *Calculus*. Addison-Wesley Publishing Company, 2nd edition, 1994. A–4

[Gab46] D. Gabor. Theory of communication. *Journal of the IEE*, 93:429–457, 1946. 8

[GvL83] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. The John Hopkins University Press, 1983. 54

[GW01] Rafael C. Gonzales and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2nd edition, 2001. 28, 32, 33, 34, 40

[Hei94] W.F. Heinz. *Diamond Drilling Handbook*. W.F. Heinz, 3rd edition, 1994. 1, 56

[Her04] B.M. Herbst. The value of the error function at infinity. Private communications, October 2004. A–1

[HL70] J.J.L. Hodges and E.L. Lehman. *Basic Concepts of Probability and Statistics*. Holden-Day, Oakland, CA, 1970. 80

[HPGM96] Jonathan Hall, Marco Ponzi, Mauro Gonfalini, and Giorgio Maletti. Automatic extraction and characterisation of geological features and textures from borehole images and core photographs. In *37th Annual Logging Symposium*. SPWLA, June 1996. 1

[HSD73] R.M. Haralick, K. Shanugan, and I. Dinstein. Texture features for image classification. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8(6):610–621, 1973. 1, 34

[HSL92] Ming-Huwi Horng, Yung-Nien Sun, and Xi-Zhang Lin. Texture feature coding method for clasification of liver sonography. *Computerized Medical Image and Graphics*, 26:33–42, 1992. 1, 34

[HTE94] Neil F. Hurley, David R. Thorn, and Sandra L.W. Eichelberger. Using borehole images for target-zone evaluation in horizontal wells. In *AAPG Bulletin*, volume 78, pages 238–246. American Association of Petroleum Geologists, February 1994. 2

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2003. 67

[JS94] Björn Jawerth and Wim Sweldens. An overview of wavelet based multiresolution analysis. *SIAM Review*, 36(3):377–412, September 1994. 38

[Kle] Dan Klein. Lagrange multipliers without permanent scarring. Internet. `http://www.cs.berkeley.edu/~klein`. A–4

[M^{+}01] Dana Mackenzie et al. Wavelets: Seeing the forest — and the trees. Beyond Discovery, http://www.beyonddiscovery.org, December 2001. 6

[Mal89] Stéphane G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989. 15, 44

[Mal99] Stéphane Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 2nd edition, 1999. x, xi, 6, 7, 11, 17, 28, 38, 84

[Mar63] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.*, 11(2):431–441, June 1963. 67

[MH96] J. Kent Martin and D.S. Hirschberg. Small sample statistics for classification error rates (ii): Confidence intervals and significance tests. Technical Report ICS-TR-96-22, Department of Information and Computer Science, University of California, Irvine, 1996. 81

[MS73] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM Journal on Numerical Analysis*, 10(2):241–256, April 1973. 54

[NS93] Morton Nadler and Eric P. Smith. *Pattern Recognition Engineering*. John Wiley & Sons, Inc., 1993. 33

[Pap65] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, Inc., 1965. 49, A–1

[Pla] Anglo Platinum. Review of mineral reserves and resources. Internet. `http://www.angloplatinum.com`. 1

[PM96] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing*. Prentice Hall, Upper Saddle River, New Jersey 07458, 3rd edition, 1996. x, 22, 25

[PS95] Devesh Patel and John Stonham. Accurate set-up of Gabor filters for texture classification. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 2501/2, pages 894–903. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, USA, 1995. 1, 37

[PTVF03] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2003. 54, 64, 65, 66, 67, 68

[PZP01] Jr. Peyton Z. Peebles. *Probability, Random Variables, and Random Signal Principles*. McGraw-Hill, Inc., 4th edition, 2001. 8, 33, 50, 64

[RBC$^+$92] M. B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael, editors. *Wavelets and their Applications*, pages 453–470. Jones and Bartlett, 1992. 40

[Skl78] Jack Sklansky. Image segmentation and feature extraction. *IEEE Transactions on Systems, Man and Cybernetics*, 8(4):237–247, April 1978. 1, 32, 85

[SN97] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1997. viii, xi, 19, 21, 23, 24, 26, 27

[SP03] Changming Sun and Stefano Pallottino. Circular shortest paths on regular grids. *Pattern Recognition*, 36(3):709–719, March 2003. 2, 62

[ST03] C. Starr and R. Taggart. *The Unity and Diversity of Life*. 10th edition, 2003. 47

[Str93] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 1993. 52, A–2

[TGF04] C.E. Thomaz, D.F. Gillies, and R.Q. Feitosa. A new covariance estimate for Bayesian classifiers in biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(2):214–223, February 2004. 49

[THK97] Baskar B. Thapa, Paul Hughett, and Kenzi Karasaki. Semi-automatic analysis of rock fracture orientations from borehole wall images. *Geophysics*, 62(1):129–137, January 1997. sinusoidal fitting by Hough transform. 2

[TK99] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 1999. A–4

[Uns95] Michael Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on Image Processing*, 4(11):1549–1560, November 1995. 1, 38

[VBL95] John D. Villasenor, Benjamin Belzer, and Judy Liao. Wavelet filter evaluation for image compression. *IEEE Transactions on Image Processing*, 4(8):1053–1060, August 1995. 39

[vdW98] Gert van de Wouwer. *Wavelets voor Multischaal Textuuranalyse*. PhD thesis, Universitaire Instelling Antwerpen, 1998. 1, 44, 84

[VH92] Martin Vitterli and Cormac Herley. Wavelets and filter banks: Theory and design. *IEEE Transactions on Signal Processing*, 40(9):2207–2232, September 1992. 26

[VvG86] C.F. Vermaak and G. von Gruenewaldt. Introduction to the Bushveld Complex. *Mineral Deposits of Southern Africa*, I and II:1021–1029, 1986. ix, 78