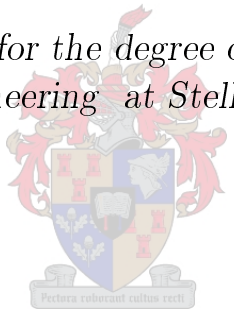
On-board Image Quality Assessment for a Satellite

by

Izak van Zyl Marais

Dissertation presented for the degree of Doctor of Philosophy in Electronic Engineering at Stellenbosch University



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

Promoters:

Prof W.H. Steyn
Prof J. du Preez

March 2009

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, and that I am the owner of the copyright thereof (unless to the extend explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
I.v.Z. Marais

Date: .....

Abstract

The downloading of images is a bottleneck in the image acquisition chain for low earth orbit, remote sensing satellites. An on-board image quality assessment system could optimise use of available downlink time by prioritising images for download, based on their quality.

An image quality assessment system based on measuring image degradations is proposed. Algorithms for estimating degradations are investigated. The degradation types considered are cloud cover, additive sensor noise and the defocus extent of the telescope.

For cloud detection, the novel application of heteroscedastic discriminant analysis resulted in better performance than comparable dimension reducing transforms from remote sensing literature. A region growing method, which was previously used on-board a micro-satellite for cloud cover estimation, is critically evaluated and compared to commonly used thresholding. The thresholding method is recommended. A remote sensing noise estimation algorithm is compared to a noise estimation algorithm based on image pyramids. The image pyramid algorithm is recommended. It is adapted, which results in smaller errors. A novel angular spectral smoothing method for increasing the robustness of spectral based, direct defocus estimation is introduced. Three existing spectral based defocus estimation methods are compared with the angular smoothing method.

An image quality assessment model is developed that models the mapping of the three estimated degradation levels to one quality score. A subjective image quality evaluation experiment is conducted, during which more than 18000 independent human judgements are collected. Two quality assessment models, based on neural networks and splines, are fitted to this data. The spline model is recommended.

The integrated system is evaluated and image quality predictions are shown to correlate well with human quality perception.

Opsomming

Die aflaai van beelde is 'n wurgplek in die afstandswaarneming satelliet-beeldverwerkingsketting. 'n Aanboord-beeldkwaliteit-bepalingstelsel kan, deur voorkeur te gee aan beelde met beter beeldkwaliteit, die beskikbare aflaaityd optimaal benut.

'n Beeldkwaliteit-bepalingstelsel, wat verlagings in beeldkwaliteit meet, word voorgestel. Algoritmes om verlagings af te skat word ondersoek. Die verlagings wat afgeskat word, is: additiewe sensor-ruis, wolkdekking en die hoeveelheid wat die teleskoop uit fokus is.

Vir wolkedeteksie toon die nuwe toepassing van heteroskedastiese diskriminant analise op afstandswaarneming aan dat die tegniek beter verrigting as vergelykbare dimensie-verlaging-tegnieke uit afstandswaarneming-literatuur lewer. 'n Gebied-groei-algoritme, wat voorheen aanboord van 'n mikrosatelliet vir wolkdekking bepaling gebruik is, word krities ge-evalueer en vergelyk met 'n meer algemene grysskaaldrempel-tegniek. Die drempel-tegniek word aanbeveel. 'n Satelliet-beeld ruisafskattingsalgoritme word vergelyk met 'n algoritme wat op beeldpiramiedes gebaseer is. Die piramiedemetode word aanbeveel. Die algoritme word aangepas met gevolg dat kleiner afskattingsfoute begaan word. 'n Nuwe hoekspektrale vergladdingsalgoritme, wat die robuustheid van spektraal gebaseerde, uit-fokus-afskattingsalgoritme verbeter, word ontwikkel. Die nuwe algoritme word met drie bestaande, spektraal gebaseerde uit-fokus-afskattingsalgoritmes vergelyk.

'n Beeldkwaliteit-beoordelingsmodel word ontwikkel wat die drie gemete verlagings op een kwaliteitspunt afbeeld. 'n Subjektiewe beeldkwaliteit-beoordelingseksperiment, waarin meer as 18000 onafhanklike menslike oordele versamel word, word uitgevoer. Twee beeldkwaliteitsmodelle, onderskeidelik op neurale netwerke en stuksgewyse polinome gebaseer, word op die versamelde data gepas. Die stuksgewyse-polinoommodel word aanbeveel.

Die geïntegreerde stelsel word getoets en lewer beeldkwaliteitskattings wat goed met menslike waarneming van beeldkwaliteit korreleer.

Acknowledgements

I thank the Lord for giving me the talents, discipline and ideas necessary for writing this dissertation. I gratefully acknowledge the funds received from the Wilhelm Frank bursary fund. I would like to thank the following people (in no specific order) for their contribution towards this project:

- Dr. Hanno Coetzer for his help with design of the dispertion measure.
- Dr. Ludwig Schwart for always being willing to explain difficult concepts in understandable language.
- All the dedicated people who helped to develop the various free, open source tools that were used: Ubuntu, Python and its various libraries, the IPython shell environment, LaTeX, the Kile and TeXnicCenter LaTeX editors, the SciTe editor, gcc, μCLinux, Doxygen.
- Eugene van Wyk for his help with getting started programming the Hico SH4 board.
- Prof. du Preez for his insights.
- My mother for tirelessly proofreading the dissertation even though she might not find it very entertaining.
- Prof. Steyn for his guidance and dependability.
- Wolfgang Lück for his guidance concerning existing algorithms and for providing many images on behalf of the Satellite Application Centre (part of the Council for Scientific and Industrial Research of South Africa).
- Corné van Daalen for his help with mathematical simplification.
- Everybody who participated in the subjective image quality assessment experiment. I decided to keep the duplicate names of those who participated in more than one experiment. Your masochism is greatly appreciated!

– *The 'blur' experiment:*
A Nonymous
Alistair Baldwin
Altus van Tonder
Andre Young
Arno Barnard
E. Hansmann
Farron Yssel
Gen Blan
Gerrit Kruger
Helgard van Rensburg
Henk Marais
Johan Schoonwinkel
Madelé van der Walt
Patricia Taylor
Patrick Duriez
Rinus Brand
Rudi Gaum
Stefan van der Walt
Tinus Stander
Waine Smith

– *The 'clouds' experiment:*
A Nonymous
Albert Swart
Eugene Pretorius
Gerrit Kruger
Helgard van Rensburg
Henk Marais
Herman Steyn
Janto Dreijer
Jemma Shipton
Johan Schoonwinkel
John Dalton
Juan Pablo Lozano
Keith Browne
Kobus Botha
Leo Herselman
Rebecca Vanderpool
Simphiwe
Steven Kriel
Suné Smith
Tinus Stander
Vian Espost

– *The 'cross-coupling' experiment:*
Albert Swart
Arno Barnard
Arnold Mulder
Barry Smith
Bernard Visser
Carlo van Schalkwyk
Cobus Stals
Eric Baker
Esti Hansmann
Francois Marais
Gerrit Kruger
Gideon Spreeth
Graham Hardie
Hannes van den Berg
Helgard van Rensburg
Henk Marais
Herman Steyn
Jaco Badenhorst
Johan Schoonwinkel
Johannes van der Horst
John Wilson
John-Philip Taylor
Jonathan Hoole
Keith Browne
Liza Baker
Neil Kruger
Nelius Rossouw
Peter Peiser
Ruan de Hart
Rudi Gaum
Steven Kriel
Susanne Kolditz
Wouter Kriegler

– *The 'alignment' experiment:*
Adam Sparks
Albert Strasheim
Andre Young
Andy Kniss
Catherine Laporte
Charl Müller
Christiaan Brand

Eduard Burger
Francois Marais
Gerrit Kruger
Gideon Spreeth
Helgard van Rensburg
Henk Marais
Herman Steyn
Hilton Gibson
Inge Blom
Johan Botha
Johan Lourens
Johan Schoonwinkel
Lara Kotze
Ludwig Schwardt
Migael Jordaan
Neilen Marais
Nelius Rossouw
Ruan Venter
seeker84
Susanne Kolditz
Tinus Stander
Troels Kofoed Jacobsen
Willie Krige
Willie van Rooyen
Zelda Doyle

– *The 'noise' experiment:*
Bernard Visser
Caitriona Murray
Dewald Mienie
Gerrit Kruger
Helgard van Rensburg
Johan Schoonwinkel
Jonathan Hoole
Marianne du Preez
Mark Byrnes
Myshele Goldberg
Paul van der Merwe
Peter Wiles
Renier Marchand
Robin van Wyk
Ruan de Hart
Susanne Kolditz
Tinus Stander
Willem Mostert
Wouter Kriegler
WPF Schonken

Table of contents

List of figures

List of tables

List of Acronyms

ADT	Abstract data type.
AVIRIS	Airborne visible/infrared imaging spectrometer.
AVHRR	Advanced very high resolution radiometer.
CB	Current boundary.
DEM	Digital elevation model.
DMOS	Difference mean opinion score.
GSI	Ground Sample Interval, also called ground instantaneous field of view (GIFOV).
HDA	Heteroscedastic discriminant analysis.
HOT	Haze optimised transform.
HVS	Human visual system.
IB	Internal boundary.
IQA	Image quality assessment.
LDA	Linear discriminant analysis.
LEO	Low Earth Orbit.
MWIR	Medium wave infrared spectrum, $3 - 5\mu$m.
NDVI	Normalised difference vegetation index.
NIR	Near infrared spectrum, $0.7 - 1.1\mu$m.
PCA	Principal component analysis.

PDF	Probability density function.
PSF	Point spread function.
PSNR	Peak signal to noise ratio.
RAM	Random access memory.
SNR	Signal to noise ratio.
SWIR	Short wave infrared spectrum, $1.1 - 2.5\mu$m.
TC	Tasseled cap (transform).
TIR	Thermal infrared spectrum, $8 - 14\mu$m.
VIS	Visible spectrum, $0.4 - 0.7\mu$m.

Chapter 1

Introduction

1.1 Motivation

The design of an on-board image processing system for a low earth orbit (LEO), remote sensing micro satellite is presented in this dissertation. The system must be able to prioritise images for download according to an image quality measure. Given that the number of images that can be downloaded in a day is fewer than the number that can be acquired, it is desirable to download only the acquired images with the best quality.

1.1.1 On-board processing in remote sensing

In the past the processing capability of satellites has been limited. As processor and memory technology advances, increased use can be made of on-board processing. It becomes feasible to undertake more complex processing tasks on board of the satellite, which increases the autonomy of the spacecraft [111].

On-board processing has been successfully used to reduce data rates, which leads to cost savings [111, pp. 539-550]. Downlink bandwidth and -time are constraints in satellite design: increasing bandwidth increases cost, while downlink time is a function of the satellite's orbital path. LEO satellites remain in contact with a single tracking station for only a few minutes. Reduced data rate requirements can lead to reduced downlink bandwidth requirements. Alternatively, given a predetermined bandwidth, reduced data rates can allow optimal use of available downlink time.

Often, not all of the data collected by the satellite is needed. By processing the data on board of the satellite, it is possible to transmit only the necessary data. For example, Europe's METEOSAT satellites store and format cloud scanner data before transmitting it to numerous ground stations at a reduced

rate. In [68] an on-board neural network classifier is used to generate thematic maps, which are downloaded instead of full resolution multi-spectral images. Different image data compression schemes also utilise on-board processing to reduce data rates [54, 36]. NASA's recent MISR satellite includes custom digital circuits that can average 4×4 pixels into a single pixel to conserve data rates [32].

The system that is the subject of this dissertation must be able to compare acquired images on board of the satellite. This processing can be done when the satellite is in eclipse, when the processing power will not be needed for image acquisition. By generating a quality score for each image, it is possible to rank images according to quality scores. Optimal use of available downlink time is ensured by downloading images from the top of this ranked list.

1.2 Ranking images

Algorithmically determining a quality score for an image is the domain of image quality assessment (IQA). The problem can be divided into feature estimation and quality estimation. During feature estimation, certain features of the image are considered and given numerical values. During quality estimation, these numerical values are mapped to an image quality score.

When ranking images and choosing features, two conceptually different approaches are available. One can either promote good quality images or penalise bad quality images. In terms of features used, this dichotomy is between measuring image content features and image degradation features. Various IQA and feature selection approaches are discussed in Chapter 5. However, some introductory information is given to provide a conceptual framework for the chapters devoted to feature estimation, Chapters 2 to 4.

1.2.1 Good quality images

What constitutes a good quality image? The answer to this question is subjective and depends upon the application [47, p. 76]. One could propose numerous image features that attempt to measure information content. For example contrast, variance, texture content, entropy or sharpness can be considered. Some of these features have been used by researchers and are discussed in Chapter 5.

The conceptual problem with content measures is that they are only objective when considering images of the same subject. For example, when comparing images taken 16 years apart by different satellites, various statistical measures, such as variance and kurtosis were used to compare carefully aligned scenes of the same targets [55]. When comparing different scenes, the scene content severely influences these features and, therefore, they cannot be used to rank different good quality images.

1.2.2 Bad quality images

The images acquired by satellite sensors are often imperfect. Sometimes design errors can cause unwanted degradations. In these cases image processing can often be used to restore such known degradations. An example is the mirror aberration on the Hubble Space telescope [56]. Even when there are no design shortcomings, environmental factors such as temperature variations, radiation, available light and weather conditions can conspire to degrade image quality. In such cases the level of degradation varies between images and, therefore, image restoration techniques have to be applied on a case-by-case basis. Furthermore, certain types of degradations cannot be restored.

It is not meaningful to rank images based on constant degradations that result from design errors or choices. Although contrast is a content measure, a design choice often causes satellite images to have reduced contrast. Satellite sensors with global coverage must view a wide range of scenes, from very low radiance to very high radiance. Therefore, the dynamic range of such sensors is set at the design stage to accommodate a large range of scene radiances. However, this range is seldom present in a single scene. Thus, images typically use less than the full quantisation range, which means the contrast is low. After the images have been downloaded, the dynamic range is increased using post-processing [94, pp. 202-227]. Another typical constant degradation that is removed by post-processing, is striping caused by sensor calibration (see Chapter 3).

For the IQA system that is the subject of this dissertation, the features selected attempt to estimate variable degradations. Cloud cover, additive noise and the amount that the telescope is out of focus (the defocus extent) are estimated.

1.3 Goals

The goals of this dissertation were:

- To investigate ways of measuring image quality. This includes investigation of possible features and quality models.
- To investigate feature estimation algorithms. These algorithms must be able to estimate features autonomously (without human intervention) and blindly (using only a single image, without a reference image).
- To evaluate the feasibility of the selected estimation algorithms and quality assessment model. This includes accuracy as well as speed evaluations.

1.4 Fundamentals

The notation used in this dissertation for an image quantised and sampled so that the digital image has L rows and M columns follows [47]:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots & f(1,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(L-1,0) & f(L-1,1) & \cdots & f(L-1,M-1) \end{bmatrix}. \quad (1.4.1)$$

The values of the coordinates (x, y) are discrete quantities represented by integer values. The right side of equation (1.4.1) is by definition a digital image. Each element of the digital image, i.e., each element of the array, is called a *pixel* (picture element). The value of $f(i, j)$ at a specific coordinate corresponds to the quantised intensity level of the pixel in row i column j.

1.5 Structure

The dissertation is structured into self-contained chapters. Each chapter has its own literature review, methodology, results and conclusions sections. Three chapters are dedicated to the three estimated features respectively: Chapter 2 covers cloud estimation, Chapter 3 covers noise estimation and Chapter 4 covers defocus estimation. In Chapter 5 the design of the quality assessment model is discussed. Finally, conclusions are summarised and recommendations made in Chapter 6.

Chapter 2

Cloud estimation

2.1 Literature

2.1.1 Introduction

Analysis of cloudy images has a long history in remote sensing. The research usually has one of two goals: distinguish cloudy pixels from cloud-free pixels or classify cloudy pixels into different cloud types.

It is essential to distinguish cloudy and cloud free pixels before automatic estimation of surface variables from remote sensing images can be done. While previous researchers have cited land surface albedo[1], -insolation and -temperature [51], Normalised Difference Vegetation Index (NDVI, used to monitor vegetation) [23] and sea surface temperatures [91] as variables that cannot be measured in the presence of cloud cover, most surface variables measured in the visual (VIS) through to thermal infrared (TIR) bands will be meaningless in the presence of cloud cover. Given that approximately 50% of the earth's surface is covered by cloud at any given moment [90], the importance of cloud detection can be easily understood.

To classify and analyse clouds, cloudy regions also first have to be distinguished from cloud free regions. This plays an important part in weather prediction and climate-ecological studies [14], such as the International Satellite Cloud Climatology Project [100]. In general, cloud analysis algorithms only need to identify pixels with more than 50% cloud cover, while cloud detection algorithms are more strict: even pixels with as low as 1% cloud cover have a significant effect on measured brightness temperature [90].

In section 2.1.2 general image processing techniques applicable to cloud

[1] The *albedo*, or reflectivity, of an object is the extent to which it diffusely reflects sunlight.

detection and used in the following sections are discussed. Cloud detection algorithms can be divided into two categories: those based on spectral techniques and those based on spatial techniques, examined in the sections 2.1.3 and 2.1.5. Spectral techniques include transforms that reduce the dimension of the data. Heteroscedasitc discriminant analysis, a novel dimension reducing transform in the context of remote sensing, is discussed in 2.1.4. In addition to spectral and spatial techniques it is possible to use time domain information in cloud detection (by comparing a cloud free image of a region with the image to be segmented), but the practical challenges associated with such a scheme are severe [51]. Finally, in section 2.1.6 a region growing technique, that combines spatial and spectral domain ideas and has been used on board a microsatellite, is considered.

2.1.2 Applicable image processing techniques

Estimation of the amount of cloud cover in an image belongs to the domain of image *segmentation*: the goal is to divide (segment) the image into cloudy and clear regions[2]. It is then easy to measure the relative size of the cloudy area. Segmentation is the process of grouping pixels in an image into homogeneous regions based on one or more properties. Most common image segmentation algorithms use one of two comparative pixel intensity properties: discontinuity or similarity [47, p. 568]. Algorithms based on discontinuity use sudden changes in intensity, such as edges of a region, to segment a region. Methods based on similarity partition the image into regions with similar intensity values.

Image thresholding is a popular segmentation method that belongs to the similarity category. Its importance can be ascribed to its intuitive nature and the fact that it requires less processing power than more advanced techniques. Thresholding is the most popular approach to cloud estimation (see section 2.1.3). Starting with a gray-level histogram such as the one in Figure 2.1, which corresponds to an image of light objects (clouds) on a dark background, Figure 2.2(a), it is possible to separate the objects from the background by associating all pixels of intensity greater than the threshold with the object. A threshold image $g(x, y)$ is defined as:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \leq T, \end{cases} \tag{2.1.1}$$

where $f(x, y)$ is the gray level of the point (x, y) and T is the threshold level. Figure 2.2(b) shows the threshold image corresponding to the threshold value

[2]It is implied that after segmentation it is known which segment is cloudy and which is clear. In the context of pattern recognition such labelling of data is called *classification*. However, often the use of a certain segmentation technique presupposes specific classes resulting from segmentation. Therefore the two terms are used interchangeably in this discussion.

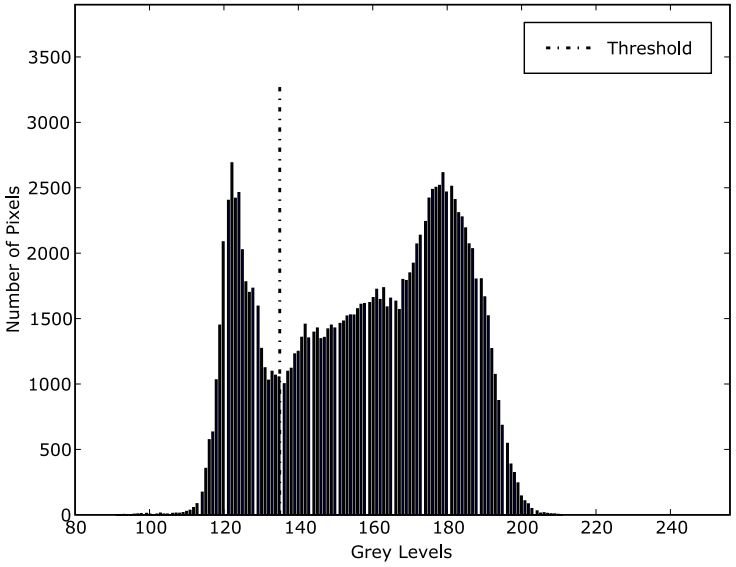
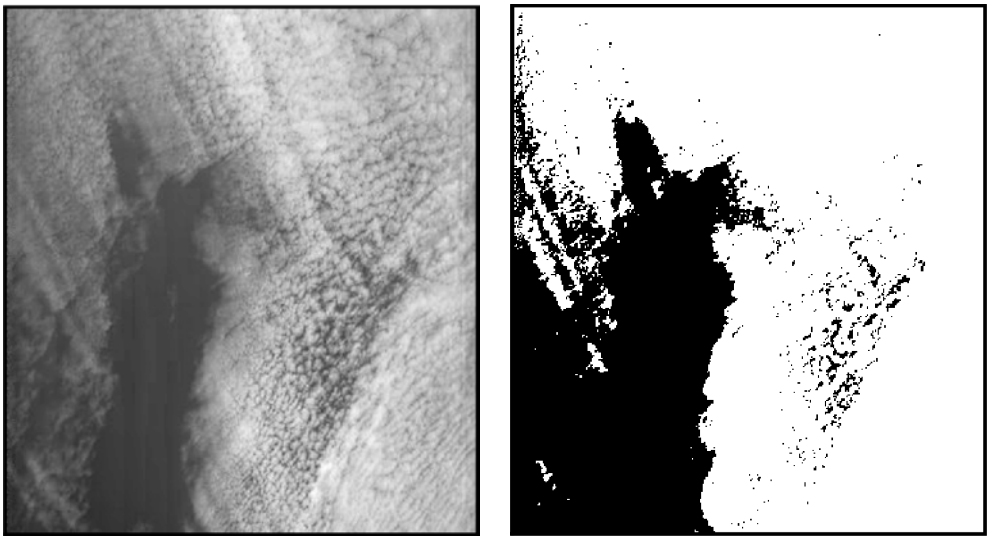


Figure 2.1: A histogram partitioned by a threshold.

from Figure 2.1. In this example, T depends only on the $f(x, y)$ and the



(a) (b)

Figure 2.2: An example greyscale image (a), with the corresponding threshold image (b).

threshold is a *global* threshold. In general, T could be of the form:

$$T = T[x, y, p(x, y), f(x, y)], \tag{2.1.2}$$

where $p(x, y)$ is some local property of the point (x, y). If T is dependent on $p(x, y)$ the threshold is *local*. An example of a local property that has been used for cloud estimation is variance (see section 2.1.3). If T depends on the coordinates (x, y), the threshold is called *dynamic*; a variation of this type of threshold has recently been introduced to cloud detection [114, 58]. In these cases thresholds were not varied on a pixel by pixel basis, but different thresholds were trained for different surface types. When evaluating an image, its surface type was first determined by using the image's latitudinal and longitudinal coordinates as entry into a global lookup table.

Considerable research has been done in the field of automatic thresholding; an overview of techniques can be found in references [88] and [65]. These techniques try to determine an optimum threshold to separate light objects from a dark background automatically. However, the methods assume that the histogram has two dominant modes, like the two peaks in Figure 2.1. Hence, the assumption is that there is always a dark and a light region in the image and that they are to be separated. This makes these techniques unsuitable for cloud estimation: all remote sensing images do not contain clouds and, conversely, some images may consist of only clouds; so two dominant histogram modes are not guaranteed.

Even if no automatic thresholding techniques are used, some measure to compare the performance of different segmentations is needed. Reference [65] introduces a measure to evaluate the performance of thresholding techniques that does not require these techniques to be automatic. It is a meaningful, objective performance criterion for global thresholding algorithms:

$$Pr(err) = Pr(O) \times Pr(B|O) + Pr(B) \times Pr(O|B), \tag{2.1.3}$$

where $Pr(B|O)$ is the probability of error in classifying object as background and $Pr(O|B)$ is the probability of error in classifying background as object. For a thresholding technique to give good performance both these probabilities must be low. $Pr(O)$ and $Pr(B)$ are a priori probabilities. To calculate these probabilities one must manually segment an image to create a reference threshold image, and measure the area of the object and background.

When using global thresholding, it is important to consider the illumination of the scene. Image intensity can be modelled as the product of an illumination $i(x, y)$ component and a reflectance component $r(x, y)$:

$$f(x, y) = i(x, y)r(x, y). \tag{2.1.4}$$

If there is a nonuniform light distribution across the scene, the shape of the histogram will be altered proportionately to the degree of non-uniformity of $i(x, y)$ [47, p. 598]. Luckily, there is normally nothing between cloud tops and the sun, so illumination is uniform. The level of uniform illumination can differ depending on the time of day and this is taken into account by many of the algorithms in section 2.1.3. Furthermore, it is possible for high layers of cloud

to cast shadows on lower layers, resulting in nonuniform illumination. This creates the possibility that lower, shaded clouds visible through gaps in higher clouds might not be identified by a global threshold.

A different segmentation technique, also based on similarity, but taking the relative position of pixels into consideration, is *region growing* [47, p. 613]. Pixels are grouped together based on predefined, application specific criteria. Algorithms generally start with one or more seed points. Neighbouring pixels with properties similar to the seed point are added to the region. Since only neighbouring pixels can be added to a region, the concept of connectivity is inherent to region growing: starting from a single seed point, a region-growing algorithm will produce a connected region. Lastly, a stopping rule is needed: a definition of when neighbouring pixels are suitably different from those already in the region. When it is satisfied the growing process stops. Stopping rules can be based on local properties such as grey levels or texture, but more powerful rules also take into account the growing history or the shape of the region. A cloud segmentation technique based on region growing is discussed in section 2.1.6.

2.1.3 Spectral methods

Applying thresholds to images has been used to detect clouds since the first satellite images were produced [87]. A large portion of the initial work in cloud identification used global thresholds based on the fact that clouds are generally bright in the visible and cold in the infrared spectra. [3] Since global spectral thresholds apply to each pixel independently, they are also referred to as per pixel techniques. In [38] it was found that all clouds exhibit approximately the same albedo, apart from thin cirrus clouds, which are semi-transparent in the VIS band. A method was formulated based on automatic thresholding; it relied on a large scale of cloud cover, so many pixels are completely filled with cloud. Given a large enough area, sea, earth and cloud surfaces will produce two or three distinct peaks in the VIS histogram. Since the histogram has distinct main modes, automatic thresholding techniques become applicable. The method starts with a preset initial threshold value that is iteratively refined by moving it toward the valleys in the histogram. During the iteration, data from the infrared channel is used to stop iteration once the mean temperature of the cloudy area starts to increase, indicating that ground pixels are being included in the cloudy area.

This assumption of two or three main histogram modes might be appropriate for satellites with very large ground sample interval (GSI), such as METEOSAT (GSI = 2500m) for which the algorithm in reference [38] was developed. A large GSI averages acres of surface detail into a single pixel. In the

[3] In this chapter the term 'spectrum' refers to electromagnetic waves with limited range of frequencies. In chapter 4, 'spectrum' refers to the Fourier decomposition of a spatial domain image.

Table 2.1: AVHRR sensor specifications.

Channel	Wavelength region [μm]	GSI [m]
1	0.58-0.68 (VIS:red)	1100
2	0.725-1.10 (NIR)	1100
3	3.55-3.93 (high-temp TIR)	1100
4	10.3-11.3 (TIR)	1100
5	11.5-12.5 (TIR)	1100

resulting satellite image, the surface of the earth appears considerably more uniform than in an image where the GSI is in the order of meters. More uniform earth, cloud and sea surfaces result in clearly defined histogram modes. Satellites with large GSI are often weather satellites, designed to acquire frequent images of large areas to monitor and map large scale climate effects. The large GSI is a fundamental difference between these and resource satellites: low earth orbit (LEO) remote sensing satellites designed for mapping the surface conditions on the earth. In satellites with a smaller GSI more detail can be discerned and frequently it is a technological goal to get the GSI as small as possible. In these images the abundance of surface detail can influence the histogram in unpredictable ways. Furthermore, as mentioned in section 2.1.2, smaller GSI increases the likelihood of having images with no cloud cover or complete cloud cover, thereby also invalidating the assumption of a modal histogram [114].

The heavy reliance of the algorithm on the infrared channel also makes it unsuitable for LEO satellites with spectral reach in the VIS and near infrared (NIR) bands, such as Sumbandilasat. The algorithm struggled to distinguish clouds from snow and ice. This is a common problem in cloud estimation algorithms, since snow and ice are also bright in the VIS and cold in the TIR bands. Over desert regions such as the Sahara, a high surface albedo necessitated using TIR exclusively.

The method was adapted and expanded upon by Saunders in [90] for AVHRR scanner (Advanced Very High Resolution Radiometer, see table 2.1). The main improvement is the selective use of AVHRR channels depending on location. If the image was taken over land, AVHRR channel 1 is used for cloud detection, while channel 2 is used when the background is predominantly ocean. The reason given is that, while clouds have a high reflectance at both these wavelength intervals, the longer channel 2 wavelengths are less affected by aerosols and Rayleigh scattering than the shorter channel 1 wavelengths. But land surfaces have a higher albedo at the longer wavelengths. Therefore channel 2 is used for cloud detection over sea, since it is less affected by scattering, while over land channel 1 is used since the lower reflectance of land surfaces increases the contrast between land and cloud. Greater contrast makes identification easier. Once a channel has been chosen, the histogram

is analysed to see if a peak exists. If a peak exists and it is below the predetermined value for a cloudy scene, the peak is assumed to be cloud free. The cloud threshold T is then set to a predetermined reflectance above the peak. In the case of coastal regions there can be two histogram peaks (one each for land and sea) and the dynamic threshold determination method fails and a predetermined single threshold is used. This test is combined with two other tests based on channel 3 and 5 respectively. In channel 5, pixels must be colder than (have an intensity less than) a predefined threshold. The channel 3 test is only useful over sea and flags pixels with high local variance as cloudy. A pixel must be flagged as cloud free by all three tests to pass. The channel 1 or 2 test is only applicable during the day, which is defined as solar zenith less than 80°. Since the tests based on different channels are not so intertwined as those from [38] it is possible to apply only the channel 1 or 2 test in a satellite with limited spectral reach, such as Sumbandilasat. However, while less dependent on the TIR band, any advance beyond a basic predetermined threshold relies on large GSI for a strong histogram peak and still uses a predetermined offset. So it is merely a different parameter that has to be predetermined. Also when a single, strong histogram peak is absent, the method falls back on a predetermined threshold.

In [91] Saunders and Kriebel add two additional tests to the three just discussed, bringing the total to five tests. The two new tests are based on the ratio of AVHRR channels 2 and 1, and the difference of channel 4 and 5. The channel intensity values are first converted to at-sensor reflectance[4] defined as:

$$\rho_n \equiv \frac{\pi L_n}{E_s \cos \psi_s}, \tag{2.1.5}$$

where L_n is the recorded radiance signal [W m^{-2} sr^{-1} μm^{-1}], E_s is the extra-atmospheric solar irradiance for the selected band, ψ_s is the solar zenith angle and ρ_n is the at-sensor reflectance. The recorded radiance signal L_n is typically calculated from raw channel intensity value C using the gain G_n and intercept Y_n for the sensor channel n: $L_n = G_n C + Y_n$. It is meaningful to normalise the raw intensity before thresholding so that the threshold value can be independent of the time of day and the specific sensor variables. The ratio used in the test is

$$Q = \frac{\rho_2}{\rho_1}. \tag{2.1.6}$$

When using more than one channel it is assumed that the radiation from different channels originates from the same place. Therefore it is paramount that the different channels be accurately aligned. The main motivation for using a ratio is that in the histogram of Q, land and sea are well separated at opposite ends of the histogram and a cloudy peak can be looked for in the central region of the histogram. It is worth noting that reference [51]

[4] At-sensore reflectance is also referred to as top-of-atmosphere reflectance

claims that a ratio test is less sensitive to differences in anisotropic properties between channels than a difference test. The ratio thresholds are determined similarly to those for VIS or NIR: cloud free peaks at the top or bottom of the Q histogram are identified and only values of Q closer than predefined levels to the peaks are labelled as cloud free. Such 'dynamic' thresholding is biased towards misclassifying cloud-free pixels as cloudy. If no clear peaks are present (often the case over land), a predefined threshold is used. Once again a pixel is only identified as cloud free if *all* five of the tests prove negative.

Various other authors advanced the work from [90, 91] discussed in the previous two paragraphs. References [23, 14] underscore the need to adapt algorithms and thresholds to local weather conditions and specific satellites. Thresholds should be based on data collected over a long period and determination of the optimum threshold is a human lead, iterative process. In [23] two cloudy images were taken each month for 11 months and used to derive representative radiometric thresholds for the Texas region. While it is imperative for TIR thresholds (since seasonal changes affect temperature levels), it remains sound advice for VIS and NIR thresholds. Since ice and snow also have high VIS and NIR albedo, work has been done that relies on TIR bands to improve discrimination of these surfaces from clouds [10]. Reference [98] uses local threshold techniques in the TIR band to improve identification of clouds over the ocean. In reference [14] the threshold value T is dependent on a digital elevation model (DEM). This allows thresholds to be adapted for mountainous, snowy regions. Also, the ratio test from equation (2.1.6) is criticised for being over eager to identify desert areas as cloudy. It is recommended that its use be limited solely to images over the ocean.

The eagerness of Saunders and Kriebel's algorithm to classify pixels as cloudy is criticised by Guttman in [51], again for the AVHRR sensor. He advocates a 'clear-until-proven-cloudy' policy as opposed to the scheme where failure of any one of five tests results in a pixel classified as cloudy. Also, all dynamic thresholds are replaced with static thresholds. While many of the tests rely on TIR channels, a useful solar glint test is introduced. Solar glint is a highlight caused by the reflection of the sun on water bodies and has the same bright white appearance in the VIS band as clouds. However, glint is only allowed in the forward scatter direction (i.e., towards the sun) and close to the principal plane (i.e., small relative azimuth). Glint is possible if the satellite zenith angle ψ_s and relative azimuth θ_s are both smaller than corresponding thresholds:

$$\text{IF} \quad \theta_s < \theta_G \quad \text{AND} \quad \psi_s < \psi_G \quad \text{THEN} \quad \text{glint is possible.} \tag{2.1.7}$$

If the possibility of glint exists, an additional test based on a channel 5 threshold is required before a pixel is flagged as cloudy.

Dimension reducing transforms

In a memory-scarce on-board environment, memory requirements can be minimised by considering spectral techniques which use only a single channel. When selecting a single channel for thresholding, there are various options present in cloud detection literature. Similar to reference [90] mentioned above, red channel at-sensor reflectance, ρ_{red} (around 670 nm), has been advocated for cloud detection over land [114]. It has also been used for cloud detection over water [31]. On the other hand, the blue channel reflectance, ρ_{blue} (around 480 nm), or alternately, the lowest wavelength band available for a given satellite, is used for an initial cloud mask by the ATCOR (atmospheric correction) program [85, 86].

Instead of selecting a single channel for thresholding, one can combine multiple channels' spectral bands into a memory efficient greyscale image, which can then be thresholded. Since RAM is limited on board of a satellite, the memory efficiency of a greyscale representation is useful. Specifically, for Sumbandilasat the multispectral bands are stored in non-volatile mass storage, with a capacity that greatly exceeds that of the RAM. Furthermore, when reading data from the non-volatile memory into RAM, the architecture requires that large blocks of data from a single channel must be read at a time. Since the architecture does not allow reading the non-volatile memory scanline-by-scanline, assembling a greyscale by adding a weighted version of the channel being read to the current weighted average greyscale in RAM allows for optimal use of limited RAM. The challenge is to determine which channels to combine and which relative weights to use for optimal detection of clouds. Various transforms have been proposed.

One option is the non-linear NDVI transform [94, p. 183], which has been used in conjunction with thresholds for cloud detection [87]. The definition is $\text{NDVI} = (\rho_{nir} - \rho_{red})/(\rho_{nir} + \rho_{red})$, where ρ_{nir} is the reflectance in the NIR channel. Since clouds are less dependent on frequency than plants, cloudy NDVI pixels have values closer to zero, while vegetated land values are positive and ocean values negative. However, bare soil also maps close to zero, and it is unlikely that a transformation derived for measuring vegetation properties will be the optimal choice for detecting clouds.

A transformation based on the NDVI was proposed for cloud detection [31]:

$$\text{D} = \frac{|\text{NDVI}|^b}{\rho_{red}^2}, \tag{2.1.8}$$

where b is chosen to separate clear and cloudy classes optimally. The value of b can be related to the slope of the decision boundary between the two classes in log-space. This decision boundary is chosen to be orthogonal to the line connecting the class centroids. The cloud detection ability of the D transform has been compared favourably against NDVI, ρ_{red} and a standard deviation-based transform over a range of images [114]. In these tests, pixels

covered with thin clouds were treated as cloud-free since the application only demanded thick, completely opaque clouds be identified.

The tasseled-cap is a linear transform also originally designed for agricultural monitoring using Landsat imagery, but found effective for detection and correction of thin clouds or hazes [94, pp. 198–201], [26]. The component of the tasseled-cap useful for these purposes is $\mathrm{TC} = 0.846 \times L_{blue} - 0.464 \times L_{red}$, where L is the recorded radiance signal for the respective Landsat channels [85].

The haze optimised transform (HOT) is a data-dependent improvement on the tasseled-cap that has recently been developed for detecting and correcting for thin clouds [120]. It also uses the blue and red channels, but determines the weight of the channels based on the image data:

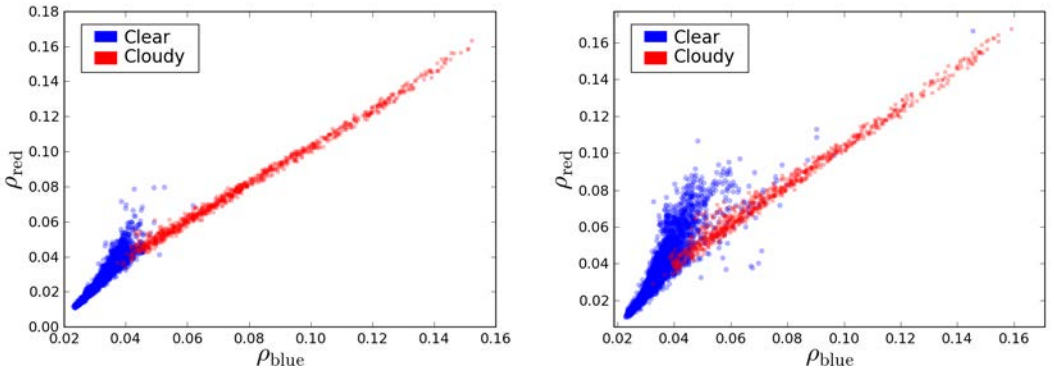
$$\mathrm{HOT} = \sin\phi \times L_{blue} - \cos\phi \times L_{red}, \tag{2.1.9}$$

where ϕ is the slope angle of a linear regression line between L_{blue} and L_{red} for clear (cloud-free) areas. For this regression L_{blue} is the independent variable and the resulting line is called the 'clear line'. The HOT transform, equation (2.1.9), projects data onto a line perpendicular to the clear line. This transform has replaced the tasseled-cap as the transform of choice for haze detection and correction in ATCOR [86].

In summary, without access to TIR it is not possible to discriminate between clouds, ice, snow or bright desert. Furthermore, cloud shadows, which also affect surface variable estimates, could not be identified even with access to TIR [23]. Methods attempting dynamic thresholding tend to be biased towards classifying pixels as cloudy and rely on large GSI. In an on-board system without access to a global land/sea classification model or a DEM, the best one can do is to use a static, predetermined global reflectance threshold. Converting an image to greyscale is a way to conserve memory that also allows for easy application of a threshold. Methods to convert a multi-channel image to greyscale include selecting single channels or applying transforms like D and HOT. Sensor data should be normalised using equation (2.1.5). If the geometry allows for solar glint according to equation (2.1.7), bright VIS pixels cannot be unconditionally accepted as cloudy and uncertainty will remain.

2.1.4 An alternative dimension reducing transform: Heteroscedastic discriminant analysis

Heteroscedastic discriminant analysis (HDA) is a transform suitable for reducing the dimension of data before applying a linear classifier. It is conceptually similar to the spectral methods from section 2.1.3: when applied to cloud detection it can combine different channels into a greyscale image. When combining the channels, each channel's weight is based on a criterion for optimal classification of clouds. To the author's knowledge, HDA has not previously



(a) Small overlap. (b) Larger overlap resulting from urban area.

Figure 2.3: Scatter plots of two images showing example spreads of class values. The image for (b) is shown in Figure 2.7.

been applied in a remote sensing scenario; its application has been mostly limited to speech classification, where it originated. It was selected from various dimension reducing techniques found in pattern classification literature.

When considering candidate transforms, it is instructive to visualise sample class distributions, as in Figure 2.3. In this example the data are four-dimensional: red, green, blue and NIR channels are present. To enable visualisation, only the red and blue channels are presented. In both images there is some overlap between classes; the increased overlap in Figure 2.3(b) is the result of bright urban structures. Classes can generally be well separated or highly overlapping, with overlap caused by either thin semi-transparent clouds or bright backgrounds, like urban structures, glint, snow and desert. It is often impossible to separate the overlapping classes perfectly using any combination of the four available bands. While the two classes are often mono-modal, cases where the clear area consists of a few large distinct regions might result in multi-modal behaviour. The shapes (spread of the data) of the two classes almost always differ significantly.

To combine four different channels into a greyscale image, the dimension of the feature space must be reduced from four to one. Classical dimension-reducing transforms include principal component analysis (PCA) and linear discriminant analysis (LDA) [15, pp. 186–191 and 561–570]. Both transforms use Gaussian probability density functions (PDF) to model the data and linear projections to reduce dimension. PCA projects data onto the axis corresponding to the direction of maximum data variance. Since PCA does not incorporate class information, it is unlikely to be optimal for classification. LDA attempts to find a projection that maximises the separation between class means (the between-class variance) while minimising the variance within each class (within-class variance). In practice this is achieved by maximising the ratio of between-class variance to within-class variance. By effectively

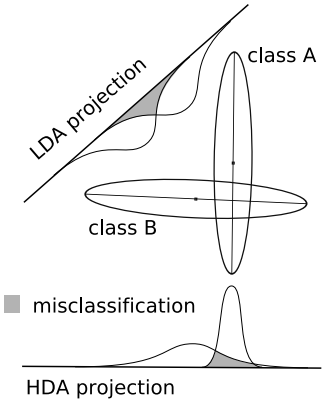


Figure 2.4: Heteroscedastic extension to LDA minimises classification error when class covariances differ.

minimising average within-class variance, it is hoped that the area of overlap between the classes (and thus the misclassification rate) will also be minimised. Since *average* within-class variance is minimised, equal covariance class PDFs are assumed. When this condition holds, the area of overlap is indeed minimised [52, p. 95], but for cloud data, classes generally have greatly differing covariances as illustrated in Figure 2.3. LDA is therefore inappropriate and gives poor results as illustrated in section 2.3.1.

Recent expansions on LDA include HDA [61, 89] and kernel discriminant analysis (KDA) [81]. KDA is a non-linear transformation that first maps data to a very high dimensional space using a kernel function before applying LDA to reduce dimension. Although non-linear decision boundaries might improve performance in highly overlapping cases, like the one shown in Figure 2.3(b), in many cases, such as the one shown in Figure 2.3(a), a linear boundary would suffice. Moreover, the computational requirements for KDA are prohibitive for on-board use: using the kernel function, an input sample is compared to each of the n points in the training set during transformation. For typical training sets used in the experiments discussed in section 2.2 $n > 6\,000\,000$ was common, which makes KDA impractical.

Heteroscedastic discriminant analysis (HDA) – sometimes referred to as heteroscedastic *linear* discriminant analysis (HLDA) – is a generalisation of LDA derived to handle unequal-covariance classes. This makes it more suitable for application to cloud detection. Figure 2.4 illustrates the difference between HDA and LDA in a simple example; since the class covariances are so different, LDA projects along a line connecting the class means, which does not minimise the area of overlap.

Equivalent formulations of HDA can be derived in different ways [61, 89]. The output is a $m \times m$ transformation matrix $\mathbf{A} = [\mathbf{A}_p \mathbf{A}_{m-p}]$, of which only the first p columns, $\mathbf{A}_p = [\boldsymbol{a}_1 \ldots \boldsymbol{a}_p]$, are used to achieve dimension reduction from $\mathbb{R}^m \rightarrow \mathbb{R}^p$. In the experiments conducted in section 2.2, $m = 4$ and

$p = 1$; i.e., only the first column $\mathbf{A}_p = \boldsymbol{a}_1$ is used. Since HDA is a linear projection, $\boldsymbol{y} = \mathbf{A}_p^{\mathrm{T}}\boldsymbol{x}$, transforming a m-dimensional input sample, $\boldsymbol{x}$, to a p-dimensional output $\boldsymbol{y}$ for a given $\mathbf{A}$ requires only a simple and fast matrix multiplication. To find an expression for $\mathbf{A}$, the implicit assumption that the discarded $(m-p)$-dimensional subspace contains no classification information is modelled and the maximum likelihood criterion is used to find $\mathbf{A}$ under this assumption [61]:

$$\mathbf{A} = \arg\max_{\mathbf{A}} \left\{ -\frac{n}{2}\log|\mathbf{A}_{m-p}^{\mathrm{T}}\bar{\mathbf{T}}\mathbf{A}_{m-p}| - \sum_{j=1}^{J}\frac{n_j}{2}\log|\mathbf{A}_p^{\mathrm{T}}\bar{\mathbf{W}}_j\mathbf{A}_p| + n\log|\mathbf{A}| \right\}, \tag{2.1.10}$$

where n is the total number of samples, n_j is the number of samples in class j and the number of classes $J = 2$ for cloud classification. $\bar{\mathbf{T}}$ is a measure of total data variance and $\bar{\mathbf{W}}_j$ is the within-class covariance matrix for class j:

$$\bar{\mathbf{T}} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^{\mathrm{T}},$$

$$\bar{\mathbf{W}}_j = \frac{1}{n_j}\sum_{i=1}^{n_j}(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_j)^{\mathrm{T}}, \quad \boldsymbol{x}_i \in \text{class } j,\ j = 1 \ldots J,$$

where $\bar{\boldsymbol{x}}$ is the data mean and $\bar{\boldsymbol{x}}_j$ is the class mean for class j. Although reference [61] went no further than stating equation (2.1.10) as a result of their derivations, one can interpret it as follows: the first term minimises the variance in the discarded subspace (this space should contain no classification information), the second term minimises the *sum* of within-class variance in the projected subspace (as opposed to the *projected average* within-class variance of LDA), while the last term prevents $\mathbf{A}$ from becoming zero. Since there is no closed form solution for equation (2.1.10), gradient descent optimisation is used to find $\mathbf{A}$. The LDA solution is taken as the initial guess for $\mathbf{A}$ required by the optimisation process.

For application to cloud detection, full covariance matrices [15, pp. 75–85] (for which equation (2.1.10) was derived) are assumed for each class. Since the spread of the data, as shown in Figure 2.3, is neither isotropic nor diagonal, this assumption is most suitable. If covariance matrices were restricted to being diagonal or isotropic, alternate, simplified expressions for HDA also derived in [61] could be applied.

2.1.5 Spatial methods

A contextual classifier

In [59] Kittler and Pairman assert that a human meteorologist considers more than radiance when identifying clouds: the size and shape, texture, global

position as well as relative position to other weather systems are taken into account. Two existing options for advancing beyond per pixel, spectral classification are discussed. One option is to divide the image into coarse cells and use texture features within a cell for additional information. Two previous studies where texture has been used are examined: in one it was found that spatial methods brought no increase in accuracy, while the other found that use of spatial methods increased the ability to discriminate between different cloud types. The second spatial domain option is to use unsupervised clustering to find groups of similar pixels in multidimensional space and draw decision boundaries between them. A study where this approach was used is commented upon: the advantage of clustering is that once pixel clusters have been formed data from the *whole* cluster can be used in classifying it, as opposed to data from a single pixel. However this is outweighed by disadvantages: small clusters can be swamped (erroneously grouped with larger clusters), each pixel is still assigned to a cluster on a individual basis so that noise at class boundaries is the same as with individual pixel classifiers. Additionally, classes exhibit great differences in within class variance, which make them unsuitable for unsupervised clustering.

To address these problems Kittler and Pairman develop a contextual classifier. It iteratively classifies pixels while taking into account already classified neighbouring pixels' classes. When applied to AVHRR data, the resulting method succeeds in reducing the noisiness of the between class boundary. Similar results were recently achieved with a k-means clustering classifier that takes into account neighbouring pixels's intensity values [12]. However, contextual classifiers are not able to correct any large incorrectly classified areas in the image. This is because incorrectly classified pixels tend to support each other. Therefore, while the method represents a small improvement over a per-pixel spectral classifier due to cleaner boundaries, it does not improve discrimination between the similar snow, ice, sun glint, desert and cloud classes. For such an improvement extra features are needed.

Texture features and neural networks

In reference [109] an attempt is made to identify useful textural features for the discrimination of cloud and various ice-covered surfaces. Sixteen Landsat MMS scenes were digitised to a 2048×2048 array with a GSI of 100m. Numerous 256×256 pixel areas from each scene were selected for textural analysis. Several textural features based on a sum and difference histogram approach were investigated. It is claimed that sum and difference histogram textural features give similar accuracy to traditional textural features based on the grey level cooccurence matrix [47, pp. 668–669], but with the advantage of decreased memory and computational time. Certain features exhibited good class separation when classifying clouds over snow- or ice-covered mountains. Good separation was also achieved for certain clouds over glaciers and sea ice.

However, there was not proper separation of broken strato-cumulus clouds or thin cirrus and ice floes.

The authors of [102] investigate texture features based on the Gabor transform for cloud detection. However, in later work where the class separation ability of different texture measures were compared, the Gabor transform did not perform favourably [106].

In [106] neural networks are used in a cloud classification system for AVHRR data. Two images were divided into 32×32 pixel sections, from which textural and brightness features were calculated. Many textural features based on spatial grey-level difference statistics, Fourier statistics, autocorrelation statistics and Gabor functions were evaluated. Fourteen texture features were compared and the five best used in the classifier along with four features based on the mean and maximum intensities of pixel segments in channels 1 and 4. Four different neural network configurations were studied and it was found that a two stage classifier gave the best performance. In the first stage a per pixel classification based only on spectral brightness is used to separate the data into broad categories of land, sea and cloud. Textural features are only introduced in the second stage to classify cloud into 10 different cloud types. Classification accuracies of 91% are reported, but the test and training data is taken from two AVHRR images only. For the classifier to be of practical use, it will have to be trained on a diverse collection of many images taken at different locations, times of day and seasons. For example, no images incorporating problem areas such as desert or ice were tested. It is notable that the algorithm effectively relies solely on spectral classification to distinguish cloudy and cloud free pixels, since the first stage classifier uses no textural information.

Neural networks and spatial texture features have also recently been used in cloud detection from the MODIS (Moderate-Resolution Imaging Spectroradiometer) sensor [99] as well as the SPOT (Satellite Pour l'Observation de la Terre) sensor [57]. For the MODIS sensor, its increased spectral resolution is utilised: seven bands ranging from VIS to TIR are selected, but insufficient detail of the implementation and results is given.

Conclusion

While there has been much work done in applying texture features and pattern classification systems to cloud classification, it is difficult to compare results from studies where different satellites are used [59, 54]. The effectiveness of texture features derived for weather satellites with large GSI will have to be tested for a resource satellite with small GSI. Also, much effort has been directed toward classifying different cloud types which does not necessarily result in improved detection of clouds.

The most relevant for the purpose of this dissertation is the work done in [109] and its follow-up article [110]. The Landsat data used there has relatively

small GSI (still more than a factor 10 larger than Sumbandilasat's) and the focus was on cloud detection over difficult terrain. Although cloud detection was improved, some types of cloud over snow remained inseparable.

The biggest problem with attempting to build such a pattern recognition system is that it will have to be trained with data from the specific satellite it is intended to be used with. Although the same can be said for thresholds, the increased complexity of the classifier and the features used mean considerably more and diverse training data is needed. A general classifier able to distinguish between cloud and snow or ice or desert based on texture will have to be presented with all (or at least most) possible textures that these surfaces can have. For a resource satellite with small GSI this problem is compounded; not only does the ability to see more detail imply a larger variety of possible textures, reduced swath width[5] means it will take much longer to acquire a general dataset that covers a large range of locations. Therefore data will have to be collected from the intended satellite over a long period and such an undertaking is beyond the scope of this dissertation.

2.1.6 A promising region-growing based method

Context

Hou *et al.* implemented an on-board cloud detection scheme for PoSat-1, a microsatellite developed at Surrey Satellite Technologies [54]. The goal of the project was similar to this one: image processing was employed to make optimal use of downlink time. In this case it was achieved by identifying cloud boundaries and adapting the standard JPEG compression algorithm so that the boundaries are compressed at a higher ratio. Since it was found that many bits are typically used in storing the transition from bright cloud centres to cloud-exterior, this increased local compression resulted in significant savings.

The project also had similar constraints to this work: since PoSat images are single-band, the authors could not use methods developed for multispectral imagers with access to TIR. Additionally, constraints of on-board implementation favour methods that are less computationally complex than using texture features with neural networks. Since the algorithm was suitable for the given constraints and had already been implemented on board a microsatellite, it was decided to investigate it further. An overview of the algorithm is given below, the experiments, results and concusions are described in sections 2.2.2, 2.3.2 and 2.4.2, while details regarding implementation are reserved for Appendix A.

[5] *Swath width* refers to the area on the ground imaged by the satellite.

Algorithm description

The region-growing algorithm used in [54] was first described in reference [53] and is particularly suitable for segmenting grey-level areas with high contrast relative to their local backgrounds, such as bright clouds on a darker background. Similar to other region-growing algorithms, the method starts with a seed point that satisfies certain criteria and expands the region in all directions until the stopping criterion is reached. Unlike most region-growing algorithms only one pixel is added at a time, making the method more predictable. A combination of two discontinuity measurements of the region being grown, namely average contrast and peripheral contrast, is used to stop the growing process.

The pixel to be added to the current region is the neighbouring pixel with the highest intensity. If the method starts at a local maximum bright pixel, this ensures that the boundary pixels added will have monotonically decreasing grey levels. If more than one neighbouring pixel has the same intensity, a first come first served strategy is adopted. To define the two region measurements, the following two boundaries are required:

current boundary (CB) the set of pixels adjacent to the current region,

internal boundary (IB) the boundary produced by the set of connected outermost pixels of the current region.

These two boundaries are illustrated in Figure 2.5. Each time a pixel is added

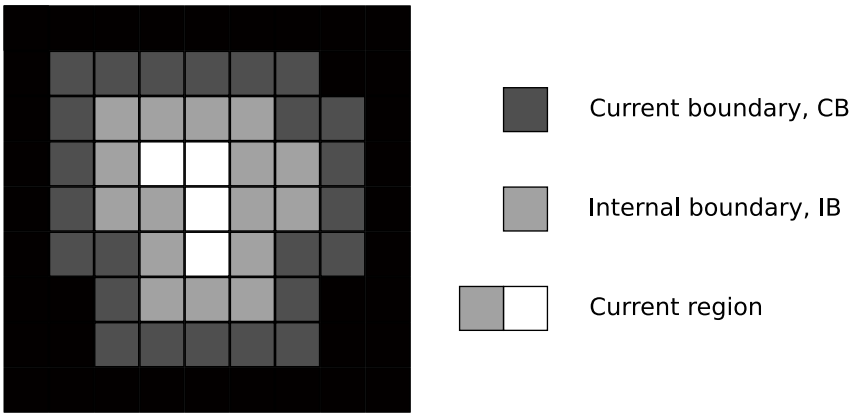


Figure 2.5: The two boundary definitions. In this schematic the current region comprises 20 pixels.

to the current region, the boundaries are updated. Next, the two region descriptors used for the stopping criterion can be defined:

average contrast the difference between the average grey level of the region and the average grey level of its CB pixels,

peripheral contrast the difference between the average grey levels of a region's IB and CB.

As the region is grown, one pixel at a time, the values of these two measurements are remembered. They can be plotted as functions of the pixel index like in Figure 2.6(c).

The average contrast will increase as long as high intensity pixels are being added to the region: once the region starts growing into the background, the rate of grey-level decrease for the boundary will be less than that for the region and the average contrast will start decreasing.

The peripheral contrast is representative of the gradient at the CB. However, it it less sensitive to noise than the gradient since it computed using the average across the two boundaries instead of the average across two pixels. For a bright blob against a uniform background, the peripheral contrast will have a well-defined maximum. However, for a textured or noisy background, it will exhibit many local maxima.

The stopping criterion is that the last local maximum in peripheral contrast before the global maximum in average contrast defines the final segmentation boundary. These concepts are illustrated in Figure 2.6 using a 2-D Gaussian shape

$$g(x,y) = Me^{-\frac{(x-u_x)^2+(y-u_y)^2}{2\sigma^2}}, \tag{2.1.11}$$

where u_x, u_y specifies the centre of the Gaussian blob, σ specifies the spread and M is a constant used to normalise the output to the maximum grey-level range. The highest magnitude gradient for a Gaussian shape is located one standard deviation from the mean. Thus the maximum peripheral contrast measure defines a circle with radius σ seen in Figure 2.6(b). The grey-level mappings in 2.6(c) behave as expected: as pixels further away from the centre of the Gaussian are added, the average grey-levels decrease. Since the current region average retains the bright centre pixels, its average decreases at a slower rate. As expected, the average contrast increases until the region starts growing into the background at approximately pixel number 12000 and then decreases. The last local maximum of peripheral contrast is in this case also the global maximum, but this is not the case with real images.

2.2 Experiments

2.2.1 Dimension reducing transforms

Data

Since Sumbandilasat had not yet been launched, the data set[6] for comparing dimension reducing transforms consisted of cloudy Quickbird and Landsat

[6] Courtesy of Satellite Application Centre, a branch of the CSIR.

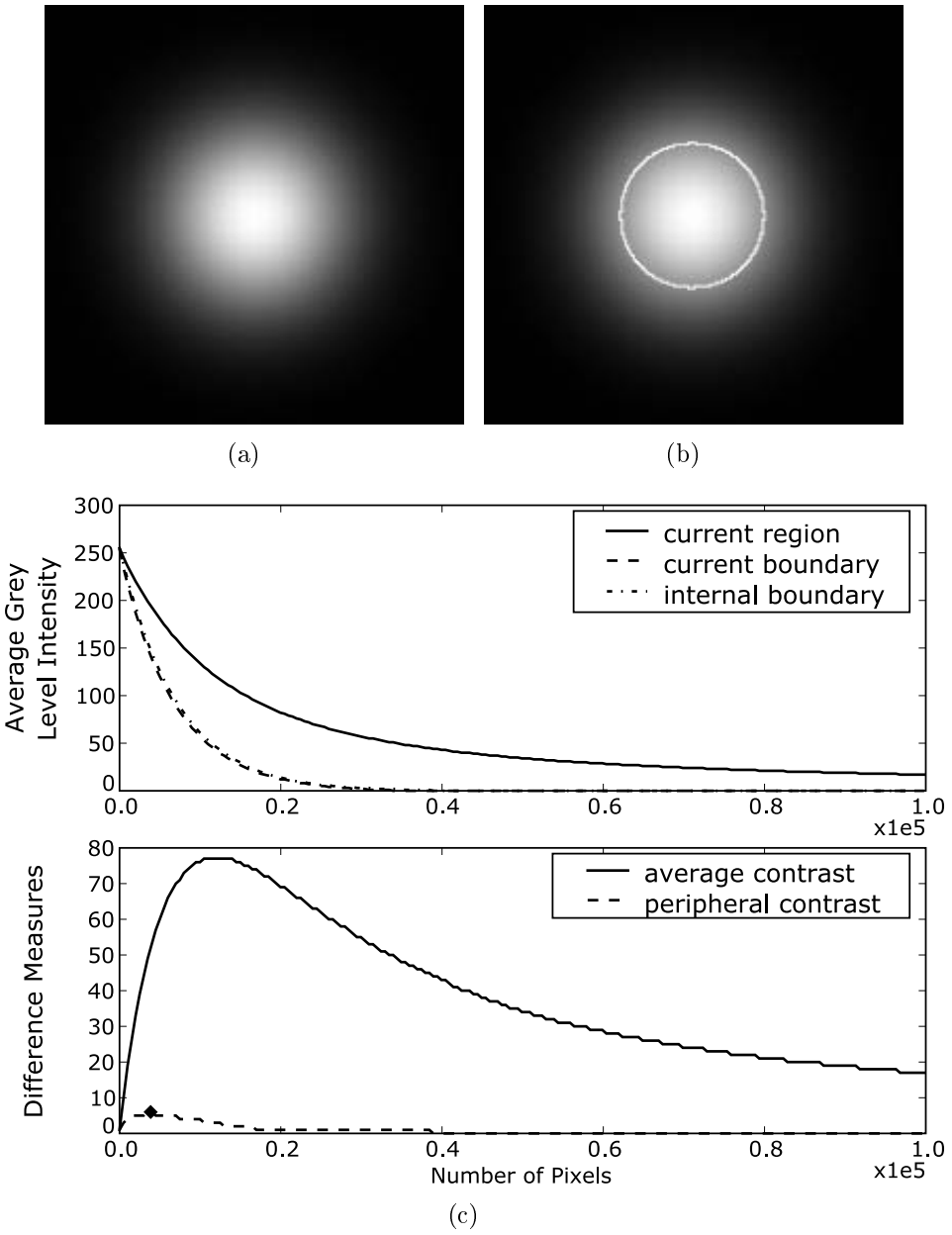


Figure 2.6: Segmentation results for Gaussian image. (a) Original image. (b) Segmentation result. (c) Mappings for grey-levels, peripheral contrast and average contrast obtained during region growing. The segmentation point is indicated with a ♦.

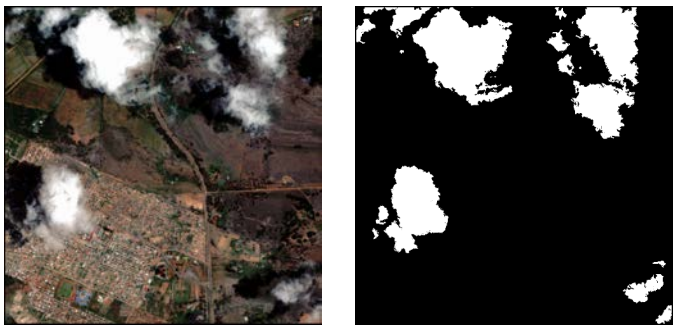


Figure 2.7: A sample Quickbird sub-scene with its cloud mask.

images. Four Quickbird multispectral scenes from which 12 sub-scenes were extracted, and nine Landsat 5 images from which 20 sub-scenes were extracted, were used. Each of the 32 sub-scenes measured 1000×1000 pixels. The sub-scenes were selected to represent different surface or cloud types present in a scene. Surface types present in the scenes were farmland, mountain and urban areas, while cloud types present were cumulus and stratus of various thickness. Images were acquired during the day with solar elevations varying between 32 and 72 degrees. Since only combinations of the visual and NIR channels are applicable to Sumbandilasat, the last three channels from the Landsat 5 images were discarded. To allow reflectance values from different images to be compared, all data was first converted to at-sensor reflectance using equation (2.1.5) and normalised with respect to mean earth sun distance [60]. Satellite specific details regarding conversion from raw sensor data to at-sensor reflectance are available for Landsat 5 [20] and Quickbird [60].

To establish ground truth cloud masks for evaluation purposes, each sub-scene was manually segmented. This was aided by segmentation tools present in the photo editing software used[7]. Segmentation was carefully checked against each channel and false colour composites to ensure that any visibly cloudy pixel be labelled as cloudy. This process took two weeks. Establishing ground truth masks are a known difficulty in evaluation of cloud-detection algorithms [48]; there will invariably be errors in these masks, but effort was made to limit these to a minimum. Figure 2.7 shows an example of a manually created cloud mask. Contrast enhancement (for display purposes only) has caused saturation in the clouds.

Cloud masks were generated by applying thresholds to transformed images. To evaluate a transform, these masks were compared against the manually created cloud masks. Both classification and evaluation were done at the individual pixel level. The performance measure for comparing image segmentation from equation (2.1.3) on page 8 was used.

[7]Krita, a free photo editing program that includes support for 16 bit images, was used [1].

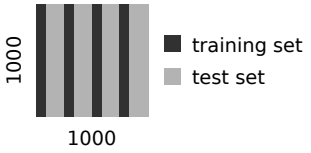


Figure 2.8: Division of image pixels into training and test sets.

Adaptive transform test

The dimension reducing transforms selected for comparison were: the HDA transform discussed in section 2.1.4, the HOT transform from equation (2.1.9) and the D transform from equation (2.1.8). Since HOT and D are data-dependent improvements of the static tasseled-cap and NDVI, also mentioned in section 2.1.3, these static transforms were not evaluated. The single ρ_{blue} and ρ_{red} channels were also included in the comparative tests.

Since HOT was derived for *interactive* masking and correction, in the first test thresholds and transformation parameters were allowed to adapt to each image. Although these adaptive tests are not as practically applicable to on-board implementation as the fixed tests of the following section, they nonetheless provide an additional comparison between transforms. Furthermore, using different parts of a single image for test and training data is common in remote sensing literature. Each 1000×1000 image was divided into a training set (30% of the pixels) and a test set (the remaining 70%). The training set consisted of four evenly spaced vertical image segments, depicted in Figure 2.8. For each image, the transformation parameters, followed by the thresholds, were trained separately. The transformation parameters – b (scalar) for the D transform, ϕ (scalar) for HOT transform and $\boldsymbol{a}$ (4×1 vector) for HDA transform – were trained as described in sections 2.1.3 and 2.1.4. Optimal thresholds were determined for each transform following the example of reference [114]: after applying the transform, a 128-bin histogram was constructed for each image. Outliers were discarded to lessen their impact on the histogram shape: the dynamic range of the image was minimised while keeping 98% of the data. The segmentation error was computed at each threshold level according to equation (2.1.3) from page 8 and the threshold that resulted in optimum segmentation was chosen.

For each image, using the transformation parameters and thresholds derived for it, the performance was evaluated on both the test and the training sets. The average performance was then calculated separately across the training sets and the test sets. Furthermore, data sets as well as results for the Quickbird and Landsat 5 images were kept separate. Images that contained only clouds (two of the 32 images) or only clear sky (four images) were not evaluated for these adaptive tests, since two classes are required to determine b and $\boldsymbol{a}$. Such discarding of images from the test sets was unnecessary for the fixed transform test, discussed below, since parameters were based on averages

over many images. For the adaptive transform test, the training sets contained 7.8 million pixels while the test sets contained 18.2 million pixels.

Fixed transform test

In the second set of tests, the transformation parameters and thresholds were kept fixed across multiple images.

For the first test using this approach, images were again segmented into training and test sets as in Figure 2.8. Since no images were discarded, the training sets contained 9.6 million pixels and the test sets contained 22.4 million pixels. The transformation parameters were again calculated for each image based on its training set, but then averaged across all images. For HDA, each $\boldsymbol{a}$ vector was normalised prior to averaging. As an alternative to averaging, one might pool all the training data from the different images before training each parameter, however averaging had previously been used to derive values for b based on multiple images [31]. Furthermore, tests were conducted that confirmed that pooling training data did not give good results with HDA. When pooling data from different images with different surface types, the 'clear' class has a multi-modal distribution. This explains why training HDA on pooled data gives poor results: HDA's single Gaussian PDF, used to model each of the two classes, cannot accurately represent the spread of multi-modal data.

To determine the optimal threshold, the segmentation error on the training set was calculated for each image, for each histogram bin level. The level that resulted in the lowest total segmentation error across all images was selected as the optimal threshold.

For the second test using fixed transformation parameters, individual images were not segmented into test and training parts. Instead, whole 1000×1000 images were randomly divided into test and training sets. During this division care was taken to ensure that all sub-scenes from the same original scene were in the same set. About half of the images were used for training and the rest for testing. Similar to the previous test, transformation parameters were averaged and the global optimal threshold determined across the training set. These values were then used to segment both sets and the segmentation errors recorded. This test was the most difficult but also the closest to the reality of the on-board application: parameters have to be determined on a set of training images (on the ground) and then applied to a completely different set of images (on board of the satellite).

Statistical significance test

When comparing the classification accuracy of two algorithms with similar performance, the difference in observed performance might be the result of sampling error: given another experiment, the results might be reversed. Using

a statistical significance test allows one to test whether a hypotheses about a population parameter is true [70]. In this case the hypotheses concerning the cloud detection algorithms that must be tested in a mathematically principled manner are:

$\mathbf{H_0}$: The algorithms are equally accurate.

$\mathbf{H_1}$: The algorithms are not equally accurate.

By calculating the probability that differences between the algorithms can be attributed to chance, one can draw conclusions with a specified certainty. This certainty depends upon the amount of data used to evaluate the algorithms. When the accuracy of the two algorithms are close to each other, a large amount of data is required to reject $\mathbf{H_0}$. Conversely, failure to reject $\mathbf{H_0}$ can be caused either by the fact that the algorithms are equally accurate, or by a lack of data to sufficiently discriminate between them.

The McNemar significance test can be used to discriminate between two algorithms classifying common data segments [45, 34]. The performance of the algorithms is first represented by a 2×2 array as shown in Table 2.2.

		Algorithm 2	
		Correct	Incorrect
Algorithm 1	Correct	$N_{00} = n_{00}$	$N_{01} = n_{01}$
	Incorrect	$N_{10} = n_{10}$	$N_{11} = n_{11}$

Table 2.2: The number of occurrences of the joint classification outcomes for two algorithms. N is the random variable and n is the outcome.

In the McNemar test, the cases where both algorithms gave identical classification results, i.e., N_{00} and N_{11}, are ignored since they describe the algorithms' common behaviour. The difference between the algorithms is described by N_{10} and N_{01}. The number of occurrences where only one of the algorithms made an error is given by the random vatiable $K = N_{10} + N_{01}$, with outcome $k = n_{10} + n_{01}$. Under the hypothoses $\mathbf{H_0}$ it can be shown [45] that N_{10} has a binomial B(k, 0.5) distribution. For large k ($k > 50$) and n_{10} not too close to 0 or k, the binomial may be approximated as Gaussian:

$$W = \frac{|N_{10} - k/2| - 0.5}{\sqrt{k/4}}, \tag{2.2.1}$$

where W is a random variable with a Gaussian distribution having a mean of zero and a standard deviation of one. The probability P of observing a given value of N_{10} can then be approximated as:

$$P = 2\mathrm{P}(W > w), \tag{2.2.2}$$

where w is the outcome of W.

Since the McNemar test only applies to a pair of two algorithms (while 5 cloud detection algorithms are to be compared) the solution is to consider all possible pairs of algorithms. Given that the algorithms classify per pixel, we assume the classification errors are statistically independent, which is a prerequisite for applicability of the McNemar test. The McNemar counts from Table 2.2 for all testing results from the various fixed and adaptive transform experiments where added together. No training results were included. The results of the pairwise statistical significance tests are presented in section 2.3.1.

2.2.2 Region growing

Since a comparative evaluation of the region growing algorithm (section 2.1.6) has not been done [54], it was compared to a single global threshold on the basis of segmentation performance.

The same data set described in section 2.2.1 on page 22 was used. The thresholding segmentation error was evaluated using the experimental set-up from the fixed transform test in section 2.2.1: test and training sets consisted of whole images. To increase the size of the experiment, the different dimension reducing transforms evaluated in section 2.2.1 were applied to generate different greyscale images. In keeping with the fixed transform test's methodology, transforms were averaged across the whole training set and the global optimal threshold for the entire training set was selected. Thus, for each base multichannel image, five greyscale images based on ρ_{blue}, ρ_{red}, HOT, D and HDA (a total of 75 training and 80 test greyscale images each with dimensions 1000×1000) were to be segmented using both thresholding and region growing. The per-pixel segmentation errors for each greyscale image were averaged across all images as well as all transforms in a set, preserving only the training–test and Landsat–Quickbird divisions.

The seed-points for the region growing algorithm were derived from the cloud masks created by applying the above mentioned global thresholds. The central point of each connected region in a mask was used. Similar to thresholding, the per-pixel segmentation errors of the resulting region growing masks were calculated using equation (2.1.3) on page 8. These errors were averaged across all images and transform types as described in the previous paragraph. Results are presented in section 2.3.2 on page 46.

2.2.3 Measuring cloud dispersion

Justification

It can occur in remote sensing that the target only covers a fraction of the image. In such a case it is possible that the image has some cloud cover but it

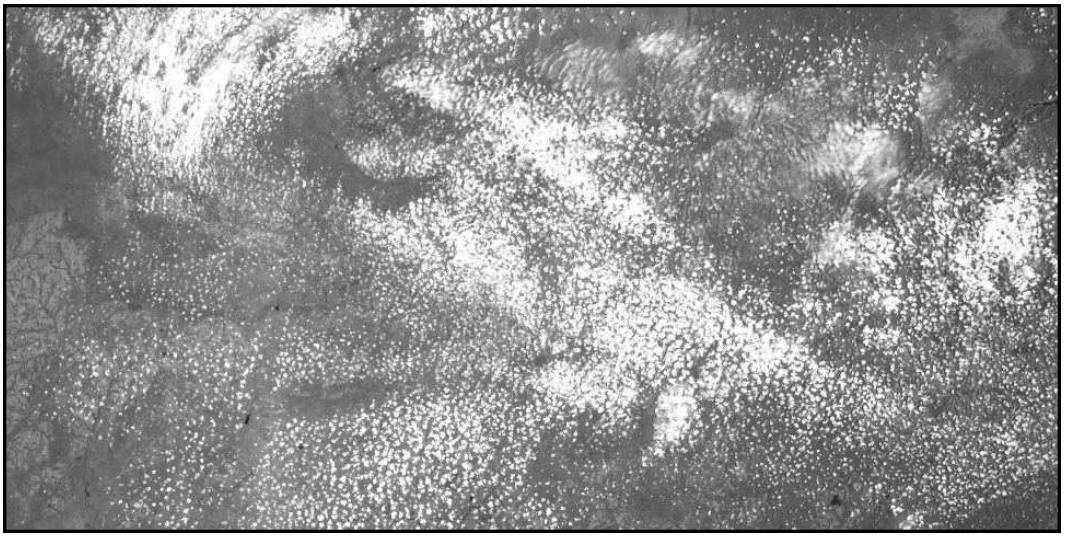
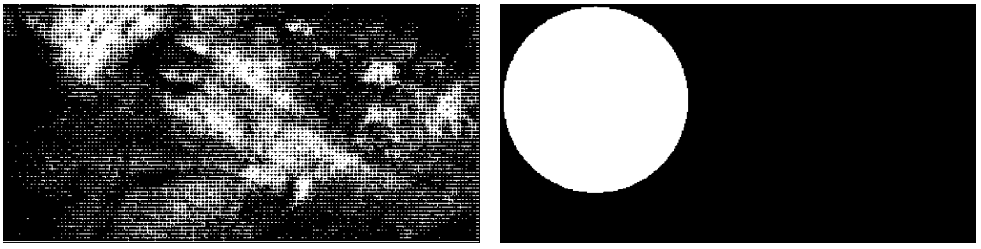


Figure 2.9: A cloudy scene with high dispersion.



(a) Dispersed cloudy pixels. (b) Concentrated cloudy pixels.

Figure 2.10: Comparison of dispersion amount using masks. (a) and (b) have the same total cloud cover.

does not occlude the desired target. Alternatively it is possible that the image has very slight cloud cover that does occlude the target. Therefore, whilst the *amount* of cloud cover is the primary indicator of a cloud-corrupted image, the specific location of the cloud is also significant. However, the location of a specific target within an image is seldom known during imaging. Thus, a more general and useful measure is perhaps the dispersion of the clouds.

Figure 2.9 shows an example of a scene with only about 20% cloud cover, but very high dispersion. It is unlikely that much useful information can be extracted from this image. In Figure 2.10 a cloud mask based on the image from Figure 2.9 is compared with a mask where the same amount of cloud cover is concentrated in a single area. It is clearly possible in Figure 2.10(b) that the target could be located in the right half of the image and therefore visible. Measuring cloud dispersion is consequently justified when one wants to determine if a given target might be occluded.

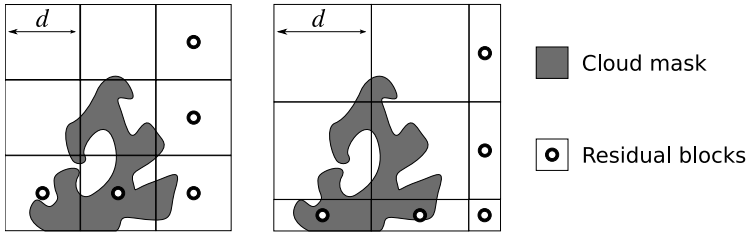


Figure 2.11: Residual blocks encountered with continuously varying block size. In the left hand figure the marked blocks are about the become residuals if d increases. In the right hand figure some of the residuals no longer have cloud cover although the number of blocks has stayed the same.

Algorithm design

An algorithm was developed to measure the dispersion of cloudy pixels throughout the image. Similarly to the noise estimation algorithm discussed in Chapter 3, the image is divided into blocks of varying sizes during different stages of the algorithm. For a given block size, the percentage of blocks that contain no cloudy pixels is calculated and this used as a dispersion measure. To understand this measure one must observe its behaviour with varying block size.

When the block size is equal to 1×1 pixel, there is no difference between an image with dispersed clouds (Figure 2.10(a)) and an image with all the cloudy pixels tightly grouped together (Figure 2.10(b)). This is logical since a 1×1 pixel block size corresponds to the conventional notion of cloud cover. At the other end of the scale, when the block size is equal to the image size, it is again impossible to distinguish between the images from Figure 2.10. Since any image with *some* cloudy pixels will have a 100% cover in this case, it is not surprising. Hence the useful information is located between these two extreme block sizes. An attempt was made to vary the block size continuously and observe the behaviour of the dispersion measure. Dividing the image into $d \times d$ pixel blocks and letting

$$1 \leq d \leq \text{image size},$$

graphs like those in Figure 2.12 can be generated, expressing the percentage of clear blocks as a function of block size. A problem arises when the image size is not an integer multiple of the block size: how does one weigh the contribution of the smaller, residual blocks (see Figure 2.11)? Initially they were weighed the same as the other blocks. This resulted in a distorted dispersion measure graph that sometimes increases (see Figure 2.12(c)). As d increases a residual block that previously overlapped with the cloud mask can decrease until it has no cloud cover, undesirably increasing the percentage of clear blocks. Although the residual block's size continues to decrease as d increases, its relative weight stays the same resulting in the stair step appearance of Figure 2.12(c). When

each block's contribution to the percentage of clear blocks is weighed by its relative size, a graph like Figure 2.12(d) is generated. It looks slightly better but the same problem persists; as d increases, residual blocks that previously had cloud cover 'slide' off the cloud mask and become clear. However, while increasing d, there is no useful information to be extracted when the total number of blocks remains unchanged. Therefore, it would be more useful to directly increase the number of blocks. Nevertheless, the problem of what to do when the image proportions are not an integer multiple of the block size remains. One option is simply to disregard residual area, but this might discard useful information. The solution devised is to down-sample the image so that the sidelengths are both powers of two and then let the number of blocks also be a power if two. This ensures that the image proportions are always an integer multiple of the block size. Although information is still discarded, it is evenly distributed throughout the image, so it does not affect the dispersion measure. A more detailed discussion of down-sampling is given in section 2.2.4. The dispersion measure can be described mathematically:

$$s(d_l) = \text{percentage of clear blocks in cloud mask} \tag{2.2.3}$$

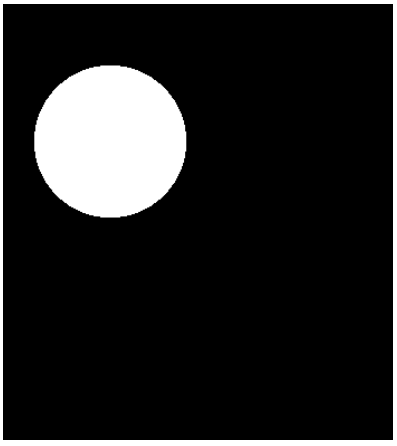
$$d_l = 2^l, l = 1, 2, 3, \ldots, n, \tag{2.2.4}$$

where $d_l \times d_l$ is the block size and 2^n is the sidelength of the shortest side of the image after down-sampling.

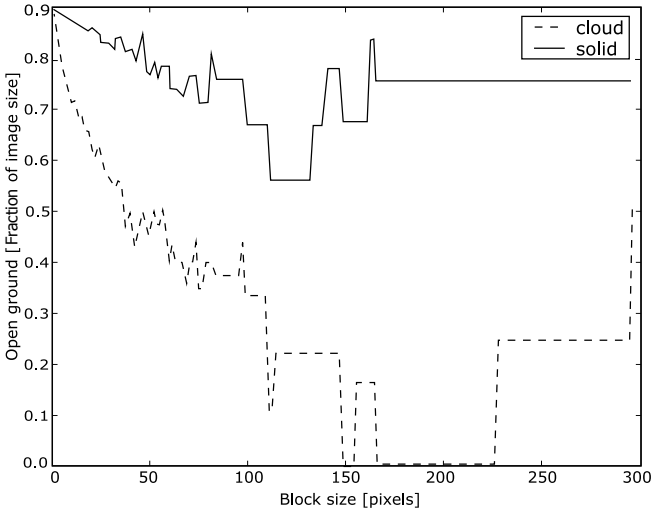
Figure 2.12(e) shows the result of this approach. Note that the maximum block size is now 256 pixels (the original image was approximately 300×300). By looking at the central area of the graph, it is clearly possible to discern the dispersed and the concentrated cloudy images. When the algorithm is applied to the two images from Figure 2.10, the difference is even more pronounced, Figure 2.13, since the dispersion of Figure 2.10(a) is more extreme. The result is that the amount of open ground rapidly decreases with increasing d_l. Also note that, because of the aspect ratio of the image, there are still two blocks in the image at maximum d_l. This explains why the `solid` graph has a minimum of 50% instead of 0%.

In its final form the dispersion measure algorithm can be seen as a variation of an image pyramid. Image resolution pyramids are an efficient way to analyse global, intermediate and local scale effects in remote sensing image processing, [47, pp. 351–354] and [94, pp. 265–271], and form the basis of multi-resolution techniques such as wavelet expansions. The noise estimation method discussed in section 3.1.4 also uses image pyramids. It is common to refer to the tessellation of the image into $2^l \times 2^l$ blocks as level l of the pyramid. In section 2.3.3 the dispersion measure is used in conjunction with thresholds to classify clouds into suitable and unsuitable categories.

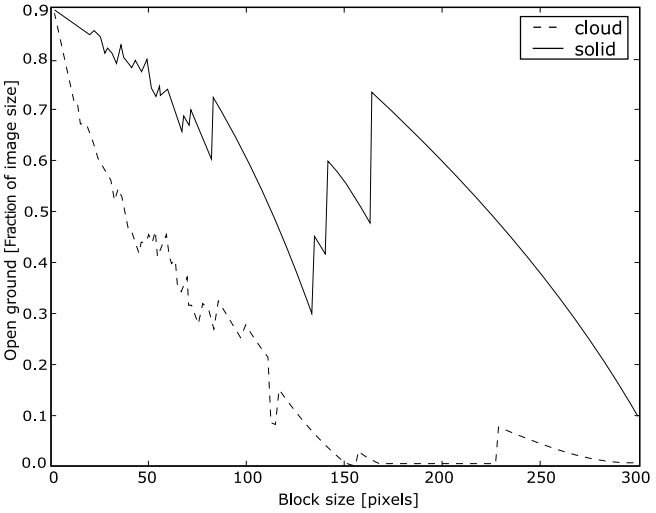
(a) Dispersed pixel cloud mask. `cloud` in graphs.



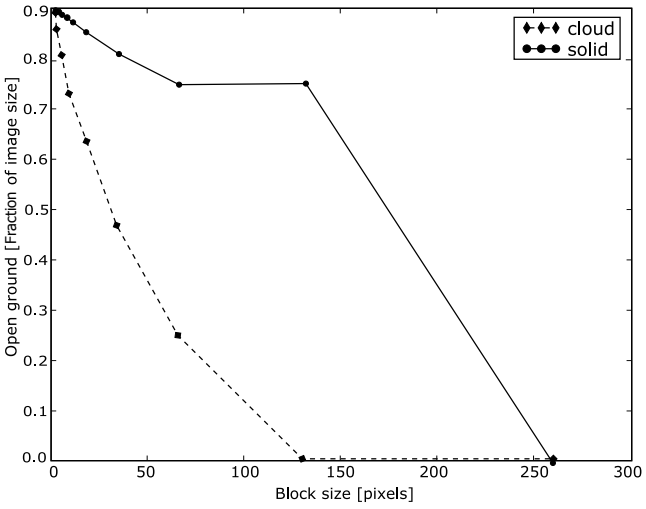
(b) Concentrated pixel cloud mask. `solid` in graphs.



(c) Residual blocks weighed evenly.



(d) Residual blocks weighed according to size.



(e) Image resampled and integer number of blocks used.

Figure 2.12: Design of the dispersion measure. (a) and (b) show the two input images. (c) and (d) show attempts at continually varying the block size, while (e) shows the final implementation of the algorithm.

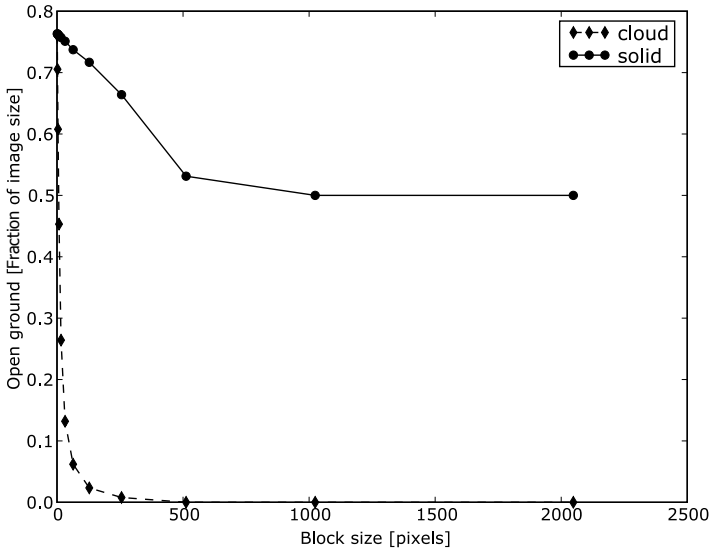


Figure 2.13: The images from Figure 2.10 show pronounced differences in dispersion measure.

Using the measure in an experiment: Introduction of thresholds

An attempt was made to use the dispersion measure to classify cloud masks into suitable and unsuitable based on cover and dispersion. Although, in its current form, this experiment cannot be integrated into the classification system developed in Chapter 5, it nevertheless demonstrated the potential usefulness of the dispersion measure.

A graph is difficult to use in a classifier; a single number is preferable. As described in section 2.2.3, it is the centre area of $s(d_l)$ (as opposed to the left or right end-points, $l = 1$ or $l = n$) that contains useful dispersion information. After examining different dispersion measure graphs similar to the ones in Figure 2.12 and 2.13, it was decided that a meaningful single number to use is the open-ground percentage of the third last entry, $l = n - 2$, in the dispersion graph, $s(2^{n-2})$. This corresponds to an image divided into at least $2^2 \times 2^2 = 16$ blocks (for a more or less square image). If the one side of an image is between two and four times the length of the other, this would correspond to division into $(2^2 \times 2^2) \times 2 = 32$ blocks.

For this experiment hard thresholds were used to classify an image as suitable or unsuitable. First, a threshold was introduced to ensure blocks are not counted as cloudy when only a few pixels of a big block are cloudy: blocks are only considered cloudy if more than T_1 pixels are cloudy. After some experimentation T_1 was set to 3%: at this level it had the desired result of suppressing the effect of blocks with very slight cloud cover.

To prevent images with extremely scarce but widely dispersed cloudcover

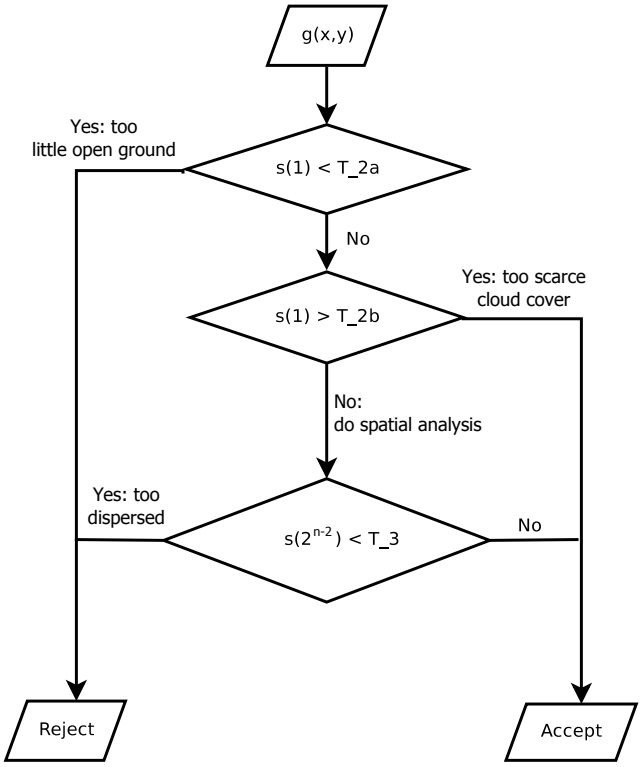


Figure 2.14: Flow diagramme of the dispersion classification algorithm.

from being rejected, or images with so much cloud cover that spatial analysis is unnecessary from being analysed, a preliminary screening is done:

$$g(x,y) = \begin{cases} \text{unacceptable (too little open ground)} & \text{if} \quad s(1) < T_{2a} \\ g_{spat}(x,y), \text{ a candidate for spatial analysis} & \text{if} \quad T_{2a} < s(1) < T_{2b} \\ \text{acceptable (too scarce cloud cover)} & \text{if} \quad T_{2b} < s(1), \end{cases} \tag{2.2.5}$$

where $g(x,y)$ is the cloud mask, $T_{2a} = 50\%$ is the unconditional reject threshold and $T_{2b} = 90\%$ is the unconditional acceptance threshold. Finally the deciding threshold, T_3, to be applied to remaining candidate images' third last dispersion measure entry, was set to 40%:

$$g_{spat}(x,y) = \begin{cases} \text{too dispersed} & \text{if} \quad s(2^{n-2}) < T_3 \\ \text{acceptable} & \text{if} \quad T_3 \leq s(2^{n-2}). \end{cases} \tag{2.2.6}$$

A flow diagramme of the decision process is presented in Figure 2.14. Thirty one cloud masks (of which 17 were artificially generated, i.e., were not derived from actual satellite photos) were classified into acceptable and unacceptable groups using the method just described. The results are presented in section 2.3.3.

2.2.4 Down-sampling options

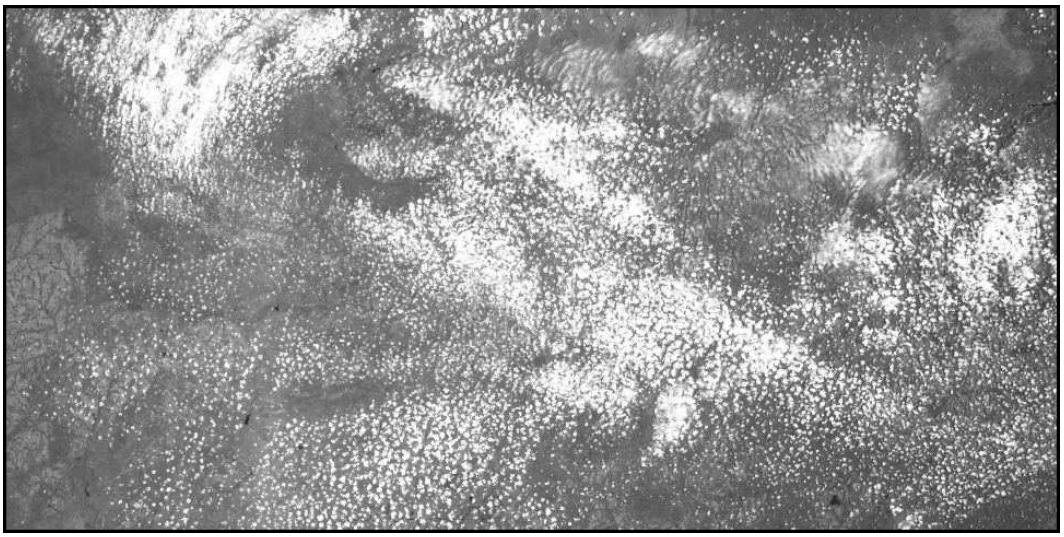
When measuring blur and noise, the assumption of spatial uniformity comes to the rescue since one can consider only a subsection of an image. However, when analysing cloud cover, the *whole* image has to be considered, as seen in the previous section. Memory constraints on board the satellite might mean it is not possible to load the entire image into memory[8]. For any technique requiring some variation of spatial analysis (such as the region-growing algorithm from section 2.1.6 or the dispersion measure from 2.2.3) this poses a problem. The solution is to retrieve parts of the image from secondary storage and down-sample them so that a reduced resolution version of the entire image can be assembled in RAM.

An experiment aimed at measuring the effect of down-sampling on cloud detection was conducted. First, suitable cloudy images were chosen. Two high resolution cloudy images[9] (Figure 2.15(a) and (b)), one medium resolution image (2.15(c)) and one low resolution Sunsat image (2.15(d)) were chosen. The images were selected to represent a variety of cloud types: (a) has a low cloud cover but high dispersion, (b) has higher cloud cover, (c) has scarce cover and (d) has solid clouds with high cover. To measure the effect of down-sampling on segmentation, a ground truth segmentation has to be established. To this end cloud masks were manually created using Corel Photo-paint™. Based on these masks an optimal threshold for each image was computed using the method described in section 2.2.1. These thresholds were then used to generate new cloud masks from the original resolution images. The images were successively down-sampled using one of two methods and the same thresholds used to generate new cloud masks at the lower resolutions. By comparing the cloud cover percentage estimates at the lower resolutions to those at the original resolution the relative error introduced by down-sampling can be assessed.

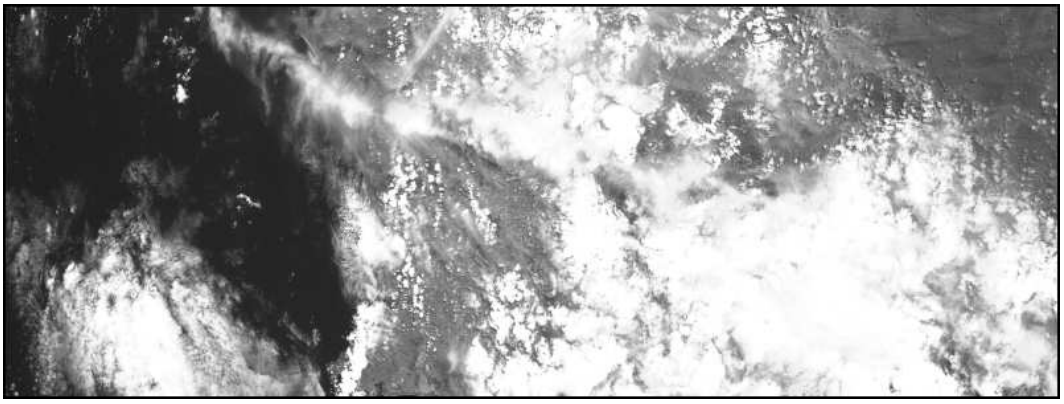
Since down-sampling maps multiple input pixels to a single output pixel, there are different methods available to weigh the input pixels. The two down-sampling methods used were the `NEAREST` and `ANTIALIAS` resampling filters of the `thumbnail()` function of the Python Imaging Library (PIL) [2]. The `NEAREST` resampling filter picks the nearest pixel from the input image and ignores all other input pixels. In the case of `ANTIALIAS`, a weighted average of the input pixels is used to produce each output pixel. Results are presented in section 2.3.4 on page 53.

[8] It is not even possible to load a *single* channel of an image completely into the 64MB RAM on Sumbandilasat.

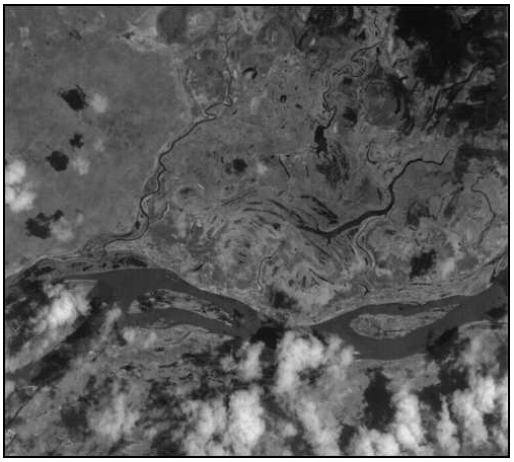
[9] Acquired from the Satellite Application Centre.



(a) 10000×5000



(b) 9866×3676



(c) 2263×2072

(d) 349×327

Figure 2.15: Images used in the down-sampling experiment. The resolution of each image is given.

2.3 Results

2.3.1 Dimension reducing transforms

Illustration of the unsuitability of LDA

As described in section 2.1.3, the popular LDA transform is unsuitable for dimension reduction during cloud detection, because of the differing covariance matrices of the two classes. An example of LDA's failure is presented here for a single image. Dimension reduction using both LDA and HDA was applied to the image shown in Figure 2.7 on page 24, with the scatterplot in Figure 2.3(b) on page 15. The training segment for this image, as defined in Figure 2.8 on page 25, was used to determine the parameters. Histograms for the projected classes are presented in Figure 2.16. Because no distinction is made between the two classes' covariance matrices, LDA's minimisation of within-class variance has resulted in completely overlapping classes in the projected space. Observe the transformation vector weighs $\boldsymbol{a}^{\mathrm{T}} = [a_{blue}, a_{green}, a_{red}, a_{nir}]$ for each transform:

$$\boldsymbol{a}_{\mathrm{HDA}}^{\mathrm{T}} = [0.92, 0.13, -0.36, 0.00], \quad \boldsymbol{a}_{\mathrm{LDA}}^{\mathrm{T}} = [-0.33, 0.68, -0.58, 0.30].$$

It is interesting to note that while HDA gave most weight to the blue channel followed by the red channel (it automatically chose the two channels used in the TC and HOT transforms), LDA gave most weight to the green and the red channels and erroneously subtracted the blue channel.

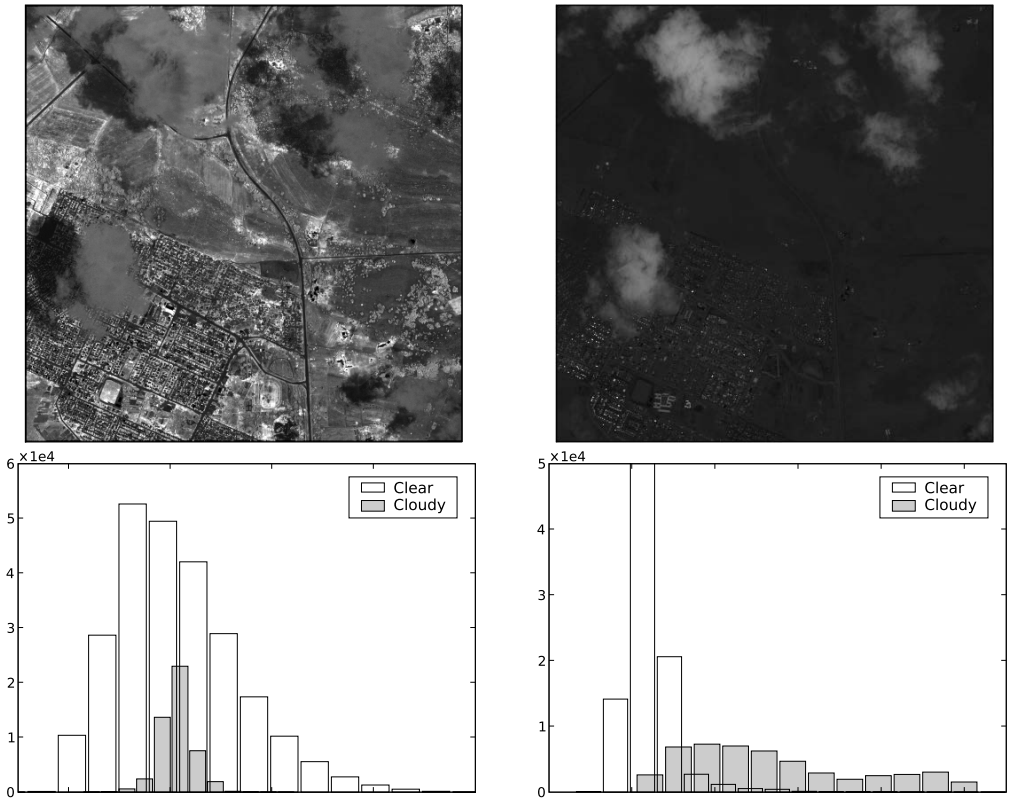
Adaptive transform test

The average per-pixel segmentation errors after applying each image's optimal transforms and thresholds are presented in Figure 2.17.

For images from both the Landsat and Quickbird satellites the performance difference between the training and test sets was small. Such similar performance is to be expected for the training–test division depicted in Figure 2.8 on page 25: since the data in both sets were taken from adjacent areas, segmentation parameters were easily generalisable.

Applying thresholds to ρ_{blue}-images gave better segmentation results than applying thresholds to ρ_{red}-images. Compared to the clouds, the backgrounds were darker in the ρ_{blue}-images. Thin clouds and hazes were included in the 'cloud' set and were more visible in the ρ_{blue}-images against these darker backgrounds. In the ρ_{red}-images, brighter backgrounds led to increasing overlap between classes.

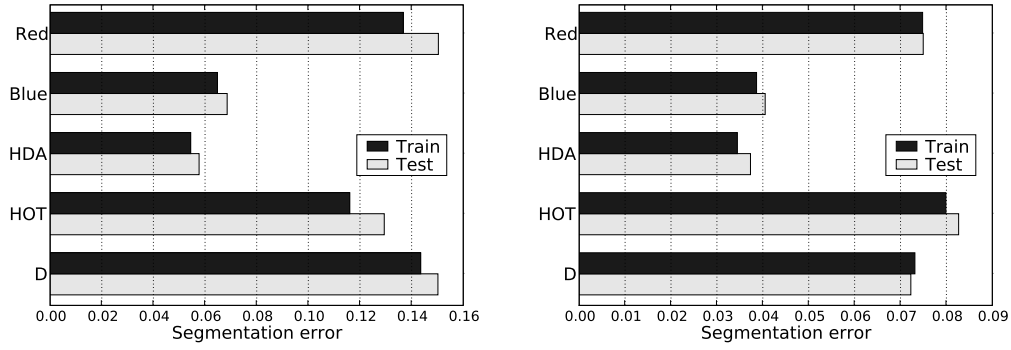
In ρ_{nir}-images, this overlap was greater than in ρ_{red}-images. Since the D-images are formed by combining ρ_{red}- and ρ_{nir}-images, it is to be expected that such a combination will not separate the 'clear' and 'cloudy' classes of data set well. The original requirements for the D transform did not include detection of thin clouds [31]; therefore its relatively poor performance on thin



(a) LDA projection. The classes overlap completely.

(b) HDA projection. The overlap between classes is minimised.

Figure 2.16: LDA fails to minimise overlap between classes in in projected space.



(a) Quickbird images.

(b) Landsat 5 images.

Figure 2.17: Test results for the adaptive transform test. Reported errors are per pixel, for example 0.04 implies 4% of pixels were incorrectly classified.

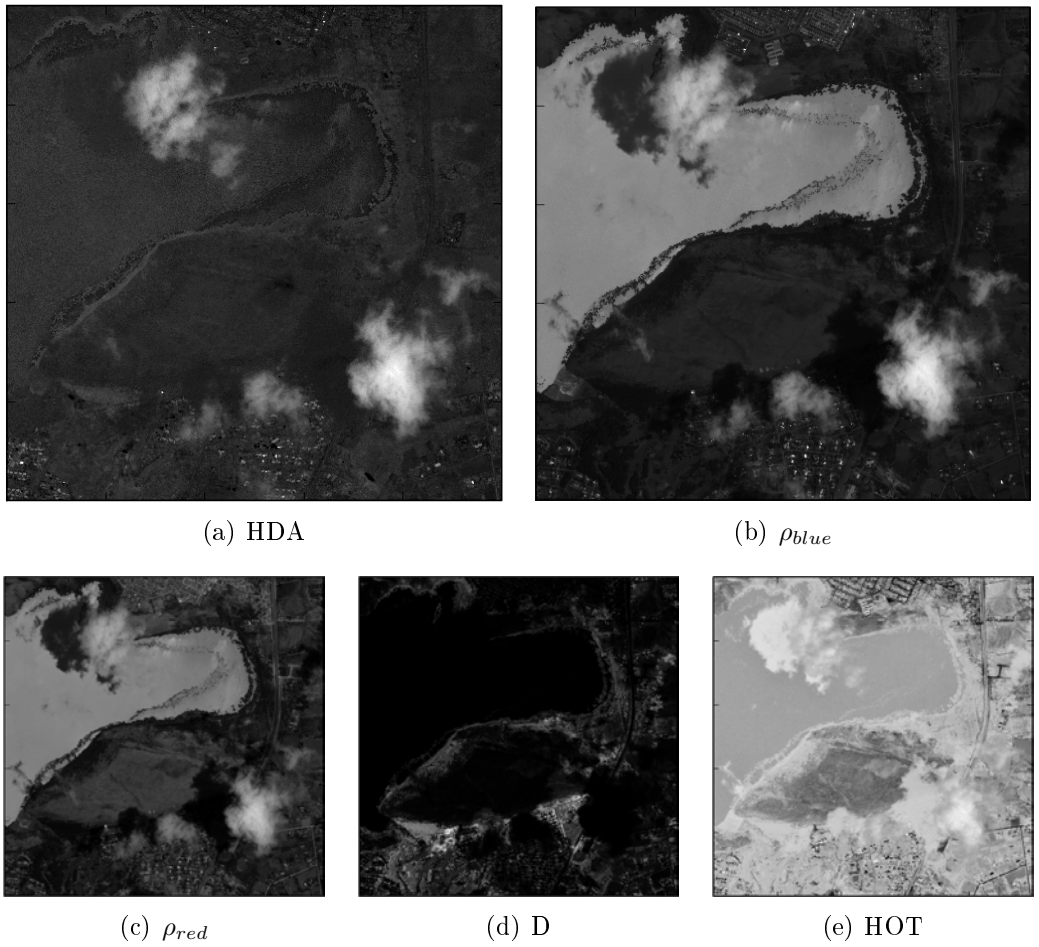
clouds is reasonable. Since the Landsat images contained more thick cumulus clouds, these images had better class separation than the Quickbird images. This is reflected in the average performance differences between the two sets in Figure 2.17(a) and 2.17(b). However it is also visible in the slight relative performance increase of the D transform in Figure 2.17(b).

The HOT transform generally did not give better segmentation performance compared to using only the ρ_{blue}-image. This might point to its emphasis on haze correction as opposed to haze and cloud detection. However, in some cases this transform failed completely in a manner similar to LDA in Figure 2.16(a). In these cases both haze and thick cloud were projected onto the same range as the clear class. It accounts for the high average segmentation error in Figure 2.17(b). Individual examples are elaborated on below.

For all four cases (test and training in Figure 2.17(a) and (b)) HDA gave the best average segmentation performance. It is useful to investigate some individual images. One Quickbird sub-scene included a difficult patch of cloud over a lake shoreline. Figure 2.18 shows the different transformed images. The lake is bright in the ρ_{blue}- and even the ρ_{red}-image (Figure 2.18(b) and (c)), resulting in an overlap with the cloud class. HOT is known to give poor results over water [120], and fails to separate the two classes (Figure 2.18(e)). D images do not visualise well: since the transform is non-linear, it compresses the dynamic range of certain areas in the image. Note that, for D, clouds should map closer to zero than other classes. While clouds do map to zero, the lake and some of the surrounding urban structures also map to zero in this case (Figure 2.18(d)). The HDA-image suppresses the bright lake, resulting in better separation between cloud and background, both visibly (Figure 2.18(a)) and quantitatively (segmentation error for HDA is 2.9% compared to 5.1%, 5.6%, 6.3% and 9% for ρ_{blue}, ρ_{red}, D and HOT respectively). This suppression is allowed by HDA's extra freedom to use any combination of the four available channels: for this image $\boldsymbol{a}^T = [0.64, -0.75, 0.15, 0.04]$. In this case the green channel, not used by any of the other transforms, played an important role in decreasing the segmentation error. In most other HDA-images the green channel did not outweigh the blue, but in cases where HOT gave poor separation, the HDA green component often outweighed the red component.

In cases where the HDA transform relied primarily on the red and blue channels, similar to the HOT transform, it is interesting to observe the differences between the two. Going back to the image from Figure 2.7 on page 24, the scatterplot from 2.3(b) is repeated in Figure 2.19 with the projection directions for both HDA and HOT superimposed. To decrease the overlap between classes, the HDA projection gave more weight to the blue channel: segmentation error of HDA is 4.2% as opposed to 4.8%, 5.8%, 10% and 12% for HOT, ρ_{blue}, ρ_{red} and D respectively.

In another image, not shown here, of thin haze over bright urban areas, all transforms struggled to separate the classes, as expected. However, the HDA



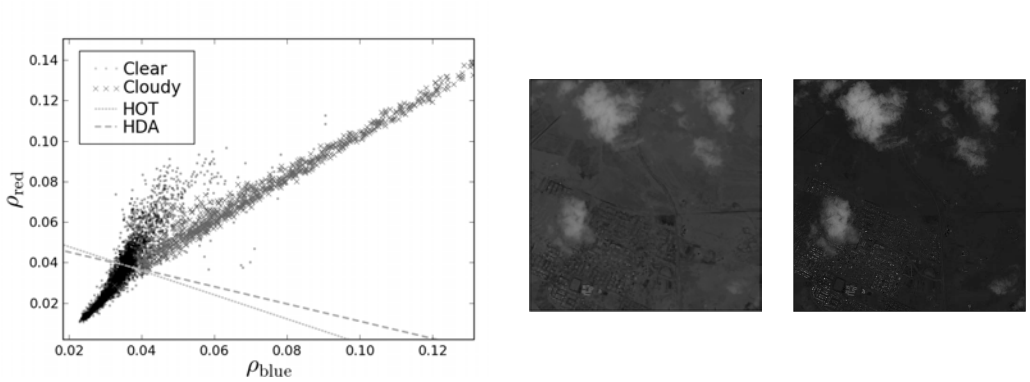
(a) HDA (b) ρ_{blue}

(c) ρ_{red} (d) D (e) HOT

Figure 2.18: HDA suppresses the lake for better segmentation, while HOT and D increases overlap between classes.

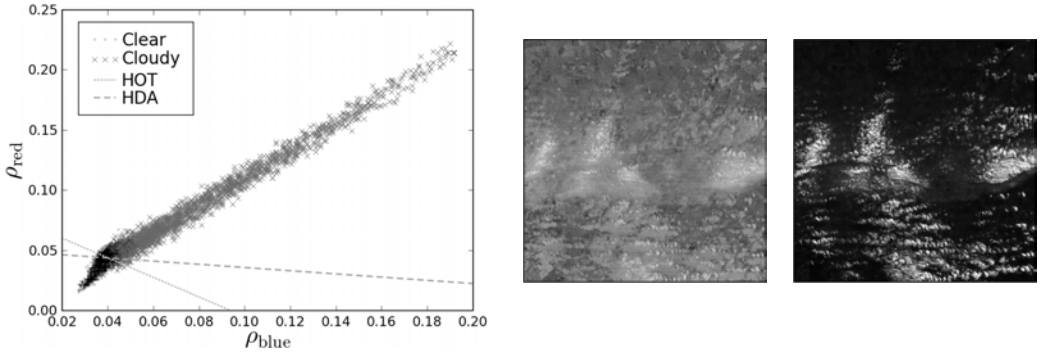
again gave better segmentation error than the single channels and the other transforms.

For five of the Landsat images, HOT increased the overlap between the classes, resulting in considerably worse performance than the other three transforms. This might be ascribed to the HOT transform's design goals: its aim was not primarily cloud detection, but rather to reduce the reflectance variation of cloud-free surfaces so that atmospheric correction for thin clouds can be attempted. For the example in Figure 2.20 the HOT segmentation error was 26% compared to the other transforms' errors of about 5%. Note that, while the HDA gave significant weight to the green channel in this case, the HDA projection line in 2.20(a) only depicts the blue and red components.



(a) Scatterplot with projection directions. (b) HOT-image. (c) HDA-image.

Figure 2.19: A different projection direction in the blue–red-space increases class separation of HDA compared to HOT. HDA gives more weight to the blue channel to decrease overlap in the projected space.

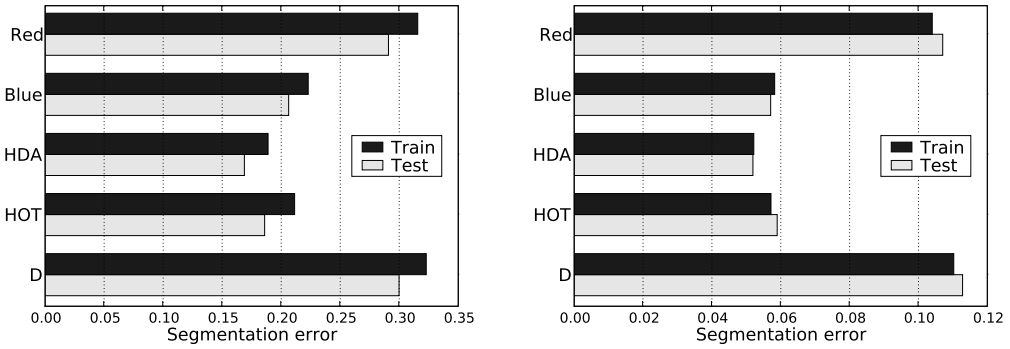


(a) Scatterplot with projection directions. (b) HOT-image. (c) HDA-image.

Figure 2.20: In certain cases HOT severely reduced separation ability, while the other transforms retained it.

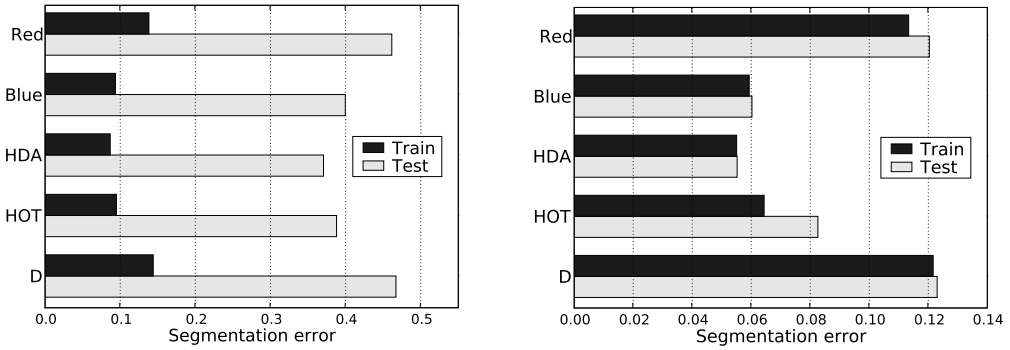
Fixed transform test

Figure 2.21 gives the average segmentation error for the test described in section 2.2.1: fixed transformation parameters and thresholds across training and test sets consisting of segmented images. Once again, for all four cases, the HDA transform gives the best average performance. The differences between training and test data for all transforms were small, indicating good generalisability in spite of a large data set. The training error is slightly larger than the test error for all transforms in the Quickbird data set, as depicted in Figure 2.21(a). This unexpected behaviour was caused by a quirk in the data: three of the 12 images had large, difficult-to-classify areas at the left edge of the image. Due to the training–test division from Figure 2.8 on page 25,



(a) Quickbird images. (b) Landsat 5 images.

Figure 2.21: Test results for the fixed transform test with segmented images in training and test sets.



(a) Quickbird images. (b) Landsat 5 images.

Figure 2.22: Test results for the fixed transform test with whole images in training and test sets.

these images resulted in disproportionately large training error. The relative performance of HOT segmentation has improved compared to the adaptive test. This was caused by its mapping of optimal segmentation points to a smaller range compared to for example ρ_{blue}, decreasing sensitivity to a global threshold. Thus, if each image were to be segmented with its own optimal threshold, the performance of the other transforms would increase more than that of HOT. However, since its optimal thresholds lay closer together, its performance when applying a global threshold was comparable to, though not better than that of HDA.

Figure 2.22 presents the average segmentation error for the test where the training and test sets consisted of completely different images. The relative

rank of the different transforms was similar to that from Figure 2.21, with HDA again giving the best performance in each of the four cases. The dramatic difference between the test and training performance in Figure 2.22(a) can be ascribed to two difficult test images, containing large areas of thin clouds, which all transforms completely failed to identify. In the previous test, figure 2.21(a), these images were segmented across both training and test sets, which decreased the training performance but increased the test performance. For the Landsat images, Figure 2.22(b), overall performance only decreased slightly compared to Figure 2.21(b). Since these images contained mostly thick clouds, class separation was better and images less sensitive to variations in thresholds.

Statistical significance test

The McNemar counts, as described in section 2.2.1 and Table 2.2 on page 26, are presented in Table 2.3. These counts are for the adaptive transform test as well as the two fixed transform tests. Only testing data results are included. One can feel intuitively that, for any given pair, the probability of n_{10} and n_{01} being drawn from a B(k,0.5) distribution is low since nowhere is n_{10} approximately equal to n_{01}. Table 2.4 shows the w outcomes, as computed using equation (2.2.1). Recall that, under $\mathbf{H_0}$, the hypothosis that the algorithms are equally accurate, w should be an outcome of a Gaussian random variable with a mean of zero and a standard deviation of one. Clearly this is not the case. The probability P (computed according to equation (2.2.2)) that observed differences between any two algorithms arose by chance rounded to 0.00 in all cases, as expected given the values in Table 2.4. Therefore, due in part to the very large number of data samples in the combined testing sets, it is possible to conclude with 99.9% certainty that observed differences between the algorithms did not arise by chance.

Table 2.3: McNemar counts for the dimension reducing transforms. The transforms evaluated are listed in the top row and right-most column. The four values at the intersection of a given pair of models represent the joint classification count, as explained in Table 2.2 in section 2.2.1

R		HDA		D		HOT		
46 767 960	973 109	50 643 395	1 204 202	46 608 717	932 613	48 424 841	1 769 840	B
4 346 817	5 512 114	471 382	5 281 021	4 506 060	5 552 610	2 689 936	4 715 383	
		46 731 102	5 116 495	47 126 422	414 908	45 711 486	4 483 195	R
		1 009 967	4 742 436	614 647	9 444 023	2 029 583	5 375 736	
				46 584 617	956 713	49 245 299	949 382	HDA
				5 262 980	4 795 690	2 602 298	4 803 021	
						45 587 150	4 607 531	D
						1 954 180	5 451 139	

Table 2.4: Outcomes w of W for the dimension reducing transforms. Clearly, the probability that any of these values have been drawn from a zero mean, 1 standard deviation Gaussian random variable is very low.

R	HDA	D	HOT	
1 462	566	1 532	435	B
	1 659	196	961	R
		1 726	877	HDA
			1 035	D

2.3.2 Region growing

Upper limit

Although it is claimed that the stopping rule makes the region-growing algorithm insensitive to the size of the upper limit, this is not always the case. In the very first test image no upper limit was specified, i.e., the region was allowed to grow to the full image size and the last local maximum of peripheral contrast before the global maximum of average contrast was used to segment the image. The resulting segmentation boundary is shown in Figure 2.23. The seed point was located inside the cloud in the lower left of the image. The final segmentation boundary identifies the entire island as cloud. By examining the grey level and difference measures in Figure 2.24 one can understand why this is the case.

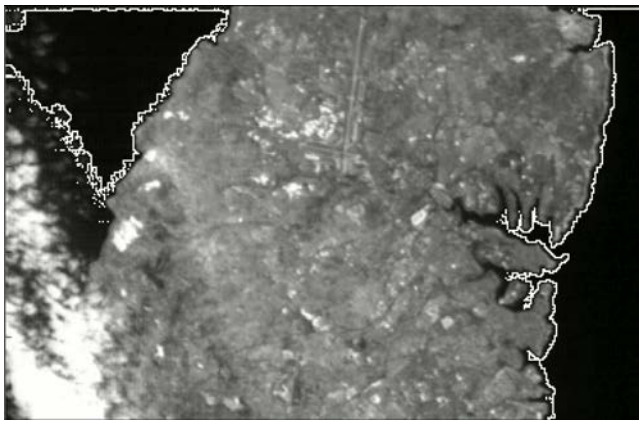


Figure 2.23: Region-growing segmentation boundary with no upper limit imposed. The white line is the segmentation boundary.

The intensity measures behave as expected, starting at a maximum at the seed point and decreasing more or less monotonically. One can see the three distinct levels the region consumes as it grows by considering the current boundary: the bright initial cloudy area (pixels 0–5000) followed by the land area (pixels 10000–70000) and finally the darker ocean (pixels 70000–92000). The current boundary grey levels take a dip at about pixel 8000 when more of the boundary is located over ocean than land.

An average contrast peak corresponding to the cloud boundary occurs early in the growing process. As the region grows into the darker background of the island, the average contrast decreases. However, once the region starts growing into the still darker ocean the average contrast between the region (now comprised of both the very bright cloud and relatively bright island) and the background increases to more than its previous levels. Therefore the segmentation occurs at this second peak in Figure 2.24 indicating that the

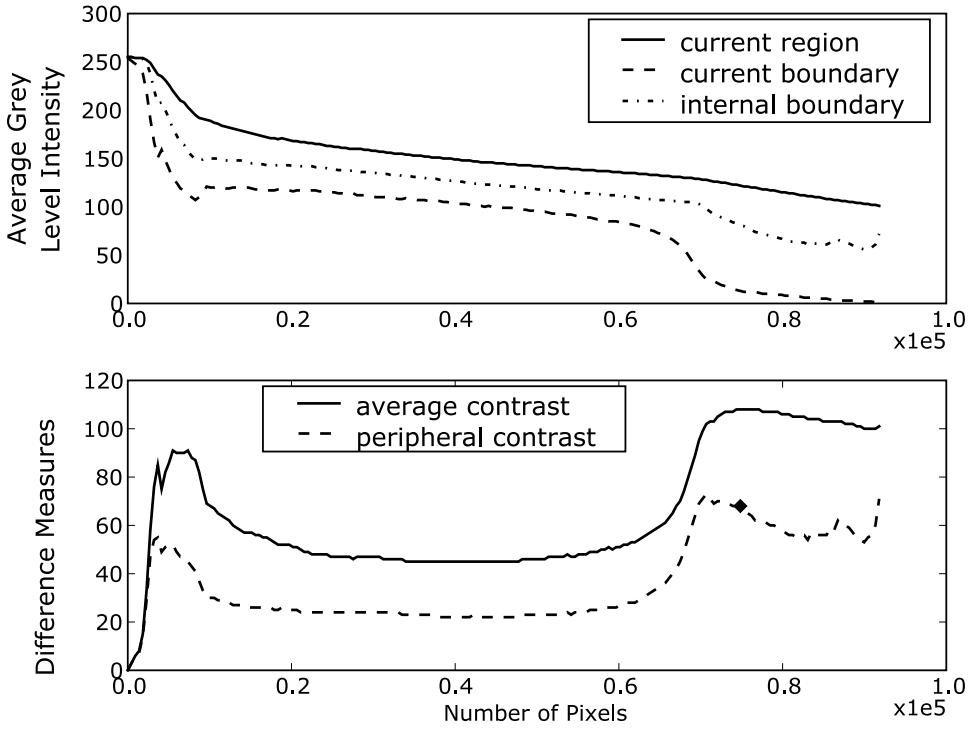


Figure 2.24: Difference measures and grey levels with no upper limit imposed. Segmentation point is indicated with a ⧫.

contrast between the island and the background is greater than the contrast between the cloud and the island.

To curb this type of behaviour it was decided to set the upper limit of the algorithm to some value smaller than the total image size. A multiple of the rough mask (used to establish the seed point) area size was used, as discussed in Appendix A.3 where implementation details of the region growing algorithm are given. Figure 2.25 shows the results of this limit. Proper segmentation occurs at the first peak, which is now the only peak, as shown in Figure 2.26. By examining the previous difference measures graph, Figure 2.24, one can see that it will not be very sensitive to the choice of the multiple, since there is a broad valley between the two peaks in the graph. Were the island in the image to be only slightly larger than the cloud, this would be more difficult to do.

Comparative test

The average segmentation errors for region growing and thresholding, from the experiment described in section 2.2.2 on page 28, are presented in Figure 2.27. Compared to simple global thresholding, the region growing method fared poorly. In three of the four data sets, thresholding outperformed region grow-

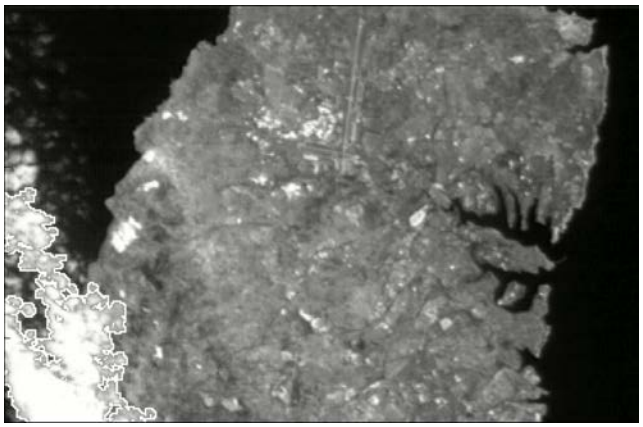


Figure 2.25: Region-growing segmentation boundary with upper limit imposed. The white line is the segmentation boundary, which is now satisfactory. Note that still only one seed point was used, so all clouds are not expected to be identified.

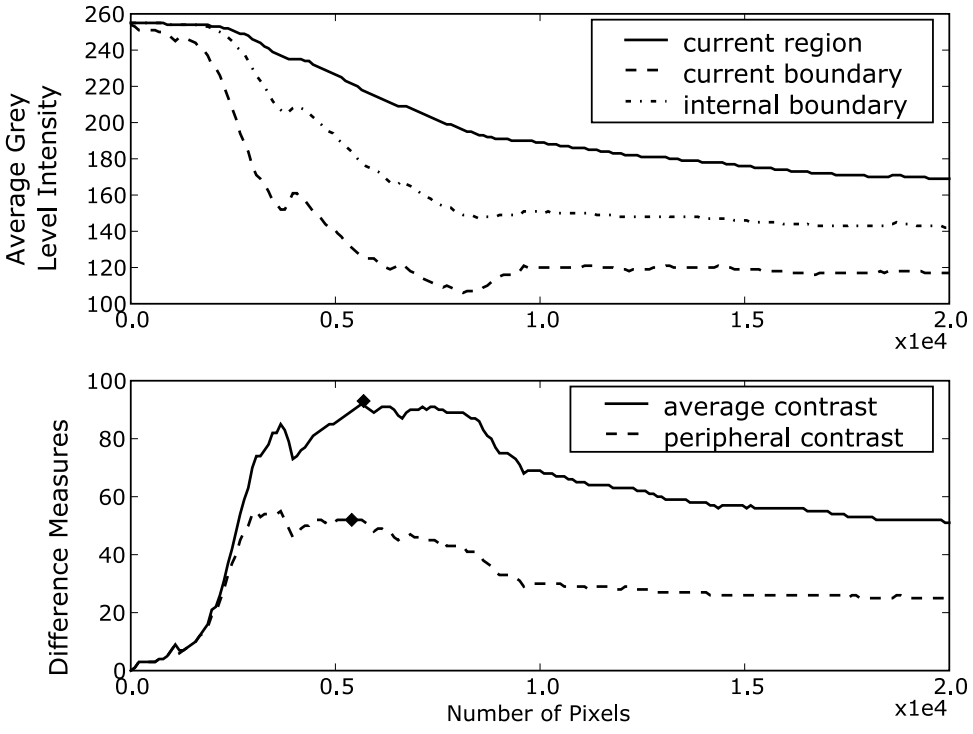
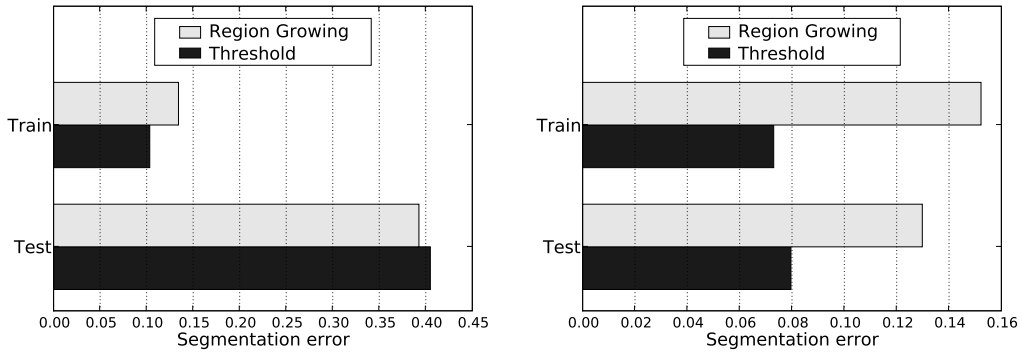


Figure 2.26: Difference measures and grey levels with upper limit imposed. The ♦ on the average contrast graph shows the global maximum, while the ♦ on the peripheral contrast graph is the segmentation point.

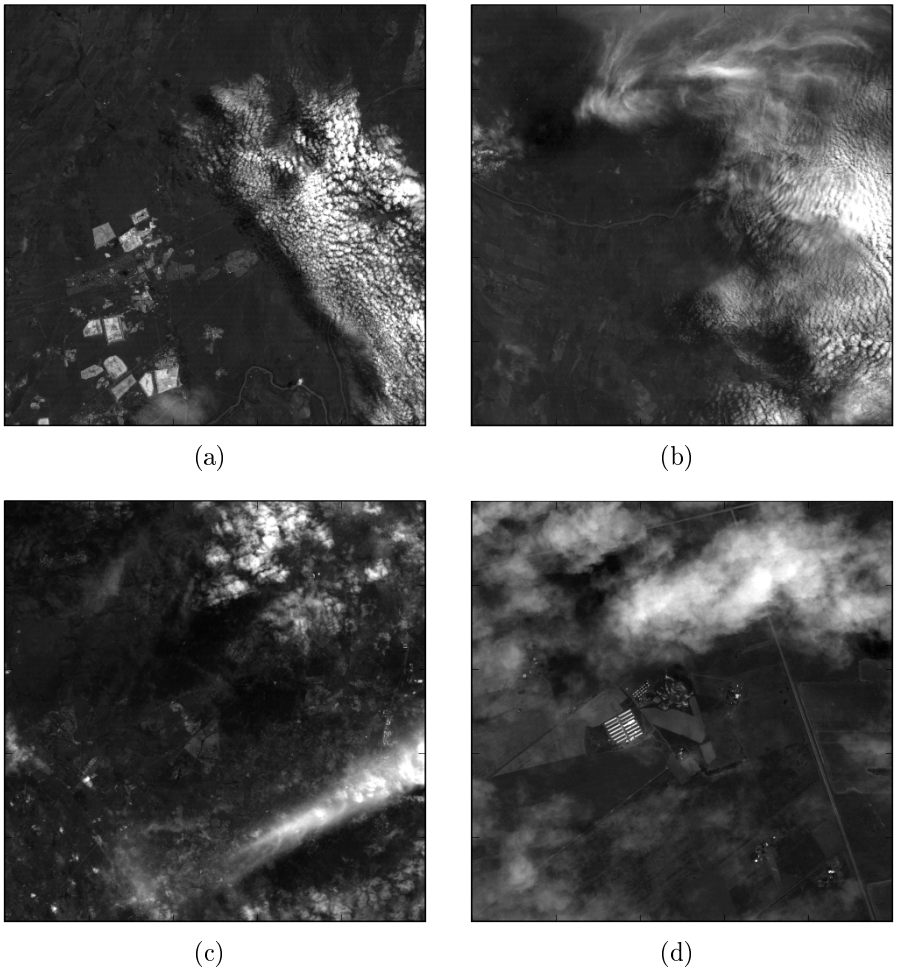


(a) Quickbird images. (b) Landsat 5 images.

Figure 2.27: Test results comparing region growing and thresholding segmentation errors.

ing by an ample margin. In the fourth data set (the Quickbird train set, from Figure 2.27(a)) both methods fared poorly, with the region growing method not able to improve significantly when segmenting difficult, thin clouds.

To gain understanding into why region growing performs worse than thresholding, one should look at individual examples. Figure 2.28 shows four example input images. In Figure 2.29, difference images are used to compare the cloudmasks generated from the input images using region growing and thresholding. In these images black and light grey indicate areas where the methods gave the same results (for background and cloud respectively). White indicates areas that the thresholding method identified as cloudy but the region growing method identified as clear. Conversely, dark grey indicates areas that the region growing method identified as cloudy but the thresholding method identified as clear. Because seed-points for region growing are derived from the threshold cloud masks, both methods struggle with some of the same areas: the bright patches of farmland in Figures 2.28(a) and (d) are erroneously labelled as cloudy in both cases (Figures 2.29(a) and (d)). The large white areas in Figures 2.29(a) to (c) clearly correspond to cloudy areas in the input images that the region growing method failed to identify. In these cases the stopping condition based on peripheral contrast was not robust enough. The contrast between the thick, bright central area of the clouds and the thin rest of the clouds was greater than the contrast between the clouds and the background, resulting in early stopping and a large segmentation errors. In Figure 2.29(a) such contrast differences are exacerbated by clouds being only 'loosely' connected: they are connected in threshold cloud mask resulting in a single seed-point, but there are dips in intensity between small adjacent clouds. The resulting intensity variation as a function of region size differs considerably from that of a smooth Gaussian blob like the one from Figure 2.6 on page



(a) (b)

(c) (d)

Figure 2.28: Sample input images for segmentation. Images (a) to (c) are Landsat 5 scenes while (d) is a Quickbird scene. In all cases the blue channel is depicted.

23. In Figure 2.29(d) the region growing method fared better by expanding beyond the threshold mask. However, one can see that the shape of the clouds' boundary are being influenced by the roads and fields beneath them, which is undesirable.

2.3.3 Cloud dispersion

The dispersion experiment described in section 2.2.3 was carried out on 31 cloud masks. Of these 17 were artificially generated to represent various possible groupings of concentrated cloud cover that might occur. The best way to analyse the results is to judge visually the classifications made. Figure 2.31 shows the images that passed the unconditional acceptance test because of cloud paucity while Figure 2.30 shows the images that were unconditionally

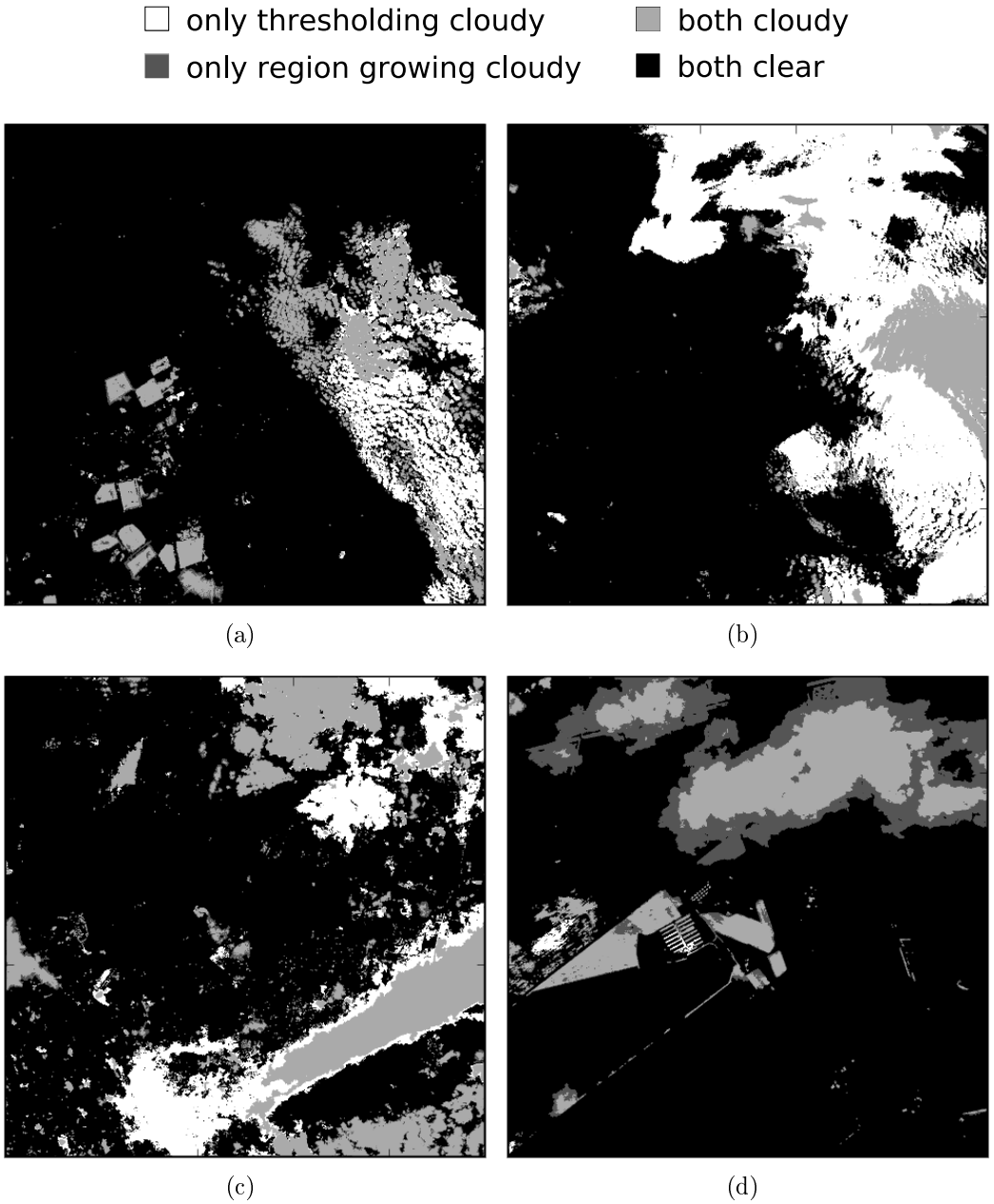


Figure 2.29: Difference images that compare region growing and thresholding cloud masks.

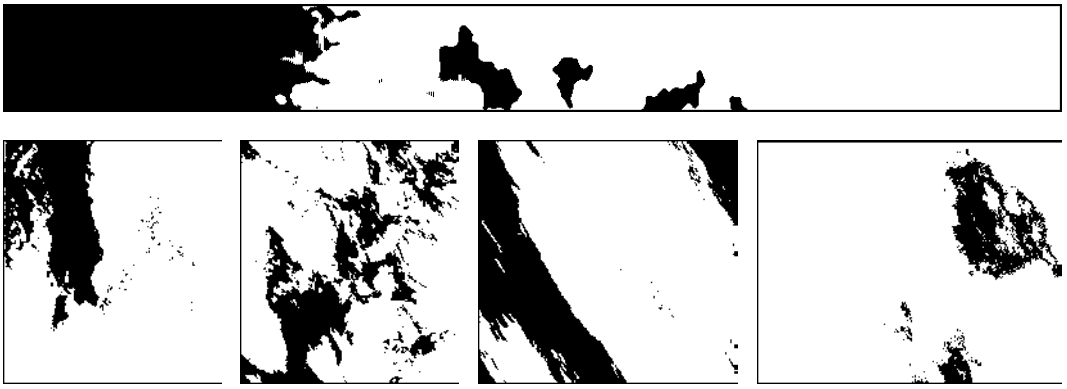


Figure 2.30: Images unconditionally rejected because of cloud abundance.

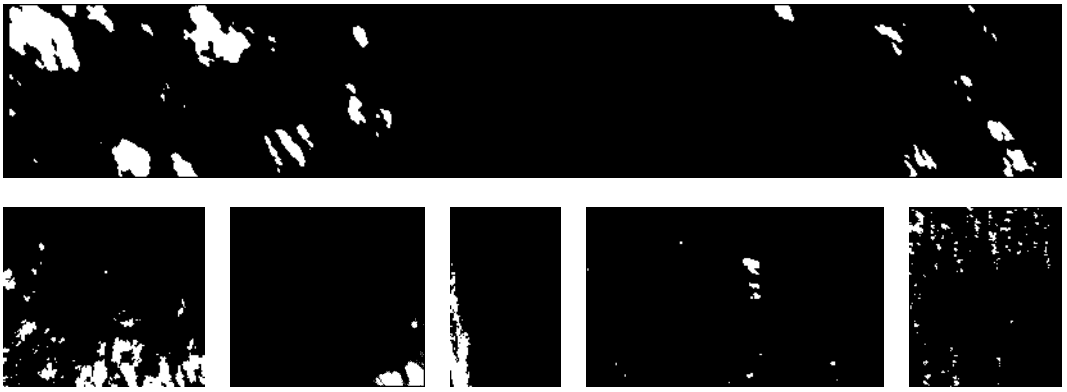


Figure 2.31: Images unconditionally accepted because of cloud paucity.

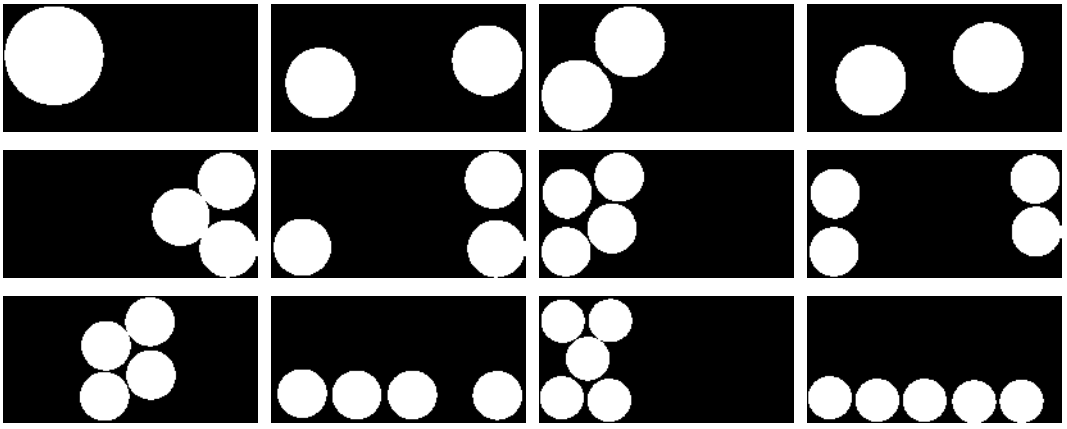


Figure 2.32: Images accepted after dispersion analysis.

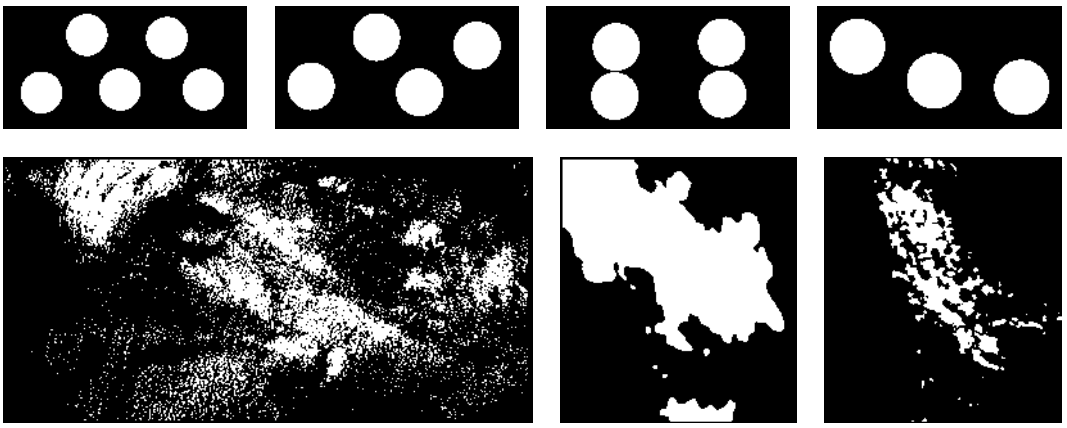


Figure 2.33: Images rejected after dispersion analysis.

rejected because of an abundance of cloud. The rest of the images were passed on to the dispersion analysis step: Figure 2.33 shows the images that failed the dispersion requirement, while Figure 2.32 shows the those images that were accepted.

The results appear meaningful: the clouds in Figure 2.33 are indeed dispersed throughout the image in such a way that would likely occlude a target, while those from figure 2.32 are more clumped together so that it is possible a target could be visible. However, the precise choice of thresholds, especially for the unconditional acceptance and rejection, is contentious. It is discussed further in the conclusions, section 2.4.3.

2.3.4 Down-sampling

The results of the down-sampling experiment described in section 2.2.4 on page 35 are presented here. The graphs in Figure 2.34 show the decrease in cover estimation as a result of down-sampling. The horizontal axis represents relative image size, thus, the point at the right of the graph corresponds to the original image size. As one moves left on the horizontal axis the image size decreases. Figure 2.35 is a different view of the same data: here the percentage difference between the true cloud cover estimate and the estimate after down-sampling is depicted.

Image 2.15(d) has the worst performance at 6% difference between the original, full resolution cloud estimate image and the down-sampled version in Figure 2.35(d). However, this is because the very low resolution of the starting image means that the 2000 times smaller version is only 6×7 pixels. This is clearly unrealistic. If one considers the next to smallest version in 2.35(d) the difference is about 1% for both `NEAREST` and `ANTIALIAS`, which is acceptable and expected, given the solidity of the cloud cover.

Image 2.15(b) has an insignificant error percentage even for a 2000 times smaller down-sampled image. This is because the high resolution original has

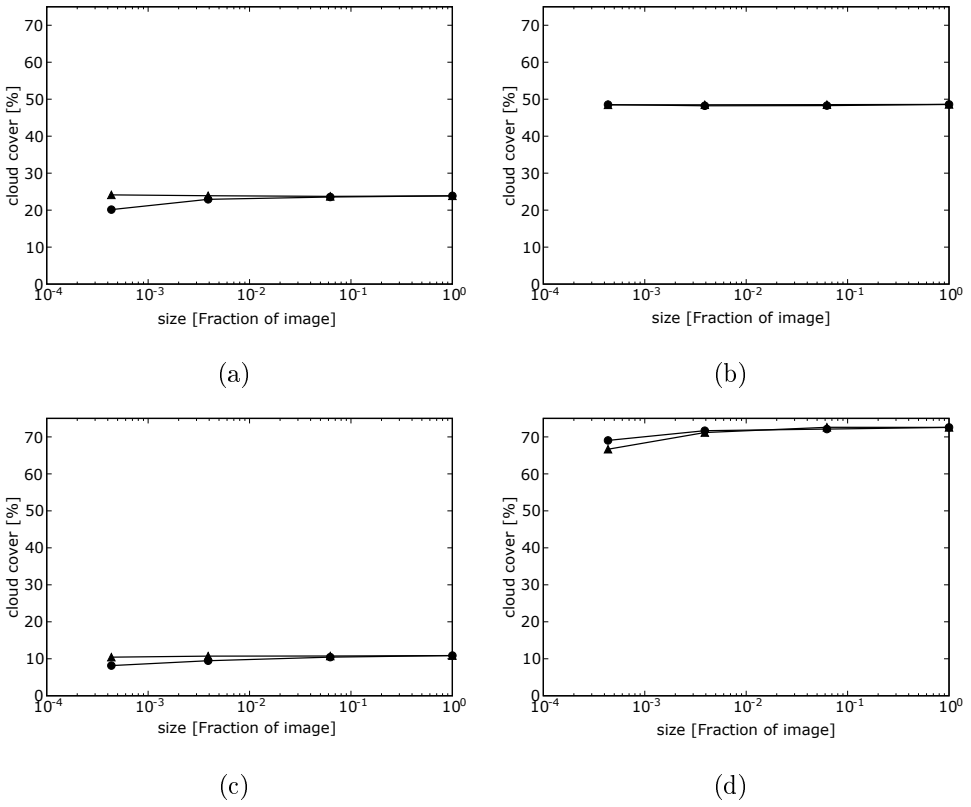


Figure 2.34: The effect of down-sampling on the cloud cover estimate. The graphs correspond to the images in Figure 2.15 on page 36, for example (a) is the graph for the image from Figure 2.15(a). The • graphs are for the `ANTIALIAS` filter while the ▲ indicates the `NEAREST` filter.

clearly defined big groupings of solid cloud.

Both images 2.15(a) and (c) have noticeably worse performance with the `ANTIALIAS` resampling filter than the `NEAREST` one. This is caused by a combination of small clouds and open ground being averaged together into a single pixel, lowering the intensity of the pixel to below the cloud threshold. The averaging together results in an underestimate of the cloud cover which agrees with previous recommendations that satellites with large GSI may lead underestimating cloud cover when detecting small clouds [48, 113]. When the `NEAREST` filter is used instead, the pixel intensity levels are not affected and statistically both clouds and open ground have a similar chance of being nearest to the output pixel. Thus, the amount of clouds and open ground in the final estimate should remain more or less the same as in the high resolution original.

In Figure 2.36 the down-sampled images and resulting cloud masks illustrate the difference. In the image in Figure 2.36(a) the intensity levels are reduced and the image appears smoother. In the resulting cloud mask the areas of high cloud concentration in the original are detected as solid cloud,

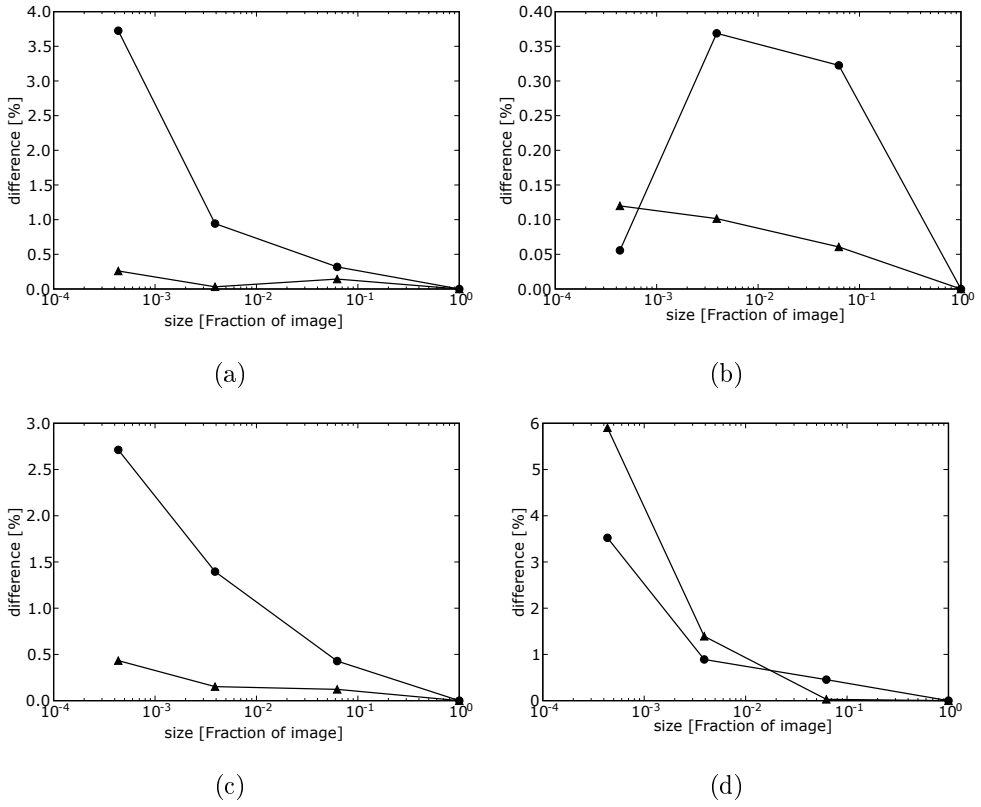


Figure 2.35: The difference between full resolution and down-sampled cloud cover estimates. Note the different scales. The graphs correspond to the images in Figure 2.15 on page 36, for example (a) is the graph for the image from Figure 2.15(a). The • graphs are for the `ANTIALIAS` filter while the ▲ indicates the `NEAREST` filter.

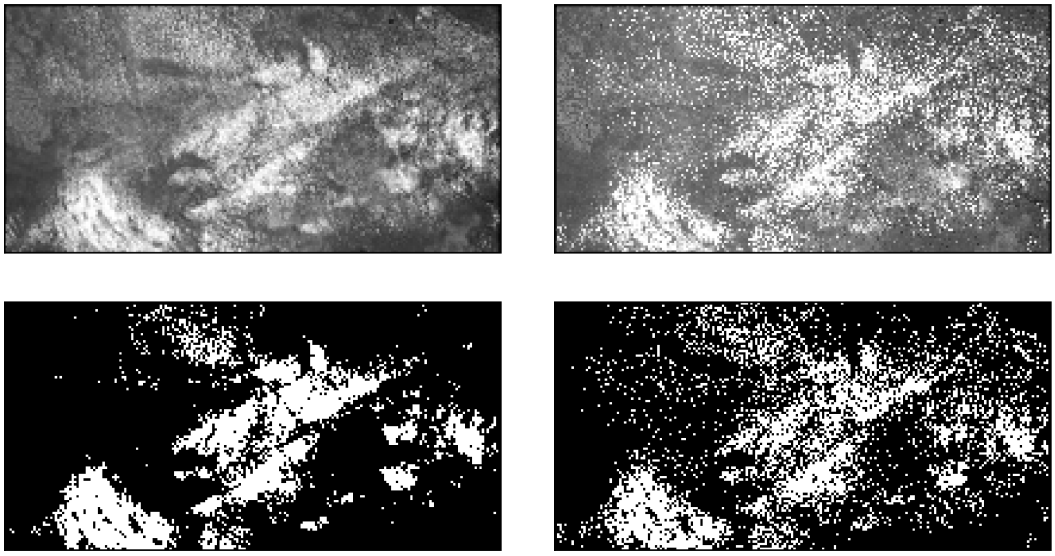
while areas with dispersed cloud in the original are detected as cloud free. In 2.36(b) the aliasing can be clearly seen in the roughness of the image. However, this is desirable since the high intensity of the cloudy pixels can still be seen dispersed throughout the image, as confirmed by the cloud mask.

2.4 Conclusion

Equations (2.1.5) and (2.1.7) should be used to normalise image brightness and eliminate the possibility of glint, if the necessary information on satellite and solar positions is available.

2.4.1 Dimension reducing transforms

HDA was shown to be a competitive image transform when the goal is to detect both thin and thick clouds in images with visual to NIR bands by applying a single threshold. It consistently gave the best average segmentation errors



(a) ANTIALIAS (b) NEAREST

Figure 2.36: Images and masks demonstrating the difference between NEAREST and ANTIALIAS.

across surface and cloud types present in a set of high resolution images, when compared to existing transforms from remote sensing literature. Although the data set used is a respectable size, limited access to images restricted the variety of surface and cloud types that could be evaluated.

Nevertheless, the flexibility of HDA allows it to weigh the available channels in an optimal way for a specific combination of surface and clouds types, based on the training data. Thus, although the data set resulted in better segmentation results on ρ_{blue}-images than on ρ_{red} ones, HDA should still give superior performance even if other surface types reversed this situation. It was also shown to be most suitable when using averaged weights across a variety of images. These two properties make it suitable for a use in a fast, global cloud detection system.

A paper based on the application of HDA to dimension reduction for cloud detection has been accepted for publication in an international journal [71].

2.4.2 Region growing

Based on the test results the region growing algorithm cannot be recommended over simple global thresholding for cloud detection. In the context of [54], where high contrast cloud boundaries are compressed using more aggressive JPEG compression, it is definitely worthwhile: the algorithm segments at the boundary with a maximum peripheral contrast, and they wanted to identify a high contrast boundary for aggressive compression.

However, when faced with a combination of thin and thick clouds where

the desired cloud boundaries are not necessarily high contrast, its performance compared to thresholding does not warrant its increased complexity.

Although not analysed as such by the authors of [54], the same arguments that applied to the contextual classifier described in section 2.1.5 can be applied to the region-growing algorithm. The continuity prerequisite inherent in region growing cleans up the segmentation boundaries in a similar manner to the contextual information from [59]. The same counter argument also applies: one can expect no improvement in misclassified areas (bright desert or snow) over normal thresholding. Therefore all the increased complexity can hope to achieve, is a marginal improvement in the precise detection of cloud boundaries.

Furthermore, if an image has to be down-sampled as suggested in section 2.2.4 and nearest neighbour down-sampling is used as suggested in section 2.4.4, then it is likely that the aliasing introduced will also negatively affect the region-growing algorithm since intensity surfaces vary less smoothly in the presence of aliasing. The aliasing introduced can also lead to many single-pixel, unconnected areas, as seen in Figure 2.36(b) on page 56. The resulting excessive number of seed points and unconnectedness of the areas will make the region-growing algorithm slow and further reduce its performance compared to thresholding.

2.4.3 Cloud dispersion

The dispersion measure developed in section 2.2.3 was tested in an example application and results presented in section 2.3.3. While the experiment proves the usefulness of the measure, hard decision boundaries based on thresholds are undesirable. The images on board the satellite must be ordered in terms of image quality, not classified into acceptable or unacceptable classes.

Rather than thresholding based on total cloud cover, $s(1)$, and dispersion, $s(2^{n-2})$, it might be better to combine these features in some statistical model to map possible combinations to an output score. To determine if dispersion is a worthwhile measure was one of the goals of the subjective quality assessment experiment discussed in Chapter 5. As will be discussed there, the contribution of cloud spread to image quality was found to be very small compared to the other variables measured.

2.4.4 Down-sampling

Based on the results from the down-sampling experiment in section 2.3.4 it is clear that the down-sampling can have an effect on cloud detection, albeit a small one. Nevertheless, it was found that nearest neighbour resampling is preferable to other, more advanced averaging schemes. Nearest neighbour is also the simplest to implement, which represents another advantage.

Chapter 3

Noise estimation

3.1 Literature

3.1.1 Introduction

Noise is an unwanted variation in sensor output that interferes with our ability to extract information from the data. It is introduced into the data by the sensor and can take a variety of forms. The performance of imaging sensors can be affected by various environmental factors: light levels and sensor temperature are major factors that affect the amount of noise present in an image taken with a CCD sensor [47]. Sensor noise determines the accuracy with which absorption features can be distinguished in the spectra and objects identified on the ground [28]. Additionally, multispectral ratios, like the normalised difference vegetation index (NDVI) used to indicate vegetation, are particularly susceptible to image noise since any noise is amplified by the ratio calculation [94, p. 185].

The simplest noise model is an *additive, signal independent* component at each pixel:

$$g(x, y) = f(x, y) + n(x, y), \tag{3.1.1}$$

where $f(x, y)$ is the input image, $n(x, y)$ is the noise term and $g(x, y)$ is the output. The function $n(x, y)$ can be tailored to describe many common types of noise. It is frequently a reasonable assumption that the noise has zero mean over a large area and is manifested as positive or negative fluctuations about $f(x, y)$. In references [28] and [11] the assumptions of a stationary, additive signal, not correlated with intensity nor autocorrelated were applied to remote sensing data. Because of its mathematical tractability in both the spatial and frequency domains, $n(x, y)$ is often modelled as Gaussian, i.e., at

each location (x, y) the noise fluctuation has a zero-mean Gaussian probability density function (PDF):

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z)^2/2\sigma^2}, \tag{3.1.2}$$

where z is the grey level and σ is its standard deviation. Sensor noise in remote sensing applications has been approximated by equation (3.1.2) in [25], [42] and [94, p. 165].

To quantify global additive noise levels in a meaningful, relative way various signal to noise ratios (SNRs) exist. The most common one is variance SNR:

$$\mathrm{SNR}_{\mathrm{var}} = \frac{\sigma_s^2}{\sigma_n^2}, \tag{3.1.3}$$

where σ_s^2 is the variance of the uncorrupted signal and σ_n^2 is the noise variance. However, this is not the only descriptor; in [28, 42] the mean signal intensity, $\overline{z}$, and noise standard deviation are used:

$$\mathrm{SNR}_{\mathrm{avg}} = \frac{\overline{z}}{\sigma_n}. \tag{3.1.4}$$

Also common in image quality assessment is the peak signal to noise ratio [29]:

$$\mathrm{PSNR} = \frac{Z^2 LM}{\sigma_n^2}, \tag{3.1.5}$$

where Z is the peak signal value (255 for 8-bit images) and L and M are the image dimensions. Given that the problem at hand is one of blind estimation, where access to the original uncorrupted image is not available, PSNR is an attractive measure. Fortunately it was found that PSNR is an excellent measure of quality for white noise distortion in images [97].

Other types of noise also occur in remote sensing. While additive noise is more common in AVIRIS (Airborne visible/infrared imaging spectrometer) and Landsat images, *signal dependant, multiplicative* noise is more frequently encountered in synthetic aperture radar (SAR) images [42]. Periodic noise is also common in high resolution pushbroom or whiskbroom scanners. This is typically manifested as *striping*, caused by differences in calibration and response of each of the detectors. Electronic interference can also cause periodic noise, visible in many Sunsat images. Isolated, local random noise or dropped scanlines can be caused by data loss during transmission.

Although periodic noise is more visible than global random noise, it is generally easier to correct. Striping can be comfortably rectified by detector matching [94, p. 301]. Other types of periodic noise can be characterised by Fourier analysis and removed using lowpass[1], bandpass or notch filters [94,

[1]Lowpass filters were successfully used to remove periodic noise from Sunsat images. Nevertheless, it is not the correct approach since these filters block all high frequency content instead of only the periodic noise's frequency.

pp. 259–264, 302–323], [47, pp. 227, 246–248]. Because of the many different types of periodic noise and its variation between different sensors, these methods are usually *ad hoc*. Furthermore, robust SNR measures for striping or local noise have not been developed yet [94, p. 135].

Because of the various reasons outlined in the previous paragraphs, it was decided to focus on estimation of global, additive Gaussian noise. The blind noise variance estimation problem is important in many branches of computer vision, image processing and remote sensing and can be stated as follows:

> Estimate σ_n^2, the noise variance, from the noisy image $g(x, y)$ without having access to *a priori* information about the original image $f(x, y)$.

In the remainder of this section, 3.1, existing noise estimation literature is presented: section 3.1.2 considers the optimal noise estimation procedure, while sections 3.1.3, 3.1.4 and 3.1.5 present practical alternatives. Existing comparative literature is discussed in section 3.1.6.

The selection, implementation and testing of the algorithms is discussed in section 3.2, while the results of the experiments are presented in section 3.3. Finally, various conclusions are drawn in section 3.4.

3.1.2 The Semivariogram: Optimal manual noise estimation

The most common method of estimating noise in images is to locate an area in the image with low variance manually, compute the variance and assume that noise is the main contributor to the variance [47, pp. 227–230]. However, the recommended method for estimating image noise in remote sensing relies on the semivariogram [94, pp. 165–166] [22] [11], a more advanced measure first introduced in [27] and applied to noise estimation in [28], where its application was called the geostatistical method.

The main advantage of the semivariogram over the popular basic method described at the start of the previous paragraph, called the 'image' method in [28], is that its noise estimates are independent of interpixel variability. Proponents of the geostatistical method claim that any method based on calculating variance of an array will estimate a noise variance that inherently contains image interpixel variance as well.

To calculate the semivariogram one must first take a transect of the image and extract the signal intensity $z(x)$ at pixels $x = 1, 2, \ldots, n$ along the transect. The relation between a pair of pixels distance h apart (called the lag distance) can be given by the variance of the differences of all such pairs. Within the transect there will be m pixels separated by the same lag. Their average is

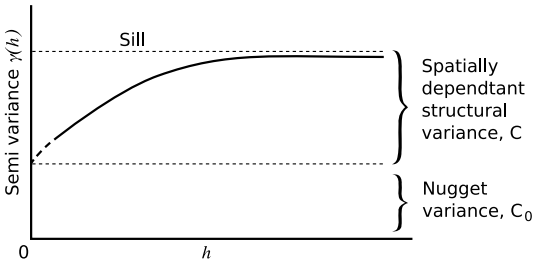


Figure 3.1: The general form of the semivariogram.

given by

$$\overline{S}^2(h) = \frac{1}{2n}\sum_{i=1}^{n}[z(x_i) - z(x_i + h)]^2.$$

$\overline{S}^2(h)$ is an unbiased estimate of the semivariance $\gamma(h)$, which is a useful measure of the difference between spatially separate pixels. The larger $\gamma(h)$ the more dissimilar pixels h apart are. A typical semivariogram is presented in Figure 3.1. The important features of the semivariogram are:

the sill: The asymptotic upper bound of $\gamma(h)$,

C_0**, the nugget variance:** The limit of $\gamma(h)$ as h approaches 0 and

the spatially dependent structural variance: The sill minus the nugget variance.

From the definition $\gamma(h) = 0$ when $h = 0$ but in practice there is an offset, C_0, caused by intrapixel variability and spatially independent noise. The statistical proof of

$$\lim_{h \to 0} \gamma[z(h)] = \sigma_n^2 \tag{3.1.6}$$

is given in [28].

Although the most statistically sound method available for noise estimation, the transect $z(x)$ has to be manually selected from a homogeneous image area to minimise the effect of intrapixel variability on the estimate. Also, $\overline{S}^2(h)$ is a discrete function with $h = 1, 2, \ldots, n$, so the limit from equation (3.1.6) must be estimated by fitting a curve on the discrete data and extrapolating to 0, which re-introduces some spatial dependence as well as dependence on the function used for extrapolation [11].

It is impossible to select a transect manually in an automatic IQA algorithm. Thus alternative methods must be investigated even if they are theoretically suboptimal.

3.1.3 Methods based on a standard deviation histogram

Gao developed an automatic method for estimating additive noise in remote sensing images [42]. The algorithm is based on the idea of detecting a peak in a standard deviation histogram and is relatively simple and intuitive.

The method consists of the following steps:

1. The image is divided into many small blocks or cells each with the dimensions $d \times d$ pixels. Cell sizes considered include $d = 4, 5, \ldots, 8$. The cells are located within the image with index $k = 1, 2, \ldots, K$, where K depends on the size of d. For each cell the local mean, $\overline{g}_k$, of the signal is calculated:

$$\overline{g}_k = \frac{1}{d \times d} \sum_{i=1}^{i=d} \sum_{j=1}^{j=d} g_k(i,j), \tag{3.1.7}$$

   where $g_k(i,j)$ is the grey level of the pixel at location (i,j) in the cell. The mean signal over the entire image, $\overline{g(x,y)}$ is computed. The unbiased estimate of local standard deviation for the kth cell, $\sigma(k)$, is also calculated:

$$\sigma_g(k) = \sqrt{\frac{1}{d \times d - 1} \sum_{i=1}^{i=d} \sum_{j=1}^{j=d} (g_k(i,j) - \overline{g}_k)^2}. \tag{3.1.8}$$

   It is asserted that homogeneous blocks with small $\sigma(k)$ provide information on the noise in the image, while inhomogeneous blocks having larger $\sigma_g(k)$ typically contain image edges or textures. The minimum and maximum of $\sigma_g(k)$ for all blocks are also computed.

2. A histogram of $\sigma_g(k)$ values is constructed. Between the maximum and minimum of $\sigma_g(k)$ a number of bins are set up. The $\sigma_g(k)$ values of all blocks are then grouped into these bins and the number of $\sigma_g(k)$ values in each bin is counted. The bin with the largest number of blocks (the peak of the histogram) is associated with the mean noise σ_n for the image.

3. The SNR is calculated according to equation (3.1.4).

Gao demonstrates the usefulness of the algorithm by testing it on a simulated constant image with added Gaussian noise, a simulated checkboard pattern image with added Gaussian noise and several sets of AVIRIS data GERIS data. AVIRIS has a GSI of 20m and 10 bands covering 0.4–2.5μm. The Geophysical and Environmental Research Imaging Spectrometer (GERIS) has 64 channels between 0.43μm and 2.5μm. To minimise the effect of image features on the noise estimate the use of 4×4 pixel blocks was recommended. The choice of bin width was found to be problematic and affected the precision of the noise estimate. Because the maximum $\sigma_g(k)$ can vary greatly, using a

fixed number of bins sometimes resulted in a relatively large bin width, which negatively impacted the noise estimate. To curb this influence, the bins were set in a range between the minimum $\sigma_g(k)$ of all blocks and 1.2 times the average $\sigma_g(k)$ of all blocks. It was recommended that 150 bins be used for images larger than 500×500 pixels.

It was concluded that the method is feasible. Furthermore, it was claimed to be superior to the 'image' and 'geostatistical' methods since it incorporates data from the entire image instead of selected areas or transects.

In [25] a very similar method is presented, again in a remote sensing context, although no reference is made to Gao's work. In this the median of a histogram of $\sigma_g(k)$ values is used. Additionally, an edge mask identifying edges in the image is constructed using Laplacian and gradient operators. Blocks that contain edge pixels are omitted from the histogram in an attempt to decrease the sensitivity of the method to image variance. The method was found to work well on simulated images containing sharp edges, but slightly overestimated the noise in Landsat images due to residual image variance. Different block sizes ($d = 4, \ldots, 9$ pixels) were also investigated; it was found that a trade-off exists between block size and ability to estimate variance. While smaller block sizes, as advocated by Gao, decrease the sensitivity to image variance they also decrease the ability to identify high levels of noise. This is because estimating variance from a small area that is not sufficiently large results in an underestimate.

3.1.4 A method based on image pyramids and order statistics

Meer *et al.* developed a method that creates a dichotomy between signal and noise by analysing the *noise statistics* at different levels of an image pyramid [76]. They cited previous examples of noise estimation algorithms that attempt to separate image and noise based on identification of *image features*. These methods struggled to identify image features in the presence of high noise levels.

The noise variance estimator

Since the method is based on an image pyramid, it is required that the image size $N \times N$ be a power of two: $N = 2^n$. The image is again divided into square cells of size $d_l \times d_l$, where $d_l = 2^l$, $l = 1, 2, \ldots, n$. The tessellation of the image into cells of size $2^l \times 2^l$ is referred to as level l of the image pyramid. The index of cells at level l is $k_l = 1, 2, \ldots, K_l$, where $K_l = 4^{n-l}$. The variance $\sigma_g^2(k_l)$ at the kth cell is computed according to

$$\sigma_g^2(k_l) = \frac{1}{4^l - 1} \sum_{i=1}^{i=2^l} \sum_{j=1}^{j=2^l} (g_{k_l}(i,j) - \overline{g}_{k_l})^2. \tag{3.1.9}$$

Since the noise is additive, from equation (3.1.1) follows:

$$\sigma_g^2(k_l) = \sigma_f^2(k_l) + \sigma_n^2(k_l). \tag{3.1.10}$$

Order statistics refer to operations on the list of variance estimates $\sigma_g^2(k_l)$ for all K_l of the cells at level l, ordered according to the variance size. The estimator for noise variance at level l is the smallest value of the sample variance $\sigma_g^2(k_l)$:

$$q(l,1) = \min_{k_l} \sigma_g^2(k_l), \tag{3.1.11}$$

where the second parameter, set to 1, emphasises that this is a first order statistic (as opposed to higher order statistics introduced later). Equation (3.1.11) is a good estimator since $\sigma_n^2(k_l)$ is a quasi constant function of σ_n^2 and the contribution of $\sigma_f^2(k_l)$ is minimised by selecting the cell with the smallest variance, i.e., the cell in the original image that was the most homogeneous.

The usefulness of the image pyramid approach becomes apparent when one considers the properties of $q(l,1)$: it increases with increasing l. The monotonic behaviour of the first order statistic is a result of the fact that sample variance is a consistent estimator: its spread (confidence interval) decreases with increasing degrees of freedom of the sample. In a uniform image, $f(x,y)$, the larger the cell size the closer the first order statistic is to the true noise variance σ_n^2. For a realistic image, more image features $\sigma_f^2(k_l)$ are also incorporated into $q(l+1,1)$ further increasing it relative to $q(l,1)$.

However, especially at small tessellations where the number of cells K_l is large, there is a possibility that $q(l,1)$ is an outlier. In this case the noise variance is severely underestimated. To guard against outliers, the first four order statistics $q(l,i), i = 1,2,3,4$ (the four smallest values for sample variance) are used. The higher the order statistic of a sample the lower the probability that it is an outlier. A slippage test is employed to determine if any of the first three values are outliers. Any outliers are discarded and the remaining values averaged to give the final variance estimate for level l, $v(l)$.

The slippage tests compare ratios of differences of order statistics to threshold values. The ratios are:

$$r_0(l,1) = \frac{q(l,2) - q(l,1)}{q(l,4) - q(l,1)} \qquad r_0(l,2) = \frac{q(l,3) - q(l,2)}{q(l,4) - q(l,2)} \tag{3.1.12}$$

$$r_0(l,3) = \frac{q(l,4) - q(l,3)}{q(l,4) - q(l,2)} = 1 - r_0(l,2).$$

The threshold tests are:

$$\text{if} \quad r_0(l,1) \leq 0.5 \quad \text{then} \quad v(l) = \frac{1}{4}\sum_{i=1}^{4} q(l,i) \tag{3.1.13}$$

else

$$\text{if} \quad r_0(l,2) \leq 0.7 \quad \text{then} \quad v(l) = \frac{1}{3}\sum_{i=2}^{4} q(l,i) \tag{3.1.14}$$

Table 3.1: The employed bounds for the ratio $v(l-1)/v(l)$.

	3	4	5	6	7	8	9
$\beta(l)$	0.2	0.6	0.8	0.9	0.95	0.975	0.9875

else

$$\text{if} \quad r_0(l,3) \leq 0.7 \quad \text{then} \quad v(l) = \frac{1}{2}\sum_{i=3}^{4} q(l,i) \tag{3.1.15}$$

else

$$v(l) = q(l,4). \tag{3.1.16}$$

For very large cell sizes $K_l = 4\,(l = n-1)$ and $K_l = 1\,(l = n)$ the slippage tests cannot be applied. $v(l)$ is instead taken as the average of the four order statistics when $K_l = 4$ or equal to the global sample variance $\sigma_g^2(1)$ when $K_l = 1$.

By tessellating, computing order statistics and applying slippage tests at each level l of the image pyramid, as described in the preceding paragraphs, a sequence of variance estimates $v(l)$ is obtained for $l = 1, 2, \ldots, n$. To achieve the dichotomy between signal and noise and to obtain a final value for σ_n^2, the ratios of consecutive variance estimates, $v(l-1)/v(l)$, are used. Based on experimental tests on a uniform image with added Gaussian noise, as well as theoretical derivations (also assuming the noise to be Gaussian), an expression for the lower bound for this ratio is obtained in [76]. The lower bounds of $v(l-1)/v(l)$ are generated by the expression:

$$\beta(l) = 1 - 0.1 \times 2^{-l+6} \qquad l = 3, 4, \ldots, n. \tag{3.1.17}$$

Values for $\beta(l)$ are given in Table 3.1.

Estimating noise variance: The dichotomy between signal and noise

To arrive at the final noise variance estimate from the sequence of variance estimates $v(l)$, a variety of rules is employed. The first rule detects an image that is not corrupted by noise:

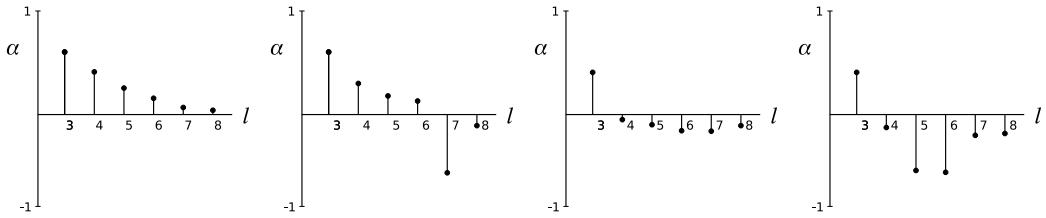
Rule 1

$$\text{if} \quad \exists l \geq 2 \quad \text{such that} \quad q(l,4) < 1 \quad \text{then} \quad \hat{\sigma}_n^2 = 0, \tag{3.1.18}$$

where $\hat{\sigma}_n^2$ is the estimated noise variance value output by the algorithm. In this rule incorrect decisions for small noise variances are avoided by comparing the fourth order statistic to a minimum variation threshold.

When noise is present the dichotomy between signal and noise is achieved by analysing a *deviation sequence*:

$$\alpha(l) = \frac{v(l-1)}{v(l)} - \beta(l) \qquad l = 2, 3, \ldots, n. \tag{3.1.19}$$



(a) Uniform image. (b) Good separation. (c) Moderate separation. (d) Poor separation.

Figure 3.2: Example deviation sequences, $\alpha(l)$, with varying levels of signal-noise separation.

where $\beta(l)$ obtained from (3.1.17). Since $v(l-1) < v(l)$ it must be true that $-1 < \alpha(l) < 1$. For fine tessellations at low l the cells used to derive $v(l)$ are quasi uniform and $\alpha(l)$ is positive since $\beta(l)$ is the lower bound for uniform regions. As the cell size increases with greater l, at a given level l_0 it becomes impossible to find a uniform region. The order statistics for sample variance start to include image variance, which causes a sudden increase in $v(l_0)$ and $-1 < \alpha(l_0) \ll 0$. Therefore, the signal-noise dichotomy can be achieved by searching for the level at which the deviation sequence $\alpha(l)$ becomes significantly negative.

To avoid false alarms for images where the signal-noise separation is poor (to be elaborated upon below) a cumulative thresholding technique is used to detect the level at which $\alpha(l)$ becomes significantly negative:

$$l_u = \min_l[\arg(\alpha(l) < 0)] \qquad l = 3, 4, \ldots, n \qquad (3.1.20)$$

$$l_0 = \min_l[\arg(\sum_{i=l_u}^{l} \alpha(i) < T)] \qquad l = 3, 4, \ldots, n \qquad (3.1.21)$$

$$\text{and} \quad \sum_{i=l_u}^{l_0+1} \alpha(i) < T \qquad l_0 = l_u, l_u + 1, \ldots, n-1. \qquad (3.1.22)$$

The threshold T is set equal to a small negative value -0.1. Thus, l_u is the first level where the deviation sequence becomes negative and l_0 is the level where a cumulative deviation sequence sum is below a threshold. The variance estimates for $l < l_0$ belong to the noise domain, while for $l > l_0$ they belong to the signal domain.

Example deviation sequences are presented in Figure 3.2, for 256×256 pixel images ($n = 8$). In the case of a uniform image with no variation the deviation sequence remains positive and all values belong to the noise domain, Figure 3.2(a). A special rule is employed in this case:

Rule 2

$$\text{if} \quad \alpha(l) > 0 \quad \forall l = 3, 4, \ldots, n \quad \text{then} \quad \hat{\sigma}_n^2 = v(n). \qquad (3.1.23)$$

In an image with good signal-noise separation, Figure 3.2(b), the transition to signal domain occurs at a high level and is sharp: $l_u = l_0$. In this case the order statistics are accurate estimates of the true noise variance and $\hat{\sigma}_n^2$ is precise. In an image with moderate signal noise separation, Figure 3.2(c), the transition occurs at intermediate levels and is not well defined ($l_0 = l_u + 1$). The sample variances have larger spreads. In an image with poor signal-noise separation, the transition occurs at low levels, 3.2(d). In this case the noise domain is severely reduced and it is impossible to get a good estimation of the noise variance. The algorithm recognises this situation by using the following two rules:

Rules 3 and 4

$$\text{if} \quad l_0 = 3 \,\text{or}\, 4 \quad \text{then} \quad \hat{\sigma}_n^2 \,\text{cannot be estimated.} \tag{3.1.24}$$

Note that for rules greater than 2 the convention is that the rule number corresponds to the deviation sequence transition level.

To characterise the separation between signal and noise further, the busyness of the signal, i.e., how much grey level variation there is in the signal domain, is analysed. If the signal domain spans multiple levels and its sequence of variance estimates increase steeply then it is a busy signal. In the rules applied for $5 \leq l_0 < n$ a busyness parameter is introduced and used during interpolation between $v(l)$ for different l.

Transition at level $l_0 = 5$. The busyness of the signal is given by the parameter:

$$\rho = \alpha(5) + \alpha(6) \qquad -2 < \rho \leq T = -0.1 \tag{3.1.25}$$

The relation from (3.1.25) holds because of the requirement enforced on l_0 by equation (3.1.22). The more negative ρ is, the more busy the original image was and the more the lower levels belonging to the noise domain are contaminated by variance from the signal domain.

The range (-2,T) of ρ is divided into four regions marked by index $i = 1, 2, 3, 4$ with ρ_{i-1} and ρ_i the bounds for the ith region. In the first region ρ is the smallest and, correspondingly, the signal is the busiest; the noise domain is strongly contaminated by image variance, reflected by the following rule:

Rule 5-1

$$\text{for} \quad \rho_0 = -2 < \rho \leq \rho_1 = -1.5 \qquad \hat{\sigma}_n^2 = v(3). \tag{3.1.26}$$

The contamination by signal is the weakest at level 3; $v(3)$ is the most reliable.

In the remaining three regions of ρ linear interpolation is used. First the interpolation variable is computed:

$$\text{for} \quad \rho_{i-1} \leq \rho \leq \rho_i \qquad \delta = \frac{\rho - \rho_i}{\rho_{i-1} - \rho_i}. \tag{3.1.27}$$

The interpolation rules are identical and result in continuous interpolation across ρ:

Rule 5-2

$$\text{for} \quad \rho_1 = -1.5 \leq \rho \leq \rho_2 = -1 \qquad \hat{\sigma}_n^2 = \delta \times v(3) + (1-\delta) \times v(4). \quad (3.1.28)$$

Rule 5-3

$$\text{for} \quad \rho_2 = -1 \leq \rho \leq \rho_3 = -0.5 \qquad \hat{\sigma}_n^2 = \delta \times v(4) + (1-\delta) \times v(5). \quad (3.1.29)$$

Rule 5-4

$$\text{for} \quad \rho_3 = -0.5 \leq \rho \leq \rho_4 = T \qquad \hat{\sigma}_n^2 = \delta \times v(5) + (1-\delta) \times v(6). \quad (3.1.30)$$

Transition at Level $l_0 = 6$ or 7. Transitions occurring at level 6 or 7 imply large cell sizes and therefore the sample variances constitute accurate estimates of noise variance. The busyness of the signal is characterised by considering $\alpha(l_0 - 1)$ and $\alpha(l_0)$. Their ranges $T < \alpha(l_0 - 1) < 1$ and $-1 < \alpha(l_0) \leq T$ are partitioned into two regions. T is still -0.1 as previously defined. Again interpolation between two levels in the noise domain is used.

If $\alpha(l_0 - 1)$ is negative (the first region), the transition from signal to noise domain is moderately-defined. The interpolation variable is $\delta = |\alpha(l_0)|$.

Rule 6-1

$$\text{for} \quad T < \alpha(l_0 - 1) \leq 0$$
$$\hat{\sigma}_n^2 = \delta \times v(l_0 - 2) + (1 - \delta) \times v(l_0 - 1). \quad (3.1.31)$$

The more negative $\alpha(l_0)$ is (the more busy the signal), the closer to $v(l_0 - 2)$ (deeper into the noise domain) is $\hat{\sigma}_n^2$.

If $\alpha(l_0 - 1)$ is positive (the second region), the transition from signal to noise domain is well-defined. Let the two regions of $\alpha(l_0)$ be bounded by ρ_{i-1} and ρ_i, $i = 1, 2$. Similar to (3.1.27), the interpolation variable is:

$$\text{for} \quad 0 \leq \alpha(l_0 - 1) < 1 \quad \text{and} \quad \rho_{i-1} \leq \alpha(l_0) \leq \rho_i \qquad \delta = \frac{\alpha(l_0) - \rho_i}{\rho_{i-1} - \rho_i}. \quad (3.1.32)$$

The interpolation rules are:

Rule 6-2

$$\text{for} \quad \rho_0 = -1 < \alpha(l_0) \leq \rho_1 = -0.5$$
$$\hat{\sigma}_n^2 = \delta \times v(l_0 - 2) + (1 - \delta) \times v(l_0 - 1). \quad (3.1.33)$$

Rule 6-3

$$\text{for} \quad \rho_1 = -0.5 < \alpha(l_0) \leq \rho_2 = T$$
$$\hat{\sigma}_n^2 = 0.5[(1+\delta) \times v(l_0 - 1) + (1-\delta) \times v(l_0)]. \quad (3.1.34)$$

The interpolation is continuous across the regions of $\alpha(l_0)$. The largest value of $\hat{\sigma}_n^2$ cannot exceed $0.5[v(l_0 - 1) + v(l_0)]$, which is still in the noise domain.

When the transitions occur at higher levels l_0 the estimate values at the transitions are used; the large cell sizes ensure these are reliable.

In summary, after the variance estimate sequence $v(l)$ has been computed, the following steps are followed:

- Compute the deviation sequence $\alpha(l)$, (3.1.19).
- Find the transition level l_0 between the signal and noise domains (3.1.21)–(3.1.22).
- Apply the correct rule to estimate $\hat{\sigma}_n^2$, (3.1.23)–(3.1.34).

A similar method applied to remote sensing

A somewhat similar method, also based on dividing the image into cells of increasing size, was recently developed in a remote sensing context [112]. It utilises the presence of multiple channels in remote sensing images by first averaging across all channels before identifying homogeneous areas in the image suitable for use in noise estimation.

However, after this step no attempt is made to separate image from noise further; unlike the image pyramid method the algorithm does not evaluate its own ability to estimate noise correctly. Instead it assumes that the most homogeneous areas contain no image variance, which is obviously not a valid assumption in all cases. Furthermore, it required interactive tuning on a scene by scene basis to give optimal results. This human intervention requirement is unacceptable for an autonomous application.

3.1.5 Dark current

If a part of the sensor is shielded from exposure, the dark current can be used to estimate noise without fear of image features influencing the estimate. In [28, 42] it is argued that it is difficult to use dark current for estimation of random global additive noise in the presence of periodic noise, since dark current pixels are not usually subjected to the same calibration processing as image pixels. Furthermore the very presence of shielded pixels is dependent on the design of the sensor; a method capable of estimating noise directly from the image is therefore generally more applicable.

3.1.6 Comparative literature survey

Because of the importance of noise estimation in image processing there have been many algorithms designed to tackle the problem. A comparative test of six methods, including Meer *et al.*'s image pyramid method from section 3.1.4, was done in reference [78]. However, Gao's remote sensing specific algorithm from section 3.1.3 post-dates this test.

The six tests used are classified into two categories:

- Those based on filtering $g(x, y)$ to suppress image structure and then derive $\hat{\sigma}_n^2$ from the filtered image. These methods are very basic. Two of the six methods tested fall into this category: simple average and median filters are tested. The median filter is used in such a manner as part of a blur estimation algorithm discussed in section 4.2.2.

- Those based on computing $\hat{\sigma}_n^2$ from the image regions in $g(x, y)$ that are initially classified as showing little image structure. The image pyramid method falls into this category.

Of the six methods evaluated, the image pyramid method was the only one that attempted to establish an image noise dichotomy so that it can warn when the algorithm is unable to give an accurate noise estimate. While Gao's method was not investigated, its predecessor [64], also belonging to the second category and developed in a remote sensing context, was.

Four synthetic and five real-life images were corrupted with varying levels of both uniform and Gaussian additive noise to generate a large set of test images. Depending on which criteria is used to evaluate the results, different conclusions could be drawn. The image pyramid method gave the best performance by far for low noise images. However, it was prone to giving large estimation errors sometimes (10% of the images). Although not mentioned by the authors, it is noteworthy that the image pyramid method showed no apparent decrease in performance with *uniform* noise even though the lower bound $\beta(l)$ (3.1.17) was developed for *Gaussian* noise statistics. The basic averaging filter method gave surprisingly good performance across a range of σ_n^2 levels while the median method performed the best at very high σ_n^2 levels. The simple filter based methods have a 10 times speed advantage over the more advanced methods. No single method could consistenly give correct estimation results in all test cases.

3.2 Experiments and implementation

3.2.1 Implementation

Selected algorithms

If the satellite performs as designed, most of the images to be analysed by the IQA system will either have no noise or low noise levels. Therefore, based on the comparative test from section 3.1.6, the image pyramid method was decided upon since it outperforms other methods in low noise situations. Since it was not developed specifically for a remote sensing context, it is evaluated on remote sensing images representative of the different types of image structures that might be encountered.

Additionally, Gao's standard deviation histogram method from section 3.1.3 is also implemented and tested for comparison. It is interesting to note that, although Gao mentions the image pyramid method in the introduction of [42], no comparative conclusions are drawn.

Embedded evaluation

Both algorithms were implemented using Python. After consideration of the comparative results, it was decided to evaluate the performance feasibility of the image pyramid method by implementing it in C on embedded SH4 hardware similar to the Sumbandilasat.

A Hico SH4 evaluation board running the μCLinux operating system was used for initial development. However, the hardware had much greater RAM limitations than the Sumbandilasat hardware, so final speed tests were done on the experimental payload development board that had the same specifications as Sumbandilasat.

Since C has less built in functionality, auxiliary functions to handle file access and memory management had to be implemented.

All embedded C code was thoroughly documented using Doxygen [3] compatible comments. This enabled convenient, html browsable as well as printable documentation to be generated using Doxygen. The printable documentation is included in Appendix C.

Details on image pyramid method's implementation

During the implementation of the image pyramid method two optimisations were made to minimise the execution speed.

Firstly, the variance calculation from equation (3.1.9) on page 63, that would normally take two passes of the data, was done in a single pass. Two passes are needed as $\overline{g}_{k_l}$ must first be computed by considering all the pixels in the cell before (3.1.9) is evaluated, again considering all the data in the cell. Through algebraic manipulation (3.1.9) can be written in a form suitable for

single pass calculation:

$$\begin{aligned}(4^l-1)\times\sigma_g^2(k_l) &= \sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} g_{k_l}^2(i,j) - \sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} 2g_{k_l}(i,j)\times\overline{g}_{k_l} + \sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l}\overline{g}_{k_l}^2 \\ &= \sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} g_{k_l}^2(i,j) - 2\overline{g}_{k_l}\sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} g_{k_l}(i,j) + (4^l)\overline{g}_{k_l}^2 \\ &= \sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} g_{k_l}^2(i,j) - 2\overline{g}_{k_l}\times(4^l)\overline{g}_{k_l} + (4^l)\overline{g}_{k_l}^2 \\ \sigma_g^2(k_l) &= \frac{1}{4^l-1}\sum_{i=1}^{i=2^l}\sum_{j=1}^{j=2^l} g_{k_l}^2(i,j) - \frac{4^l}{4^l-1}\overline{g}_{k_l}^2. \end{aligned} \tag{3.2.1}$$

(3.2.1) can be evaluated in a single pass through the data: at each point, (i,j), values are added to the sum of squares and to the average. After the last pixel has been considered the two terms can be subtracted and $\sigma_g^2(k_l)$ computed.

For a big image the sum of squares term can become large enough to cause overflow in the embedded implementation if the C variable used to store it is too small. However, the 64-bit `unsigned long long int` data type is capable of storing the large numbers without overflow. Consider:

$$\text{(maximum integer size)} = \text{(maximum intensity)}^2 \times \text{(image size)}.$$

For an image with a bit-depth of sixteen bits[2] and an image size of 8000×80000 where every pixel is at the maximum intensity, the maximum integer number that must be representable is:

$$(2^{16})^2 \times (8000\times80000) = 2.75\times10^{18} \approx 2^{61} \ll 2^{64}.$$

The second optimisation is typical in image pyramid algorithms: the results obtained at the previous level of the pyramid are stored so they can be used in the next level. In this case the sum of squares (left hand term in term (3.2.1)), as well as the average (right hand term in term (3.2.1)) for each cell, is stored. When computing $\sigma_g^2(k_l)$ for level l these intermediate results from $l-1$ can be used to determine the sum of squares and average, drastically reducing computation speed. In the embedded implementation care was taken to free the memory of the intermediate results from level $l-2$ since these are redundant. This optimisation increased the performance speed by a factor of three for the test image from Figure 3.3.

3.2.2 Experiment

From the structure of Gao's standard deviation histogram algorithm described in section 3.1.3, it was suspected that it might struggle with noise estimation

[2]The Sumbandilasat sensor is capable of capturing at this bit depth.

in noiseless images. Since the histogram peak is used, it was surmised that it would be especially sensitive to images where large parts of the image contain fine texture.

As an initial test image, Figure 3.3(a) was used. The image has large areas of high variation caused by waves on the surface of the water. Scattergrams, like the one in 3.3(b), are used extensively by Gao in [42] to analyse image structure. Each point on the scattergram represents a cell in the tessellated image. The position of the cell in the scattergram is determined by its local mean, $\overline{g}_k$, and local standard deviation. The dark concentrations in the lower left hand side indicate that there are many blocks with low variance and low intensity, corresponding to the soil region in the image. The stripe running from left to right along the bottom of the image results from the the choppy dam surface. The effect of noise on a scattergram can be observed in Figure 3.3(c): although the entire scattergram is shifted towards the right, the movement is most visible at the left end of the graph. While the standard deviation histogram considers all the data points, the image pyramid method is only concerned with cells at the left end of the graph.

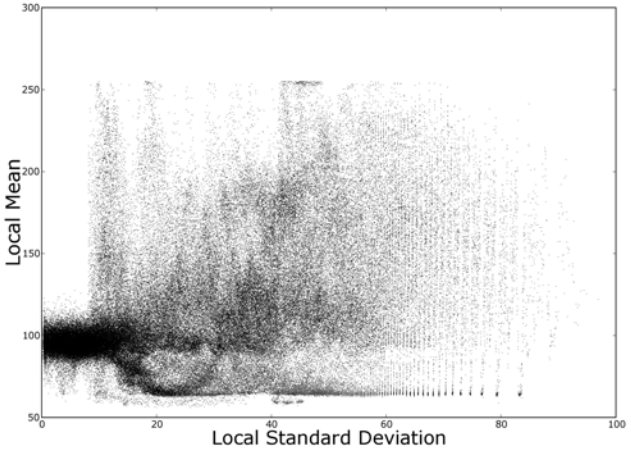
Without considering the scattergram, it was thought that this large area of high variance might be sufficient to prove the superiority of the image pyramid method. However, both methods proved to be equally capable of identifying noise, or the lack thereof, in Figure 3.3(a). As can be seen in the scattergram, the concentration of clear cells proved higher than that of cells in the choppy dam area. This results in a very low standard deviation histogram peak, enabling correct identification of noise free images.

It was clear that a more thorough test was needed to differentiate between the two methods. Furthermore, since the image pyramid method contains many different rules, a variety of image types is needed to test the algorithm thoroughly. The five remote sensing images shown in Figure 3.4 were chosen since they cover a broad spectrum of remote sensing image types. Figure (a) has low spatial detail. Figure (b) has sharply defined edges typical of coastal regions. High spatial detail typical of city scenes can be seen in figures (e) and (d), while (c) contains a combination of dense spatial detail and open spaces. All images are 8-bit greyscale with resolutions shown in the figure and were acquired by the KITSAT imager.

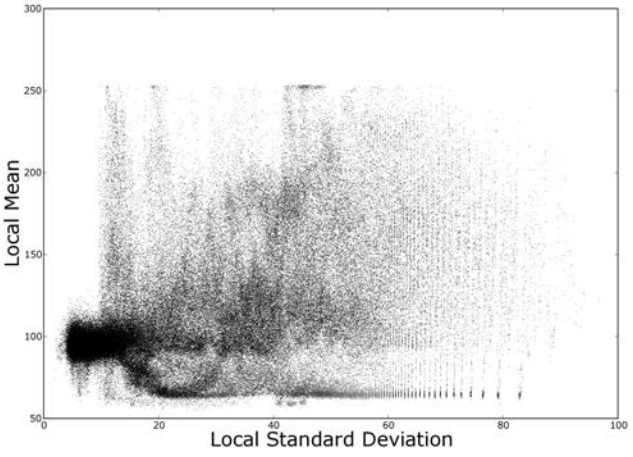
To give a different perspective on the vastly differing spatial structures encountered in remote sensing, scattergrams for the `Lasvegas` and `Redsea` images are depicted in Figure 3.5(a) and (b). From 3.5(a) one can see that there are very few blocks with low standard deviation; a high concentration of cells is located at approximately $\sigma_g(k) = 6$. In Figure 3.5(b), on the other hand, the many homogeneous areas mean that there is a concentration of low variance cells. The semi-periodic bands visible to the right of the graph are caused by quantisation.

As will be discussed in section 3.3.2 on page 3.3.2, the dynamic range of the images from Figure 3.4 was reduced to allow fair comparison between the two

(a)

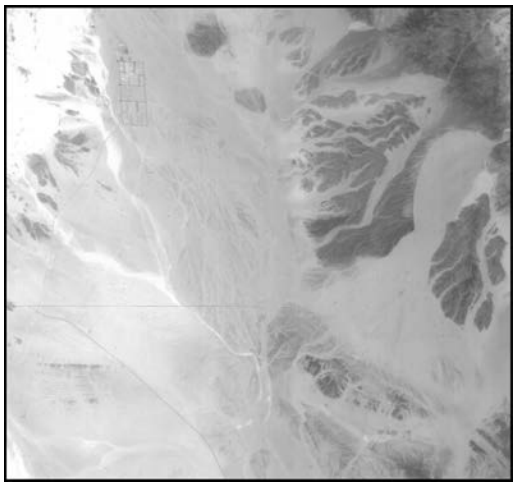


(b)

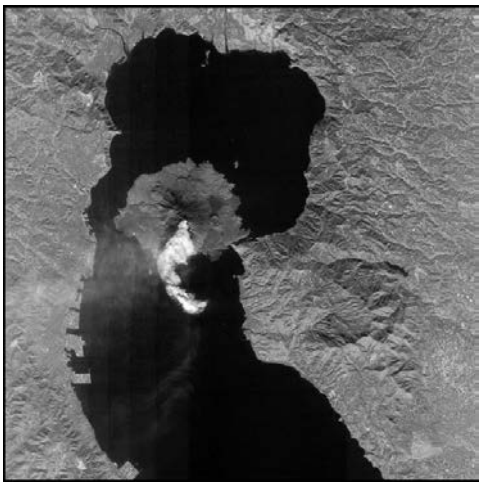


(c)

Figure 3.3: Image of dam with large high variance area with scattergrams. The scattergrams were constructed using 6×6 pixel cells. (b) was derived from the noise-free image shown, while for (c) Gaussian noise with $\sigma_n = 5$ was added.

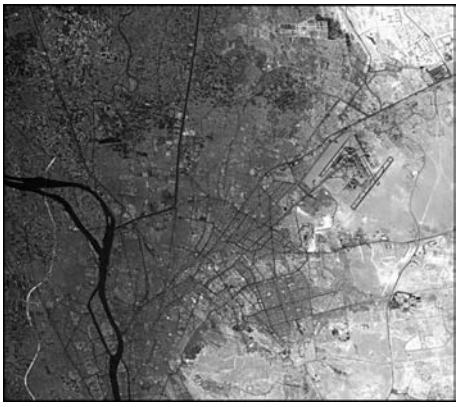


(a) `Redsea`. 1420×1330

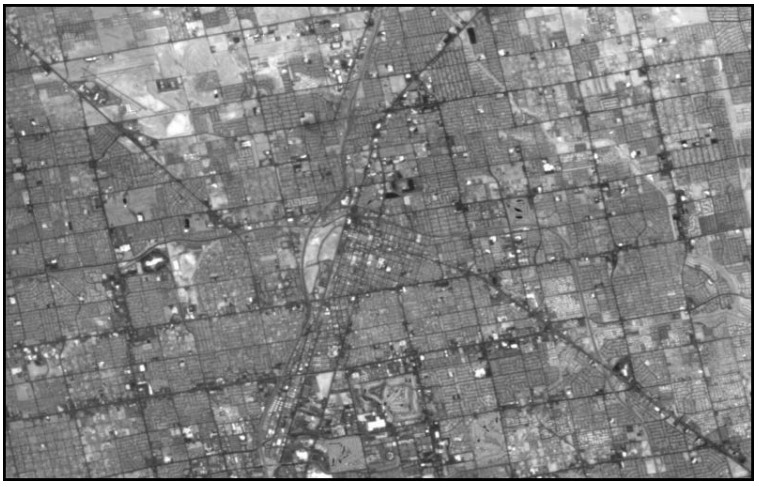


(b) `Volcano`. 1554×1556

(c) `Kuwait`. 2800×2200

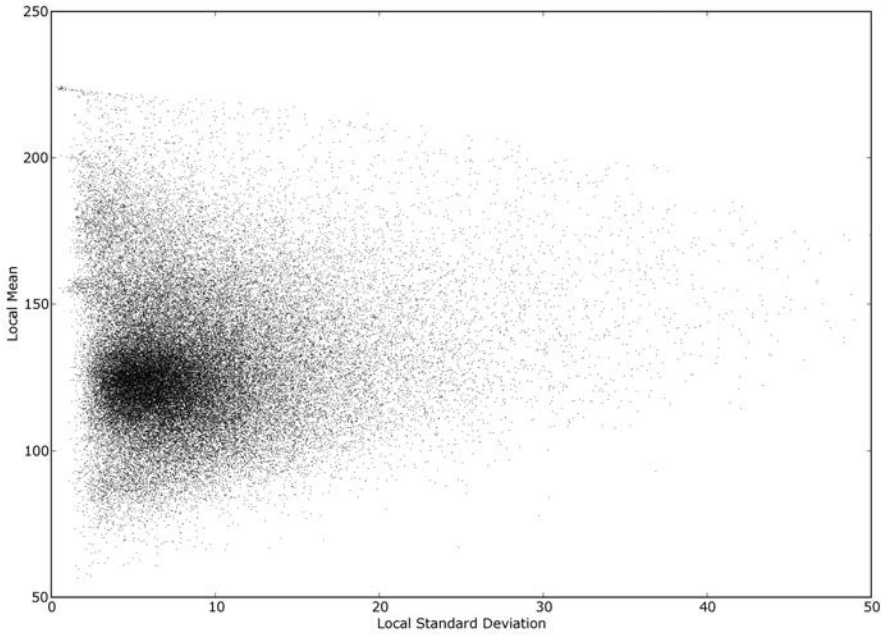


(d) `Cairo`. 1918×1686

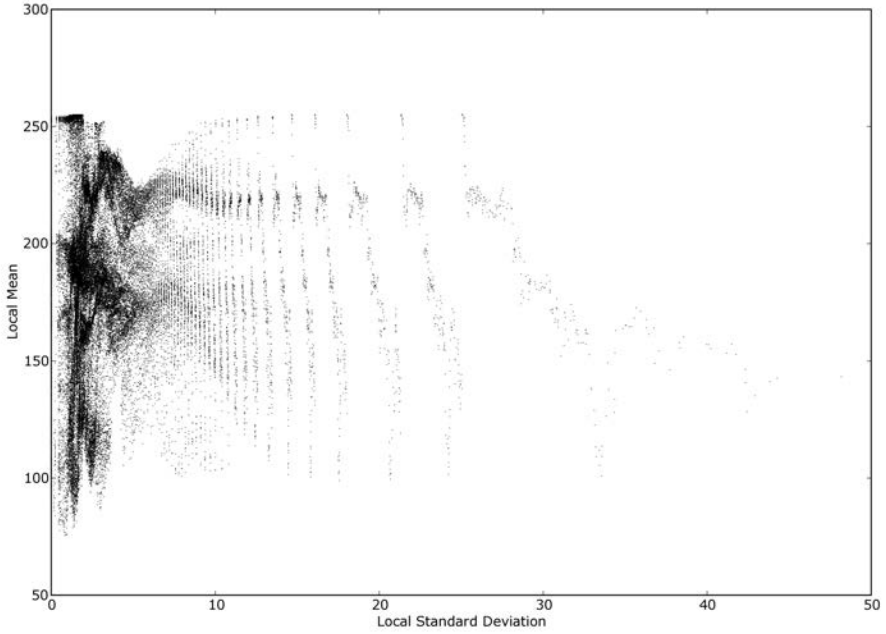


(e) `Lasvegas`. 1228×780

Figure 3.4: Base images used during experiment, with their resolutions.

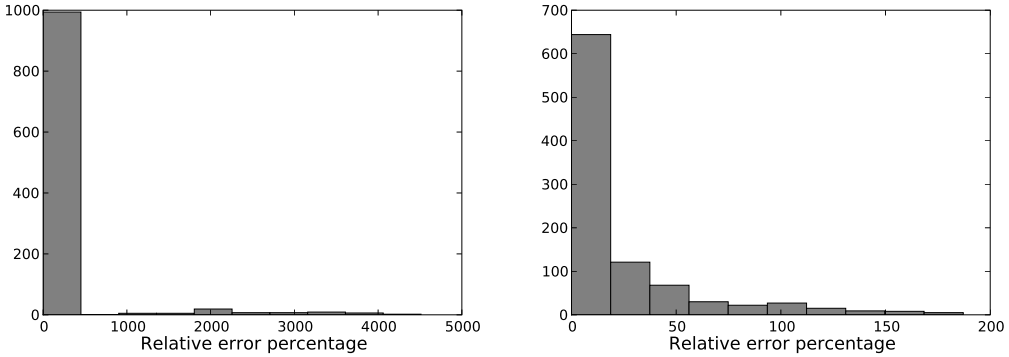


(a) `Lasvegas`



(b) `Redsea`

Figure 3.5: Scattergrams clearly show the differences in local statistics.



(a) All errors. (b) Greatest 10% of errors discarded.

Figure 3.6: Histogram of relative error percentages for the standard deviation histogram method. In (b) the greatest 10% of errors have been discarded to allow for a more detailed view of the distribution of the remaining errors.

algorithms. Varying levels of Gaussian noise was added to these images to test the estimation algorithms' abilities in differing circumstances. Values chosen were $\sigma_n = \{0, 1, 3, 5, 7, 9, 14, 20\}$, which broadly correspond to the range of noise levels usually used for testing algorithms in the literature encountered. For each noise level ($\sigma_n = 0$ excluded), 30 different outcomes were generated. The total number of images used in the test was therefore 1055.

3.3 Results

3.3.1 Standard deviation histogram method

The accuracy of the noise estimates was measured by computing the relative error. The error percentages were calculated using:

$$\frac{|\hat{\sigma}_n^2 - \sigma_n^2|}{\sigma_n^2} \times 100\%,$$

where $\hat{\sigma}_n^2$ is the variance estimate and σ_n^2 is the known added Gaussian variance. If the true image is noiseless and any noise is reported, the error percentage is taken as 100%.

Application of the standard deviation histogram method resulted in the relative errors shown in Figure 3.6 in histogram form. While 70% of the estimates had better than 25% error, some images resulted in unacceptably large errors of 100% and greater.

The errors for each individual image (averaged over the 30 noise outcomes) are presented in Table 3.2. Since the method has no rule to treat a noiseless[3].

[3]An image free of additive Gaussian noise; all images possess at least quantisation noise

Table 3.2: Average error percentage per image over 30 instances of each image. Results are for the standard deviation histogram method.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	100	0	0	0	100
1	3225	37.9	1514	4.83	1801
9	325	13.8	20.1	9.40	143
25	111	6.32	7.29	4.68	61.0
49	64.1	8.50	4.08	3.23	38.7
81	40.1	10.3	3.52	3.00	27.8
196	18.1	9.34	5.03	3.31	14.5
400	8.08	3.36	6.22	5.75	5.57

Table 3.3: Estimated noise variance per image over 30 instances of each image. Results are for the standard deviation histogram method.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	29.1	0.00	0.00	0.00	18.9
1	33.2	1.38	2.52	1.04	19.0
9	38.2	10.2	10.8	9.78	21.9
25	52.8	26.4	26.6	25.9	40.2
49	80.4	53.1	49.3	48.7	67.9
81	113	89.3	80.6	81.1	103
196	231	214	188	194	224
400	429	410	377	377	421

image as a special case, all images were estimated as having some finite noise. This noise variance was rounded to the nearest 10^{-2}. Entries in the $\sigma_n^2 = 0$ row with variance after rounding > 0 have 100% error. Note that only one noiseless image was evaluated for each input image instance since the outcome is deterministic. To gain better insight into the behaviour of the algorithm on noiseless images, the actual estimates, $\hat{\sigma}_n^2$, similarly averaged, are presented in Table 3.3. The three images containing large areas of low spatial detail were correctly identified as having negligibly small noise. However, in the `Lasvegas` and `Cairo` images, with their lack of large homogeneous areas, large noise estimates were erroneously made.

For $\sigma_n^2 = 1$, the relative error is very large for all the images except for the *Kuwait* image. Since σ_n^2 is so small one might argue that the *relative* error is an over sensitive indicator. From the average estimates in Table 3.3, one can see that the same two images are the problem cases. Generally, with increasing noise levels,the accuracy of the method increases since the increasing contribution of σ_n^2 to σ_g^2 starts to outweigh that of σ_f^2.

To allow insight into the algorithm's variation in behaviour, a measure

Table 3.4: Standard deviation of error estimates from Table 3.2.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	603	10.4	56.8	5.19	367
9	60.1	6.26	8.87	4.17	34.8
25	18.3	4.15	5.07	3.66	11.8
49	11.6	4.56	3.13	2.52	8.64
81	8.06	6.89	2.18	2.47	5.67
196	5.92	4.23	3.35	2.16	4.17
400	5.19	2.20	3.53	2.87	2.97

Table 3.5: Standard deviation of noise variance estimates from Table 3.3.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	6.04	0.10	0.57	0.06	3.67
9	5.41	0.56	0.80	0.50	3.14
25	4.58	1.22	1.44	1.17	2.95
49	5.71	2.33	2.49	2.00	4.23
81	6.53	5.68	3.33	3.14	4.59
196	11.6	8.33	8.96	7.49	8.17
400	24.7	12.5	17.4	12.1	13.5

of spread about the averages (listed in Tables 3.2 and 3.3) is also reported. The standard deviation across the 30 images at each noise level is shown in Table 3.4 for relative estimate error and Table 3.5 for noise variance estimates. Generally, each measure's standard deviation increases with an increase in the measure. The spread of data does not refute the claim that the method performs poorly for the `Lasvegas` and `Cairo` images.

3.3.2 Image pyramid method

Dynamic range saturation

In initial tests, the image pyramid method severely underestimated the amount of noise present in the images. Although initially surprising, it is a known weakness of the method when dealing with images where the dynamic range is saturated. The problem occurs when the lowest variance cells selected through order statistics have very high or very low average intensity, i.e., grey level values saturated at or close to 0 or 255 for 8-bit images. Since the average value is already at the limit of the dynamic range, and the Gaussian noise added has a zero mean value, half of the noise signal is 'clipped'. This reduces the variance in these cells to far below typical values. Because the method

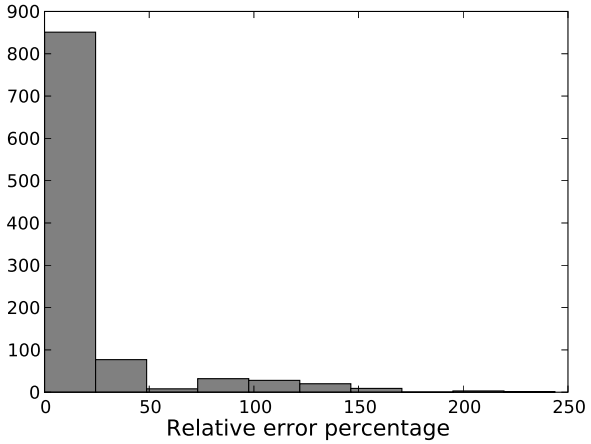


Figure 3.7: Histogram of relative error percentages for the image pyramid method.

uses order statistics, it is these low variance cells that are then used to derive the noise estimate.

The remote sensing images from Figure 3.4, used in the test, had already been subjected to severe contrast stretching. The resulting images all contained areas of saturated white or black. In the case of the `Volcano` and *Kuwait* images, 3.4(b) and 3.4(c), these areas were very large.

As mentioned in the introduction, it is a typical form of image processing applied to satellite images before they are viewed by humans, since it enables us to make full use of the available dynamic range of monitors or printers during analysis. It is also more aesthetically pleasing. However, unprocessed satellite images rarely make full use of the dynamic range of the sensor, since this would imply possible information loss at the very start of the remote sensing chain.

Therefore, without great loss of generality, the input images were scaled so all intensity values fall in the 30–225 range, leaving ample headroom for the noise. The same noise levels were applied again and the algorithms rerun. The histogram of error percentages is presented in Figure 3.7, while a more detailed view of the results is presented in Table 3.6. By comparing the histogram in 3.7 with the one from Figure 3.6, it is clear that there are fewer large errors with the image pyramid method than with the standard deviation histogram method. If one considers Table 3.6, it is apparent that in each case noiseless images were correctly identified as such, even for the `Lasvegas` and `Cairo` images with high spatial density of details throughout the image. For the images where $\sigma_n^2 = 1$ a 100% error was recorded, since application of *Rule 1* meant that $\hat{\sigma}_n^2 = 0$ in every case. Considered objectively, this is not a problem. Images where $\sigma_n^2 = 1$ are indistinguishable from $\sigma_n^2 = 0$. Additionally, even though noise with $\sigma_n^2 = 1$ was added, it is suppressed by integer quantisation of the image, so $\hat{\sigma}_n^2 = 0$ is probably a better characterisation of the image

Table 3.6: Average error percentage per image over 30 instances of each image. Results are for the image pyramid method.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	100	100	100	100	100
9	216[a]	11.0	14.6	14.7	39.1
25	139	17.6	7.47	4.68	34.3
49	101	16.3	9.36	2.14	12.9
81	89.0	12.4	12.4	1.72	13.0
196	21.6	2.01	8.07	4.65	11.9
400	26.5	5.48	3.85	5.39	8.04

[a] Only 4 instances evaluated.

Table 3.7: Standard deviation of noise variance estimates from Table 3.6.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	0	0	0	0	0
9	16.0[a]	0.95	4.15	0.64	6.04
25	10.5	0.88	1.55	2.03	15.8
49	5.31	0.66	6.24	0.34	9.32
81	9.59	1.51	5.52	0.30	7.81
196	2.27	1.52	1.66	5.37	1.59
400	1.42	0.28	2.41	2.22	5.23

[a] Only 4 instances evaluated.

noise. These 'errors' were taken as 0 when generating the histogram.

In the low noise, $\sigma_n^2 = 9$, `Lasvegas` image, signal noise separation was deemed too poor to enable an accurate noise estimate in 26 of the 30 cases. In the remaining four cases, the noise estimate resulted in a large error of 216%. Considering this large error, as well as the results of the standard deviation histogram experiment on the image, refusing to estimate would be the correct decision in this case.

Given the variation in the algorithm's behaviour, i.e., rejecting some of the `Lasvegas` images while estimating others, the spread of the data is reported to quantify some of this variation. The standard deviation of the error estimates are presented in Table 3.7. Note that the standard deviation is the largest for the image where only 4 instances were evaluated.

While most of the other error percentages seem acceptable, especially the `Lasvegas` but also the `Cairo` images result in large errors at low noise levels. The results from the histogram agree with the findings from the comparative test [78]: the image pyramid method performs better than other methods

at low noise levels, but still gives very large errors in approximately 10% of images tested. Furthermore, while the algorithm correctly refused estimation in most of the cases for the `Lasvegas` image with $\sigma_n^2 = 9$, some estimates were attempted with poor results.

Making the algorithm more conservative

It is suspected that the reason for the image pyramid algorithm performing worse than in the original paper [76] is simply because the test images are more difficult. The busiest image from the test suite in [76] appears to have less dense spatial structure than the extremely dense `Lasvegas` image or one of the synthetic test images from the comparative test. Everyday photographs typically have considerably lower spatial density than remote sensing images. This is why testing the algorithm on remote sensing images is important.

Rules were examined to determine which rules were used to classify which images, as well as the average errors associated with each rule. It was found that *Rules 5* were only being used for the low noise `Lasvegas` and `Cairo` images, with very large average errors relative to the other rules. It was also interesting to note that the uniform image rule, *Rule 2*, was never used, as is to be expected. However, when testing the algorithm on a uniform image it was used.

In an attempt to make the algorithm more conservative, the deviation sequence, $\alpha(l)$ was 'shifted to the left':

$$\alpha_c(l) = \alpha(l+1),$$

where $\alpha_c(l)$ is a more conservative deviation sequence. The reasoning behind this shift is that images that would previously be detected as having the noise to signal transition at level $l = 5$, would now transition at $l = 4$, so *Rule 3 and 4* would apply. Therefore, by changing α, one's view on what constitutes good signal-noise separation is effectively becoming more conservative. By shifting the entire deviation sequence instead of modifying individual rules, the more conservative approach filtered through to all levels of the image pyramid.

The algorithm was re-evaluated on the same test images. The histogram is shown in Figure 3.8, while Tables 3.8 and 3.9 gives the detailed results. Upon comparing the histogram to Figure 3.7, the improvement is obvious. If one considers Table 3.8 one can see the cause: four images were classified as having signal noise separation that is too poor to attempt estimation. The four images are specifically those in which the noise variance was overestimated previously, confirming the success of the modification. *Rules 3 and 4* were also applied to some of the the $\sigma_n^2 = 25$ and $\sigma_n^2 = 49$ `Cairo` images; the average in Table 3.8 was computed from the remaining images, as detailed in the table footnotes.

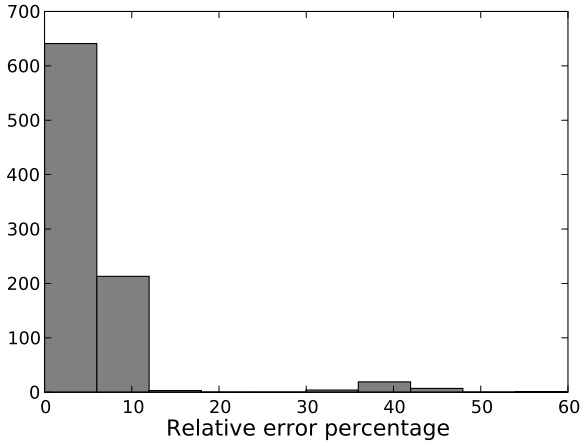


Figure 3.8: Histogram of relative error percentages for the image pyramid method using $\alpha_c(l)$.

Table 3.8: Average error percentage per image over 30 instances of each image using $\alpha_c(l)$.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	100	100	100	100	100
9	-	2.71	3.24	0.61	-
25	-	2.42	8.08	1.96	1.99[b]
49	-	0.73	5.59	2.42	4.95[c]
81	59.9[a]	0.46	3.22	2.13	8.04
196	39.40	3.38	1.70	3.21	4.24
400	6.33	9.49	6.92	9.61	6.39

[a] Only 1 instance evaluated.
[b] Only 13 instance evaluated.
[c] Only 29 instance evaluated.

3.3.3 Feasibility of embedded implementation

The embedded C implementation of the image pyramid method was tested on images of various sizes. For each image size the algorithm was repeated 10 times and the average execution time recorded. The results are presented in Figure 3.9. The time taken for each of the 10 runs at a given size was almost identical.

Given the image pyramid structure with calculations being repeated at each level, one would expect the algorithm to be $O(n \log n)$; however, the graph in 3.9 appears more linear. This is the result of the optimisations previously described: reusing the results from the previous level means that for each of successive level of the $\log n$ levels, the time spent decreases drastically. The

Table 3.9: Standard deviation of noise variance estimates from Table 3.8.

σ_n^2	Lasvegas	Volcano	Redsea	Kuwait	Cairo
0	0	0	0	0	0
1	0	0	0	0	0
9	-	0.12	0.19	0.06	-
25	-	0.16	0.26	0.18	0.47[b]
49	-	0.35	1.93	0.17	0.95[c]
81	0[a]	0.41	3.49	0.28	2.53
196	5.34	1.00	1.91	1.88	5.00
400	6.14	2.23	11.98	3.45	12.04

[a] Only 1 instance evaluated.
[b] Only 13 instance evaluated.
[c] Only 29 instance evaluated.

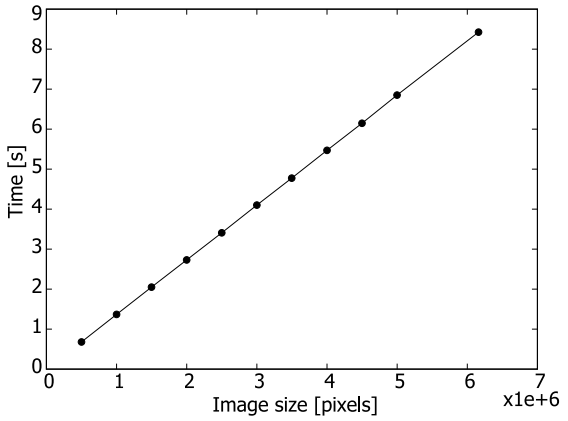


Figure 3.9: The execution time of the embedded image pyramid implementation.

largest image tested, the 6.1 megapixel image at the end of the graph, had dimensions 2200×2800 pixels and took 8.43 seconds.

Therefore, even though the algorithm was developed with parallel architecture in mind, it still performs more than adequately on sequential SH4 chip.

3.4 Conclusion

3.4.1 Choice of method

It is evident from the experiments conducted that the conservative image pyramid method is preferable to the standard deviation histogram method. Table 3.10 presents the average and median relative error percentages for each method. Since there are large errors in the standard deviation method, which can have a disproportionate influence on the average, the median is presented

Table 3.10: A summary of the comparative results: average and median relative error percentages.

	SDH[a]	IP[b]- $\alpha(l)$	IP - $\alpha_c(l)$
Average	158	21.4	5.24
Median	11.8	11.4	3.45
Standard deviation	613	33.1	7.52

[a] Standard deviation histogram.
[b] Image pyramid.

as well.

The experimental results confirm the findings of the comparative study discussed in section 3.1.6, namely that the image pyramid method gives the best results in low noise conditions. This is an important requirement: it is crucial for the chosen algorithm to be able to successfully identify a noiseless image, lest a good image erroneously be given a low priority.

The standard deviation in a comparative experiment can be used as an informal way to evaluate whether the compared methods' confidence intervals overlap. Although, in this case the it appears as if the confidence intervals indeed do overlap, an alternative interpretation is available: the standard deviation is an additional measure of algorithm performance. As suggested in the histograms, Figures 3.6, 3.7 and 3.8, the probability density function of relative error is not symmetrical: higher standard deviation instead corresponds to a longer tail on the high-error, right hand side of the histogram. Since the SDH method results in very large relative errors, the spread of these errors are also greater than the IP methods. Furthermore, the conservative IP method resulted in the smallest errors and thus these errors have the smallest spread.

By incorporating different cell sizes into the algorithm the image pyramid method avoids the cell-size trade-off, discussed in section 3.1.3, faced by methods that rely on a single tessellation of the image.

There is one area that the standard deviation histogram algorithm has the upper hand: it does not have the sensitivity to images containing saturated areas that the pyramid method has. Because it used information from the entire image and not just from the areas with the least variance, the contrast stretch applied to the images in Figure 3.4, did not affect it. Although remote sensing images do not usually contain saturated areas, depending on the calibration of the sensor, saturation can occur occasionally. It is most likely to happen when imaging cloudy scenes: the intensely white clouds can cause saturation at the upper limit of the sensor's range.

However, it is preferable to have a method that sometimes underestimates noise levels than one that often overestimates them.

3.4.2 The saturation problem

If, in the system where the algorithm is to be implemented, it is found that saturation caused by clouds occurs often, the image pyramid method could be adapted. It was recommended in [76] that the average intensity of the cells used in the order statistics be checked and a warning given when it is too close to the edges of the sensor's dynamic range.

Alternatively it is possible to first discard cells with an intensity that is too low or too high. If one considers the scattergram, this is equivalent to clipping the top and bottom off the graph. So long as the remaining points are still representative of the noise, i.e., the image contains unsaturated homogeneous areas, the working of the algorithm should not be affected.

However, as cloud abundance increases, more of the points in the scattergram will be concentrated near the top of the graph and it will become difficult to find representative cells when the saturated cells have been discarded.

Luckily there are two mitigating factors. Firstly there is the ability of the algorithm to assess its own estimation capability: if the signal noise separation is too poor, *Rules 3 and 4* detect that. Secondly, as will be discussed in Chapter 5, cloud cover is weighed much more heavily in the final quality score than noise. Therefore, when the amount of cloud cover has increased to such a degree that it becomes difficult to find homogeneous cloud free cells, cloud cover will dominate in the final quality score and the amount of noise is of small account.

3.4.3 Use of multiple channels

In the final implementation, all the different channels in the satellite image will have to be considered, since the noise in different channels of a colour image is independent [47, p. 339]. Indeed, different parts of the detectable spectrum often have distinctly different noise characteristics; and hence, in remote sensing, noise estimation algorithms are applied to each channel individually [42, 112].

3.4.4 Choice of SNR

Given the variety of available SNRs, discussed in section 3.1.1, which one should be used? The most useful measure is the variance SNR, equation (3.1.3). However, the fact that access to the uncorrupted image, $f(x, y)$, and therefore to the signal variance, σ_f^2, is unavailable, eliminates the variance SNR.

Using the average signal intensity, $\overline{z}$, as in (3.1.4), does not seem to add any relevant information not in the PSNR. If images with higher $\overline{z}$ were indeed preferable to images with low $\overline{z}$, then simply adding a constant offset to the image intensity would be a common image enhancement or preprocessing technique. It is not.

As mentioned in the introduction, PSNR is suitable since it does not require unavailable image information and has been proven to be a good quality measure for white noise distortion. However, there is no useful extra information in PSNR not contained in the noise variance, σ_n^2. Larger images should not receive a higher quality score simply because of their size. While it is true that they might contain more information, they will also take proportionately longer to download from a satellite, negating any advantage. Furthermore, the peak signal value adds no information useful for ranking images: all images acquired by the same sensor will have the same maximum possible intensity. Therefore, it is recommended that the noise variance estimate itself be used as a feature.

Chapter 4

Defocus estimation

4.1 Introduction

An image captured by a sensor is not an exact reproduction of the scene viewed. If a point of light is viewed, the optical system will blur or spread the light to some degree, characterised by the *point spread function* (PSF). In the mathematical model of the imaging system, the PSF is a weighting function for spatial convolution [94, pp. 78–91]. It can be considered as the spatial responsiveness of the imaging system. If the imaging system becomes defocused, the PSF shape will be affected and the image will become blurry.

Estimation of image sharpness, focus or blur spans a variety of fields. Many different techniques can be employed depending on the application.

Measurement of image sharpness is a crucial part of edge detection algorithms [47, pp. 572–585]. These algorithms make no assumptions about the image acquisition chain; they simply evaluate the sharpness of edges in an image.

Estimation of optimal focus is an important problem with practical application in autofocus algorithms for cameras [121, 115]. These algorithms rely on a *sequence* of images from the same subject. Typically an image sharpness measure is computed for each image in the sequence. The measure is compared across all the images in the sequence. The sharpest image is assumed to correspond to an in-focus imaging system. More advanced methods use focus measures that are invariant to illumination changes [115]. Others estimate the PSF of the imaging system so that the focus measure is invariant of the imaged object [121].

Estimation of PSF is also required during restoration of blurred images (called deconvolution). Sometimes it is possible to estimate the PSF by using

knowledge of the imaged object [47, pp. 256–257] [17]. When the imaged object is unknown, blind estimation of PSF is used. These techniques use only a single potentially degraded image to estimate PSF and are discussed further in section 4.2.1.

In the remainder of the introduction the degraded image model is explained (section 4.1.1) and defocus estimation in the context of image quality assessment is discussed (section 4.1.2). An overview of existing literature is presented in section 4.2: section 4.2.1 introduces the field of blur estimation; section 4.2.2 discusses the theoretical base of the class of blur estimation algorithms used; section 4.2.3 explains some of the shortcomings of the methods in existing literature. Angular spectral smoothing, a new technique based on the method discussed in 4.2.2, is presented section in 4.3. The various experiments conducted are described in section 4.4: different window functions are considered in section 4.4.1; the PSF estimation methods from sections 4.2.2 and 4.3 are compared in section 4.4.2; the effect of reduction in image dynamic range is investigated in section 4.4.3 and the embedded implementation is discussed in section 4.4.4. Finally, results are presented in section 4.5 and conclusions are drawn in section 4.6.

4.1.1 Degraded image model

A linear model is commonly used to model image degradation:

$$g(x,y) = f(x,y) * h(x,y) + n(x,y), \tag{4.1.1}$$

where $f(x,y)$ is the original image, $g(x,y)$ is the degraded image and $h(x,y)$ is the PSF. Additive noise is modelled by $n(x,y)$ (see chapter 3). The two dimensional convolution operator is $*$. If one expands the $*$ operator, (4.1.1) becomes:

$$g(x,y) = \sum_{\alpha=-\infty}^{\infty} \sum_{\beta=-\infty}^{\infty} f(\alpha,\beta)h(x-\alpha, y-\beta) + n(x,y).$$

The PSF of an optical system can be decomposed into many parts. However, when the optical system becomes defocused, the blur PSF dominates. This can be caused by temperature variations on board of a satellite. The PSF of the defocused lens system with circular aperture can be approximated by a uniform function with two-dimensional (2-D) circular support and radius R [18]:

$$h(x,y) = \begin{cases} 0; & \sqrt{x^2+y^2} > R \\ 1/(\pi R^2); & \sqrt{x^2+y^2} \leq R. \end{cases} \tag{4.1.2}$$

4.1.2 Defocus estimation in the context of image quality assessment

The concept of measuring image blurriness as part of image quality assessment is not new; edge sharpness has been used as a measure of image quality in the past.

In [66] edge sharpness level is proposed as one of three objective measures to aid in image quality assessment. The other two proposed measures were random and structural noise. Although no such distinction is made in [66], it is important to separate image *degradation* measures and image *content* measures. Random and structural noise are degradation measures, whilst edge sharpness level is a content measure. Degradation measures are generally more objective than content measures: an image with low SNR is almost always worse than an image with high SNR. The same cannot be said for content measures: in remote sensing there is little reason to believe images containing more sharp edges in the global structure are more useful than images containing few or none. However, if the telescope becomes defocused, it will produce a useless set of images that contain no sharp edges.

The problem is therefore to distinguish between images where the subject has few or no sharp edges and images blurred by the imaging system. Such images might achieve the same score using an edge sharpness level measure. Indeed, this problem resulted in outliers in the image quality assessment experiments in [75], where inappropriate use of an edge sharpness measure caused some images to receive disproportionately bad scores.

Hence, it was decided to measure only degradation measures for image quality assessment: the defocus blur extent of the PSF is chosen instead of edge sharpness level. To the author's knowledge, the fields of PSF estimation and quality assessment have not been combined before. The relationship between defocus extent and image quality is discussed further in Chapter 5.

4.2 Literature

4.2.1 Point spread function estimation

Blind estimation of the PSF is a subset of the blind image deconvolution problem, which, given the linear degradation model from (4.1.1), attempts to recover the original image, $f(x, y)$, and PSF, $h(x, y)$, using only the degraded image, $g(x, y)$, and partial information about the imaging system. The problem is difficult. Firstly, it is ill-conditioned: small changes in input conditions can cause large changes in results. Secondly, solutions may be non-unique and, therefore, assumptions about the PSF and image structure are often necessary. Because of these difficulties many approaches have been suggested. These cover a broad range in terms of computational complexity and applica-

bility, with some tailored to specific scene types and others to specific PSFs. An instructive overview can be found in [62, 63]. Although many techniques are discussed, they follow one of two approaches:

1. Identify the PSF first and then use a classical technique such as Wiener filtering to restore the image. This approach is called *a priori* or *direct* blur identification. Algorithms in this category are computationally simple, but sensitive to SNR.

2. Identify the PSF and true image simultaneously. An iterative process is used that estimates the PSF, restores the image, evaluates the result and then repeats. Many algorithms fall into this *indirect* category. They are generally computationally complex and often have ill-convergence.

Some of the methods belonging to the second category include techniques based on autoregressive moving average (ARMA), nonparametric deterministic image constraints restoration as well as higher order statistics. ARMA techniques use a statistical autoregressive model for the image. They are less sensitive to noise than direct techniques, but the model is not suitable for images containing sharp edges. Nonparametric deterministic image constraints do not assume parametric models for either blur or image, but require that the image have finite support, i.e., the image is an object against a uniform background. Higher order statistical models are suitable for images with 'spike'-like structure, such as astronomical images. Methods based on these models are robust against noise, but are very computationally expensive. None of these models are applicable to all remote sensing images.

In [19] edges in the image are identified and used to derive blur characteristics. If no edges are present ARMA techniques are used. This is a good compromise, but still retains the $O(N^2)$ computational complexity of ARMA techniques. Another method that attempts to identify edges in the image and use these for blur estimation is [37]. This method claims to be able to handle spatially variant blurring. Neither of these methods make provision for images that naturally contain blurry edges in the image structure.

Recent advances postdating [62] include an autocorrelation based direct method for motion blur identification capable of identifying both linear and accelerated motion [116, 117, 118]. In [41] the difficulty of PSF estimation and restoration is acknowledged. To sidestep it, image features that are invariant with respect to blur are identified and used to recognise objects in a degraded scene. If access to multiple instances of the same image blurred by substantially different PSFs is available, [44] can be used in either direct or indirect configuration. An alternative approach to direct identification uses vector quantisation to train a classification system to recognise various types of PSF [80]. This would allow identification of Gaussian blur PSFs, which can be used to model atmospheric turbulence. However, the method requires the system be trained for specific images.

If the motion path is known, images degraded by complicated, non-linear motion PSFs can be successfully deconvolved [93]. This is utilised in [13]: the motion path is identified using two cameras to achieve high spatial and temporal resolution. An iterative technique that imposes a piece-wise smoothness constraint on the image is introduced in [119]. Although this technique represents an improvement over previous techniques, its iterative nature, non-unique solutions and especially its piece-wise smooth image structure make it unsuitable for this application.

For the purposes of this application, the class of direct methods based on spectral techniques was chosen [18, 21, 92]. Despite recent advances discussed above and in [62], these methods remain popular because they make no assumptions about the true image structure and are computationally simple. These attributes make them suitable for an on-board implementation in a remote sensing system, since earth images do not conform to a template and processing of very high resolution images using limited memory and processing power is required.

4.2.2 Blur identification based on spectral techniques

Since no constraints are placed upon the true image, assumptions about the blur PSF shape are required to make the blind deconvolution problem solvable.

Two common blur types, linear motion blur and defocus lens blur, can be represented by simple, spatially invariant, parametric models. For motion blur a 1-D rectangular (also called *box-cart*) PSF is used. For defocus blur the model from equation (4.1.2) is used. Although blurring caused by space-variant PSFs can be restored [103], it cannot be identified using direct blind deconvolution. Fortunately the spatial-invariance assumption is valid in the remote sensing context; since the objects imaged are approximately the same distance from the sensor, defocus blur will be uniform throughout the image.

Even though the motion blur and defocus blur problems are often solved using similar methods, the motion blur model is not applicable in a remote sensing application with a linear push broom sensor. Since the sensor is not an array, motion cannot cause smearing of scene content across multiple pixels in the acquired image, unless the motion is exactly in the cross-track direction. Therefore it was decided to concentrate only on defocus blur.

Power spectrum and power cepstrum

Cannon proposed the use of the power cepstrum for blur identification [18]. If the power spectra are considered, (4.1.1) becomes:

$$P_g(u,v) = P_f(u,v)\,|H(u,v)|^2 + P_n(u,v). \tag{4.2.1}$$

The frequency response of the PSF, $H(u,v)$, is of the form $J_1(Rr)/(Rr)$ where R is the PSF radius, $r = \sqrt{u^2+v^2}$ and $J_1(\cdot)$ is the first-order Bessel function of

the first kind (which has nearly periodic, radial zero-crossings). The function $J_1(x)/x$ is also known as the *jinc* function, for its similarity to the function $\text{sinc}(x) = \sin(x)/x$, and is discussed in detail in [16, pp. 347–381]. Welch's method [108], which averages small sections to reduce the variance of the P_g estimation, is used:

$$P_g(u,v) = \frac{1}{N}\sum_{i=1}^{N} |G_i(u,v)|^2 . \tag{4.2.2}$$

The image is subdivided into square sections. The size of the section must be greater than the size of the PSF. Each section, $g_i(x,y)$, is windowed [84, pp. 623 – 630] and the periodograms, $|G_i(u,v)|^2$ (where $G_i(u,v) = \mathcal{F}\{g_i(x,y)\}$), of all sections are averaged to arrive at a power spectrum estimate, $P_g(u,v)$. The radial zero-crossings of $H(u,v)$ are zeros in $|H(u,v)|^2$ and local minima in $P_g(u,v)$. Blur identification in the spectral domain proceeds by identifying the first local minimum.

The type of window function to use was not specified in [18]. Various options exist. An optimal window for image restoration is derived in [101]. However, this window function requires knowledge of the PSF, which is unavailable since estimation of the PSF is the goal of blur identification. Various window functions are investigated in section 4.4.1.

If the power cepstrum is considered instead,

$$C_g(p,q) = \mathcal{F}^{-1}\{\log P_g(u,v)\}, \tag{4.2.3}$$

where $\mathcal{F}^{-1}$ is the inverse Fourier transform[1], the defocus blur radius is characterised by a ring of large negative spikes at $2R$ from the origin in $C_g(p,q)$ (as shown in Figure 4.3(a) on page 99). These are assumed to be the result of periodic zeros in $P_g(u,v)$. During blur identification only the negative part $C_g(p,q)$ is considered. Using the cepstrum instead of the spectrum has some advantages: it is algorithmically easier to identify a global negative maximum rather than a first significant local minimum. Furthermore, since the ring of spikes is the result of *periodic* minima in the spectrum, it is more robust to noise than identification based on only *one* minimum.

Bispectrum and bicepstrum

In an attempt to increase the reliability of blur identification in the presence of Gaussian noise, Chang *et al.* [21] turned to the bispectrum [77]. Just as the Wiener-Khintchine theorem relates the power spectrum to the Fourier transform of the autocorrelation (second-order moment sequence) [82], so the bispectrum of a signal $X(k)$ can be defined in terms of the third-order moment

[1]Some power cepstrum definitions use the forward transform instead [24]. They are functionally equivalent.

sequence:

$$B(u,v) = \sum_{\tau_1=-\infty}^{+\infty} \sum_{\tau_2=-\infty}^{+\infty} m_3(\tau_1,\tau_2) e^{-j(u\tau_1+w\tau_2)}, \tag{4.2.4}$$

where $m_3(\tau_1,\tau_2) = E[X(k)X(k+\tau_1)X(k+\tau_2)]$ is the third order moment sequence of $X(k)$, a real, discrete, zero-mean, stationary process. Since the third-order moment sequence is zero for a stationary, zero-mean Gaussian process, the bispectrum is invariant to Gaussian noise.

The bispectrum of a one-dimensional (1-D) signal is 2-D. Similarly, the bispectrum of a 2-D image is 4-D. Chang *et al.* suggests that for blur identification it is sufficient to consider only the 2-D "central slice" or 1-D "central-line", which can be computed more efficiently than the full 4-D bispectrum. The recommended direct 2-D estimator for the ith sub-segment is

$$B_g^{(i)}(u,v;0,0) = G_i(u,v)G_i(0,0)G_i^*(u,v). \tag{4.2.5}$$

The mean of the entire observed image has to be removed before it is segmented. In a similar fashion to (4.2.2), averaging is used to reduce the variance:

$$B_g(u,v;0,0) = \frac{1}{N}\sum_{i=1}^{N} B_g^{(i)}(u,v;0,0). \tag{4.2.6}$$

Since defocus blur has circular symmetry, the local minima in $B_g(u,0;0,0)$, a 1-D function, give the same information as in the 2-D $B_g(u,v;0,0)$ and are used for identification.

Savakis and Easton [92] rely for identification on negative peaks in the bicepstrum:

$$D_g(p,q) = \mathcal{F}^{-1}\{\log B_g(u,v;0,0)\}. \tag{4.2.7}$$

They argue that the use of bicepstrum over bispectrum holds the same advantages as cepstrum over spectrum, but inherits robustness against noise from the bispectrum.

Spectral subtraction and comb filtering

As an alternative to higher order spectra, Fabian and Malah [40] proposed adding pre and postprocessing to the cepstral method to increase robustness in the presence of noise. The preprocessing is based on a spectral subtraction technique [67], which attempts to estimate the spectrum of the blurred, noiseless image, $A(u,v) = F(u,v)H(u,v)$, by subtracting a weighted estimate of the noise power spectrum from the degraded image power spectrum:

$$|\hat{A}(u,v)| = \begin{cases} \sqrt{P_g(u,v) - \alpha\hat{P}_n(u,v)} & \text{if } P_g(u,v) > \alpha\hat{P}_n(u,v) \\ \epsilon & \text{otherwise} \end{cases} \tag{4.2.8}$$

$$\angle\hat{A}(u,v) = \angle G(u,v), \tag{4.2.9}$$

where ϵ is a small constant used to avoid numerical issues when taking the logarithm and α is the weight given to the noise power spectrum estimate. This approach rests upon the idea that, for white Gaussian noise, P_n is a constant offset proportional to the noise variance. This offset "obscures the zeros" [40]: when taking the cepstrum, it hinders the logarithm's ability to accentuate local minima in the power spectrum. Since the logarithm is non-linear, the degree to which minima are accentuated is highly dependent on their proximity to zero. A median-complement filtered image is used as an approximation of the noise-image from which $\hat{P}_n(u,v)$ is computed. Fabian and Malah claim, and this was confirmed by the author's tests, that the method gives better results if the image is not subdivided.

The cepstrum, $C_a(p,q)$, which has circular symmetry, is computed from $\hat{A}(u,v)$. Although this implies that using the most negative peak in $C_a(p,0)$ is sufficient for blur identification, the variance caused by noise makes such an approach unreliable. Instead, an angular average $C_a(r)$ is computed by converting $C_a(p,q)$ to polar coordinates $C_a(r,\theta)$ and averaging over θ.

Aside from the main pulse at $2R$, $C_a(r)$ exhibits harmonics at values of r approximately multiples of 2R. In the presence of noise there are also spurious peaks at other values which may dominate the true peak at $2R$. The postprocessing step employs an adaptive comb-like filter that amplifies peaks which have harmonics (like the true peak) and suppresses peaks which do not have harmonics (like the spurious peaks). The filter is:

$$\begin{aligned} C_l(r) &= \frac{|C_a(r)|}{\sqrt{\frac{1}{M}\sum_{i\in A_r}(C_a(i))^2}} \qquad (4.2.10) \\ \text{for quefrency r, } A_r &= \{i | i > r_0 \text{ and } i \notin (kr-1, kr, kr+1), k = 0,1,2\ldots\}. \end{aligned}$$

The "disturbance set", A_r, is the set of quefrencies[2] where harmonics of r are not expected. This set resembles a comb-filter with 3-point stop bands. The total number of points in A_r is M. To avoid an A_r consisting only of stop bands, which would be an empty set, the value of r_0 is selected as 3. The output of the filter is therefore limited to values of $r > 3$.

4.2.3 Problems with methods in existing literature

Lack of comparative defocus tests

In spite of popularity of direct methods [62], little or no comparative literature exists on the subject. Although [118] compares the cepstral method with their method for images with a variety of PSF blur extents and SNRs, their method is only applicable to motion blur. Reference [92] compares the methods of [21] and [18] with their own and applies only the postprocessing from [40]

[2] In a playful inversion of existing terminology, cepstral domain frequencies are referred to as *quefrencies*. The word *cepstrum* is derived from *spectrum* in a similar manner.

(in the author's experience it is the preprocessing that is responsible for most of the method's performance). Their comparison is based on degraded images generated from a single test image and only motion blur identification is tested, in spite of the fact that defocus blur is more difficult to identify [40].

Ability to operate at low SNR is commonly used as a measure of algorithm performance, with comparison to other methods often based on best achieved variance SNR, equation (3.1.3) on page 59. It was found that, for this class of methods, SNR is a poor indicator of performance. Although, for a given image, the SNR is highly correlated with the identification capability (increasing noise variance σ_n^2 or decreasing signal variance σ_s^2 have a detrimental effect), when comparing different images σ_s^2 does not play as important a role as signal frequency content. This is because, the more the high frequency content in $P_f(u,v)$, the better the periodic zeros at higher frequencies in $|H(u,v)|^2$ are visible in $P_g(u,v)$. For example, blur could be correctly identified in a city scene, Figure 4.1(c), which has dense spatial activity and good high frequency content, up to SNR as low as 2.8 dB. Using a desert scene, 4.1(a), with sparse spatial structure, 17 dB was the best that could be achieved. This confirms the need for a comparative test across a variety of image types, since comparing methods based on best reported SNR is of little use. The results of such a test are presented in section 4.5.

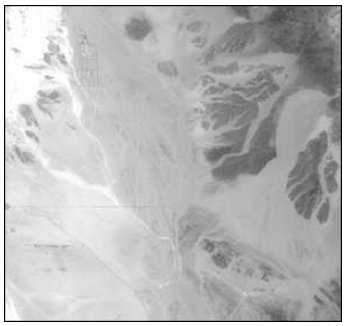
Inappropriate generalisation from 1-D to 2-D

Both [21] and [92] use 1-D, 256 pixel strip image sections for the averaging in (4.2.6). They only give results for motion blur, which, if sections are taken along the blur direction, is a 1-D function. However, they suggest that 1-D sections are equivalent to 2-D square image sections and it is implied that their results are valid for defocus blur as well. Experiments conducted showed that this is not the case. Define relative error distance as:

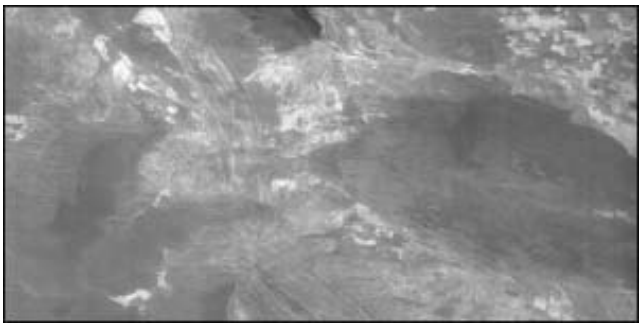
$$e_d = \frac{|R - \hat{R}|}{R}, \tag{4.2.11}$$

where $\hat{R}$ is the estimated defocus blur extent, i.e., the estimate of the R parameter from equation (4.1.2) on page 89. Using the five test images in Figure 4.1, defocus blur was added using blur extents $R = \{2, 3, 4, 9, 15\}$, resulting in 25 blurred images. Since no noise was added, the cepstral method could be used. 1-D sections of 256 pixels and 2-D sections of 128×128 pixels were both used and e_d was compared, with results in Table 4.1. It is clear that when identifying a 2-D PSF, using 2-D sections provides a significant advantage.

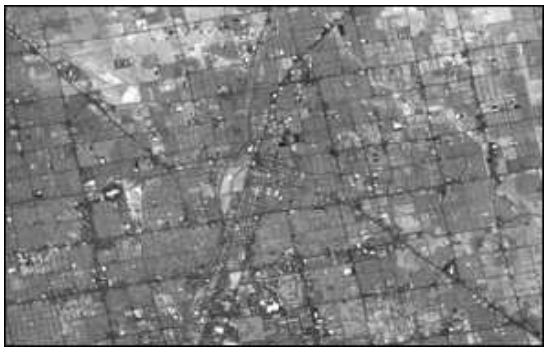
As mentioned in section 4.2.2, [40] uses a 3×1 median-complement filter to estimate $\hat{P}_n(u,v)$ for spectral subtraction. However, it was found that the filter imposes an unwanted structure on the power spectrum visible in Figure 4.2(b); it does not estimate the Gaussian noise as white, but instead concentrates noise energy at high frequencies. Therefore, it cannot "uncover the zeros" present in



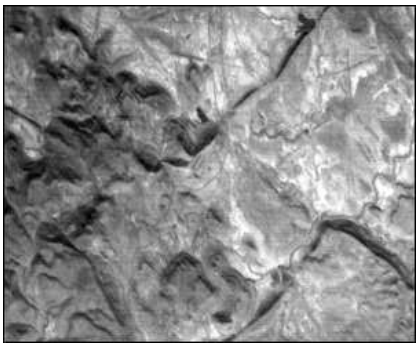
(a) 1420×1330



(b) 2205×1100



(c) 1228×780



(d) 800×653

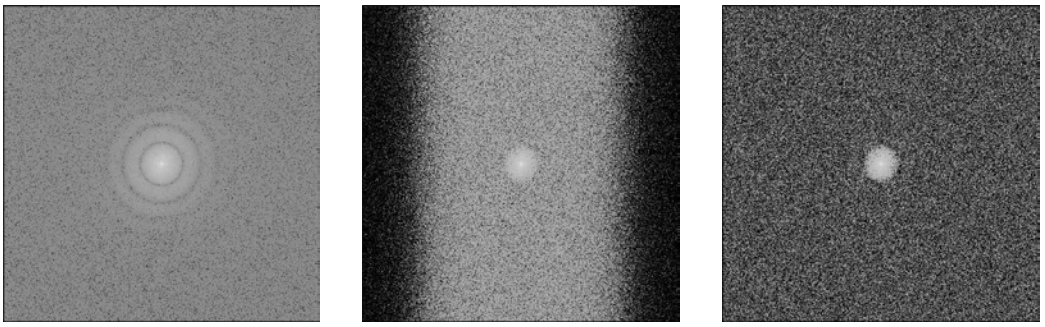
(e) 1554×1556

Figure 4.1: Base images used during experiment, with their resolutions.

Table 4.1: Comparison of defocus blur classification accuracy using 1D and 2D image sections.

Sections	Images with $e_d < 10\%$
1-D	13 (52%)
2-D	25 (100%)

noiseless, blurred power spectrum, 4.2(a), at low frequencies. If the $\hat{P}_n(u,v)$ estimate is averaged over all frequencies prior to subtraction, the location of the first low-frequency zero can be uncovered 4.2(c), greatly improving estimation accuracy.

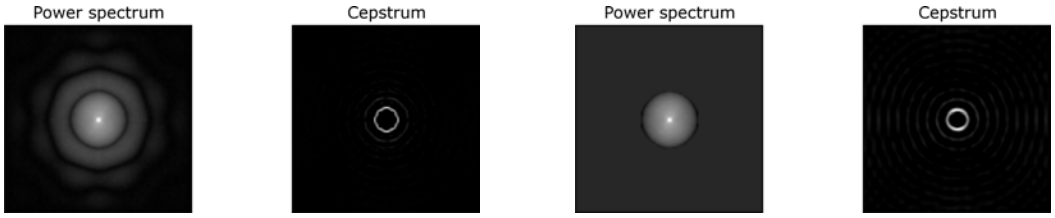


(a) $|A(u,v)|^2$ (b) $|\hat{A}(u,v)|^2$ (c) $|\hat{A}(u,v)|^2$

Figure 4.2: Power spectra estimated by spectral subtraction. In (b), $\hat{P}_n(u,v)$ is estimated by median-complement filter. In (c) that estimate is averaged.

That power spectra similar to 4.2(c) can be used in cepstral blur identification is surprising, since the cepstral peak has been previously assumed to result from radially periodic zeros of the $J_1(Rr)/(Rr)$ function [18, 40, 92]. Although periodic zeros are a requirement for a cepstral peak when using 1-D image strips, it was found that, when using the 2-D cepstrum, periodicity is not a requirement. It does, however, increase the relative height and accuracy of the peak. This was confirmed in an experiment on a blurred image where the periodicity of the power spectrum was removed. First the angular average of the first significant local radial minimum was computed. Then all power spectrum content at radial frequencies greater than the first local minimum was set equal to this average. Figure 4.3 shows the effect of this non-linear low-pass-filter on the power spectrum. Taking the cepstrum of this filtered power spectrum still allowed identification of the defocus blur extent in spite of *no* periodicity in the spectral domain (Figure 4.3). The resulting peak was however slightly shifted (10-15%).

This observation strengthens the case for using 2-D sections when identifying defocus blur. Consequently, 2-D sections are used in the angular smoothing method in section 4.3 as well as the comparison test in section 4.4.2.



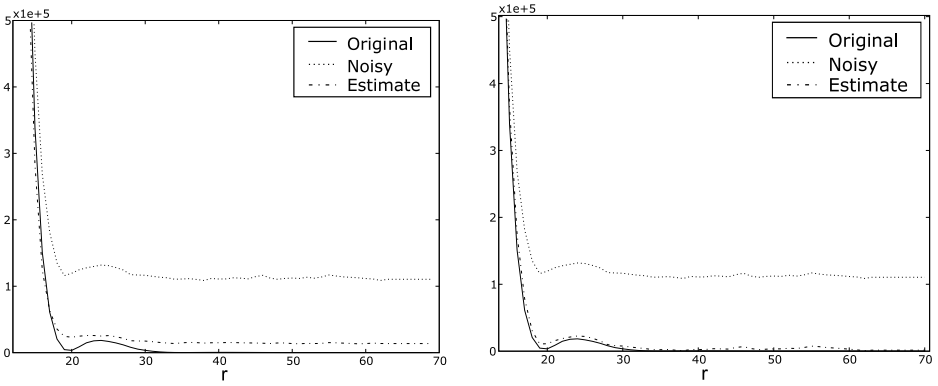
(a) Original (b) Filtered

Figure 4.3: Effect of removing radial periodicity in $P_g(u, v)$ on $C_g(p, q)$. Note that only the zero-clipped negative part of $C_g(p, q)$ is shown: white indicates large negative values and black indicates zero.

4.3 Angular spectral smoothing

4.3.1 Avoiding power spectrum distortion

The clipping element of the spectral subtraction technique is inherent in equation (4.2.8). However, this clipping of negative values to ϵ effectively distorts the shape of the power spectrum: the average of radial minima is increased relative to the rest of the signal. This effect can be seen in Figure 4.4(a). This



(a) (b)

Figure 4.4: Clipping distortion in power spectra. (a) shows the distortion resulting from clipping during spectral subtraction. (b) shows the effect of angular smoothing prior to subtraction.

figure was made by converting $P_g(u, v)$ into polar coordinates $P_g(r, \theta)$ and averaging over θ to give $P_g(r)$. The original blurred, noiseless power spectrum $|A(r)|^2$, the noisy power spectrum $P_g(r)$, and the estimate by spectral subtraction $|\hat{A}(r)|^2$, are shown. The distortion at the first local minimum is clearly visible.

By decreasing the subtraction extent to $(\min(P_g(u, v)) - \epsilon)$, one could avoid this distortion. However, even with the use of Welch's method, the variance

in $P_g(u,v)$ (attributed to variance in $P_n(u,v)$) means that such a restriction severely limits the amount of subtraction and, therefore, noise mitigation possible. This variance is quantified in section 4.3.3. Instead of limiting the spectral subtraction extent, it is attempted to reduce the variance of $P_g(u,v)$ further, prior to subtraction.

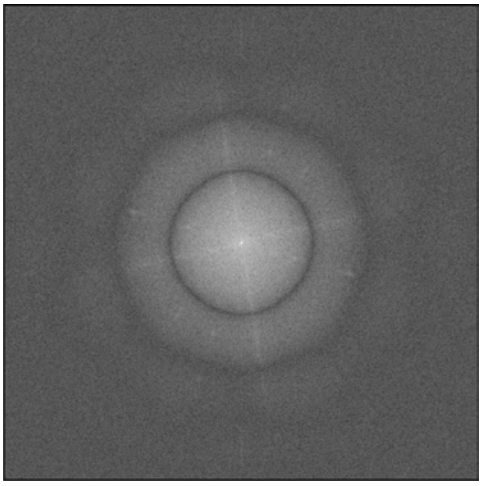
4.3.2 Smoothing procedure

To achieve this reduction in variance, the same Cartesian to polar conversion used to generate the plots in Figure 4.4 is used as a starting point. First $P_g(u,v)$ is estimated according to (4.2.2). It is converted to $P_g(r,\theta)$ using bilinear interpolation [16, p. 248]. $P_g(r,\theta)$ is averaged to $P_g(r)$ to reduce variance. The extent of the variance reduction is quantified in section 4.3.4. $P_g(r)$ values for $r > M/2$, where $M \times M$ is the size in pixels of the sections used, are set equal to the average of $P_g(M/2)$, $P_g(M/2-1)$ and $P_g(M/2-2)$. This is done because values beyond $r = M/2$ map to the corners of $P_g(u,v)$. These values are therefore estimated over fewer angles, making the averaging less reliable. The higher frequencies are also dominated by noise power, which further increases their variance. Next, the 1-D $P_g(r)$ is used as a profile to create a surface of revolution: the 1-D sequence is swept around the polar origin, $r = 0$, of the 2-D space in the θ direction. This process creates a 2-D power spectrum $\overline{P}_g(r,\theta)$, which is angular smoothed: all pixels at the same radius, say $r = k$, have the same value, namely the value of the angular average at radius $r = k$ in $P_g(r,\theta)$. $\overline{P}_g(r,\theta)$ is converted back to Cartesian $\overline{P}_g(u,v)$, using linear interpolation over r. The process is illustrated in Figure 4.5. While the averaging to 1-D reduces variance, use of a surface of revolution enforces circular symmetry on the power spectrum. The inherent circular symmetry of the defocus blur power spectrum is therefore strengthened while features of the image power spectrum $P_f(u,v)$ are further suppressed. This suppression can be seen in Figure 4.5: the diagonal stripes visible in 4.5(a) are image features and are not present in 4.5(b). The effect of the corners on values of $r > M/2 = 400$ can be seen in figures 4.5(c) to (d). The averaging for values of $r > M/2$ has not yet been applied in (c) and (d), to show its necessity.

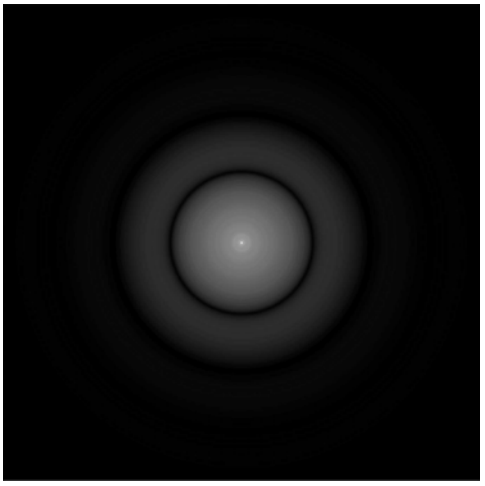
Following angular smoothing, the spectral offset created by $P_n(u,v)$ can be maximally removed in a manner similar to equation (4.2.8), but without the clipping:

$$\hat{P}_a(u,v) = |\hat{A}(u,v)|^2 = \overline{P}_g(u,v) - \min\{\overline{P}_g(u,v)\} + \epsilon. \tag{4.3.1}$$

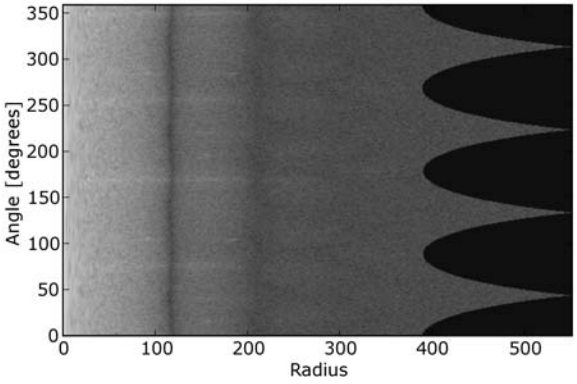
Figure 4.4(b) shows the amount of distortionless spectral subtraction made possible by this technique. Note also that *no* noise estimate $\hat{P}_n(u,v)$ is required. Although replacing the median filter, used to estimate $\hat{P}_n(u,v)$ in the spectral subtraction technique, with a more sophisticated estimator (like those discussed in Chapter 3) might also improve results, a comparative test



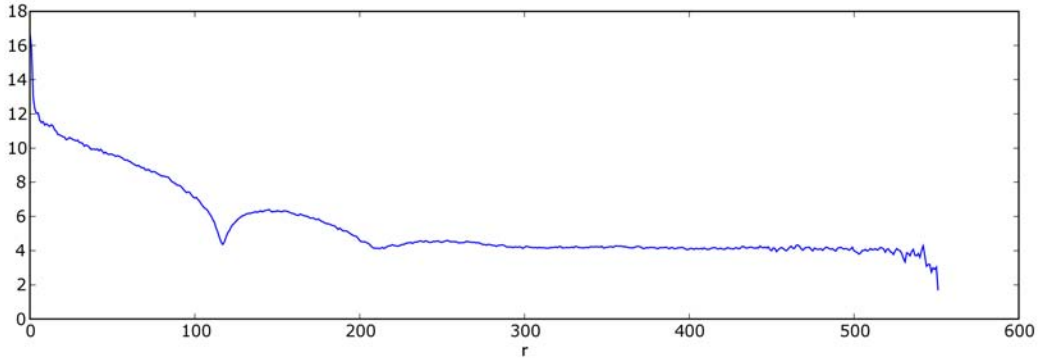
(a) $P_g(u, v)$



(b) $\overline{P}_g(u, v)$



(c) $P_g(r, \theta)$



(d) $P_g(r)$

Figure 4.5: Use of angular smoothing to reduce variance and enforce circular symmetry. Note that, for illustration purposes, $P_g(u, v)$ was estimated from the whole image and not using Welch's method. Furthermore in all images the log is taken to aid visualisation, as is the norm when depicting 2-D power spectra.

has shown that no noise estimator is capable of consistently accurate estimates across a wide range of image types and noise levels (see section 3.1.6). Thus, removing the need for noise estimation increases the robustness of the author's method.

The value of ϵ should ideally be as small as possible to maximise noise mitigation. However, it was found that, especially for noiseless images, choosing ϵ too small disproportionately accentuates the regions of $\overline{P}_g(u,v)$ closest to zero when taking the logarithm. These regions are typically at high frequencies and are not the local minima that must be accentuated. It was found that $0.001 < \epsilon < 0.01$ gives good results. This spectral offset can be interpreted as the offset that would result from Gaussian noise with $0.001 < \sigma_n^2 < 0.01$.

The cepstrum $C_a(p,q)$ is calculated from $\hat{P}_a(u,v)$ and clipped so that only negative values are considered. Except for quantisation effects, this exhibits perfect circular symmetry. To get $C_a(r)$, therefore, a central slice can be used. The postprocessing filter from (4.2.10) can be used on this 1-D sequence to enhance the height of the desired peak. A simple peak picking algorithm identifies the blur radius.

An unexpected advantage of this angular smoothing approach is that in-focus images have a peak at $r = 2$ that can be detected *prior* to comb-filtering $C_a(r)$ (as previously explained, the output of the filter is restricted to $r > 3$). Since image power spectra are generally exponential in shape, the surface of revolution created from a profile in which the corner frequencies are set equal to a constant, is a circular shape with the first radial local minimum at the edge of $\overline{P}_g(u,v)$. This reliably maps to a cepstral sequence similar to the one in Figure 4.12 and is a boon in the context of blind image quality assessment, since in-focus images can be easily identified.

4.3.3 The variance of a noise image's power spectrum estimate

The periodogram is used as an estimate for the power density spectrum during blur estimation in equation (4.2.2). In this section results from [79] are generalised to two dimensions to show that there is significant variance in the periodogram based power spectrum estimate. It is this variance that angular spectral smoothing attempts to remove.

Consider a 2-D finite duration, discrete noise image $n(x,y), 0 < x < L - 1, 0 < y < M - 1$ obtained by sampling a single realisation of a continuous random process $\eta(x_c, y_c)$ at a constant rate in both the x_c and y_c directions. Let $\eta(x_c, y_c)$ be a real, zero-mean, white process with Gaussian probability density function. The additive noise in the degraded image model from equation (4.1.1)

is modelled as $n(x, y)$. The 2-D Fourier transform of $n(x, y)$ is[3]:

$$\mathcal{F}\{n(x,y)\} = N(u,v) = \sum_{x=0}^{L-1}\sum_{y=0}^{M-1} n(x,y)e^{-j2\pi ux}e^{-j2\pi vy},$$

or, using a radial frequency variables $\mu = 2\pi u$ and $\nu = 2\pi v$ for ease of notation:

$$N(\mu,\nu) = \sum_{x=0}^{L-1}\sum_{y=0}^{M-1} n(x,y)e^{-j\mu x}e^{-j\nu y}.$$

The periodogram is then defined as:

$$\begin{aligned} P_n(\mu,\nu) &= \frac{1}{LM}|N(\mu,\nu)|^2 \\ &= \frac{1}{LM}\sum_{k=0}^{L-1}\sum_{l=0}^{M-1}\sum_{x=0}^{L-1}\sum_{y=0}^{M-1} n(k,l)n(x,y)e^{j(\mu k+\nu l)}e^{-j(\mu x+\nu y)}. \end{aligned} \tag{4.3.2}$$

The periodogram is an asymptotically unbiased estimate of the true power density spectrum:

$$\lim_{L,M\to\infty} E[P_n(\mu,\nu)] = \Gamma_n(\mu,\nu),$$

where the true power density spectrum, $\Gamma_n(\mu,\nu)$, of the stationary random process $\eta(x, y)$ must be calculated by Wiener-Khintchine theorem:

$$\Gamma_n(\mu,\nu) = \int_{-\infty}^{\infty} E[\eta(x_c,y_c),\eta(x_c+\tau_1,y_c+\tau_2)]e^{-j\mu x_c}e^{-j\nu y_c}dx_c dy_c,$$

since $\eta(x_c, y_c)$ does not have finite energy and therefore does not have a Fourier transform. The unbiased nature of the periodogram estimate is proven in [84, pp. 902–905] for 1-D signals and is easily generalised for the 2-D case.

However, while unbiased, the periodogram is not a consistent estimate of the true power density spectrum; the variance of the estimate does not converge to zero:

$$\lim_{L,M\to\infty} \text{var}[P_n(\mu,\nu)] \neq 0.$$

Therefore the periodogram does not converge to the true power density spectrum.

To derive an expression for the variance,

$$\text{var}[P_n(\mu,\nu)] = E[|P_n(\mu,\nu)|^2] - |E[P_n(\mu,\nu)]|^2, \tag{4.3.3}$$

[3]The Fourier transform of an aperiodic function implies continuous frequency variables, u, v, μ and ν.

equation (4.3.2) is substituted into the first term of equation (4.3.3), using the shortened notation $\sum_{a,b=0}^{L} \equiv \sum_{a=0}^{L}\sum_{b=0}^{L}$:

$$E[|P_n(\mu,\nu)|^2] = \frac{1}{L^2M^2}\times \sum_{k,x,p,r=0}^{L-1}\sum_{l,y,q,s=0}^{M-1} E[n(k,l)n(x,y)n(p,q)n(r,s)]e^{j\mu(k-x-p+r)}e^{j\nu(l-y-q+s)}. \quad (4.3.4)$$

For zero mean, jointly Gaussian random variables, X_1, X_2, X_3, X_4 it can be shown that

$$E[X_1,X_2,X_3,X_4] = E[X_1,X_2]E[X_3,X_4]+ E[X_1,X_3]E[X_2,X_4] + E[X_1,X_4]E[X_2,X_3].$$

Therefore

$$E[n(k,l)n(x,y)n(p,q)n(r,s)] = E[n(k,l)n(x,y)]E[n(p,q)n(r,s)]+ E[n(k,l)n(p,q)]E[n(x,y)n(r,s)] + E[n(k,l)n(r,s)]E[n(x,y)n(p,q)],$$

which, for white noise reduces to:

$$E[n(k,l)n(x,y)n(p,q)n(r,s)] = \begin{cases} & k=x, l=y, p=r \text{ and } q=s \\ \sigma_n^4 & \text{or } k=p, l=q, x=r \text{ and } y=s \\ & \text{or } k=r, l=s, x=p \text{ and } y=q \\ 0 & \text{otherwise.} \end{cases} \quad (4.3.5)$$

Substituting equation (4.3.5) into (4.3.4) yields:

$$\begin{aligned} E[|P_n(\mu,\nu)|^2] &= \frac{\sigma_n^4}{L^2M^2}\left\{\sum_{x,r=0}^{L-1}\sum_{y,s=0}^{M-1} 1 + \sum_{p,r=0}^{L-1}\sum_{q,s=0}^{M-1} 1+ \sum_{p,r=0}^{L-1}\sum_{q,s=0}^{M-1} e^{j2\mu(r-p)}e^{j2\nu(s-q)}\right\} \\ &= \frac{\sigma_n^4}{L^2M^2}\left\{2L^2M^2+ \sum_{p=0}^{L-1}[e^{-j2\mu}]^p\sum_{r=0}^{L-1}[e^{j2\mu}]^r\sum_{q=0}^{M-1}[e^{-j2\nu}]^q\sum_{s=0}^{M-1}[e^{j2\nu}]^s\right\}. \end{aligned} \quad (4.3.6)$$

Using the geometric series expansion:

$$\sum_{p=0}^{L-1} a^p = \begin{cases} L, & a=1 \\ \frac{1-a^L}{1-a} & a\neq 1 \end{cases},$$

one can simplify the factors of the second term in equation (4.3.6):

$$\begin{aligned}\sum_{p=0}^{L-1}[e^{-j2\mu}]^p \sum_{r=0}^{L-1}[e^{j2\mu}]^r &= \left(\frac{1-e^{-j2\mu L}}{1-e^{-j2\mu}}\right)\left(\frac{1-e^{j2\mu L}}{1-e^{j2\mu}}\right) \\ &= \frac{2-(e^{-j2\mu L}+e^{j2\mu L})}{2-(e^{-j2\mu}+e^{j2\mu})} \\ &= \frac{2-2\cos(2\mu L)}{2-2\cos(2\mu)} \\ &= \left(\frac{\sin(\mu L)}{\sin(\mu)}\right)^2,\end{aligned}$$

which simplifies equation (4.3.6) to:

$$E[|P_n(\mu,\nu)|^2] = \sigma_n^4\left\{2+\left(\frac{\sin(\mu L)\sin(\nu M)}{LM\sin(\mu)\sin(\nu)}\right)^2\right\}. \tag{4.3.7}$$

Substituting equation (4.3.7) into equation (4.3.3) and using $E[P_n(\mu,\nu)] = \sigma_n^2$ the expression for variance is:

$$\text{var}[P_n(\mu,\nu)] = \sigma_n^4\left\{1+\left(\frac{\sin(\mu L)\sin(\nu M)}{LM\sin(\mu)\sin(\nu)}\right)^2\right\}. \tag{4.3.8}$$

From equation (4.3.8) one can see that the variance of the power density spectrum estimate is of order σ_n^4. Furthermore increasing the image size to infinity does not reduce this variance to zero:

$$\lim_{L,M\to\infty} \text{var}[P_n(\mu,\nu)] = \sigma_n^4.$$

This result can be generalised for a nonwhite Gaussian process by approximating the nonwhite random sequence as the output of a linear system excited by white noise. The squared magnitude of frequency response of the linear system is selected to be equal to the power density spectrum of the random process, $\Gamma_n(\mu,\nu)$. An argument for the 1-D case is presented in [79] and is also valid for the 2-D case. The variance of the periodogram is:

$$\text{var}[P_n(\mu,\nu)] = \Gamma_n^2(\mu,\nu)\left\{1+\left(\frac{\sin(\mu L)\sin(\nu M)}{LM\sin(\mu)\sin(\nu)}\right)^2\right\}, \tag{4.3.9}$$

which shows that in general, for the nonwhite case, the periodogram is still not a consistent estimate of power spectrum density.

4.3.4 Reducing the variance of the power spectrum estimate

The power spectrum estimate, $P_n(u,v)$, of a discrete Gaussian white noise image $n(x,y)$ has a mean $E[P_n(u,v)] = \sigma_n^2$ and a variance $\text{var}[P_n(u,v)] \approx \sigma_n^4$. The effect of angular averaging on these values is investigated in this section.

Since the derivation in section 4.3.3 used the Fourier transform, a finite, aperiodic, discrete $n(x,y)$ implied a continuous (and periodic) $P_n(u,v)$. In practice, the discrete Fourier transform is used, which results in a discrete $P_n(u,v)$ (and a periodic $n(x,y)$). To emphasize the discrete nature of the periodogram, it is the convention of signal processing literature to use alternate variables, such as $P_n(k,l)$ where the periodogram is sampled at discrete frequencies $u_k = k/L$ and $v_l = l/M$ and one period of $n(x,y)$ has size $L \times M$ as usual. However, the convention in this dissertation (section 1.4), which is common to image processing literature, is to assume 2-D images are discrete unless otherwise noted. Outside of section 4.3.3, u and v will continue to refer to discrete variables. The choice of Fourier transform over discrete Fourier transform for the derivation in section 4.3.3 is simply for notational convenience.

Consider the discrete power spectrum estimate $P_n(u,v)$ of a square noise image ($L = M \Rightarrow n(x,y)$ is size $M \times M$), which is converted to polar coordinates $P_n(r,\theta)$, $0 < r < M/2 - 1$, $0 < \theta \leq 359, \theta \in \mathbb{Z}$. Averaging over θ can be written as:

$$\begin{aligned} P_n(r) &= \frac{1}{360} \sum_{\theta_i=0}^{359} P_n(r,\theta_i) \\ &= \frac{1}{180} \sum_{\theta_i=0}^{179} P_n(r,\theta_i), \end{aligned} \tag{4.3.10}$$

since the power spectrum is symmetric with respect to the polar origin.

The expected value of the angular averaged periodogram is unaffected:

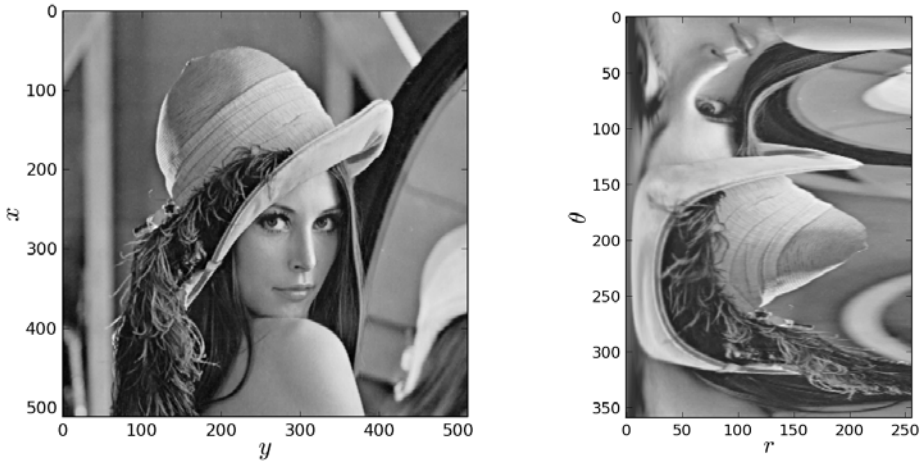
$$\begin{aligned} E[P_n(r)] &= E\left[\frac{1}{180} \sum_{\theta_i=0}^{179} P_n(r,\theta_i)\right] \\ &= \frac{1}{180} \sum_{\theta_i=0}^{179} E[P_n(r,\theta_i)] \\ &= \frac{1}{180} \sum_{\theta_i=0}^{179} \sigma_n^2 \\ &= \sigma_n^2. \end{aligned} \tag{4.3.11}$$

If it is assumed for the moment that $P_n(r,\theta_i)$ is uncorrelated with $P_n(r,\theta_j)$

for $i \neq j$, then the variance of the angular averaged periodogram is:

$$\begin{aligned} \text{var}[P_n(r)] &= \text{var}\left[\frac{1}{180}\sum_{\theta_i=0}^{179} P_n(r,\theta_i)\right] \\ &= \frac{1}{180^2}\sum_{\theta_i=0}^{179} \text{var}[P_n(r,\theta_i)] \\ &\approx \frac{1}{180^2}\sum_{\theta_i=0}^{179} \sigma_n^4 \\ &= \frac{\sigma_n^4}{180}. \end{aligned} \tag{4.3.12}$$

Thus the variance appears to be reduced by the number of discrete uncorrelated samples in the average, from σ_n^4 to $\sigma_n^4/180$. In practice the $P_n(r,\theta_i)$ and $P_n(r,\theta_j)$ are not uncorrelated since converting from Cartesian to polar coordinates requires interpolation, which implies samples at neighbouring θ values contain shared information. This is most noticeable at small values of r since few pixels close to the centre of the image (the origin of the polar coordinate system) are used for many values of θ, as shown in Figure 4.6.



(a) Cartesian coordinates.

(b) Polar coordinates.

Figure 4.6: Neighbouring pixels close to the origin $r = 0$ of the polar coordinate system are highly correlated.

Another way of thinking about the correlation is from a sampling perspective. Sampling occurs at regular intervals in Cartesian coordinates. On the other hand, sampling in polar coordinates implies denser sample spacing closer

to the polar origin. Since the samples are regularly spaced in Cartesian coordinates, the extra samples needed in high density polar areas must be derived by interpolation.

The true number of discrete uncorrelated samples used in the angular average varies with distance from the polar origin: at a certain radius r it is half of the number of Cartesian samples touching a circle with radius r. Close to the polar origin constant values of r describe small circles in Cartesian coordinates, while further from the polar origin, larger values of r describe larger circles. Only half the samples at radius r are uncorrelated because of the symmetry of the power spectrum. The number of samples touching a circle of radius r is approximately equal to the circumference of the circle:

$$(\text{Cartesian samples at radius } r) \approx 2\pi r.$$

Replacing 180 in equation (4.3.12) with the approximate true number of uncorrelated samples yields:

$$\text{var}[P_n(r)] \approx \frac{\sigma_n^4}{\pi r}. \tag{4.3.13}$$

The validity of equation (4.3.13) was tested by generating 100 unit variance Gaussian noise images (each of size 100×100). The angular average of each noise image's periodogram was calculated. Finally the variance of the 100 angular averages was computed as a function of r and compared to the variance predicted by equation (4.3.13). As Figure 4.7 shows, the results are similar.

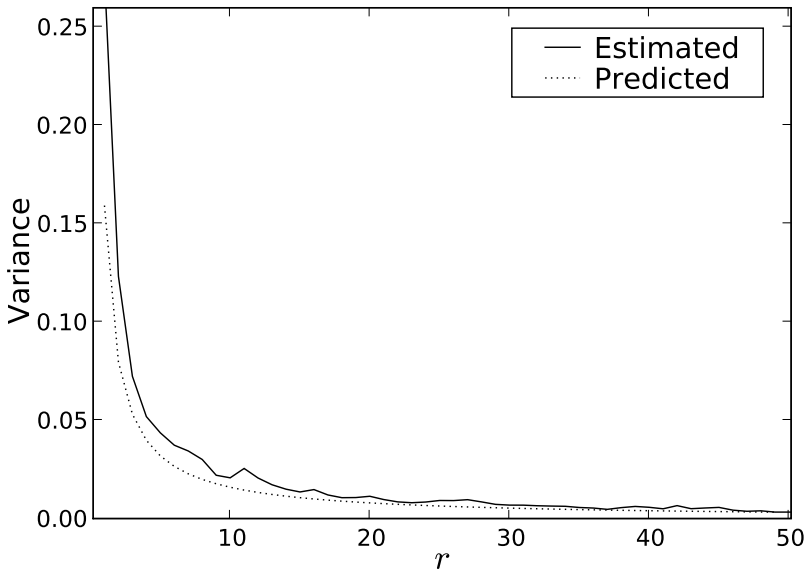


Figure 4.7: Variance of angular averaged periodogram predicted by equation (4.3.13) agrees with estimated variance.

As described in equation (4.3.1), this reduction in variance allows spectral subtraction to remove the offset caused by noise, $E[P_n(u,v)] = \sigma_n^2$, without distorting the power spectrum at local minima of $P_g(u,v)$. Angular spectral smoothing can be categorised as a nonparametric method for power spectrum estimation [47, pp. 908–920], similar to Welch's method, but with narrower applicability. Welch's method reduces the variance of the power spectrum estimate at cost of resolution. Since the data must be divided into sections for averaging, spectral resolution is lowered from the size of the data to the size of the segment. Resolution is further limited by spectral leakage introduced by windowing at the section-level. Angular spectral smoothing reduces variance by discarding angular information, while maintaining radial resolution. Since the power spectrum to be determined, namely $P_h(u,v)$ the power spectrum of the blurring function, possesses circular symmetry, it is desirable to maintain radial resolution while angular information is superfluous.

4.3.5 Estimate confidence

Any deconvolution technique will fail, given a low enough SNR, and concerns have been expressed about the sensibility of using blind (and hence unreliable) image quality assessment in a scientific environment [46]. Therefore, during blur identification, it is useful to have an indication of the confidence in the blur estimate; in a fully automated, scientific environment images should only receive a bad score if it can be stated with certainty that they are out of focus. $C_a(r)$ will always have some maximum. At low noise this maximum is at a prominent peak 2R from the origin. However, as SNR decreases, spurious peaks appear and eventually dominate $C_a(r)$. The concept of relative energy, E_r, is introduced to enable differentiation between true and spurious peaks:

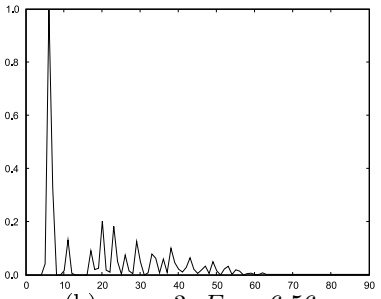
$$\begin{aligned} E_r &= \frac{E_{peak}}{E_{rest}}, \\ E_{peak} &= \sum_{i \in P} (C_a(i))^2, \\ P &= \{r | r_p - 1 \leq r \leq r_p + 1\}, \\ r_p &= \text{argmax}[C_a(r)], \\ E_{rest} &= \sum_{i \in Q} (C_a(i))^2, \\ Q &= \{r | r \notin P, 0 \leq r \leq r_{max}\}. \end{aligned} \tag{4.3.14}$$

The relative energy in the peak of the cepstral sequence $C_a(r)$ is computed. It can be tested against a threshold relative energy level and, if it is too low, the identification is probably wrong and the results should not be trusted; otherwise the defocus blur estimate is $\hat{R} = r_p/2$. For an image (Figure 4.1(a) from page 97) blurred with $R = 3$ ($r_p = 6$), Figure 4.8 shows the effect of adding noise with an increasing standard deviation, σ_n, on E_r. 4.8(b) and

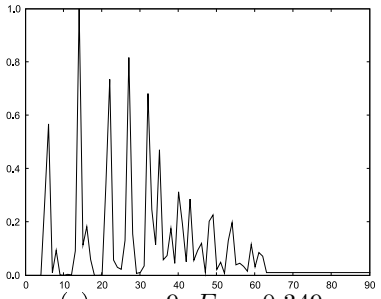
4.8(c) show $C_a(r)$: the prominent, correct peak in (b) results in high E_r, while spurious peaks in (c) result in low E_r.

σ_n	E_r
0	206
1	125
3	6.56
5	6.48
7	0.662
9	0.349
20	0.334

(a)



(b) $\sigma_n = 3$, $E_r = 6.56$



(c) $\sigma_n = 9$, $E_r = 0.349$

Figure 4.8: Spurious peaks dominate at higher σ_n and result in lower E_r.

When using Cannon's original cepstral method or the bicepstral method, low E_r is also typical of images that have no blur. Distinguishing in-focus and out-of-focus images is important for blind image quality assessment. This is discussed further in section 4.5.

4.4 Experiments

4.4.1 Choice of windowing function

As mentioned in section 4.2.2, the specific window type was not specified in the description of the cepstral method [18]. In addition to the rectangular window (the effective window when data is sectioned and no other window applied), Hamming and Hanning windows were also investigated [84, pp. 624–627].

In the spatial domain, the rectangular window has abrupt discontinuities that give its frequency response high sidelobes. These sidelobes commonly result in undesirable ringing effects in the frequency domain when data is sectioned with a (implicit) rectangular window. The classical solution is to multiply the sectioned data with a windowing function that has less severe discontinuities prior to taking the Fourier transform. Commonly used windowing function are the Hamming and Hanning windows. These functions result in smaller frequency domain sidelobes, but the cost is an increased width of the frequency domain main lobe. The windowing function's increased main lobe width can result in undesirable smoothing of data's frequency response. Figure 4.4.1 shows the (spatial domain) 1D windows for a data segment of length M.

It was found that the spectral leakage, or ringing effect, introduced by the high frequency domain sidelobes of the rectangular and Hamming windows produce unwanted artefacts in the cepstrum that could obscure the observation of the negative cepstral peak. The low sidelobes of the Hanning window

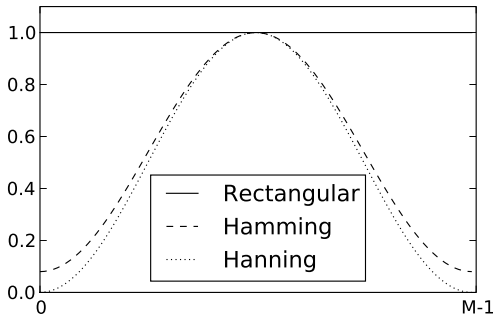
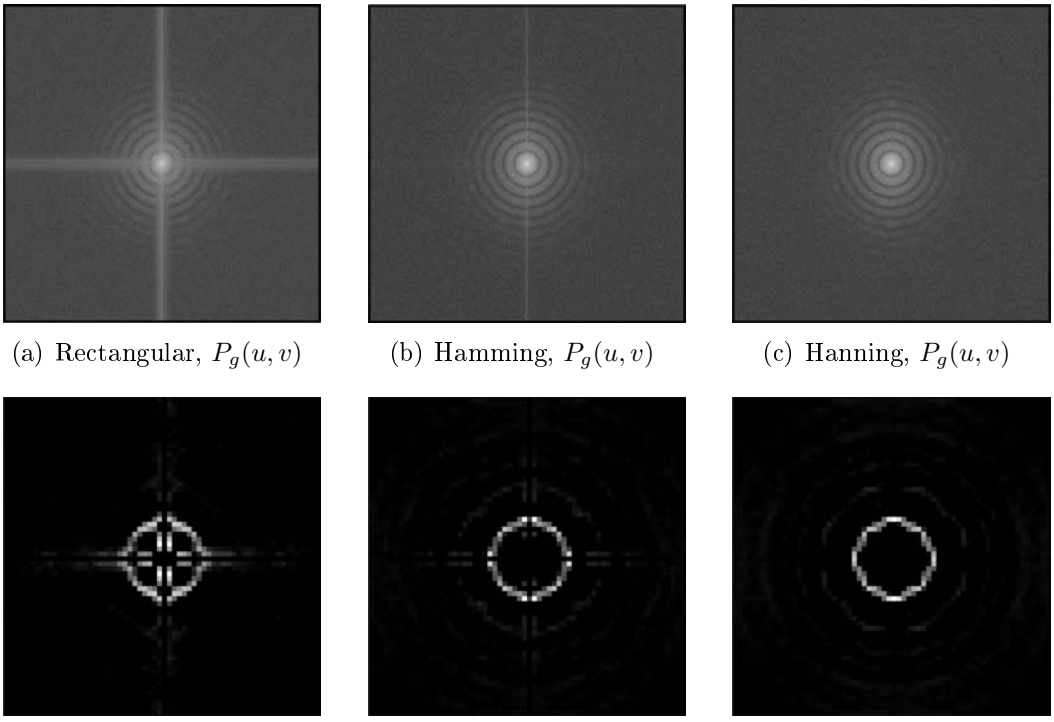


Figure 4.9: Different window types for data of length M.



(a) Rectangular, $P_g(u, v)$ (b) Hamming, $P_g(u, v)$ (c) Hanning, $P_g(u, v)$

(d) Rectangular, $C_g(p, q)$ (e) Hamming, $C_g(p, q)$ (f) Hanning, $C_g(p, q)$

Figure 4.10: The effect of window function on the power spectrum and cepstrum.(a) to (b) show the spectral leakage in the power spectrum, which is not visible in (c). (d) to (e) depicts the corresponding cepstra.

produced no such artefacts and this window was therefore selected. Figure 4.10 shows the effect of the different window functions on the frequency domain and the negative part of the cepstral domain. Note that, for illustration proposes, the whole image is considered and the cepstral domain views are zoomed.

The disadvantage of using the Hanning window is that its frequency re-

sponse has a wider main lobe, resulting in a smoothed frequency domain response. This will affect the observability of large blur PSFs: large R values in the spatial domain decrease the period width of the $J_1(Rr)/(Rr)$ frequency domain response, making it more susceptible to smoothing.

The solution is simply to choose the window size large enough, since this decreases the width of the window function's frequency domain main lobe. Fortunately, there is ample data available and, therefore, the only disadvantage of increased window size is an increased execution time. 128×128 pixel windows were chosen. These are more than wide enough to observe severe blurring such as occurs when $R = 15$.

4.4.2 Comparative experiment

The performances of the angular smoothing method and existing spectral based, direct methods were evaluated in a comparative experiment on a range of images. The five base images from Figure 4.1 were chosen, since they represent a wide variety of remote sensing image types. Figures (a), (b) and (d) have low spatial detail and varying degrees of edge gradient levels, (e) has sharply defined edges typical of coastal regions and (c) has high spatial detail typical of city scenes. All images are 8-bit greyscale with resolutions shown in the figure. These images where blurred with defocus radii $R = \{0, 2, 3, 4, 9, 15\}$ and white, zero mean, Gaussian noise was added with standard deviations $\sigma_n = \{0, 1, 3, 5, 7, 9, 20\}$, resulting in a total of 210 images.

The cepstral, bicepstral, spectral subtraction and angular smoothing methods were all used to identify the defocus blur radius. For the spectral subtraction method, $P_g(u, v)$ was estimated from the whole image and $\alpha = 1$ was used, as recommended in [40]. For the other methods, 128×128 pixel sections were averaged according to (4.2.2) and (4.2.6) to estimate $P_g(u, v)$ and $B_g(u, v; 0, 0)$ respectively. In all cases the resulting 2-D (bi)cepstral sequence was averaged to 1-D, to facilitate peak picking. The comb-filter postprocessing was applied in all cases except for the basic cepstral technique. After peak picking, E_r thresholds for each method were varied from minimum to maximum using deciles (10 equally spaced E_r indices were chosen from a sorted list of output E_r values). Based on these E_r thresholds images were separated into "classified" and "unclassified" groupings, illustrated in Figure 4.11 for the angular smoothing method. The number of classified images is not zero for maximum E_r, because in-focus images were classified differently.

Images that were possibly in focus were classified using special rules, since none of the methods (barring spectral subtraction, discussed below) gives $r_p = 0$ for in-focus images. Since the comb-filtering technique zeros all values for $r < 4$, identification of in-focus images was done prior to comb-filtering. Given an in-focus image, cepstral sequences for the different methods are shown in Figure 4.12. The bicepstrum typically gives results similar to the cepstrum for in-focus images: neither method gives a peak at a characteristic location,

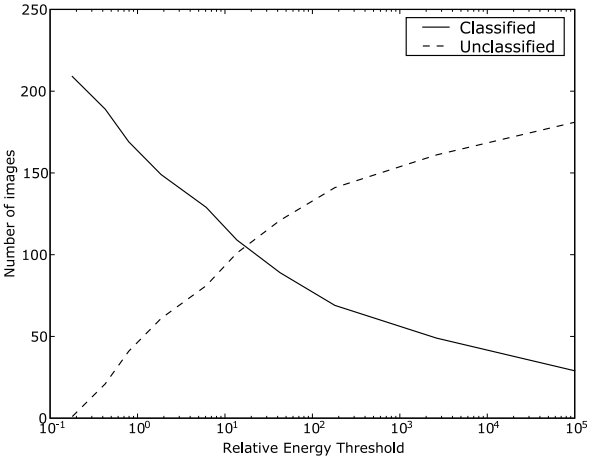


Figure 4.11: Effect of varying E_r on number of classification.

but the E_r level is normally low. As already discussed, the angular smoothing method results in a peak at $r = 2$. Images processed using the cepstral, bicepstral and angular smoothing methods with $r_p \leq 2$ were assumed to be in focus, irrespective of E_r level. In the author's experience, the spectral subtraction method has a large peak at $r = 0$ for *all* image types (both in-focus and blurred). This can be explained by looking at Figure 4.2(c): clipping a large part of the power spectrum to the same $\epsilon < 1$ value results in a large low frequency (quefrency) component in $C_a(p, q)$ that is negative, since $\log(\epsilon) < 0$. Therefore, identification by spectral subtraction could only be done after comb-filtering, which restricts the range of the output cepstral sequence to $4 \leq r \leq r_{max}$. Consequently, images processed with the spectral subtraction

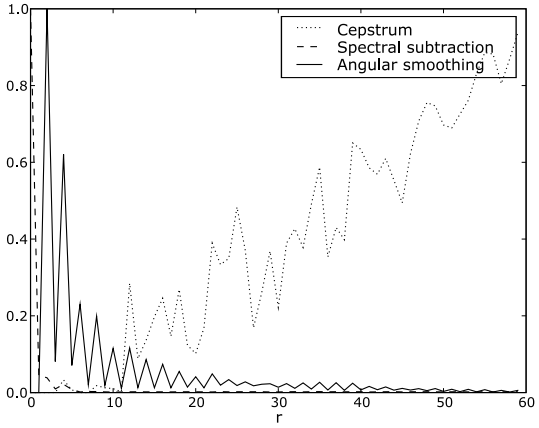


Figure 4.12: Normalised cepstral sequences for an in-focus image prior to comb filtering.

method were assumed in-focus if $r_p = 4$.

For each classification, the error distance was calculated according to (4.2.11) and averaged across all classifications for a given method. If $R = 0$ and $\hat{R} \neq 0$, $e_d = 100\%$, which corresponds to an in-focus image, incorrectly classified as out-of-focus. Since the number of classifications vary with E_r threshold, methods are compared using average e_d against number of classified images.

4.4.3 Effect of reduced dynamic range

As previously mentioned, sensor calibration and atmospheric effects often cause remote sensing images to have reduced dynamic range. Since the algorithm will likely be applied to images with reduced dynamic range, its effect on the algorithm performance was observed.

Starting with an image with full dynamic range, the dynamic range was reduced by subtracting the mean, multiplying the result with a fraction and adding the mean again, as shown in Figure 4.13. The images were blurred with PSF with extent $R = 4$, and cepstral response evaluated. The results are discussed in section 4.5.2.

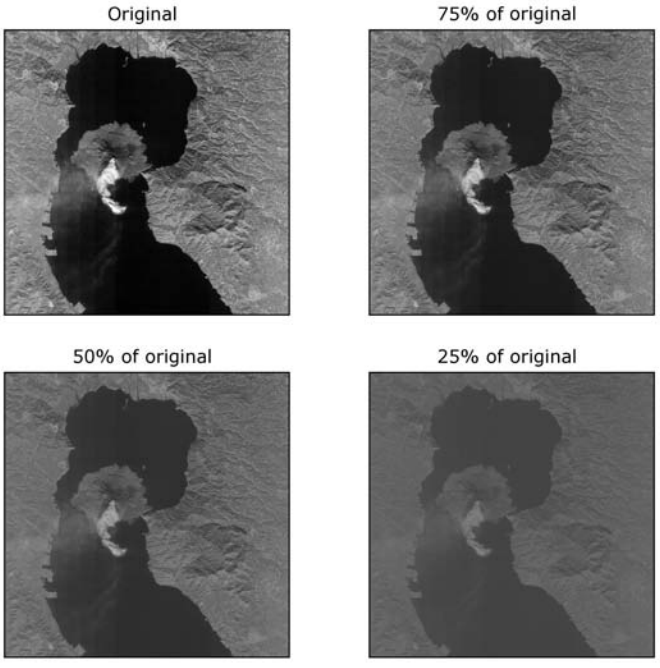


Figure 4.13: An image with successively reduced dynamic range.

4.4.4 Embedded evaluation

All algorithms were implemented using Python. The performance feasibility of the angular smoothing method was evaluated by implementing it in C on

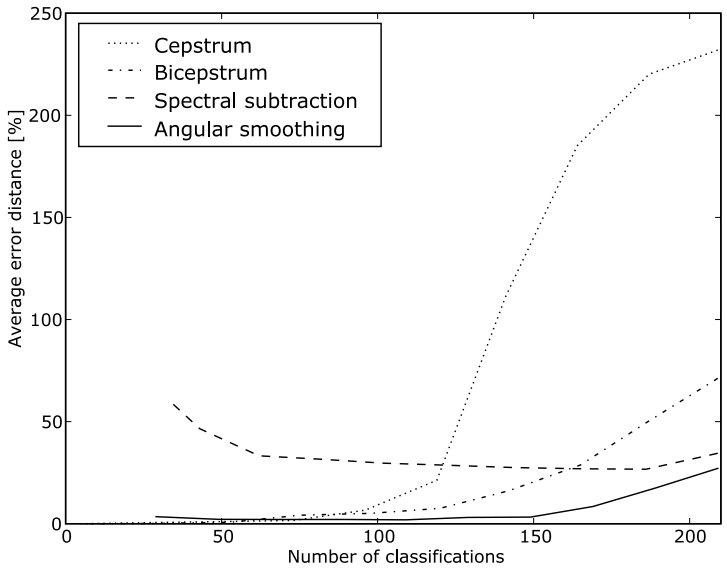
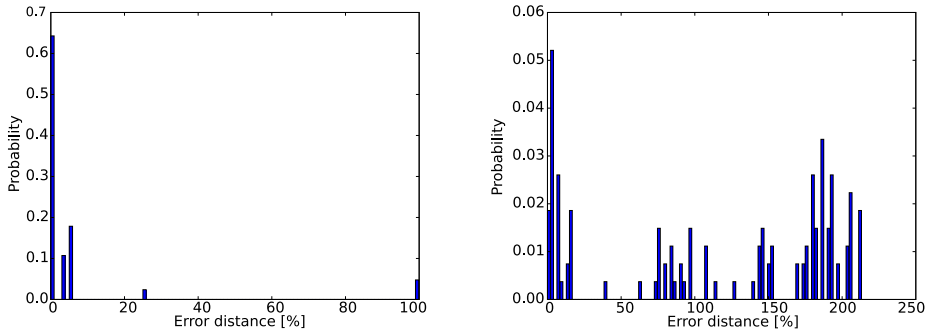


Figure 4.14: Comparison between direct blur identification techniques.



(a) Classified images (b) Unclassified images

Figure 4.15: Average errors in an example classified-unclassified split based on E_r.

the same embedded SH4 hardware used to test the noise algorithm in section 3.2.1. In addition to converting the existing code, the fast Fourier transform had to be implemented.

Documentation is provided in Appendix C.

4.5 Results

4.5.1 Comparative results

Figure 4.14 shows the result of the comparative test. As discussed in section 4.4.2, for each method deciles were selected from its range of E_r values, and

these values used to separate classified from unclassified images. Since different E_r values were applied to different methods, the horizontal axis in Figure 4.14 is "number of classifications" instead of "E_r" to allow for direct comparison. For small number of classifications (the left hand side of Figure 4.14), the strict E_r thresholds allowed all methods to classify only very low noise images. Conversely, on the right hand side of the graph no images are rejected; all methods had to attempt blur estimation on many images with high levels of noise. The usefulness of the E_r measure is clear from the fact that the classification accuracy generally increases with decreasing number of classifications, corresponding to decreasing noise levels. It is confirmed by the histograms in Figure 4.15, that show the errors for an example classified-unclassified split of the cepstral method. The unclassified images have greater errors. The trade-off is that, to achieve higher accuracy, more images have to be rejected. A way to select an appropriate level for E_r is discussed in section 5.3.8 on page 158.

First consider the right side of the graph in Figure 4.14 where all 210 images are classified. The results confirm the high noise sensitivity of the cepstral method to additive noise; the many noisy images present result in a large average e_d of 232%. The bicepstral method shows a big improvement with average e_d of 71%, while the spectral subtraction and angular smoothing give $e_d = 36\%$ and $e_d = 29\%$ respectively. As the number of classifications decrease, e_d generally improves, except in the case of spectral subtraction. The increase in this case is caused by the inability of the method to distinguish between in-focus images and images blurred with $R = 2$ ($r_p = 4$), which is not affected by increasing the E_r threshold. Instead these incorrectly classified images with $e_d = 100\%$ just start to make up a bigger portion of the total classifications, resulting in an increase in average e_d. Many in-focus images were also classified as having $r_p = 5$, 6 or 7 and generally the method has lower accuracy when used with small blur radii.

The significant area of the graph in Figure 4.14 is the right hand side, as will be justified below. The bi-cepstrum, spectral subtraction and angular smoothing methods are adaptations of the cepstral method aimed at improving robustness against noise. Therefore:

- tests on the complete set, including noisy images (right hand side of the graph), are required to differentiate between them.
- when no noisy images are present (left hand side of the graph), all methods should yield indistinguishable, correct results, as is the case in Figure 4.14.

Depending on the application, the rules for in-focus classification and the use of $R = 2$ as one of the defocus blur radii might seem to tip the advantage unfairly in the direction of angular averaging. As an additional test, all-in focus images and images classified as in-focus were discarded and the same plots generated. Figure 4.16 shows the results. The maximum number of

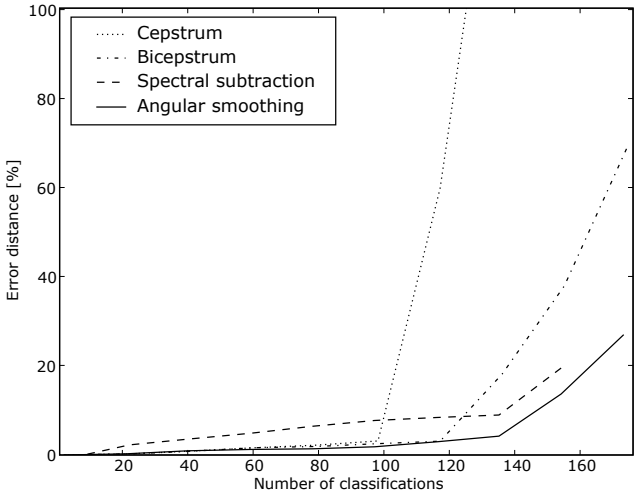


Figure 4.16: Comparison results when in-focus images and classifications are discarded.

classifications achieved with the spectral subtraction technique is in this case fewer than the others, since more images had to be discarded. The angular spectral smoothing technique still compares favourably.

These results confirm the usefulness of the angular smoothing technique, especially in the context of blind image quality assessment.

4.5.2 Effect of reduced dynamic range

The cepstral sequences corresponding to images with reduced dynamic range, Figure 4.13, are shown in Figure 4.17. While the accuracy of the estimate is not affected, the relative energy in the peak is reduced. This increases the susceptibility of the estimate to the effects of additive noise. The usefulness of a defocus estimation algorithm that is robust against noise is emphasised.

4.5.3 Feasibility of embedded implementation

The embedded C implementation of the angular smoothing method was tested on images of various sizes. For each image size the algorithm was repeated 10 times and the average execution time recorded. The results are shown in Figure 4.18. The time taken for each of the 10 runs at a given size was almost identical.

The slowest part of the algorithm is the computation of the fast Fourier transform, which has $O(n \log n)$ complexity. The 6 megapixel image had dimensions 2449×2449 and took 128,6 seconds. The slight bump in the graph at 2 megapixels is caused by the choice of section size. The images were divided

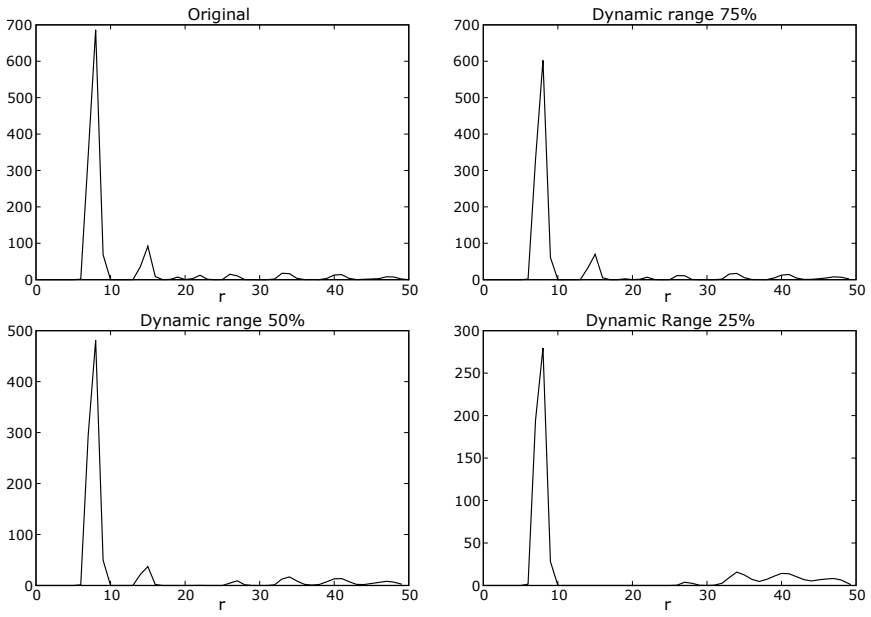


Figure 4.17: The effect of reducing the dynamic range on $C_a(r)$.

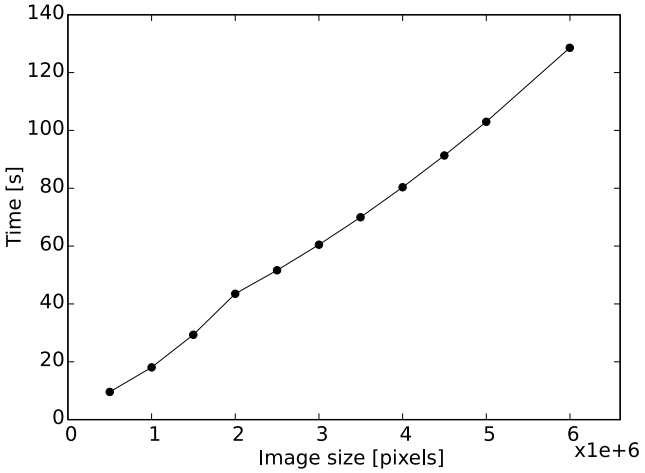


Figure 4.18: The execution time of the embedded angular smoothing implementation.

into 128×128 pixel sections, but the images' sizes were not integer multiples of 128. The edges that remained were discarded. At 2 megapixels (image size 1414×1414) these edges were the smallest; therefore less image content was discarded than for neighbouring images.

While the execution time is longer than the noise estimation algorithm's, it is still reasonable. Furthermore, the execution time can be limited, as is discussed in the conclusion.

4.6 Conclusion

Across 210 test images, the angular smoothing method gave the best average classification error of all the direct, spectral based, defocus blur identification methods evaluated. The increased robustness of the method can be ascribed to the fact that no noise estimate is needed, as well as the reinforcement of the circular structure typical of defocus blur. The method's characteristic response to in-focus images represents a further improvement. This makes it especially suitable for use in situations where examined images may be either in-focus or out-of-focus, such as during blind image quality assessment. The usefulness of the relative energy threshold was proven for situations where certainty about the estimate accuracy is required.

If the images are subject to geometric distortion due to satellite instability, it is likely that the method will fail. It relies on a spatially invariant, parametric model (4.1.1), which will be invalidated when the linear sensor moves in a non-uniform manner. If the satellite instability proves to be a greater problem than sensor noise, it might be preferable to use a method that works when image segments are 1-D strips instead of 2-D blocks. As discussed in section 4.2.3, defocus blur identification benefits greatly from 2-D blocks, but if satellite instability is a big problem, the advantages of increased performance accuracy and robustness against noise will have to be weighed against potential inability to estimate blur in conditions of non-uniform sensor motion.

The algorithm has been shown to be practically feasible for embedded use. Since the blurring is spatially uniform, it is not necessary to evaluate the entire image; a section of the image is sufficient. Increasing the size of this section reduces the effect of the image structure on the variance of $P_g(u, v)$, because the averaging of Welch's method, equation (4.2.2), is carried out over more subsections.

Although a square section was used in the experiment in section 4.4.4, this is not recommended. Instead the entire sensor swath-width should be used and the section size varied in the along track direction. Keeping the along track section size small, for example using a 7800×256 section, where 7800 is the swath-width, will minimise the effect of any non-uniform sensor motion that might be present. This section size will still allow for 120 128×128 subsections for the averaging in (4.2.2), which is sufficient [18]. Since this section is less

than 2 megapixels, the estimation algorithm will be executed speedily.

The PSF of a satellite's optical system will vary from channel to channel. However, these variations are fixed according to the design of the optical system. Furthermore, telescope defocus will affect all channels equally. It is therefore recommended that a single representative channel be selected as input.

A paper based on the work presented in this chapter has been published in an international journal [72].

Chapter 5

Quality assessment model

5.1 Introduction

Given various degradation measures, the question arises: how does one combine these features into a single quality score? While one could construct any model, it is desirable to have a scientific justification for choices regarding how much weight to assign to each variable. This is the domain of image quality assessment (IQA).

In section 5.2 the literature study is presented: 5.2.1 discusses IQA in general as well as specific examples of existing work; 5.2.2 examines model fitting in general and touches on its previous application in IQA; in subsections 5.2.3 and 5.2.4 the two types of models evaluated in this chapter are introduced. Section 5.3 describes the various experiments conducted during the acquiring (5.3.2 - 5.3.3) and processing of the data (5.3.4), the fitting of the models (5.3.5 - 5.3.6) and the interpretation of the results (5.3.7). The results of these experiments are presented in section 5.4. Finally conclusions are drawn in section 5.5.

5.2 Literature

5.2.1 Image quality assessment

The machine evaluation of images is an important problem in image processing that continues to be actively investigated. The goal of image quality assessment is to enable a machine to make an objective judgement on the quality of an image that corresponds to a subjective evaluation of the same image by humans. A significant application, for example, is the efficient evaluation of

new compression algorithms; instead of having to conduct an expensive survey to establish the perceived performance of a new algorithm, an IQA algorithm can be used [105].

Frequently when a machine's judgement is to match that of a human, statistical models are trained to mimic data collected in a subjective experiment. Alternatively, a model can be constructed without making use of subjective data. In this case a fixed, non-linear, monotonic mapping between the output of the model and subjective data is allowed when testing the model against subjective data.

Full-reference quality assessment

The majority of existing literature focuses on *full-reference* quality assessment algorithms, where the original image, $f(x, y)$, is assumed to be available for comparison with the degraded image, $g(x, y)$. Most full-reference IQA models are constructed without training on subjective data. Instead, the algorithms use objective mathematical models and therefore, as previously mentioned, output can be modified by non-linear mappings to validate against subjective data.

Many of the full reference algorithms use a sophisticated human visual system (HVS) model. An HVS model attempts to take the way humans perceive visual information into account when determining a quality score [96]. For example, instead of merely quantifying noise in an SNR, which only considers noise power, the HVS model incorporates the noise frequency as well as the position of the noise in the image. Also modelled in the HVS model is the tendency of humans to perceive contrast better at certain spatial frequencies: a contrast sensitivity function [29], as well as the effects of luminance and contrast masking [46].

In [105] the similarity between $f(x, y)$ and $g(x, y)$ is evaluated using fuzzy set theory to combine neighbourhood-based and histogram-based similarity measures. A comparison of existing full reference IQA algorithms is made in [97].

The need for a reference image limits the application of these algorithms and differentiates them from humans, who can easily determine the quality of an image without a reference.

Blind image quality assessment

This has led to the formulation of the *blind image quality assessment* (also called *no-reference* or *univariant*) problem, in which an attempt is made to appraise the quality of an image without reference, i.e., using only $g(x, y)$. Most blind IQA algorithms incorporate statistical models which are trained on subjective data.

Applications discussed for blind image quality assessment include, among others, intelligent memory management in digital cameras [66] and evaluation of compression algorithms at the receiver [75]. In [43] the use of blind methods is justified in the context of measuring the performance of contrast enhancing algorithms. Existing full reference algorithms assume differences between the reference image $f(x, y)$ and the image in question $g(x, y)$ are *degradations*. This is not the case when $g(x, y)$ was created by applying image *enhancement* algorithms to $f(x, y)$.

In [66] three quantities are proposed as objective measures to aid in blind image quality assessment: edge sharpness level, random noise level and structural noise level. A fuzzy-logic model is trained on subjective data to give meaningful quality scores when presented with these features [107]. Similarly, specific attributes of known likely degradations are often used; in [104] the statistical properties of compressed video is used in blind quality assessment to estimate noise and subsequently compute the PSNR.

The problem of assessing contrast enhancing algorithms using blind IQA was dealt with by selecting numerous features and training an ensemble of neural networks [43]. Many features were compared using statistical analysis and those that gave the best separation between the original and contrast enhanced images were selected.

Outcome based quality assessment

Generally, the goal of IQA has been to match subjective human evaluation as closely as possible. However, depending on the context, the concept of a good quality image can differ. When images are meant for human consumption, human appraisal is the final criterion. But when images are input to some classification or recognition algorithm, the outcome of the algorithm should be the criterion by which image quality is judged.

For example, in agriculture automated processes often rely on image processing. In [74] the exposure levels during image acquisition must be adjusted to ensure optimal image quality. The performance of three existing algorithms (mushroom counting, pig-monitoring and weed identification) are measured. The image quality model must map to the performance of the algorithms.

However, the environment in [74] is very controlled: the model selected is a single measure, entropy, and is optimal when comparing various images *of the same subject*. Entropy has its origins in information theory and is defined as $H = \sum_{i=0}^{255} p_i \log(1/p_i)$, where p_i is the probability of intensity grey level i appearing in the image. These probabilities are equal to the normalised histogram bin values. Entropy has been described as a measurement of average global information content in terms of average bits per pixel. If an image has a bit-depth of 8, an entropy approaching 8 indicates that pixel intensities cover the full range and do so throughout the image [95, p. 26]. While it measures information content when comparing images of the same subject,

when the imaged scene differs, the concept of information content becomes too broad to capture in such a simple equation. For example, city scenes with high variation in pixel content will always outperform desert scenes with low variation, irrespective of the desirability of the scenes. In this sense, it is similar to the simple focus measures discussed in Chapter 4. Furthermore, while the feature corresponded to optimal performance for two of the algorithms in [74], the third algorithm required that human knowledge of the specific problem be incorporated.

In the field of medical imaging similar problems are encountered. When evaluating image quality in medical images, the evaluation should ideally be based on the diagnostic success rate [39]. However, this is a practical impossibility and, therefore, in [39] subjective preference experiments are used instead. Many viewers compare images with varying degrees of degradation or restoration and the results are used to evaluate image quality.

Conclusion

While a sophisticated HVS model is appropriate when images are meant for human consumption, it is not required in a scientific application where the peculiarities of human perception do not have to be compensated for. Although any model that tries to match the output of a subjective experiment will inherently be influenced by human perception, it is undesirable to model human perception explicitly. Moreover, an HVS model relies on the presence of the reference image, $f(x, y)$, which is not available in the proposed application.

Although proponents of full reference quality assessment disapprove of the use of blind IQA in scientific applications [46], for the application presented in this dissertation, there is no alternative. Since access to reference images is unavailable, blind image quality assessment must be used. This is why it is crucial for the feature extraction algorithms to be able to assess their own estimation ability, a characteristic inherent in the chosen noise estimation algorithm and introduced in the blur estimation algorithm with the relative energy measure in equation (4.3.14).

Using a generalised approach to feature selection, of testing a myriad of features and culling those which do not perform adequately [43, 30], is not appropriate for this application. Rather, the more considered, but also more common, method of decomposition of global image distortion into single effects is followed [46]. By concentrating on single effects the model is more objective and justifiable.

Whereas no distinction is made in [66] when proposing edge sharpness level, random and structural noise levels as quality features, one must distinguish between image degradation measures and image content measures. The motivation behind using degradation measures instead of content measures was given in section 4.1.2: degradation measures are more objective. Hence, for image quality assessment in this project only specific degradation measures

are considered: cloud cover, additive noise and defocus-extent.

Outcome based quality assessment would be ideal for this application: one would want to base a remote sensing image quality model on the performance of the algorithms to be applied to the images. However, in practice this is very difficult. In [74] there were three *pre-existing* algorithms, *one* variable (exposure) and *one* feature (entropy) that had to be justified in a machine vision context. There are a myriad of algorithms that will potentially be applied to Sumbandilasat images. Not only is it beyond the scope of this dissertation to attempt to implement all the potential algorithms, but the problem of how to combine the results from different algorithms is also not trivial. In [74] the only question was whether the chosen feature generates optimal exposure or not, where optimality is measured in an outcome based manner. In this case the relative weights of three different features are called in question.

Given the limitations discussed, it was decided to follow the same route as [39] and use a large subjective experiment. In spite of the numerous different algorithms that are applied to remote sensing images, the final analysis is almost always done by a human. It is not unreasonable to assume human judgement on image quality is a valid criterion in a remote sensing context. To make a quality assessment model suitably general, it must be based on the average opinion of many users. Ideally one would want to use remove sensing analysts' opinions, however, given the difficulty of finding more than 100 remote sensing analysts to take part in an experiment, members of the public were used instead.

5.2.2 Model fitting

General notes on statistical learning

It is illuminating to phrase the problem in terms of commonly used terminology. There are three measured variables, namely noise variance, cloud cover and defocus extent, which are the *inputs* (also called *features*, *predictors* or *independent variables*). These have some influence on the *output* (also called *response* or *dependent variable*), image quality. The goal is to use the inputs to predict the output. This is called *supervised learning*. [52, pp. 9–39]

The output variable is a *quantitative* measurement: image quality can be compared, based on the value of the measurement and values close to each other correspond to images with similar quality. It is also possible to have *qualitative* output, where output values assume a finite set, for example if images were labelled either suitable or unsuitable. Based on the type of prediction output, the prediction task is named differently: prediction of quantitative outputs is called *regression*, while prediction of qualitative outputs is referred to as *classification*. Both these tasks can be viewed as function approximation problems.

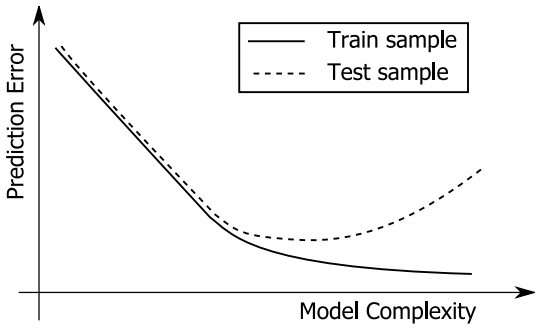


Figure 5.1: General effect of model complexity on testing and training error.

The learning task can be stated as: given the value of an input vector X, make a good prediction of the value of the output Y, denoted by $\hat{Y}$. For the $p = 3$ inputs in question, the input vector will have 3 elements. Generally the statistical prediction model is assumed to be of the form $Y = f(X) + \varepsilon$, $X \in \mathbb{R}^p$ is a real valued random input vector and $Y \in \mathbb{R}$ is a real valued random output variable. ε is a random error independent of X with expected value $E[\varepsilon] = 0$. It represents the deviation of the true input output relationship from the deterministic function $Y = f(X)$. The set of N data pairs, $\{x_i, y_i\}$, $i = 1, \ldots, N$, needed to construct the prediction model, is known as the *training data*.

In function approximation terminology, $f(x)$ has a domain equal to the p-dimensional input space. The model can be expressed as $y_i = f(x_i) + \varepsilon_i$. The goal is to obtain a useful approximation $\hat{f}(x)$ for all x in some region of $\mathbb{R}^p$ given the training data.

Model complexity and prediction error

The amount of data needed is determined by the model complexity, which is often dependent on the input space dimension. Selection of model complexity is related to the generality of the model. As model complexity is increased, the model can fit the training data better. However, if model complexity is increased too much, *overfitting* occurs: when presented with new input-output pairs, called *testing data*, the estimation is poor. The model has adapted itself too closely to the training data and loses generality. On the other hand, when the model is not complex enough, it will *underfit* and not be able to approximate the training data accurately enough. Figure 5.1 depicts the general relationship between model complexity and prediction error for both training and testing data.

Furthermore, as model complexity decreases (imposing more constraints on the solution of the function approximation), the solution becomes more sensitive to the specific choice of model: the error introduced by the model bias becomes more significant. Since different models make different assumptions

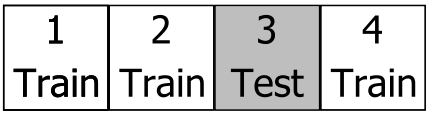


Figure 5.2: Data divided into parts for 4 way cross-validation.

about which type of constraints are suitable, it is meaningful to test more than one type of model.

As Figure 5.1 suggests, it is imperative to use different data for the training and testing of a model. In practice three datasets are needed [52, p. 196]:

training set used in the training algorithm to minimise the prediction error, i.e., used to fit the model,

validation set used to estimate the prediction error for model selection, i.e., the model complexity is chosen so that prediction error on the validation set is minimised,

testing set used to test the generalisation error of the final model.

If an abundance of data is available, data can be set aside for testing purposes. However, in practice, data is often scarce. Because of the desire to use the available data optimally, cross-validation is the most widely used method for estimating prediction error [52, p. 214]. In K-fold cross-validation the data is divided into K equal parts. One part is used for testing and the other for training. In Figure 5.2 the $k = 3$ part has been selected for testing. To complete the cross-validation procedure, k is set equal to $1, \ldots, K$ and the resulting K prediction errors are averaged to get the final prediction error.

Previous databases used in the training and evaluation IQA algorithms have differed in size. As already mentioned, the amount of *training* data required depends on model complexity. However, using copious numbers of images is also recommended during *comparative tests*: in [97] and [9]. The importance of using many images when comparing IQA algorithms is emphasised. The more images used, the finer one is able to distinguish between methods while retaining statistical significance. In [97] 779 images were used to distinguish between 10 existing quality assessment algorithms (the biggest study of its kind). In [46] 168 images were used for training and 176 for testing a model with three input features. Recently, in [105] the performance of a proposed IQA was tested on merely about 20 images; the only criterion was that the ordering obtained from the algorithm should correspond to the correct ordering, which was well-defined since the images differed markedly in quality. In [107] the training and the test set for a fuzzy-logic IQA model of three input features each consisted of 25 images. The IQA model in [43], having 32 input features and consisting of an ensemble of four neural networks, each with 10 hidden neurons, was trained and tested using a dataset of 480 images.

Table 5.1: A simple full factorial experiment.

A	B
0	0
0	1
1	0
1	1

Evaluating the entire model space

In the IQA algorithms encountered the performance of the algorithm is often measured only in the presence of a single degradation type. Even if the model has more than one input feature, only one is presented to the model at a time: in [96] the "cross distortion performance" of various IQA models is compared, but the performance of the models when presented with various degradation types, *one at a time*, is measured. If one considers the model a function in p-dimensional space, where p is the number of input distortion types, by testing only one distortion type at a time while setting the others to zero, the function is effectively being evaluated only in the planes corresponding to the axes of the space. No cross-coupling effects are modelled.

For the author's model, an attempt was made to model the complete feature space (see section 5.3.2 from page 136). However, this is difficult since the so-called 'curse-of-dimensionality' means that training and evaluating functions in higher dimensional space requires exponentially more data [52, pp. 22–28].

When attempting to measure the cross-coupling of input variables, one cannot simply vary multiple inputs in any manner. Suppose we have two input variables A and B each with only two levels, 0 and 1, for which we are interested in the output. If we take one output measurement where both inputs are low and one where both are high, it will not be possible to ascribe the behaviour of the output to any one of the variables. In this example A has been *confounded* or *aliased* with B: since the combined effect of two inputs are measured, it is not possible to tell which input has caused the effect on the output.

The simplest experimental design that avoids confounding and can model the effect of cross-coupling of input variables, is a *full factorial* experiment [50]. This simply measures every possible combination of input variable levels. Continuing the example from the previous paragraph, a full factorial experiment is shown in Table 5.1. The problem with full factorial experiments is that, as the number of input variables or the number of levels for each variable increases, the experiment size increases exponentially. For example an experiment with 4 variables each evaluated at 8 levels would require $8^4 = 4096$ observations. However, it is possible to design *fractional factorial* experiments in which the confounding variables are chosen to be unimportant.

As the number of input variable increases above two, the order of possible

confounding interactions increases. For example, with three input variables, A, B and C, each of the input variables can be confounded with one another, but higher order interactions can also be confounded with one another or with primary variables. For example, a two-factor interaction, AB, can be confounded with a primary variable C. This means that it is impossible to discern the combined effect variables A and B from the effect of variable C. Alternatively, two-factor interactions can confound one another, for example AB and AC. The purpose of conducting an experiment with more than one variable active at a time, is to determine which higher order interactions are present. However, although higher order interactions can exist, ordinarily the main effects and two-factor interactions provide the main information on the effects of factors in a response. Fractional factorial experiment design utilises this fact to allow smaller experiments in which higher order interactions may be aliased with one another, but main effects and two-factor interactions may not.

5.2.3 Piecewise polynomials and splines

Piecewise polynomials are a useful modelling tool [52, pp. 117–137]. While normal polynomials are flexible, they are limited by their global nature: it can be very difficult to find a polynomial that fits sufficiently well in all areas of the training data. This problem is solved by piecewise polynomials. Different polynomial functions are used to model different parts of $f(X)$ in different regions of the domain of X. The boundaries between the regions are known as *knots*. Various continuity restrictions are placed at the knots, for example the function must be continuous and have continuous first derivatives. These continuity restrictions place linear constraints on the parameters of the polynomial functions, effectively reducing the number of parameters (or degrees of freedom or model complexity).

Splines are piecewise polynomial functions that obey specific constraints. A commonly encountered spline is the *cubic spline*:piecewise cubic polynomials that are continuous and have continuous first and second derivatives at the knots. The total degrees of freedom in a spline function can be calculated according to:

$$\begin{aligned}(\text{degrees of freedom}) = (\text{number of regions}) \times (\text{parameters per region}) \\ - (\text{number of knots}) \times (\text{constraints per knot})\end{aligned} \quad (5.2.1)$$

Therefore increasing the number of regions or the order of the polynomial in each region increases model complexity, while increasing the constraints per knot decreases model complexity.

Generally an order-M spline with knots ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order M, and has continuous derivatives up to order $M - 2$. Cubic splines (order 4) are claimed to be the lowest order splines for which the

human eye cannot perceive the knot discontinuity. The most commonly used splines are orders $M = 1, 2$ and 4.

In these splines the knots are fixed and one needs to select the order of the spline, the number of knots, as well as their placement. These splines are known as *regression-splines*. It is a common approach to choose the number of knots and order of the spline, but let the position of the knots be determined by the position of the observations x_i, for example by dividing the area of the domain X for which data x_i is available into equal parts. Once the knots sequence is fixed, the piecewise polynomial fits can be computed using least squares approximation.

Another type of spline is the *smoothing spline*. Smoothing splines have knots at every data point and a single parameter ρ that controls the effective degrees of freedom of the model. By varying ρ between 0 and 1 the smoothing spline's behaviour changes from a normal linear regression across all data to complete interpolation between each data point. While there are also other types of splines in existence, basic regression splines can be used to construct an adequate image quality model.

5.2.4 Neural networks

Structure and terminology

Neural networks are non-linear statistical models [52, pp. 348–367] [35]. Although there are many variations, a basic configuration of the most widely used one is sufficient for the image quality model and is briefly described here. It is known as the single hidden layer, feed-forward, back-propagation neural network. Figure 5.3 depicts a network diagramme of such a neural network[1].

Although neural nets can be used for both regression and classification, the schematic in Figure 5.3 is typical for regression since a single quantitative response is modelled by one output node, Y_1. Derived features, Z_m are formed from linear combinations of the inputs X_p, and the output response is again a linear combination of the Z_m:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), \quad m = 1, \ldots, M, , \\ f(X) &= \beta_0 + \beta^T Z, \end{aligned} \tag{5.2.2}$$

where $Z = (Z_1, Z_2, \ldots, Z_m)$, $\{\alpha_m; m = 1, \ldots, M\}$ are p-dimensional weight vectors, β is a m-dimensional weight vector and $\sigma(v)$ is the activation function. The activation function used is the hyperbolic tangent sigmoid: $\sigma(v) = 2/(1 + e^{-2v}) - 1$, shown in Figure 5.4. Although it is also possible to have a different output function, a linear combination of Z_m is common in regression. Note that if σ is the identity function, the neural network collapses to a linear model

[1] The model was initially based on the human brain with each unit representing a neuron and the connections representing synapses.

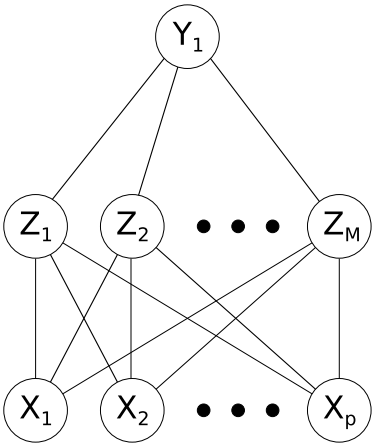


Figure 5.3: Schematic of a single hidden layer, feed-forward neural network with one output.

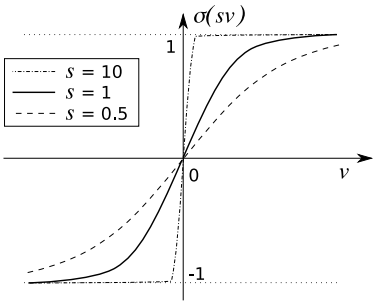


Figure 5.4: The hyperbolic tangent sigmoid function. The scale parameter s in $\sigma(sv)$ controls the activation rate.

of the inputs. In Figure 5.4 one can see that the rate of activation depends on the norm of the weight vector, $|\alpha_m|$. Hence, when $|\alpha_m|$ is small the unit is operating in the linear part of its activation function.

The network diagramme in Figure 5.3 is simplified and therefore the intercepts, a_{0m} and β_0, are not depicted. They can be drawn as additional bias inputs feeding into the hidden layer and the output.

The layer computing the derived features Z_m is called hidden since the values of Z_m are not directly observable at the output. Neural networks can have more than one hidden layer, but these are usually used where a hierarchical model of the inputs is appropriate, which is not the case for the quality assessment model. Feed-forward refers to the fact that there are no feedback paths in the network (typically used to model time dependent systems), while back-propagation relates to the training algorithm used to derive the weights from the training data.

A neural network is an example of *universal approximator*: given enough degrees of freedom the model can approximate any continuous function in $\mathbb{R}^p$

arbitrarily well. However, this generality comes at a price: interpretation of the final model is difficult. Each input enters into the model in a complex manner and its path through the network is opaque. Thus, while valuable for prediction, neural networks are not so useful for building an understandable model.

Working with neural networks

Unlike spline models, a least squares solution for neural network models does not exist. Instead, the complete set of weights and biases, denoted by θ:

$$\begin{aligned} &\{\alpha_{0m}, \alpha_m; m = 1, 2, \ldots, M\}, \\ &\{\beta_0, \beta\}, \end{aligned} \tag{5.2.3}$$

must be determined iteratively. Typically the weights are given random starting values and adjusted after each iteration (or training *epoch*) to minimise the error function, $Q(\theta) = \sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$, where n is the total number of data points. Since the existing MATLAB® implementation of back-propagation was used, its implementation is not presented here; see [52, pp. 353–355] for more detail.

The training of neural networks is not straightforward. The training algorithm is not guaranteed to converge at a global minimum of the error function $Q(\theta)$. When convergence at a local minimum far from the global minimum occurs, the model is a bad fit. The outcome of the training is dependent on the starting conditions. The simplest solution to this problem is to randomise the starting values of the weights repeatedly, repeatedly train the network and select the best solution from all training sessions.

Another problem is the tendency to overfit when the model complexity is too high. Fortunately there are various methods available to combat this problem. These methods are generally referred to as *regularisation*.

Firstly, one can choose a more than adequately complex model in conjunction with validation data and early stopping. During training the prediction error on the training data decreases after every iteration while the prediction on validation data decreases initially and then increases as overfitting occurs, like in Figure 5.1. Although, in this case, model complexity stays the same, the model becomes increasingly tailored to the training data with each epoch, making it less general. Regularisation is achieved by stopping the training process as soon as the model validation error starts to increase.

The second regularisation option is to add a penalty term based on the size of the weights to the error function $Q(\theta)$:

$$Q_{reg}(\theta) = \gamma(Q(\theta)) + (1 - \gamma)\frac{1}{n}\sum_{j=1}^{n}\theta_j^2, \tag{5.2.4}$$

where $\{\theta_j; j = 1, \ldots, n\}$ are the individual vector elements for all the weights as well as the biases. This penalty term inhibits the increase in weights' val-

ues during training and will ideally allow only those weights necessary for the model to assume significant values. Validation data is used to determine the optimum size of the penalty variable γ. This form of regularisation is recommended in [52] and exists in MATLAB®, where γ is called the performance ratio.

A third option, automatic Bayesian regularisation [69], is available in MATLAB®. This method assumes that the weights and biases are random variables with specific distributions. No tuning of parameters or validation data is required. This is the regularisation method recommended in the MATLAB® documentation. The algorithm provides a measure of how many of the network parameters are effectively being used. If the algorithm works, this should remain constant even as more units are added to the neural network.

The last regularisation option is to vary the number of hidden units (and therefore model complexity) and use validation data to select a model. A graph like the one in Figure 5.1 is used to aid the selection process.

It is recommended that the input and output data be normalised to have zero mean and a standard deviation of 1. This ensures that all the inputs are treated equally during the training process.

5.3 Experiments

5.3.1 Introduction

As discussed in section 5.2.1, it was decided to conduct a large subjective image quality experiment. The collected data would enable a model to be formed, mapping three measured features into a quality score.

Specifically, it was hoped that the experiment would provide insight into the following aspects:

- The relationship between perceived image quality and varying amounts of a single distortion type. Even when considering only one distortion type it is unlikely that the image quality will be a linear function of measured feature.

- The relative weights of the different types of distortion, when only one is present at a time. It is unlikely that noise, clouds and blurring will have an equal effect on the perceived image quality.

- The joint effect of two or three simultaneous distortion types on image quality. It is unlikely that this will simply be an additive model. By measuring the joint effect one can determine if a certain distortion type dominates the others.

To ensure that the collected dataset is valid, the methods described in [97] were closely followed during the design of the experiment and processing of

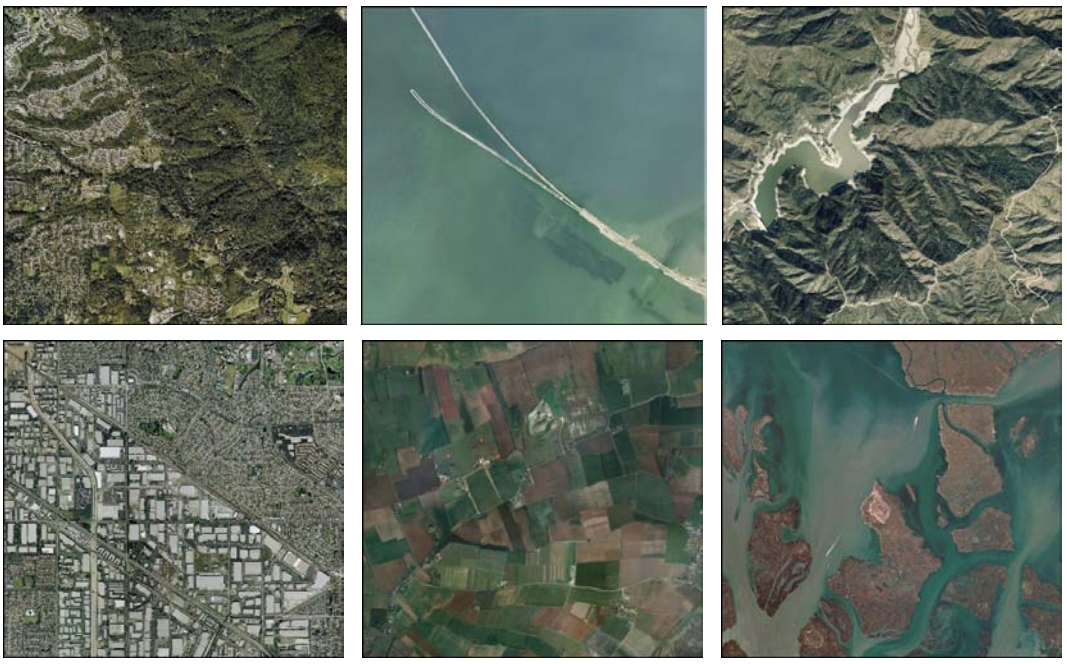


Figure 5.5: A selection of the input images used.

the data. However, even though a web-based interface was used in [97], the experiment in [97] was conducted in a test centre where all PCs were identical. The data was collected over the course of two years. To enable rapid collection of data the example of [43] was followed and the experiment conducted over the internet. This allowed the data to be collected within about one month.

5.3.2 Image database

Input reference images

It is important to have a diverse range of input reference images that adequately reflect the scope of remote sensing image types. Images from a variety of locations and ranging from smooth to dense spatial activity were selected. An important consideration was to use remote sensing images of approximately the same GSI as Sumbandilasat. 30 images with 500×500 resolution and GSI of 8m were acquired from Terraserver [4]. The relatively small size of the images allows an entire image to fit into the display area of a screen. Figure 5.5 shows a selection of the images used.

Degradation of images

The images were corrupted to varying degrees using the three degradation types. To ensure the generality of the model, the levels of degradation were varied to create images with a wide range of quality, from barely perceptible

to highly degraded. White Gaussian noise with $\sigma_n = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 13, 15, 20, 25\}$ intensity quantisation levels was added. Ten images were generated at each of the 14 noise levels, so that a total of 140 noisy images was generated. For each instance of each noisy image generated, the input image was randomly selected from the 30 base images, to average out the influence of underlying image structure.

Images were blurred through convolution with a circular disk of radii $R = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ pixels. Once again 10 images were created at each radius using the same random selection policy. Thus 140 unfocused images were created.

To have similar control over the amount of cloud cover, an algorithm was developed to add cloud cover to images. It is discussed in the next section. Using this algorithm 145 images with various levels of cloud cover and dispersion were generated.

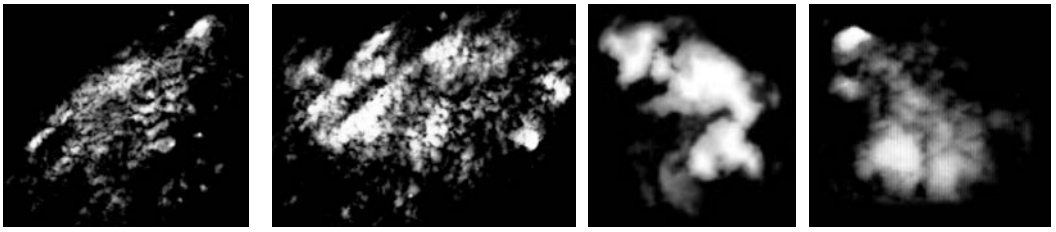
An additional set of 64 test images was created that combined different distortion types in single images. This is discussed in more detail on page 136.

Adding clouds to images

It was desirable to be able to specify two input parameters for the cloud generation algorithm: the amount of cloud cover (in total percentage of the pixels) as well as the dispersion (discussed in Chapter 2). Dispersion was controlled by specifying the number of clouds added and distributing the clouds according to a uniform random variable. For example, by specifying 50% cloud cover and two or three clouds the dispersion is less than when specifying the same cover with 50 clouds. This allowed different cover scenarios to be tested, with the hope of determining the effect of cloud cover as well as dispersion on perceived image quality.

To determine the size of each cloud, imagine the total cover percentage as a range, for example for 50% the range is $\{0, 50\}$. This range is divided $m - 1$ times at random places, where m is the total number of clouds specified. For example for three clouds the division could be $\{0, 13, 43, 50\}$. The size of each cloud is the range of each division. In the example, the cloud sizes would be $\{13, 30, 7\}$ %.

To generate the clouds, sample clouds were manually extracted from existing remote sensing images. These served as the input set. An example of the input set images is presented in Figure 5.6. Images were randomly selected from the input set, scaled, rotated and superimposed on the input image to satisfy the cover and number of clouds specifications. To avoid excessive scaling, which could result in unrealistic-looking clouds, images in the input set were divided into large and small classes. To avoid artificial looking sharp cloud edges, each cloud was blended into the background using a mask to specify the cloud's spatially varying transparency. Nevertheless, the resulting clouds look slightly artificial since they do not cast shadows on the ground.



(a) (b) (c) (d)

Figure 5.6: Example input masks for the cloud generation algorithm. (a) and (b) are large clouds while (c) and (d) are small.

Since clouds could overlap or be cut off by the edges, decreasing the cover below the amount specified, the algorithm is repeated iteratively and more clouds added until the total cover is within 10% of the specified cover.

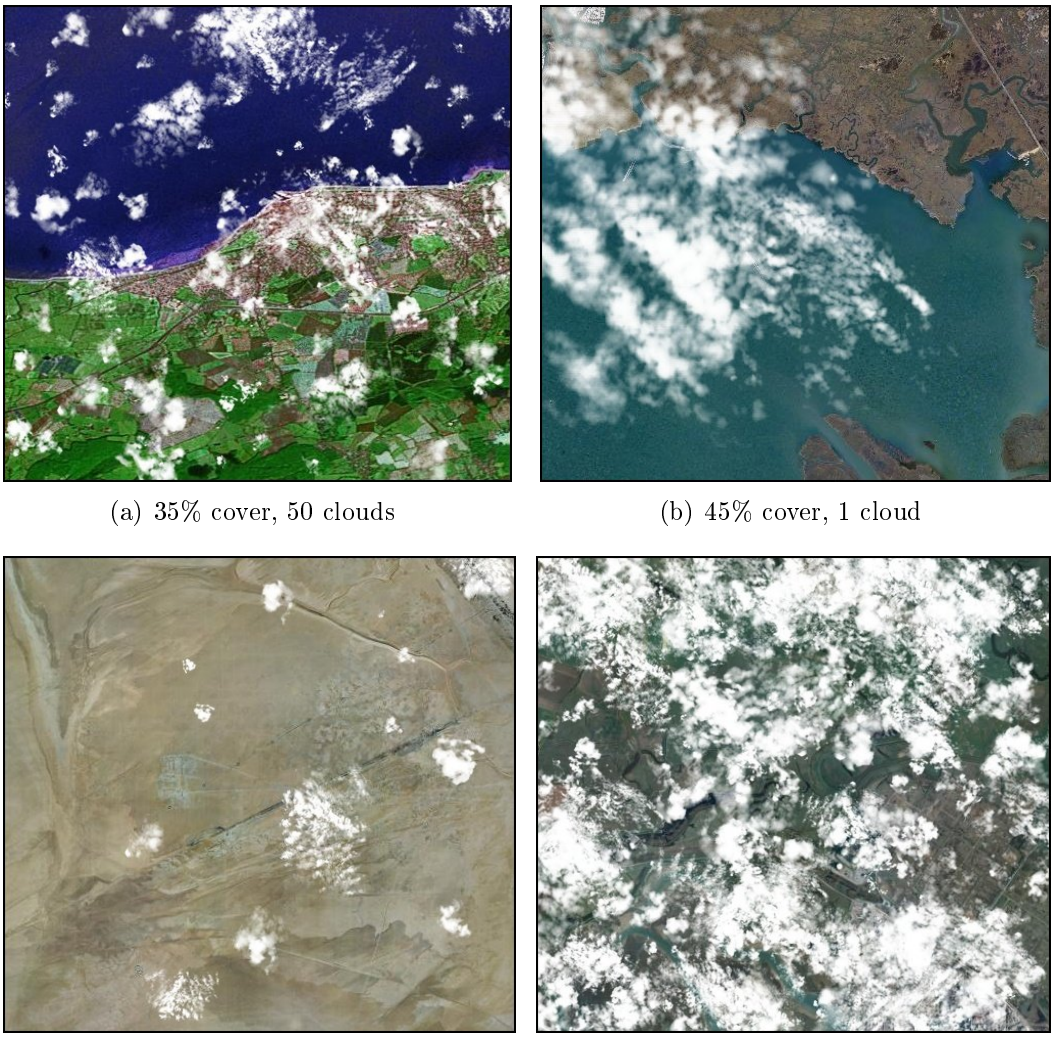
In Figure 5.7 examples of the cloudy images generated with the algorithm are shown. Using the algorithm a set of 140 cloudy images were generated with cloud cover = {1, 2, 3, ..., 16 ,18, 20, 22.5, 25, 27.5, 30, 35, 40, 50, 60, 70, 90} and number of clouds = {1, 3, 7, 15, 50} at each cloud cover level. A similar randomisation method to the noise and blur was used to select the images.

Multiple distortion types in a single image

In [97] the various models' performance were tested on one degradation type at a time. It is possible that more than one degradation type can be present in the same satellite image, although the coincidence of degradations is less likely than a single degradation. This means that it becomes necessary to model the possible cross-coupling interactions between the different inputs in the IQA model. As discussed on page 128 under the heading *Evaluating the Entire Model Space*, this type of behaviour cannot be detected in a "one factor at at time experiment". Measuring cross-coupling requires the design of a factorial experiment [50].

As previously mentioned, full factorial experiments rapidly grow in size as the number of inputs or levels of inputs increase. However, there is a constraint on the number of images that a subject can examine in a single session before fatigue sets in. It is recommended that sessions be limited to 30 minutes [97], which is enough time to evaluate approximately 170 images.

Two-level inputs, like those from Table 5.1 on page 128, allow only linear responses to be evaluated. If quadratic or cubic responses of the inputs are to be assessed, one must measure more than two levels. To ensure highest possible resolution of the IQA model it is desirable to maximise the number of levels used in the experiment of each of the three variables. Therefore fractional factorial experimental design options were investigated in an attempt



(a) 35% cover, 50 clouds

(b) 45% cover, 1 cloud

(c) 9% cover, 15 clouds

(d) 82% cover, 50 clouds

Figure 5.7: Different cloudy images generated by the cloud-adding algorithm. Notice the difference in dispersion between (a) and (b), both with relatively similar cloud cover.

to maximise the number of levels used given a fixed ceiling to the number of images. To use the experimental design procedure outlined in [50] the number of levels must be a power of two. This allows the variable for which more than two levels are to be modelled to be decomposed into multiple two-level variables. This, in turn, allows the design procedures for two-level variables to be applied (which are also available in MATLAB®).

It is possible to design a full factorial experiment with four levels for each variable. The number of observations required would be $4^3 = 64$. A quarter-fractional factorial experiment, where each variable has eight levels, would require $\frac{1}{4} \times 8^3 = 128$ observations, which is also acceptable. However, by following the method described in [50] it was determined that this would result in primary interactions aliasing with main effects, which is unacceptable. Therefore, the data was generated for the full factorial experiment.

This experiment was conducted after the data for the single variable experiments had been collected and analysed. Given that the modelling resolution is limited by the fact that only four levels of each variable are allowed, it was desirable to collect data in an area of the model that would conceivably be used. To this end, the upper limits for the cloud cover and blur were lowered to 50% and 10 pixels respectively. This is justified by the thought that images with more than 50% cloud cover are unlikely to be of any use. Furthermore, the blur response from the single variable experiment started to flatten after reaching $R = 10$ pixels.

The final range of input degradations was: cloud cover = $\{10, 23, 37, 50\}$, noise standard deviation = $\{5, 12, 18, 25\}$, blur radius = $\{2, 5, 7, 10\}$.

5.3.3 Test methodology

When designing a subjective image quality experiment, it can be either *single-* or *double-stimulus*. In a single-stimulus experiment, the subject must give a quality score to only one image at a time, while the double-stimulus case, a reference and altered version of the same image presented in succession and score must be given to both. There is a parallel to objective blind and full-reference image quality assessment. While a double-stimulus experiment more accurately captures the effect of the alteration on image quality, the experiment typically requires 3-4 times more time per image than a single-stimulus experiment.

Closely following the example of [97], a hybrid approach was followed. Single-stimulus methodology was used, but the 30 reference images were included in the same experimental session as the test images. This allows for more images to be evaluated, while still permitting many images to be evaluated within the 30 minutes time limit of each session.

Table 5.2: Experimental sessions.

Session	Images	Subjects
Blur	170	20
Noise	170	20
Clouds	170	21
Alignment[a]	51	32
Cross-coupling	148	33

[a] Double stimulus setup implies 204 images viewed and 102 images evaluated.

Equipment and software

As previously mentioned, the data was collected through the internet. The disadvantage of this approach is that there is no control over the type of monitor or the ambient illumination in the subject's room. However, given a time constraint, the advantage is that data from a large group of subjects can be collected, and that the group of subjects represent a better random sample from the population. Additionally, subjects were instructed to adjust the colour depth and resolution of their monitor to standard levels.

The web based interface consisted of various php [5] scripts to generate the html pages to be displayed to the subject. The main php script had to step through all the images in a specified directory on the server and display them one after the other, in a random sequence, to the subjects. A javascript based slider-applet [6] was adapted to allow the subjects to report their quality evaluations by dragging the slider on a quality scale. As recommended in [9] and [97] the quality scale is unmarked numerically. It is divided into five equal portions labelled as "Bad", "Poor","Fair","Good" and "Excellent". The position of the quality slider is converted into a raw quality score: an integer in range 1-100. The position of the slider resets after each evaluation. Figure 5.8 shows a screenshot of the interface. The slider bar is important since it allows for faster and more 'instinctive' evaluation than if the subjects were asked to assign a number to each image.

The quality evaluations were recorded in a MySQL [7] database. Each experimental session was stored in a different table. Python scripts were used to generate SQL queries to extract difference scores from the tables (difference scores will be discussed below). The difference scores were written to ASCII text files for further processing in MATLAB®.

The experiments were conducted in five sessions: one each for the individual degradation types, a realignment experiment and a cross-coupling experiment. The full set of reference images were randomly placed among the degraded images in each experiment. Table 5.2 shows the number of images in each experiment, as well as the number of subjects.

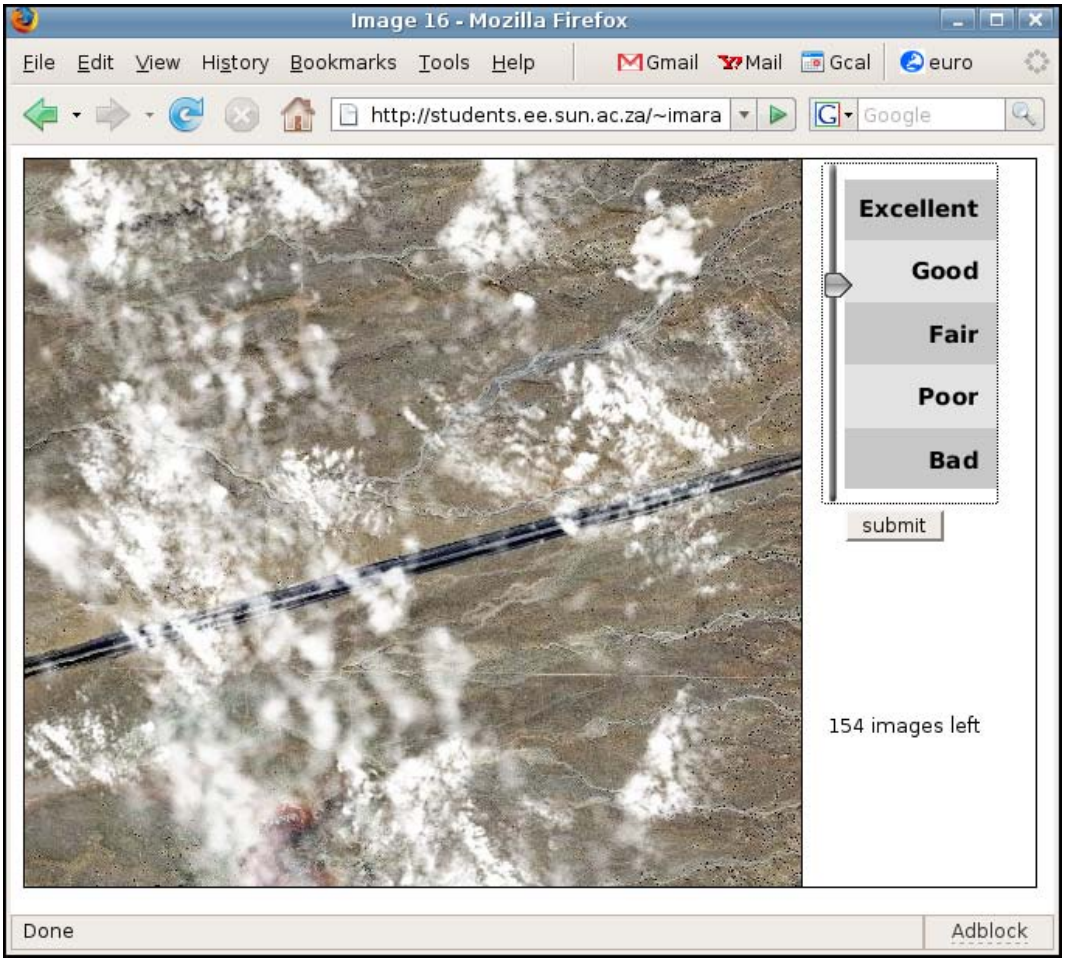


Figure 5.8: An example of the user interface to the experiment.

Single-variable sessions

A brief description about the goal of the experiment, as well as instructions, and an explanation about the type of degradation present was given at the onset of the experiment. The subjects were shown the approximate range of quality that would be present in the experiment to ensure that they used the entire quality scale. The example images were not contained in the experiment itself. Each subject sees the images in a different random sequence to ensure that the order of the images does not affect the average quality scores.

Realignment session

Ideally all the data would be collected in a single session. Since subject fatigue makes this impossible, multiple sessions have to be used. However, when using more than one session, the scales for the raw quality scores from the different sessions will not be the same. This is due to the fact that different distortion types are used in different sessions and that subjects' expectations

were 'normalised' at the start of each experiment as described in the previous paragraph. To combine data from these different sessions into a single dataset, realignment is necessary. This necessitates a re-alignment experiment where data from all the different individual sessions is present.

After the completion of the three single experiments, 17 images covering the entire quality spectrum were selected from each group. These 51 images, along with their reference images, were presented using a double-stimulus setup. The images were chosen so that all 30 reference images were used. The double-stimulus, as well as increased number of subjects, ensures more accurate quality measurements for realignment purposes. The images were presented using the *view A, view B, score A, score B* method, where A and B were, randomly, either the reference or the degraded image. Once again the order of the images was randomised differently for each subject.

Cross-coupling session

Finally, a cross coupling-experiment was performed, using single-stimulus again. As mentioned previously, data from the single experiments was analysed to determine the range of input degradations for this experiment. In addition to the 64 full factorial images, the 30 reference images and all 51 realignment images were also included in the experiment. This is necessary to align the quality scores from this experiment with the database of quality scores already collected.

In total the number of subjective human judgements collected is:

$$(170 \times 20) + (170 \times 20) + (175 \times 21) + (51 \times 32)2 + (148 \times 33) = 18623,$$

where the extra multiple of 2 in the realignment experiment accounts for the double stimulus. The number of unique degraded images is:

$$140 + 140 + 140 + 64 = 484,$$

which is comparable with existing IQA literature.

5.3.4 Processing the raw data

Outlier detection and rejection

The outlier detection and rejection values from [97] were used. A raw difference score for an image, calculated according to equation (5.3.1) defined in the following section, was considered an outlier if it was outside an interval Δ from the mean raw difference score (across all subjects) for that image. $\Delta = 2.33 \times \sigma_i$, the standard deviation for raw difference scores for that image. If, for any sessions, more than 16 evaluations of a single subject were rejected, all the evaluations for that subject were rejected. The outlier rejection procedure was run twice. About 4.4% of all images were rejected. This is the same as in

[97], suggesting that the use of Internet did not lead to some viewers having radically different viewing environments. Two subjects were rejected.

Difference mean opinion scores

The first step in calculating difference mean opinion scores (DMOS) is to compute the raw difference score:

$$d_{ij} = r_{i\text{ref}(j)} - r_{ij}, \tag{5.3.1}$$

where r_{ij} is the raw quality score for the ith subject and the jth image, and $r_{i\text{ref}(j)}$ denotes the raw quality score assigned by the ith subject to the reference image corresponding to the jth distorted image. Subtracting the reference image score ensures that only degradation effects are measured in the DMOS. Note that a larger DMOS score corresponds to worse perceived image quality.

Z scores were computed from the raw difference scores according to :

$$z_{ij} = (d_{ij} - \bar{d}_i)/\sigma_i, \tag{5.3.2}$$

where $\bar{d}_i$ is the mean of the raw difference scores over all images ranked by the subject i and σ_i is the standard deviation. Computing Z scores effectively normalises over subjects' sensitivity; for example, if a subject used only half of the scale, the range of his scores would be increased by dividing by σ_i. Or, if a subject had a tendency to give higher than average quality scores, this would be rectified by the subtraction of $\bar{d}_i$. The Z scores were averaged across all subjects to yield $\bar{z}_j$ for the jth image.

Z scores were mapped to DMOS using the results from the realignment experiment. DMOS scores were first computed from the realignment data by calculating difference scores according to (5.3.1) and then averaging across all subjects to produce DMOS_j for the jth realignment image. Figure 5.9 shows the relationship between the Z scores derived from the individual sessions and the DMOS scores from the realignment session. Each marker represents an image.

To convert the Z scores from the individual sessions into DMOS scores for a unified database, linear mappings were learned: $\text{DMOS}(\bar{z}) = p_1\bar{z} + p_2$. The values for p_1 and p_2 were computed by doing a least squares linear regression between $\text{DMOS}(\bar{z}_j)$ and DMOS_j. One mapping for each of the individual sessions was learned. These mappings are shown in Figure 5.10. The exact same process was applied to map the cross-coupling data from raw difference scores to realigned DMOS, except that all 51 images, corresponding to realignment data on each of the three axes in the input space, could be used in a single mapping instead of only the 17 used in the single variable experiments.

5.3.5 Creating a spline model

Because of the sessioned nature in which the data was collected, the dataset has different density of input observations in different areas of the input space

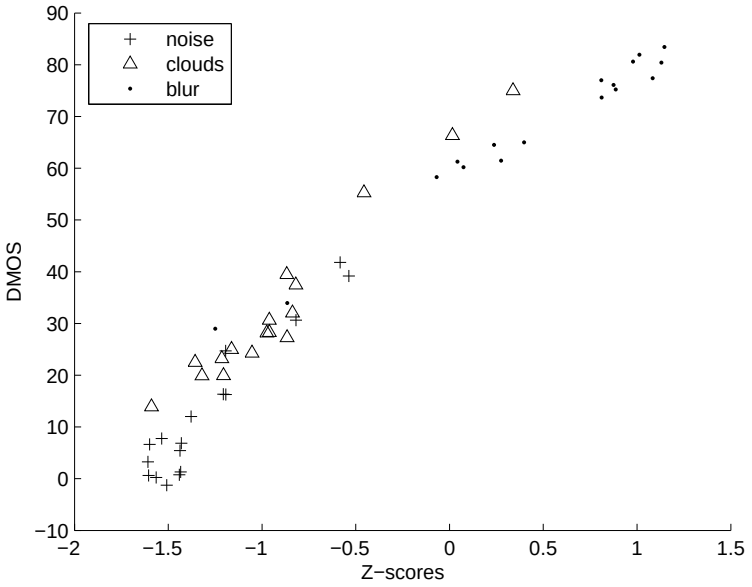


Figure 5.9: DMOS values and Z scores for images used in the realignment experiment.

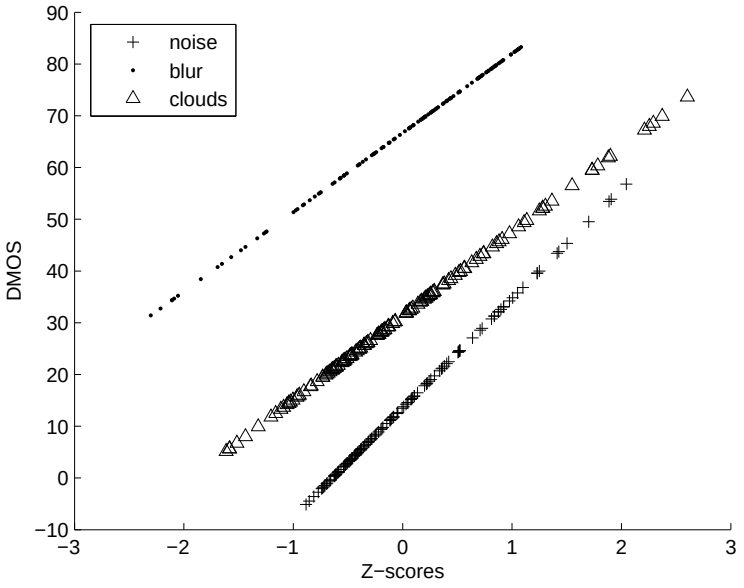


Figure 5.10: The linear realignment mappings obtained for the individual variable sessions.

$X \in \mathbb{R}^3$, where the input vector is of the form:

$$X = [X_c, X_b, X_n]^T, \tag{5.3.3}$$

where X_n is the noise variance in intensity levels, X_c is the cloud cover percentage and X_b is the defocus extent in pixels. Specifically the density in the central area, where X_n, X_c and $X_b > 0$, corresponding to the data collected in the cross-coupling experiment, is lower than on the axes of the input space. The axes are those areas where only one of the elements is non-zero and correspond to the data collected in the single variable experiments. Varying the density in this manner is an appropriate use of resources (experimental collection time), since it is likely that most of the images encountered will have no or only one degradation present. Therefore the model needs to be the most accurate on the axes near the origin of the input space.

Spline models were used in a way that allowed varying degrees of freedom in different parts of the model. This was achieved by using different two-dimensional (line) spline models on each axis of the input space and a three-dimensional (surface) spline model in central area. In the areas between the different models interpolation is used, so that the resulting final model is a continuous, smoothly varying function approximation $\hat{f}(x)$.

In the following sections the spline regularisation options available in MATLAB® are first discussed. Then the model selection process for each of the axes, as well as the central area, is described. Finally interpolation procedure used to combine the different models is described.

Available regularisation options

MATLAB's spline toolbox has two options available for varying the model complexity. When using the basic piecewise polynomial regression splines discussed in section 5.2.3 on page 129, one can manually choose the number of knots and order of the polynomials to vary the model complexity according to equation (5.2.1).

Alternatively, smoothing splines can be used. This might seem easier since the model complexity can be changed by altering a single parameter, ρ, between 0 and 1. However, smoothing splines have two disadvantages for this application. Firstly, they are considerably more complex to implement in an embedded system than regression splines. Secondly, it is in fact difficult to evaluate various model complexities since the tuning parameter ρ affects the model in a counter-intuitive manner. For example, the model will be insensitive changing ρ from 0 to 0.9 and then suddenly change its behaviour at $\rho = 0.999$. It was therefore decided to use regressions splines

Cloud axis

The DMOS data in the area of the input space where $X = [X_c, 0, 0]^T$ was divided into four equal groups for cross validation purposes (recall section

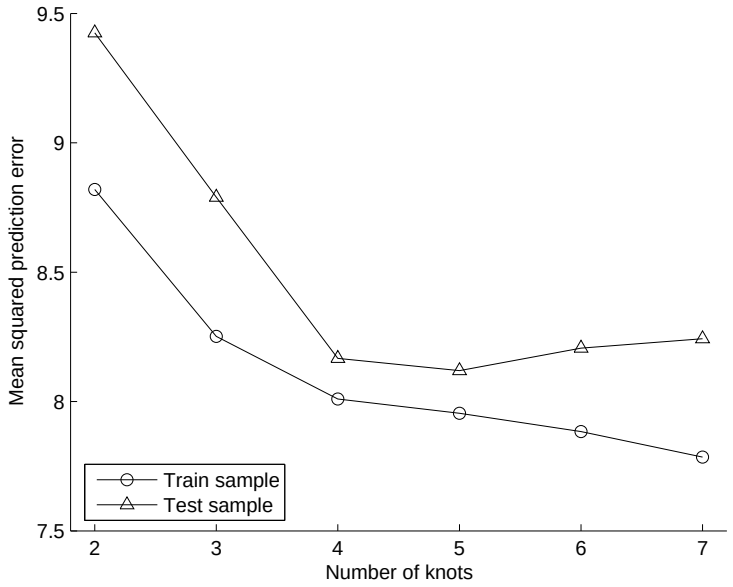


Figure 5.11: The effect of increasing the number of knots on testing and training data.

5.2.2 and Figure 5.2). The four-way split was necessitated by the structure of the data in the central area and requirements of one of the MATLAB® fitting functions, as discussed in the *Central Area* section to follow on page 148. Since data from this split had to be used for both model selection (validation) and model evaluation (testing), it was decided that two of the splits, $k = 1$ and 2, would be used for validation and the remaining two, $k = 3$ and 4, for testing. The mean squared prediction error is calculated according to:

$$MSE_k = \frac{1}{N_k} \sum_{i=1}^{N_k} (y_{ki} - \hat{f}(x_{ki}))^2, \quad k = 1, 2, 3, 4 \tag{5.3.4}$$

where $\{y_{ki}, x_{ki}\}$ are the N_k input-output testing data pairs for cross-validation run k. Thus the model selection was done by considering $\frac{1}{2}(MSE_1 + MSE_2)$.

Different order polynomials, as well as different number of knots, were experimented with. The testing and training errors behaved as expected, i.e., in a manner consistent with Figure 5.1. Figure 5.11 shows an example of how increasing the number of knots affects the training and testing mean squared prediction error for a cubic spline. As expected, the training data prediction error continues decreasing with increasing model complexity, while the test prediction error shows that optimum model complexity is at 5 knots. Note that the number of knots indicated here include the knots at the start and end-points.

To compare different order polynomials the test graphs were combined in a single plot as shown in Figure 5.12(a). Increasing model complexity generally

leads to lower test error. What is interesting is that only in the cubic spline case with 6 or 7 knots does the prediction error start to worsen. A different view on the same data can be obtained by letting the horizontal axis be the model degrees of freedom, according to equation (5.2.1) on page 129. The result is shown in Figure 5.12(b). Here one can see that the higher order polynomials have more degrees of freedom than the lower order ones.

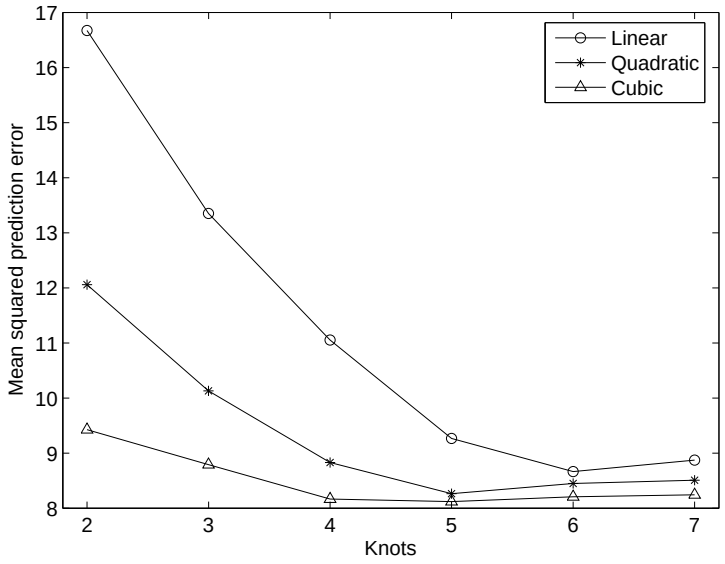
If one were to consider only the test prediction error, it would seem that the optimal spline to select is the cubic spline with 5 knots. However, other factors must also be taken into account. Firstly, the scale of the vertical axis is mean squared DMOS error; so differences of less than 1 unit are not very significant. Secondly, the final fit must always be a non-decreasing function of the degradation type; it is nonsensical to have decreasing DMOS (increasing perceived quality) with increasing degradation. And lastly, the function must preferably extrapolate well. This is not so important in the cloud cover case, since there is a finite maximum cloud cover (100%) and the highest observed datapoint is not too far from that maximum. However, in the blur and noise cases, it is more important. Generally it is desirable to have the model's degree of freedom as low as possible while allowing adequate freedom to fit the data.

Therefore, by considering figure 5.12(a) one can see that the linear fit does not improve markedly beyond 5 knots, while for the quadric fit the similar point is 4 knots and, in the case of the cubic fit, it is 3, or even 2, knots. Additionally, cubic fits of 3 and 4 knots extrapolated poorly, with the 3 knots fit decreasing and the 4 knots fit increasing too sharply. The fits as well as the test data for the first cross-correlation case, $k = 1$, for the linear 5-knot, quadratic 4-knot, and cubic 2-knot cases are shown in Figure 5.13. Given that the cubic 2-knot fit requires one less degree of freedom and gives a comparable fit, it was selected as the model to be used. Since the only 2 knots are at the end points, this is basically a single cubic polynomial fit.

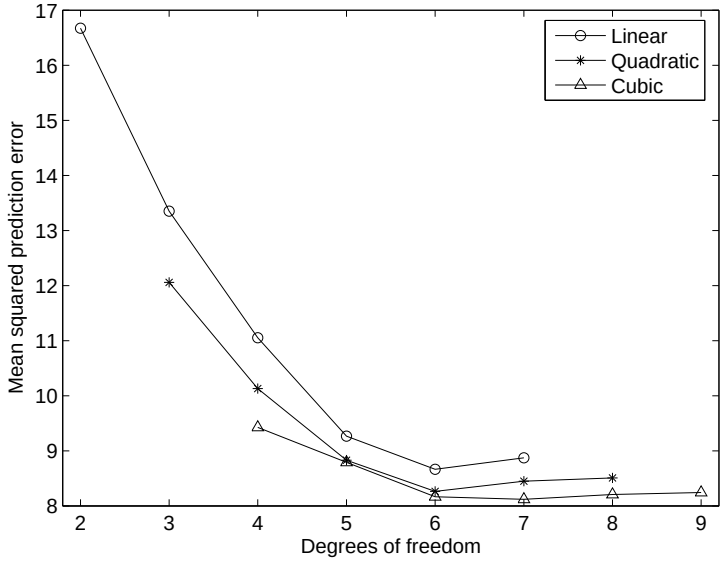
Blur axis

For the blur axis, where $X = [0, X_b, 0]^T$, the comparative test prediction error plot is shown in Figure 5.14. The linear 4-knot, quadratic 3-knot and cubic 2-knot fits all give approximately the same test error. This is not surprising since these models have exactly the same degrees of freedom.

By considering the individual fits in Figure 5.15, the cubic fit, 5.15(a), is eliminated since it extrapolates too poorly. While the quadratic fit (b), appears adequate, the linear fit (c) was chosen since it is the safest option from an extrapolation point of view; quadratic functions will grow unacceptably fast once outside the range of the training data.

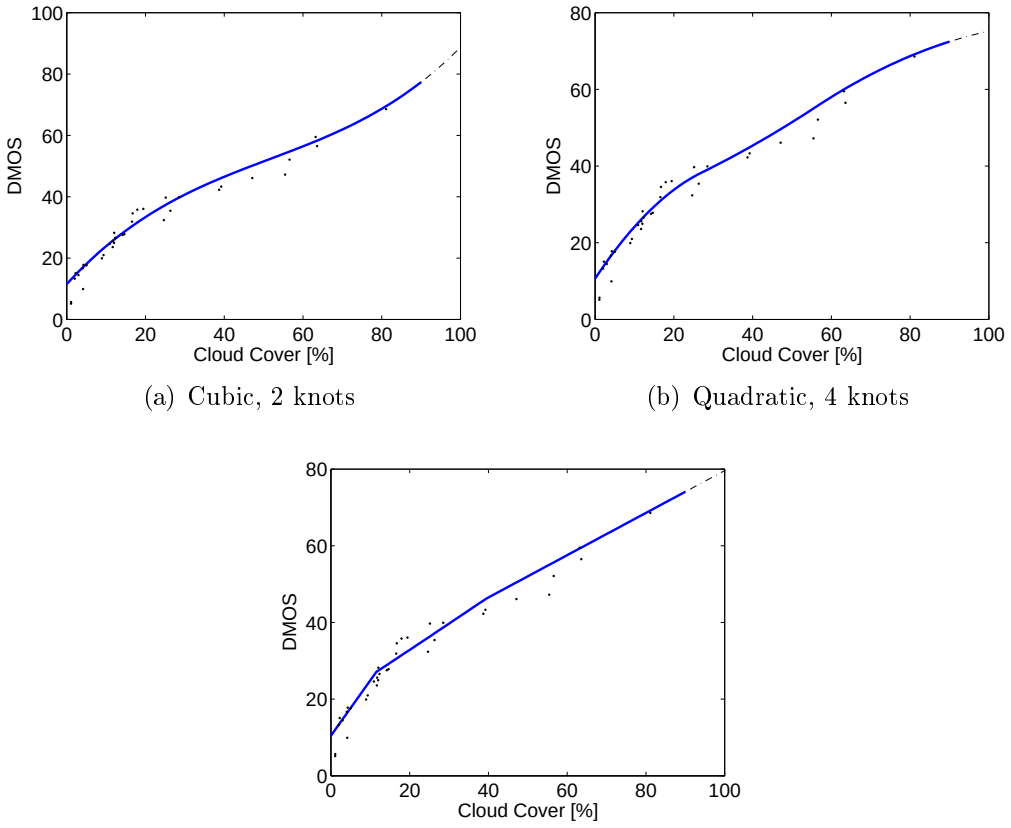


(a)



(b)

Figure 5.12: The effect of altering the polynomial order on the test prediction error of the spline cloud cover IQA model. By comparing (a) and (b) one can also see the effect that increasing the polynomial order has on the degrees of freedom.



(a) Cubic, 2 knots

(b) Quadratic, 4 knots

(c) Linear, 5 knots

Figure 5.13: Different spline fits on cloud data. The datapoints shown are from the test data, not training data. The dashed line on the end shows the extrapolation of the spline.

Noise axis

The noise data has higher variance than the other datasets, a fact which will be discussed further when the results are considered in section 5.4.2. However, this made fitting a model with high degrees of freedom difficult and unwise, as can be seen from the test prediction error plot in Figure 5.16. In this case the model with the lowest degrees of freedom is clearly the best choice.

Examining individual fits also confirmed this, with higher complexity models behaving unacceptably. The resulting linear fit is shown in Figure 5.17. Once again, only two knots are used so it becomes simple linear regression.

Central area

As will be discussed in section 5.4.2, the effect of noise in the central area of the input space, where all the elements of the vector X are non-zero, was not very significant. Additionally, the higher order spline fitting function in MATLAB®

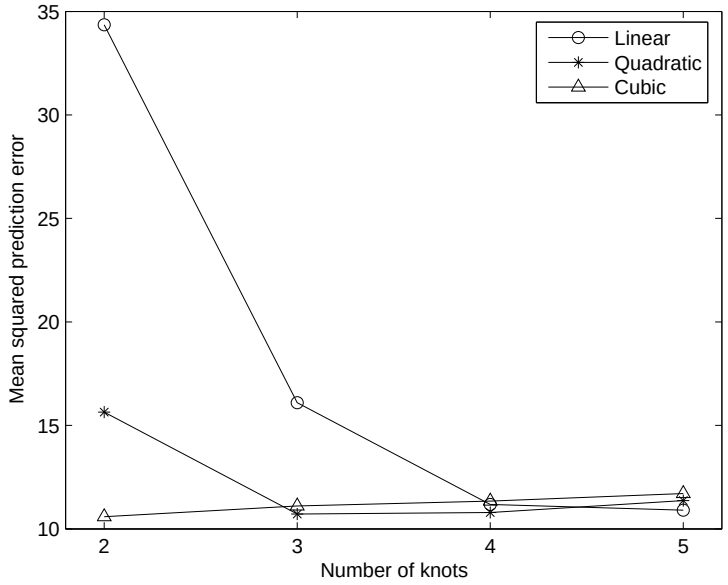
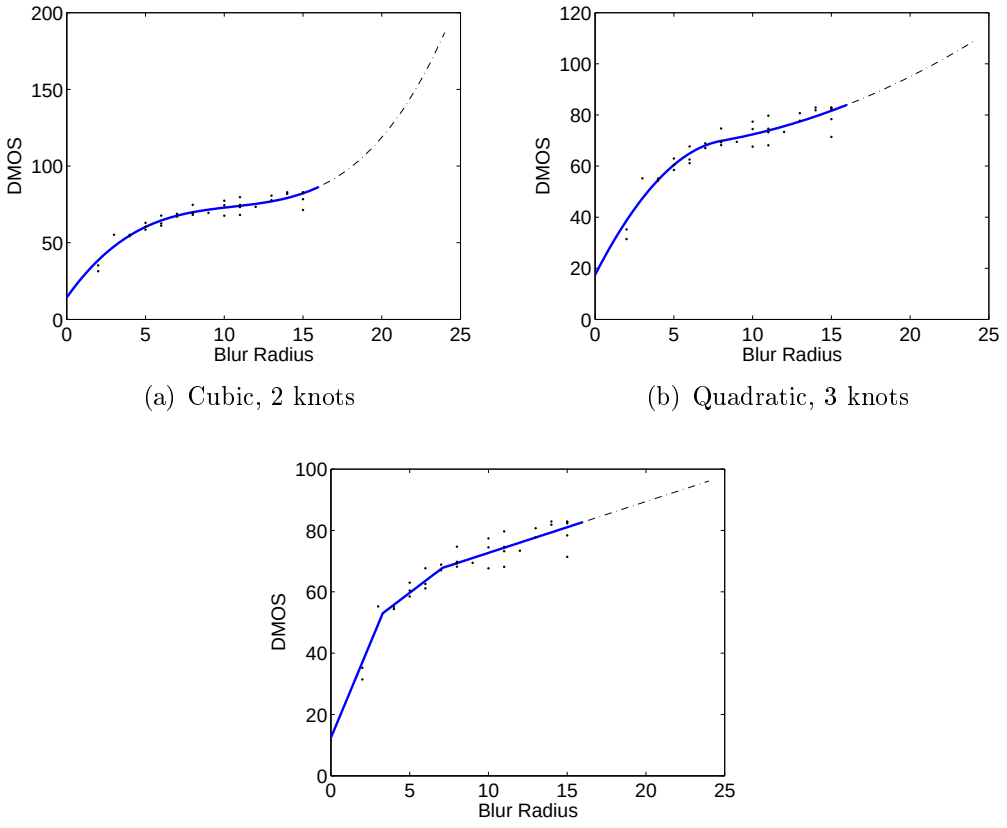


Figure 5.14: The prediction error of various spline models fit to blur data.

required that the input data be gridded. If the full three-dimensional input space were to be modelled, the 64 datapoints could not be divided into test and training data randomly as this would break the grid. To keep the training data gridded, the test data would have to come from specific planes, for example all points for which $X_b = 7$. However, this could considerably increase the testing prediction error, since one would be testing an area of the model where no training data points were nearby.

It was decided to ignore the effects of noise for the spline model of the central area. As analysis of variance tests in section 5.4.2 will show, this is not an unfounded idea. Furthermore, it enables proper separation of test and training data for cross-validation. Since noise is not modelled at all, the central area now consists of a data on a 4×4 grid with 4 datapoints (corresponding to 4 noise levels) at each grid point. For each of the $K = 4$ cross validation runs, a different point at each grid point can be selected. This preserves the gridded nature of the training set while ensuring suitable randomness in the selection of test data. Figure 5.18 shows one example division.

In addition, since it is desirable that the central area model blend smoothly with the axis plane models, data from the single variable sessions was also included into the training set for the central area. This data had to align with the existing grid, i.e., have cloud cover levels {10, 23, 37, 50} and blur levels {2, 5, 7, 10}. Since the cloud adding algorithm does not give precise cloud levels, data that was within 10% of the specified levels was accepted. Furthermore, care was taken to avoid polluting the test data with training data, so that the $k = 3$ and 4 cross validation sets could still be used to evaluate the performance



(a) Cubic, 2 knots

(b) Quadratic, 3 knots

(c) Linear, 4 knots

Figure 5.15: Different spline fits on blur data. The datapoints shown are from the test data, not training data. The dashed line on the end shows the extrapolation of the spline.

of the final model.

By averaging over the training data at each grid point, one can get a better idea of the shape of the curve being approximated, Figure 5.19. Undesirable decreasing behaviour is visible in parts of the average training data graph. As mentioned previously it is nonsensical to have decreasing DMOS with increasing degradation. Therefore this behaviour is ascribed to high variance in the collected data; if more samples could be collected at each point of the grid, the surface would be smoothly increasing. Unfortunately "the curse of dimensionality" means that it becomes increasingly difficult to collect enough data as the dimension of the input space increases. Also notice that in Figure 5.19, data from the single variable experiments has already been included into the training set.

Therefore, the model that one attempts to fit to the central area must have sufficient constraints, so that it remains monotonic.

Linear splines with 2 to 5 knots on each axis as well as quadratic splines

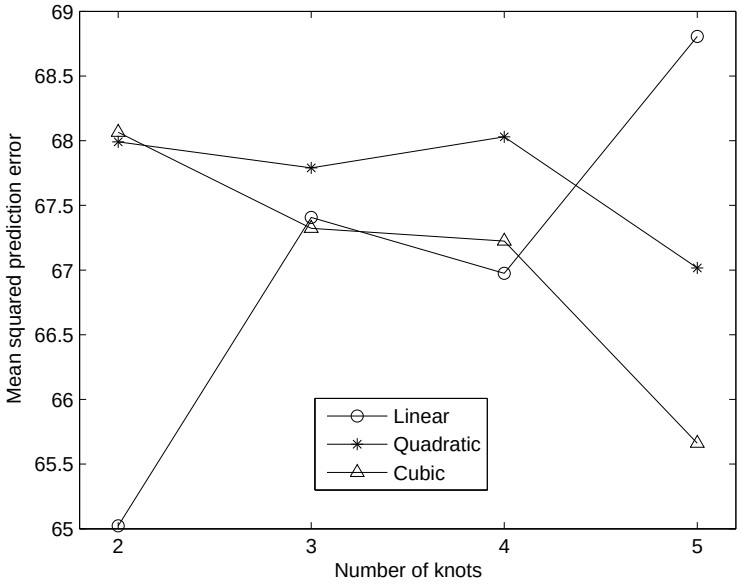


Figure 5.16: The prediction error of various spline models fit to noise data.

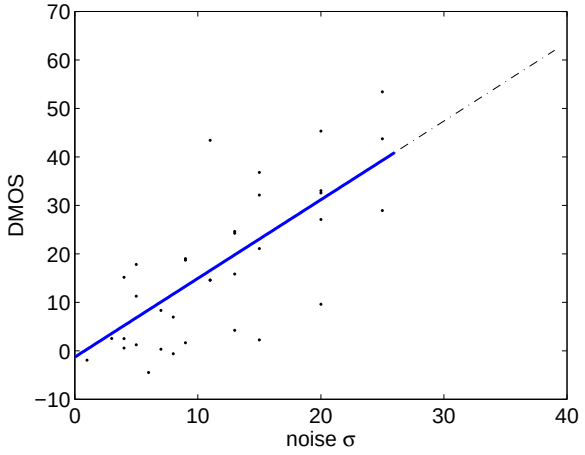


Figure 5.17: The linear regression noise fit. The dashed line on the end shows the result of extrapolation.

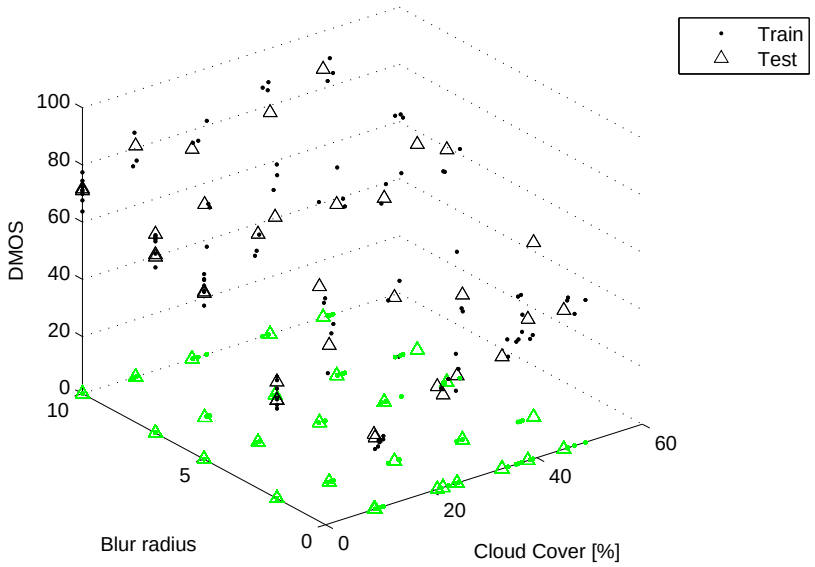


Figure 5.18: Division of the full factorial data into test and training sets for cross-validation. The green markers are a projection of the black markers onto the blur-cloud plane to aid in the spatial interpretation of the black markers.

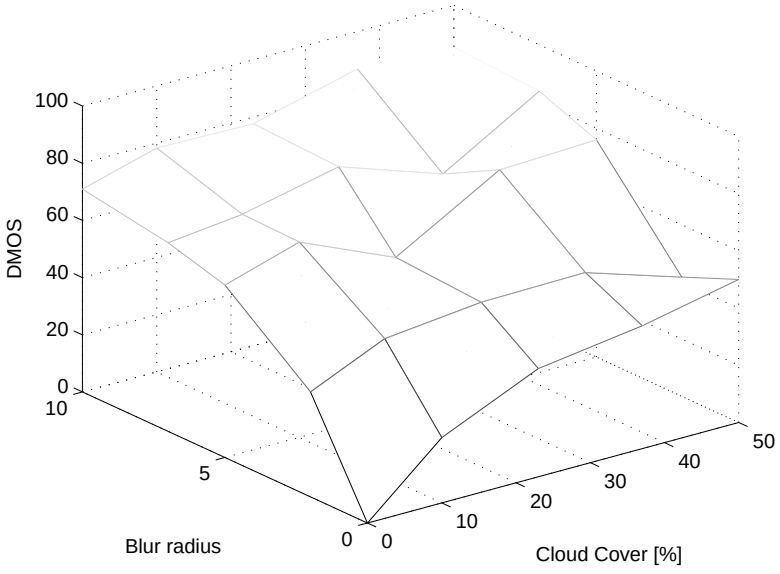
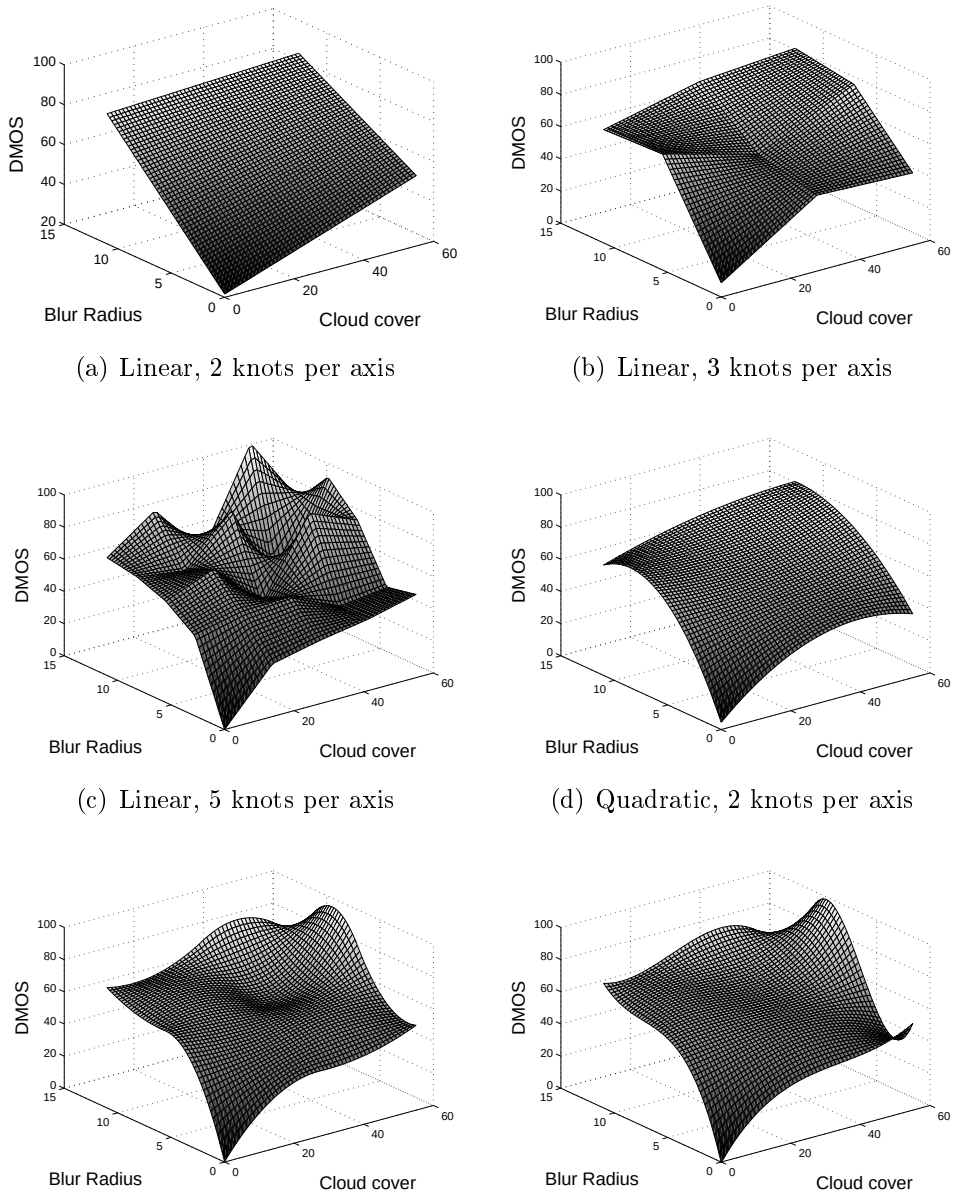


Figure 5.19: Average across training data. Note the unwanted dips in the graph.



(a) Linear, 2 knots per axis

(b) Linear, 3 knots per axis

(c) Linear, 5 knots per axis

(d) Quadratic, 2 knots per axis

(e) Quadratic, 3 knots per axis

(f) Cubic, 2 knots per axis

Figure 5.20: Different spline fits on central area data.

with 2 to 4 knots and cubic splines with 2 to 3 knots were evaluated. A selection of these fits is shown in Figure 5.20. Unfortunately the linear splines with one internal knot already show decreasing behaviour, 5.20(b). As the number of internal knots increase, it becomes worse, (c). The quadratic 5.20(d)-(e) and cubic splines, (f), show similar decreasing tendencies and it is clear that they will also extrapolate poorly.

By considering figures 5.20(a) and 5.20(b), one can see that 5.20(b), the

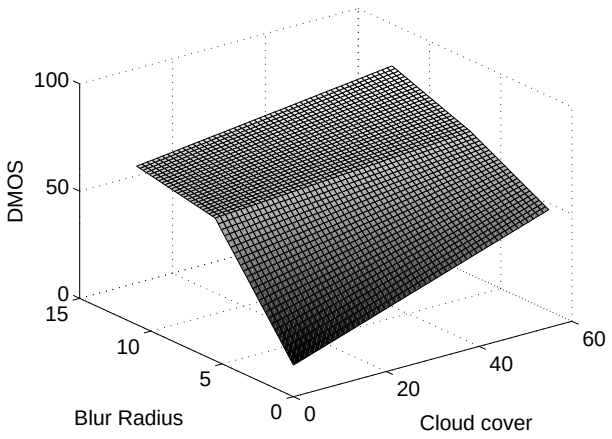


Figure 5.21: A spline model with unequal number of knots in the each axis. Note that monotonic behaviour is retained.

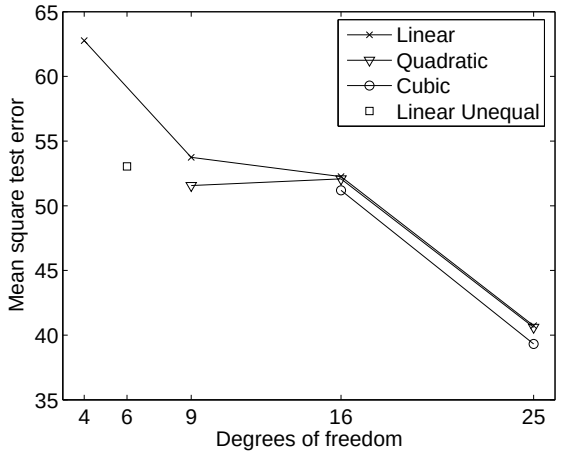


Figure 5.22: Prediction errors for different splines models fitted to the central area.

spline with one internal knot, is monotonic in the blur axis, while 5.20(a) is monotonic in the cloud axis. This led to the idea of a spline model with 3 knots in the blur axis and 2 knots in the cloud axis. The resulting spline is shown in Figure 5.21. It appears to be a good compromise: the extra freedom in the blur axis is used to model the steep initial increase and subsequent flattening of the blur data while the cloud data is more linear. Monotonic behaviour is retained.

This model also performs relatively well when evaluating the prediction error, as shown in Figure 5.22. Note that it has 6 degrees of freedom and has better prediction results than the linear 2-knots per axis model and similar results to the linear 3- and 4-knots per axis models, as well as the quadratic 2- and 3- and cubic 2-knots per axis models.

The test error continues decreasing for the non-monotonic higher order models. As mentioned previously, this is because the dataset itself is non-monotonic due to insufficient number of observations. Given the data scarcity the linear model is a good choice that is monotonic and will extrapolate well.

Combining individual models

There are now four individual models for different areas of the input space that must be combined into a single model. Three two-dimensional models ($f_c(X_c), f_b(X_b), f_n(X_n)$ for the cloud, blur and noise axes) and one three-dimensional model ($f_{centre}(X_c, X_b)$ for the central area) must be combined in the four-dimensional model space. Recall that the input space is three-dimensional so the input–output model is four-dimensional. Furthermore it is desirable to have a smoothly varying function between the different two-dimensional parts and the three-dimensional area in the middle.

A weighed sum approach is followed with weights varying between 1, where a model is fully active, and 0 where it is inactive. At every point $x = [x_1, x_2, x_3]^T$ in the input space $[X_c, X_b, X_n]$ two intermediate weights are calculated:

$$\begin{aligned} w_i &= \begin{cases} 1 - \frac{x_i}{d_i} & 0 < x_i < d_i \\ 0 & x_i > d_i, \end{cases} \\ \text{where} \quad & d = [10, 2, 5]^T \\ \text{and} \quad & i = 1, 2, 3. \end{aligned} \tag{5.3.5}$$

The vector representing the smallest observed input point from the full factorial experiment is d. For points closer than d to the axes, interpolation must be applied. Upon considering the definition of w_i in equation (5.3.5) one can see that w_i has a range $\{0 - 1\}$. Specifically, w_i is 1 for $x_i = 0$ (the axis plane, where one of the two-dimensional models will be active) and 0 for $x = d$ (where the three-dimensional model is to become active). These intermediate weights are converted into final weights as follows:

$$\begin{aligned} \gamma_c &= w_2, \\ \gamma_n &= w_1 \times w_2, \\ \gamma_b &= w_1, \end{aligned} \tag{5.3.6}$$

where γ_c is the weight that decreases as the distance from the axis $[X_c, 0, 0]$, where $f_c(X_c)$ should be active, increases. Since $f_{centre}(X_c, X_b)$ is only a function of cloud cover and blur extent, only these two variables are used when measuring distance from the axes. Thus distance from the axis $[X_c, 0, 0]$ is determined only by x_2, the value of a point in the X_b direction. This value is therefore converted into intermediate weight w_2 and finally a cloud axis weight

γ_c. These final weights are used to construct the final, combined function, $\hat{f}_{spline}(X)$, in a cumulative manner:

$$A = \gamma_c f_c(X_c) + (1 - \gamma_c) f_{centre}(X_c, X_b) \tag{5.3.7}$$

$$B = \gamma_b f_b(X_b) + (1 - \gamma_b) A \tag{5.3.8}$$

$$C = \gamma_n f_n(X_n) + (1 - \gamma_n) B \tag{5.3.9}$$

$$\hat{f}_{spline}(X) = \max(f_n(X_n), C). \tag{5.3.10}$$

Equation (5.3.7) blends the cloud model with the central model, (5.3.8) blends the result with the blur model and (5.3.9) blends the result of that with the noise model. Finally (5.3.10) is necessary to ensure monotonic behaviour in the centre of the graph when dealing with high noise values, since noise is not modelled by $f_{centre}(X_c, X_b)$.

It is enlightening to investigate some example plots. This is done when the results are discussed in section 5.4.3 on page 165. Furthermore, the two remaining cross validation data divisions, $k = 3$ and 4, were used to test the prediction performance of the model. These results are also presented in section 5.4.3.

5.3.6 Creating a neural network model

As mentioned in section 5.2.4 there are four common regularisation options available to prevent overfitting. These are critically evaluated in appendix B. The conclusion is that the most reliable way to prevent overfitting in neural networks is manual selection of model complexity using validation data.

A single layer feed forward model, as discussed in section 5.2.4, is appropriate for the image quality assessment data. No time dependencies exist and, therefore, feedback paths are unnecessary. Furthermore, there is no hierarchical ordering in the data that would benefit from using multiple layers.

The same cross-validation setup used for the spline fitting was used for the neural networks: the $k = 1$ and 2 splits are used for model selection and the $k = 3$ and 4 splits for testing. Following the recommendations discussed in section 5.2.4, the networks were trained 10 times for each new random starting position of the weights. The input and output data was normalised prior to training. The number of units in the hidden layer was varied to determine the optimal model complexity. The resulting errors for the training data and validation data are shown in Figure 5.23. While the training error decreases and stabilises, the validation error reveals that considerable overfitting occurs at a high number of hidden units. Based on the validation data, a network with 5 hidden units was selected.

Visualisations of the model, as well as performance measurements are presented section 5.4.3.

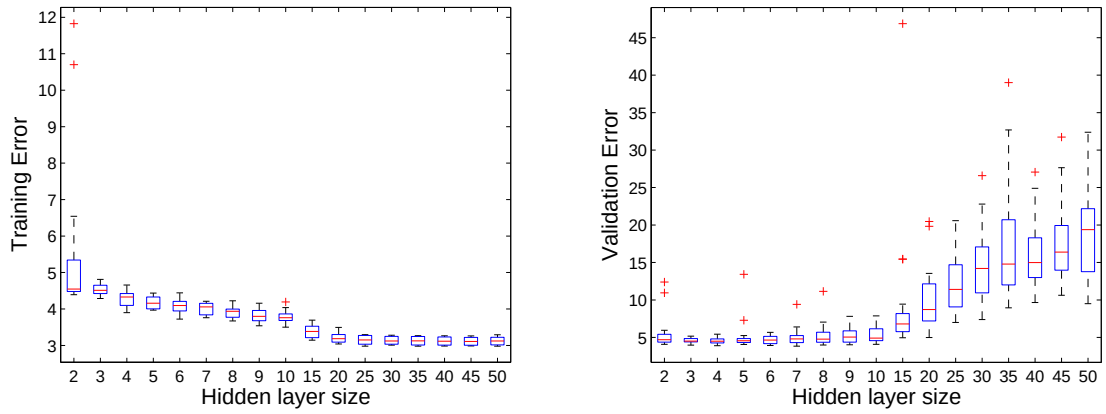


Figure 5.23: Prediction errors encountered during neural network training.

5.3.7 Hypothesis tests

Model selection was based on the relative performance of the two models: the models were provided with known true levels of the three input features and the models' ability to match the subjective quality scores compared. References [97] and [9] recommend statistical hypotheses tests, similar to those described in section 2.2.1 on page 26, be done when the results of the fitted models are compared. Since the performance of image quality models are similar, there is a chance that the difference in observed performance is just the result of sampling error: given another experiment, the results might be reversed. Hypotheses tests allow one to test whether a statement about a population parameter is true, to a specified statistical significance [70].

The hypotheses test, recommended in [9] and followed in [97], to discriminate between model performances is based on the F-test. The F-statistic, a ratio of residual variances, is computed. The residuals for both models are computed according to $(y_i - \hat{f}(x_i))$ across all the $\{x_i, y_i\}$ validation data observations. For each model, the variance of these residuals is computed. F-statistic is the ratio these two variances.

To test whether differences between the two models are significant to a specific level, the F-statistic must be greater than the relevant F-value for an F-distribution curve with the correct degrees of freedom. The shape of the F-distribution curve is determined by two degrees of freedom, the degree of freedom for the numerator and the degree of freedom for the denominator. Since the $k = 3$ and 4 cross validation test sets have 121 samples each, the residuals were calculated at 242 observations for each model. The F-distribution curve that must be used has degrees of freedom, $df = (242, 242)$. The 5% statistical significance F-values for selected F-distribution curves can be found in tables commonly included in statistical texts [70]. The F-value for 5% statistical significance is the point x on the F-distribution curve where the area under

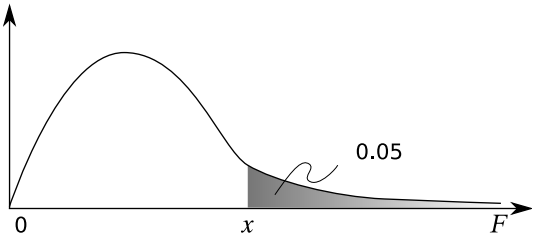


Figure 5.24: How the 5% F-value, x in the figure, is determined.

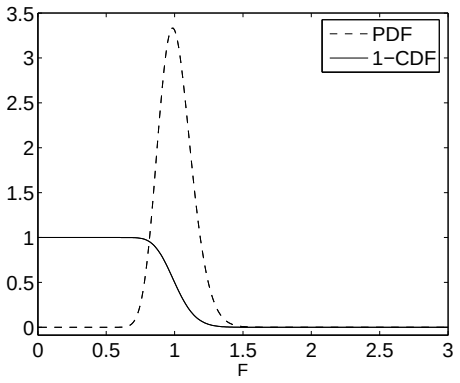


Figure 5.25: Probability density function and cumulative distribution functions for the F-distribution with degrees of freedom $df = (242, 242)$.

the curve, to the right of x is equal to 0.05, as shown for a general F-curve in Figure 5.24.

MATLAB was used to compute the precise F-values. Figure 5.25 shows a probability density function of an F-distribution with $df = (242, 242)$. To aid with the area calculations, the graph for $(1 - CDF)$ is also shown, where CDF is the cumulative distribution function. Since the cumulative distribution function is the integral of the probability density function, the $(1 - CDF)$ curve in Figure 5.25 represents the remaining area to the right of a point on the PDF graph. The results of the hypotheses test are presented in section 5.3.7.

5.3.8 Testing the integrated system

After selecting a model based on its performance for known true levels of input features, the different parts of the system were integrated and tested together in a single test. The performance of the complete system was evaluated using the 242 artificially degraded images from the test cross validation sets $k = 3$ and 4. Inputs for the spline model were computed by applying the three feature estimation algorithms to these images.

Values for the two required feature estimation parameters (cloud threshold

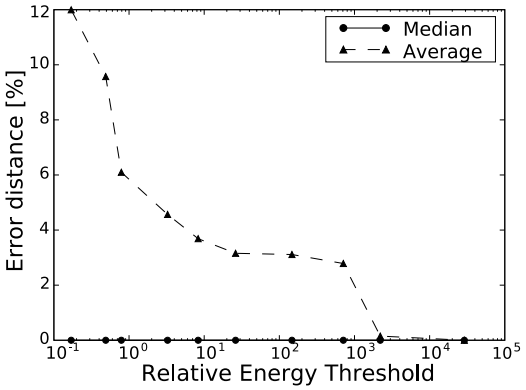


Figure 5.26: The effect of the relative energy threshold, E_r, on classification error.

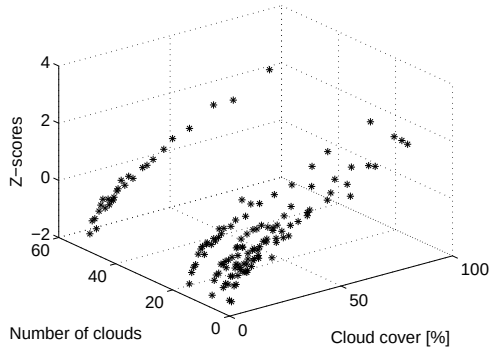
and the relative energy threshold, E_r, from equation (4.3.14) on page 109) were derived from the training cross validation sets $k = 1$ and 2 to avoid contamination of test data. Cloud thresholds were derived using methods previously described in section 2.2.1. When adding clouds to the images (see section 5.3.2 on page 135), precise cloud masks could be generated since the absolute location of all cloudy pixels added was known. These masks were used as the ground truth when training thresholds and measuring performance. Since the spectral characteristics of the artificial clouds are not important and the images contained only three visual channels, it was decided to simply use the blue channel for cloud detection.

To select a single value for E_r, its effect on the average classification error was investigated, as shown in Figure 5.26. The average relative error initially decreases rapidly with E_r and starts to flatten out. Since the number of rejected images steadily increases as E_r is increased (requiring more energy in the true cepstral peak and becoming less tolerable of spurious peaks cause by noise), the goal was to select an E_r value as small as possible to reject the least amount images but large enough to avoid the big estimation errors. Since the average error starts to grow rapidly as E_r is reduced to less than 1 (while median error stays zero, which implies large errors occurring in a few images), a value of $E_r = 1$ was selected.

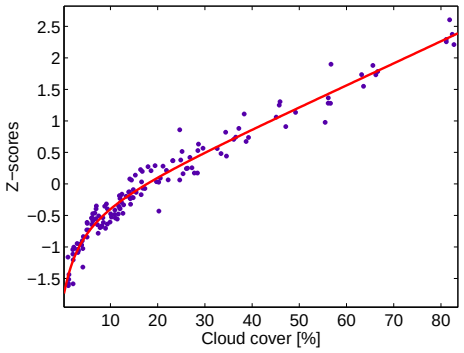
5.4 Results

5.4.1 Cloud dispersion

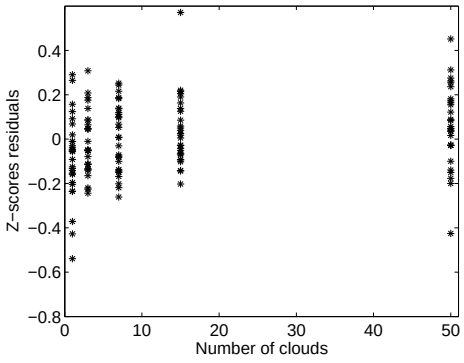
The clouds that were added to the test images possessed varying levels of dispersion, corresponding to the number of clouds specified in the cloud adding algorithm. In section 2.2.3 on page 28 a cloud dispersion measure was motivated. Using the subjective quality experiment, justification for this measure



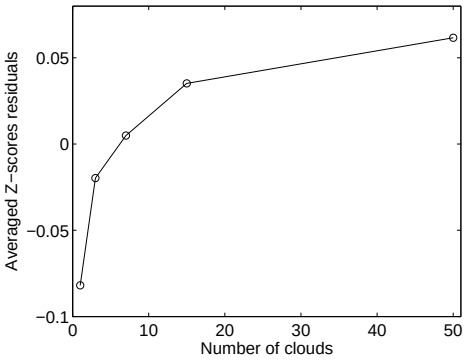
(a) Two variables influencing Z-scores.



(b) Rational fit on projection of data.



(c) Residuals.



(d) Averaged residuals.

Figure 5.27: The process followed in an attempt to observe the effect of cloud dispersion on image quality.

was sought in perceived human image quality. By plotting the scores from the cloud cover experiment as a function of both cloud cover and number of clouds, a graph like the one in Figure 5.27(a) can be generated. The primary variable determining quality score is clearly cloud cover. Note that these tests were applied to the cloud data before it was mapped to realigned DMOS to allow it to be combined with data from the other experiments. Since the mapping form Z-scores to DMOS is linear, it does not affect the conclusions drawn here.

In an attempt to see if the number of clouds had any effect on the quality score, it was desirable first to remove the effect of the cloud cover from the output. This was done by fitting rational function, $f(x) = (ax^3 + bx^2 + cx + d)/(x + e)$, to what is effectively a projection of Figure 5.27(a) onto the cloud cover plane as in Figure 5.27(b). By considering the residuals of this fit at the different number of cloud levels, the graph in (c) is obtained. Any effect that the number of clouds might have is still difficult to discern. By averaging the data at each of these points, the effect of the number of clouds can be seen in

Figure 5.27(d). Here there is a monotonic increase in the Z-score with number of clouds, which translates to a monotonic decrease in perceived image quality with increased dispersion.

However, the effect is at a very small scale relative to the effect of cloud cover. The difference between the Z-scores at the extremes of dispersion is 25 times smaller than Z-scores at the extremes of cover. To test the significance of a variable [50] recommends that an analysis of variance (ANOVA) be conducted. Analysis of variance is a procedure that can be used to test the null hypotheses that the means of two or more populations are equal [70]. MATLAB's `anova1` was used to analyse the Z-scores from the cloud experiment, divided into four sets based on the number of clouds. The ANOVA results confirm that it is not possible to distinguish between the sets, i.e., their means are equal.

Based on these results is was decided not to model the effect of cloud dispersion in the image quality assessment model.

5.4.2 DMOS scores

Individual variable sessions

Figure 5.28 shows the individual realigned DMOS scores obtained during the three different single variable experiments. By investigating the data from the individual sessions separately, one can discern non-linearities in the mapping of degradation levels to perceived image quality and comment on the spread of the data.

The noise scores show a great variance in the DMOS values for a specific noise level σ_n. At low σ_n up to as high as $\sigma_n = 8$ negative DMOS values can be found. While this might seem to indicate that subjects perceived the degraded image to be of a slightly better quality than the original, the average DMOS values across all images at low noise levels are approximately zero. Therefore, at low noise levels, noise is imperceptible. The relatively greater variance of the noise DMOS values compared to those of other degradation types can also be attributed to the fact that perception of noise is influenced by image structure. Noise becomes less discernable in the presence of a busy spatial signal structure; conversely it is more perceivable in images that contain large homogeneous areas. This is the same characteristic used by noise estimation algorithms and modelled by HVS models and, to a lesser extent, SNRs. Since information on image structure is not estimated, it is good that several different input images were used at each noise level, to ensure the generality of the resulting noise model.

The blur data has the strongest non-linear behaviour of the three sets. It indicates that the perception of the blur defocus radius increases sharply and then flattens off. Therefore, after the knee of the graph has been reached, the images are so blurred that it becomes difficult for observers to discern between

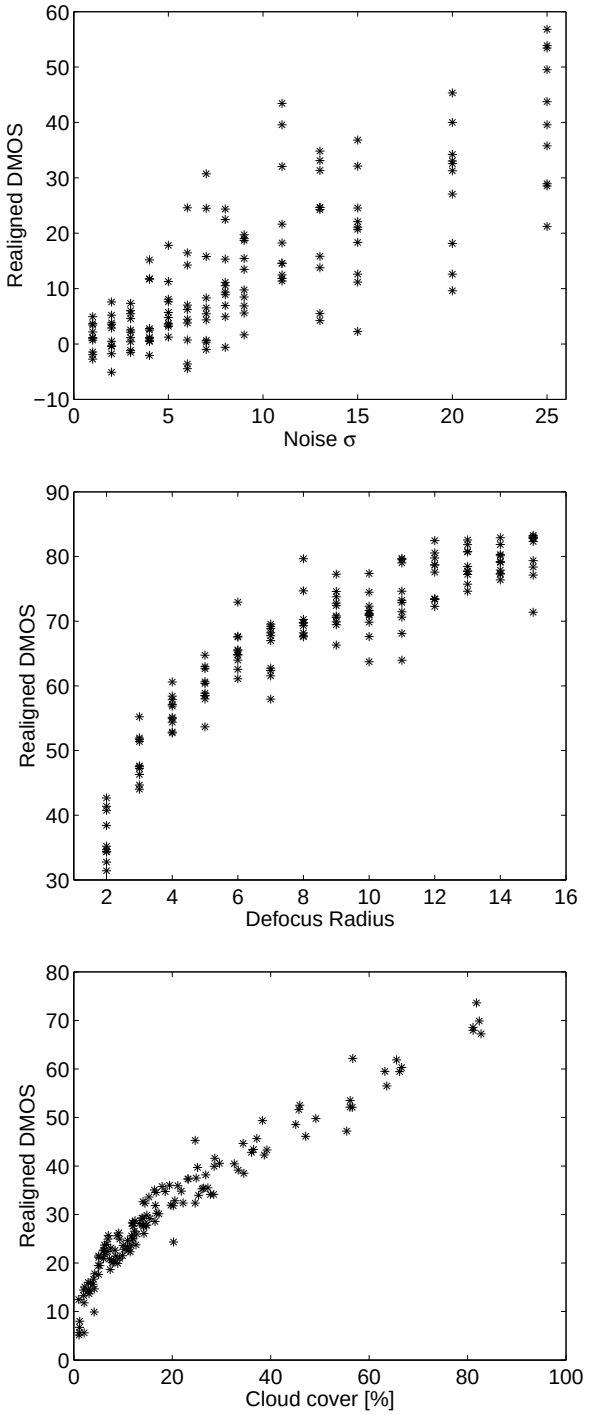


Figure 5.28: The results of the single variable sessions of subjective IQA experiment.

Table 5.3: The average zero mean (AZM) main effects of each of the single variables, cloud cover, noise σ_n and defocus extent, R.

σ_n	AZM
5	-0.55
12	-2.3
18	1.2
25	1.6

Cloud	AZM
10	-1.1
23	-4.1
37	0.48
50	4.8

Blur R	AZM
2	-18
5	0.36
7	3.9
10	14

different blur levels; all images are perceived as having approximately equal, poor, image quality. In images with little spatial structure blurring should be more difficult to perceive in principle (it is definitely more difficult to estimate algorithmically), but, in practice, the variance of the graph is much less than the noise graph. Image structure affects the perception of blur less than the perception of noise.

The cloud data appears linear for high cloud cover levels, but has some non-linearity at low cloud cover. The initial sharp increase might be attributed to a sensitivity to the *presence* of any clouds. No flattening occurs as with blur; higher levels of cloud cover are always proportionally worse than lower levels of cover.

Full factorial experimental data

Two analysis methods were applied to the full factorial experimental data to determine the significance of the different variables and primary interactions.

Firstly, the method described in [50] was used to analyse the effect of each individual variable, one at a time. It consists of subtracting the global mean from the data (making it zero mean) and then averaging the data across the variables not being considered. The resulting spread of the output levels as a function of the variable under consideration is contemplated. It gives an indication of the relative importance of the input variable.

These averaged zero mean output levels as a function of input variable are presented in Table 5.3. It is clear that blur input dominates the output response with a strong monotonic increase. The relative effect of cloud cover is approximately three times smaller than that of blur. The non-monotonic behaviour here is undesirable and is due to the relatively small influence of cloud cover. Hence, random changes in blur perception can affect the outcome more than small changes in cloud cover. Lastly the effect of noise is the smallest, approximately 10 times smaller than that of blur. Although the trend is positive, the effects of more dominant variables have caused non-monotonic behaviour.

Alternatively, N-way ANOVA can be used to analyse the relative effect of N input variables. Its use for determining if there are significant primary interactions between variables in a full factorial experiment is recommended

Table 5.4: The results of a 3-way ANOVA.

Source	p-Value
noise	0.3022
blur	1.02×10^{-12}
clouds	0.0047
noise × blur	0.1374
noise × clouds	0.0142
blur × clouds	0.0005

in [50]. Recall that the point of conducting a full factorial experiment is to determine if there are significant primary interactions. The resulting p-value for each variable (or primary interaction) can be interpreted as the probability that the outputs for the different input levels of the variable under consideration could be the result of taking random samples from the same population, i.e., the probability that the input variable has no effect on the output. Ideally $p < 5\%$ for the variable to be pronounced significant with 5% statistical significance. The p-values from the 3-way ANOVA is presented in Table 5.4. Neither the effect of the noise variable, nor the primary noise-blur interaction, is statistically significant to the 5% level. The other two interactions are significant. This confirms the usefulness of conducting a full factorial experiment to model the central area of the input space: cross-coupling between variables does occur.

Realignment of scores

To depict the result of the outlier rejection and realignment procedures on the three single variable sessions, the mapping from raw input difference scores to realigned DMOS is shown in Figure 5.29. The most noticeable is the relative shift in position of the blur data. This confirms the necessity of the realignment; when participants' judgements were calibrated to use the entire scale, the best unfocused image had an average difference score of about 20. However, in comparison to the other degradations from the realignment experiment, the best out-of-focus image has a DMOS of 30. This shift is not just confined to the bottom of the blur scale either; all blurred data is shifted relative to the other two degradation types. The effect of the double stimulus method must also be taken into account: when comparing the reference image to the blurred image in a double-stimulus situation, the effect of the blurring is more pronounced than in a single-stimulus experiment.

The same mapping for the full-factorial data is depicted in Figure 5.30. One can see that the main effect of this mapping is that the low values have been slightly lifted. This makes sense: even images with slight degradations of all three types present will have worse perceived quality when compared to images with only one degradation type present. What is more surprising, is the

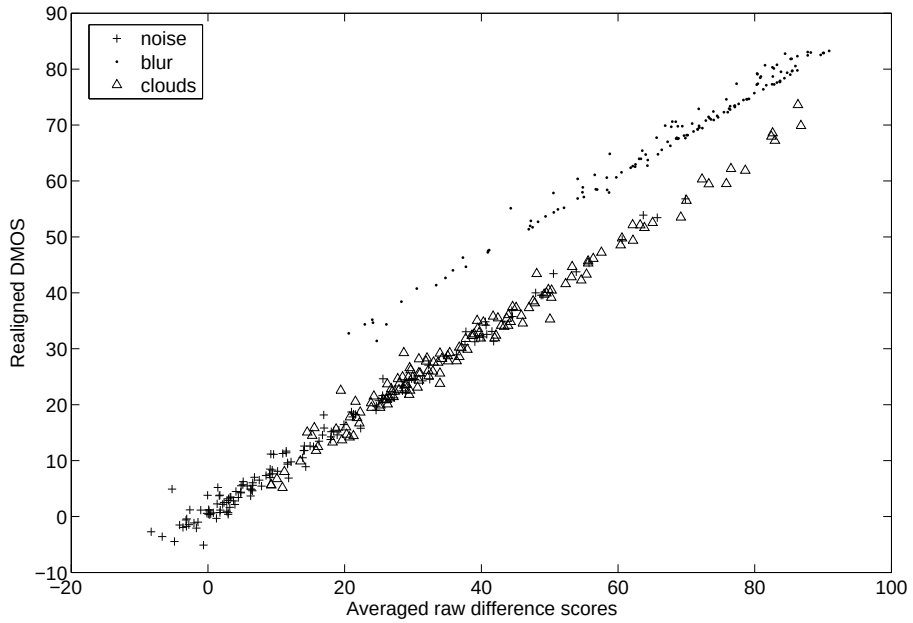


Figure 5.29: The relationship between raw difference scores and the realigned DMOS values for the individual variable session.

lack of movement of the data at the top end of the scale. This can be ascribed to the dominating effect of blurring, revealed in the previous section on page 163. In images with the maximum amount of blur from the full factorial experiment, the blur had the greatest influence on the quality score. When compared to images degraded only by severe blurring the relative quality score is similar. However, one must remember that the greatest defocus extent in the full factorial experiment is 10 pixels while in the blur experiment it is 15 pixels, it is therefore not unreasonable for these images to have similar values.

5.4.3 Comparison between models

Visual comparison

It is enlightening to attempt to visualise both the spline model $\hat{f}_{spline}(X)$ and the neural network model, $\hat{f}_{nn}(X)$. Since the model space is four-dimensional it is difficult to represent. The best approach is to keep one input variable constant and generate surface plots in three dimensions. Some example plots were generated.

Figure 5.31 shows a comparison between the training data and the resulting composite spline model $\hat{f}_{spline}(X)$ when $\sigma_n = 0$. Notice the smooth transitions from the central area model to the models used in the axis planes. This shows that the weighting method devised in section 5.3.5 from page 155 was

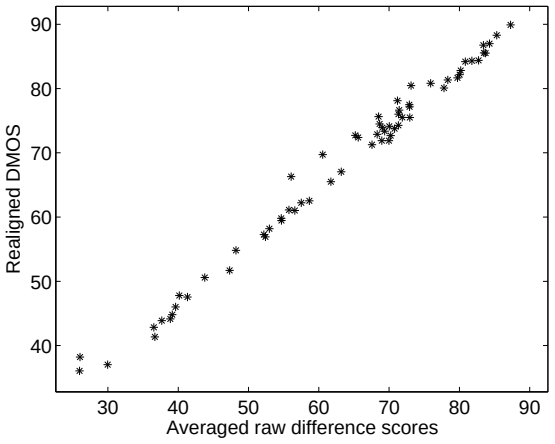
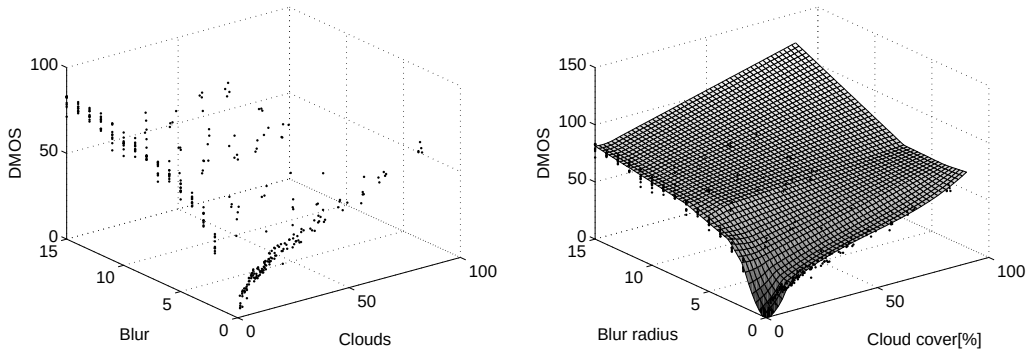


Figure 5.30: The relationship between raw difference scores and the realigned DMOS values for the cross-coupling session.

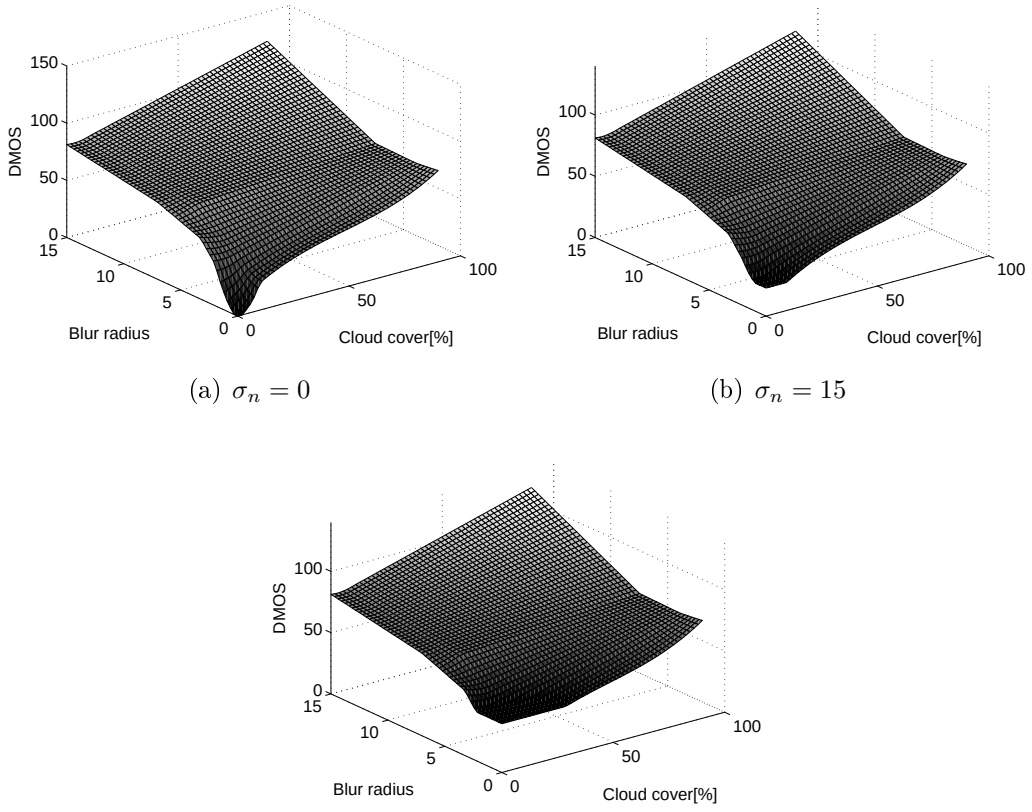


(a) (b)

Figure 5.31: A comparison between the training data and resulting surface. (a) shows all the training data, bar that from the noise only session. In (b) the $\hat{f}_{spline}(X)$ surface for $\sigma_n = 0$ has been superimposed.

successful. However, the surface deviates slightly from the cloud data at low levels of cover. This might be improved by altering the cumulative manner in which the graphs are combined in equations (5.3.7) to (5.3.10). The good extrapolation performance of the piecewise linear model used in the central section is also evident.

By keeping the noise level constant and varying the cloud cover and blur extent, the different surfaces in Figure 5.32 can be generated. These are intuitively interpretable. As the noise levels are raised from Figure 5.32(a) to (c), the best possible quality score an image can receive decreases. Furthermore, the area of the surface where that score is determined wholly by noise increases.



(a) $\sigma_n = 0$

(b) $\sigma_n = 15$

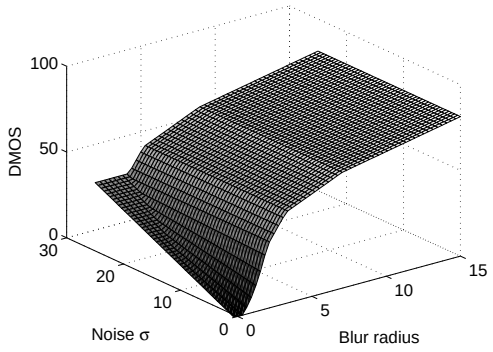
(c) $\sigma_n = 25$

Figure 5.32: Surfaces of $\hat{f}_{spline}(X)$ at fixed noise levels.

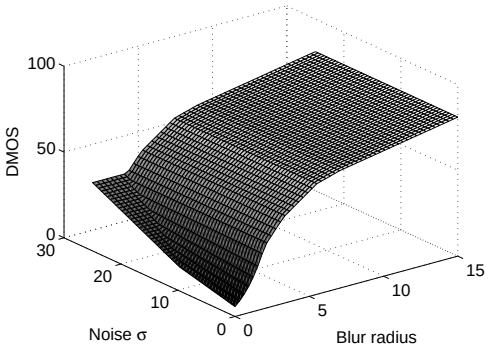
Similarly, Figure 5.33 shows the surfaces generated by keeping the cloud cover constant at certain levels while varying blur and noise. When cloud cover is at 0%, 5.33(a), the surface is a weighed combination of the two single variable curves $f_n(X_n)$ and $f_b(X_b)$. However, as the cloud cover increases, Figure 5.33(b) to (c), the surface used to model the area where $X_n > 0$ and $X_b > 0$ gradually changes shape to a slice of $f_{centre}(X_c, X_b)$ with X_c at a constant level. Also, as cloud cover increases, the DMOS floor lifts: the best possible score an image can receive decreases. Eventually cloud cover begins to dominate the effect of noise in a blur-free image, Figure 5.33(d).

Similar graphs can be generated for constant blur levels. However, due to the strong relative weight of blur discussed in section 5.4.2, the effect of noise is quickly dominated by the effect of blur, Figure 5.34.

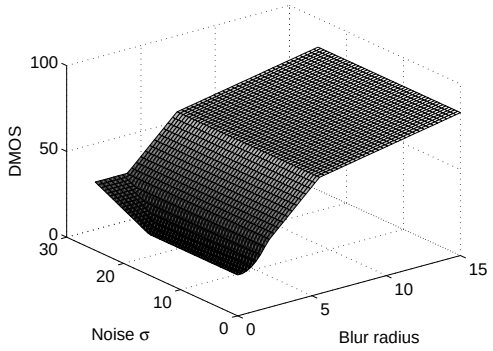
For the neural network model the same approach is followed to generate the surfaces: keep one input variable constant and vary the remaining two. In Figure 5.35 the noise level is kept constant. The results are broadly similar to Figure 5.32. In 5.35(a) one of the tansig activation functions models the non-linearity in the blur axis particularly well. The more linear structure of the



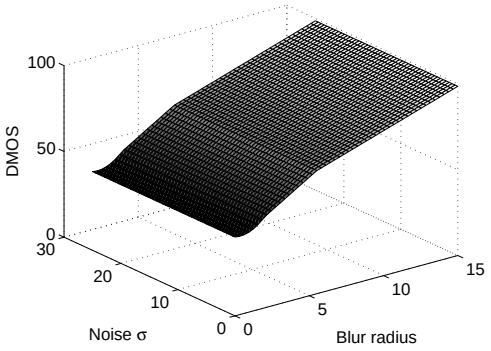
(a) Cloud cover = 0%



(b) Cloud cover = 5%

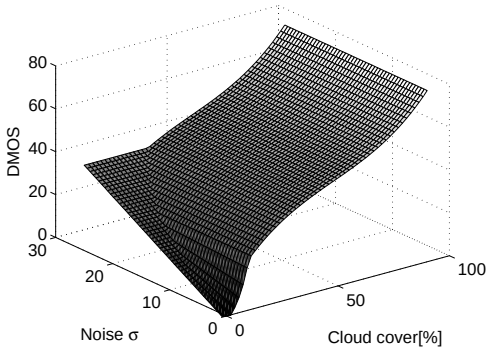


(c) Cloud cover = 10%

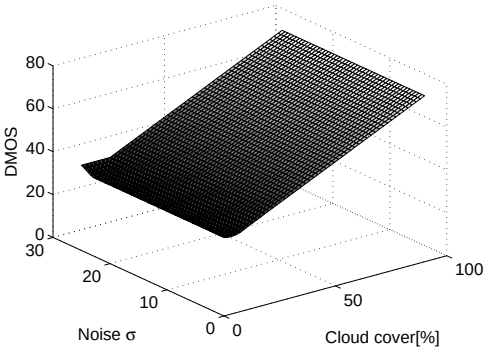


(d) Cloud cover = 40%

Figure 5.33: Surfaces of $\hat{f}_{spline}(X)$ at fixed cloud cover levels.

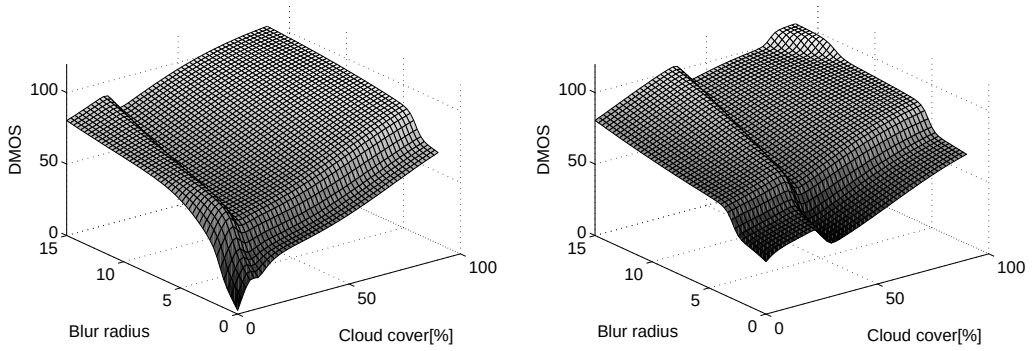


(a) Blur radius, $R = 0$



(b) Blur radius, $R = 2$

Figure 5.34: Surfaces of $\hat{f}_{spline}(X)$ at fixed defocus extent levels.



(a) $\sigma_n = 0$ (b) $\sigma_n = 10$

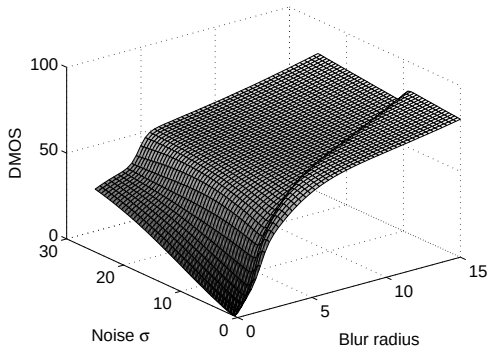
Figure 5.35: Surfaces of $\hat{f}_{nn}(X)$ at fixed noise levels.

cloud axis data with its slightly sharp non-linear increase at cloud cover levels is also visibly modelled. However, there is unwanted non-monotonic behaviour at some of the 'ridges' in the surfaces. This is caused by the tansig activation functions and is unavoidable.

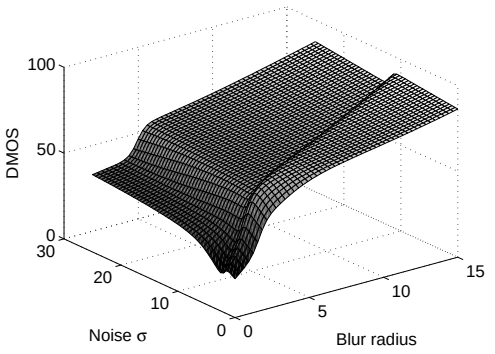
When examining $\hat{f}_{nn}(X)$ at fixed cloud cover levels, Figure 5.36, the results are again similar to those for $\hat{f}_{spline}(X)$ from Figure 5.33. As the cloud cover increases, the central area flattens and the effect of the noise variable decreases. Once again there are unwanted non-monotonic areas, for example at about $\sigma_n = 10$ on the noise axis of (c).

In the constant blur-level surfaces, Figure 5.37, the effect of noise is also quickly dominated by that of blur in a manner similar to 5.34.

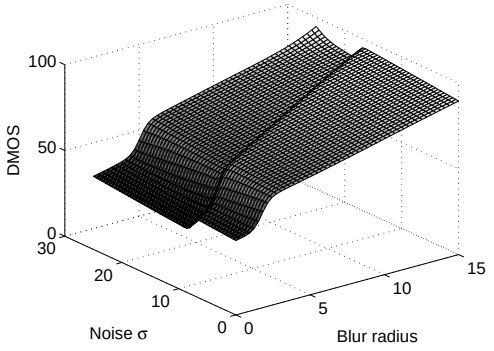
Care was taken during the construction of the spline model, $\hat{f}_{spline}(X)$, to select the sub-models, $f_c(X_c), f_b(X_b), f_n(X_n)$ and $f_{centre}(X_c, X_b)$, so that they would extrapolate well. This is ensured by relying mostly on linear, or piecewise linear, models. It is important since the training data covers a finite region of the possible input space and one cannot guarantee (except for cloud cover) that the input variables will stay in that space. Therefore the extrapolation ability of the neural network model was investigated visually. Since cloud cover will not have to be extrapolated beyond 100% a surface of constant cloud cover is presented here (although other surfaces were also investigated). Figure 5.38 shows the surface for zero cloud cover extrapolated to noise levels $\sigma_n = 80$ and blur levels $R = 50$. The surface extrapolates well with increasing blur levels, continuing to increase, but at a reduced rate. This matches the behaviour of the collected data. However, in the noise axis the surface behaves in a similar way although the noise data did not suggest such a trend. Here, a linear model would have been more appropriate.



(a) Cloud cover = 0%

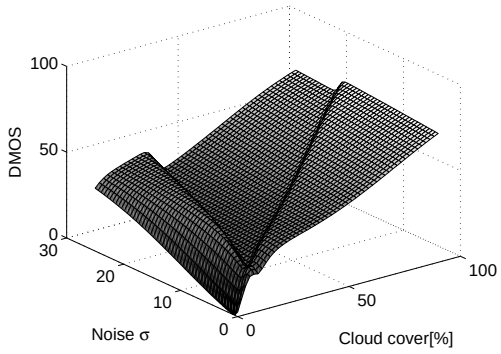


(b) Cloud cover = 10 %

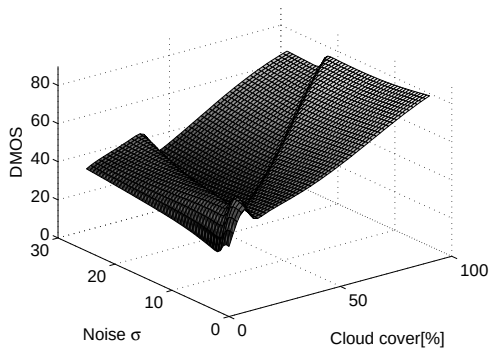


(c) Cloud cover = 40%

Figure 5.36: Surfaces of $\hat{f}_{nn}(X)$ at fixed cloud cover levels.



(a) Blur radius, $R = 0$.



(b) Blur radius, $R = 2$.

Figure 5.37: Surfaces of $\hat{f}_{nn}(X)$ at fixed defocus extent levels.

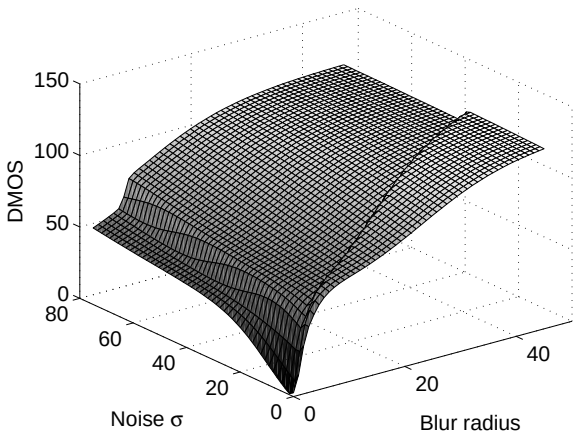


Figure 5.38: Extrapolation of the neural network model.

Table 5.5: A performance comparison between the models based on test data.

	$\hat{f}_{spline}(X)$	$\hat{f}_{nn}(X)$
RMSE[a]	7.10	10.1
STD[b]	7.06	10.1
LCC[c]	0.965	0.927

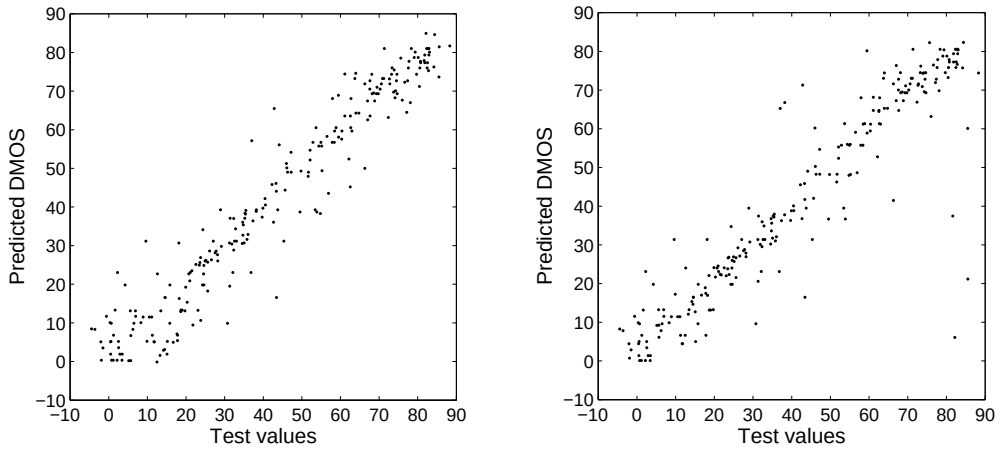
[a] Linear correlation coefficient.
[b] Standard deviation of error.
[c] Root mean squared error.

Test data comparison

The two models were tested using the two $k = 3$ and 4 cross validation datasets. The resulting mean absolute prediction error and linear correlation are presented in Table 5.5. The spline model slightly out-performs the neural network model. Figure 5.39 shows the input output relationship across both the tests sets. While both models clearly have a strong positive correlation, there are definite outliers in the neural network model. The model fared poorly with these images. This might be because the images mapped to areas in the input space close to one of the undesirable ridges in the model surface.

The calculated F-statistic for the test is 2.05, while the 5% F-values obtained from Figure 5.25 on page 158 is 1.24. Therefore the difference in performance between the two models is statistically significant to the 5% level.

However, one of the requirements for using the F-test is that the data be normally distributed. Although the residual errors used appear Gaussian, they fail mathematical Gaussianity tests. In [97] the same problem was encountered. Nevertheless, the authors claimed that, because of the large number of samples used, the Central Limit Theorem comes into play and the distribution of the variance estimates (on which the hypotheses tests are based) approximates



(a) Spline model, $\hat{f}_{spline}(X)$. (b) Neural network model, $\hat{f}_{nn}(X)$.

Figure 5.39: Correlation between expected output y_i and model prediction $\hat{y}_i$ for test data.

the Gaussian distribution. This claim was proved in [97] by running Monte Carlo tests and changing the actual distribution of the residuals from Gaussian to uniform. The effect on the results of the hypotheses tests was negligible when a large number of samples were used (90), but noticeable when fewer samples were used (10). Since our test has 242 sample points, the results of the statistical significance test hold.

5.4.4 Testing the integrated system

The previous section gave the results when the *known true* degradation levels are the inputs to the quality assessment model. In this section the *estimated* degradation levels are used as inputs instead. Therefore the feature estimation algorithms are combined with the quality assessment model to form the integrated system.

Since the integrated system starts by estimating features and rejecting difficult blind estimation cases, not all images made it to the quality estimation stage. Of the 242 images, 93 images could not be evaluated due to features estimation difficulty: 62 images were rejected by the noise estimation algorithm (where distinction between image and noise variance was poor) and 49 by the defocus estimation algorithm (where $E_r < 1$) with an overlap of 18 images. For the remaining 149 images DMOS was estimated with a root mean square error of 12.1 and a linear correlation coeficcient of 0.888, which still indicates a high degree of correlation between predicted and true DMOS values. The correlation between true and predicted DMOS levels is depicted in Figure 5.40.

As expected, prediction performance decreases when estimating the three

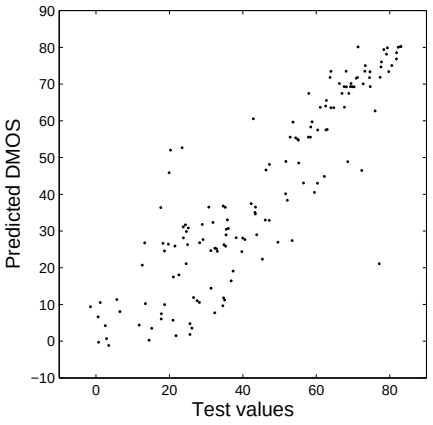


Figure 5.40: Correlation between true DMOS, y_i, and DMOS predicted by model, $\hat{y}_i$, for input feature levels x_i estimated from test images.

features in addition to the quality score. Because contrast optimisation had been applied to the input images prior to artificial degradation, noise levels were slightly underestimated and cloud levels overestimated, especially in cloud free images. These two factors cause the increase in variance at low levels of true DMOS visible when comparing Figure 5.40 to Figure 5.39. The three outliers visible at $(y_i, \hat{y}_i) \approx (20, 50)$ originate from the same bright desert base image, which caused considerable overestimation of cloud cover. The outlier at $(77, 21)$ is the result of an out-of-focus image classified as in-focus in addition to its cloud cover and noise levels being underestimated.

5.5 Conclusion

An image quality assessment model was constructed to combine the effect of the three measured degradation types into a single quality score. A regression model was used instead of a classification model to allow images to be ranked according to quality level. The model is based on a large subjective experiment to ensure generality.

The behaviour listed in section 5.3.1 was supported by the collected data: non-linear relationships between the measured features and image quality was supported; different relative weights for the different variables were supported and modelling of variable cross-coupling was also supported.

Two possible models were compared, a neural network model and a spline model 'manually assembled' from different component splines. In the construction of the spline model care was taken to adjust the degrees of freedom available to the model in different areas, based on the available data. In the central area of the input space this concept led to disregarding the effect of one of the input variables. The models were also selected to ensure monotonic

behaviour, which is appropriate for the IQA model.

The benefit of this approach over a more typical 'black-box' modelling method like neural networks was proven by the test results; the spline model outperformed the neural network model. Furthermore the spline model was designed with extrapolation in mind and appears more suited for this than the neural network, based on visual inspection.

Finally, tests applied to the entire system confirm its usefulness. The feature estimation algorithms circumvent blind estimation pitfalls by rejecting difficult images. The quality estimates for the remaining images correlate well with the subjective image quality scores from human participants.

Work regarding the creation of the quality assessment model was presented at the *2008 IEEE International Geoscience and Remote Sensing Symposium* [73] in Boston, Massachusetts. An overview of the system including results of the integrated system has been submitted for *7th IAA Symposium on Small Satellites for Earth Observation* in Berlin.

Chapter 6

Conclusion

Possible quality features were investigated and the use of degradation measures over content measures was defended in Chapters 1 and 5. Each of the selected features was justified in its corresponding chapter, Chapters 2 to 4.

For each feature existing estimation algorithms were investigated. The algorithms that were most promising and appropriate for on-board implementation were implemented and compared. Where algorithms' performance on embedded architecture could be an issue, the selected algorithms were implemented and tested on an embedded system similar to Sumbandilasat's.

6.1 Summary of chapter conclusions

In Chapter 2 cloud cover estimation was investigated. It was found that down-sampling can be used to fit the entire image into the limited memory of the embedded system and that nearest neighbour down-sampling is preferable to averaging. A region growing method, which had previously been used on-board a micro-satellite for cloud cover estimation, was critically evaluated and compared to thresholding, which is commonly used for cloud detection. The simpler thresholding method was recommended. Dimension reducing image transforms were investigated as a means of using information from multiple image channels in a memory-scarce on-board environment. The novel application of heteroscedastic discriminant analysis gave promising results and outperformed comparable transforms from cloud detection literature.

Noise estimation was considered in Chapter 3. An existing remote sensing noise estimation algorithm was compared to an estimation algorithm based on image pyramids. The two methods had not been previously compared. Based on the comparative experiment, the image pyramid method was recommended.

It has superior performance accuracy in estimating low levels of noise. It is the only method that is able to assess its own estimation ability and give warnings when it fails to discriminate between signal and noise, which is an advantage in the context of blind, autonomous noise estimation. This aspect of the algorithm was adapted to be more conservative, which resulted in better performance on remote sensing images with high levels of detail. Since the algorithm is more complex than the simple normalisation, thresholding and down-sampling methods used in cloud estimation, its embedded feasibility was evaluated. Its performance was acceptable.

In Chapter 4 algorithms for estimating the defocus extent of the PSF were investigated. A novel angular spectral smoothing method for increasing the robustness of spectral based direct blur identification was introduced. Its variance reducing properties were investigated mathematically and verified empirically. A comparative test between three existing spectral based PSF estimation methods and the angular smoothing method was conducted. The three existing methods had not previously been compared in a test of this scale. The angular smoothing method performed favourably. A novel relative energy measure was introduced and was able to separate images for which the PSF cannot be accurately estimated from those where PSF estimation is possible. The angular smoothing algorithm was implemented on the embedded system and its feasibility demonstrated.

In Chapter 5 image quality assessment methods were investigated. Since no models for blind estimation of satellite image quality existed, a subjective experiment was conducted to gather data. The experiment was suitably large to allow the construction of a generally applicable model: 18623 human judgements were collected and 484 unique degraded images evaluated. The data supported non-linear relationships between the measured features and image quality, different relative weights for the different variables as well as modelling of variable cross-coupling. A spline model was constructed that preserves monotonic behaviour and makes optimal use of available data by varying the model complexity with data density. A neural network model was also constructed. The two models were compared visually and based on test data. The spline model's performance was superior. Since evaluation of the model is simply evaluation of a piecewise polynomial function of three variables, execution time is negligible compared to that of feature estimation.

Finally, the feature estimation algorithms were integrated with the quality assessment model and the entire system was tested. For images not rejected due to blind estimation difficulty, quality was estimated successfully: there was a high linear correlation coefficient between quality estimates and the subjective image quality scores from human participants.

6.2 Recommendations

A list of recommendations is presented here and elaborated on in the following paragraphs. To develop the system into an operational system, the following is recommended:

- Develop a strategy for handling rejected images.
- Implement the cloud estimation and quality assessment model in embedded code.
- Validate the system by testing it on board the satellite.

To improve upon the system the following is recommended:

- Train different transform and threshold parameters for different regions and times to improve cloud detection.
- Compare the effects of resolution and multi-spectral use on cloud estimation.
- Incorporate attitude determination and control system data to detect geometric distortion owing to non-uniform satellite motion.
- Use the system in combination with an image acquisition scheduling system.

When Sumbandilasat becomes operational, access to many multi-spectral cloud-contaminated scenes from the same sensor will cease to be a problem. To achieve satisfactory performance across various surface types, it is recommended that different HDA parameters and thresholds be trained on region and time specific cases. These region specific transforms can be trained off-line on pooled, downloaded data and then used in a lookup table for on-board implementation. Similar approaches have been successfully applied to less flexible transforms for global threshold-based cloud detection [33, 114, 58]. Its application to HDA remains the subject of future work.

The combined effect of resolution and multi-spectral use on cloud detection could be compared. By increasing the down-sampling factor (reducing the resolution) it would be possible to fit more channels into RAM simultaneously (increasing multi-spectral use). An optimal combination of channels and resolution could be identified, allowing the cloud detection to be tailored to Sumbandilasat's spectral capabilities and memory constraints.

Images which are rejected by the noise or PSF estimation algorithms (based on estimation difficulty) could be kept in a separate list. A strategy for handling rejected images must be selected. A possibility is to sort the rejected images according to those remaining features that can be estimated. Finally, the system should be validated by testing it on board the satellite.

If geometric distortion owing to non-uniform satellite motion is sometimes present, this could represent an additional degradation feature. However, without a reference image it would be extremely difficult to autonomously estimate geometric distortion from the captured image. It would be preferable if satellite orientation information could be obtained from some other source, such as the attitude determination and control system. Incorporating such information will make the system dependent on the specific satellite.

In addition to the IQA system, an image acquisition scheduling system could further improve the use of downlink time by avoiding acquisition in cloudy conditions. Such a system was implemented for Landsat 7 [83].

6.3 Contribution

The main contributions of this dissertation to academic knowledge are:

- The novel angular smoothing blur identification algorithm, which increases robustness against noise.
- The novel application of HDA to cloud detection, which allows optimal threshold based cloud detection in in a memory scarce environment.
- The quality assessment model and integrated system, both of which are novel in the remote sensing context.

The contributions listed above have been published or submitted for publication in international journals or conference proceedings, as highlighted in the relevant chapters.

Appendices

Appendix A

Implementing the region-growing algorithm

A.1 Languages, data structures and optimisation

The language of the final implementation of all algorithms to be used on-board must necessarily be embedded C. However, to evaluate algorithms an environment more suited to rapid development is needed. Such an environment should handle memory management and provide tools for visualisation and image processing. The initial implementation of the region-growing algorithm was done in MATLAB®, since the author had a good working knowledge and it has a well-documented image processing toolbox.

During the early implementation no thought was given to execution speed. The only data structures used were

image array an $L \times M$ unsigned integer array representing the grey-scale intensity values of the $L \times M$ digital image $f(x, y)$, as defined in equation (1.4.1),

mask array an $L \times M$ boolean array $b(x, y)$ where

$$b(i,j) = \begin{cases} 1 & \text{if } f(i,j) \text{ belongs to the current region and} \\ 0 & \text{otherwise.} \end{cases}$$

Using these structures a working implementation of the growing part of the algorithm was devised. This included the calculation of average- and peripheral contrast, but not the implementation of stopping rules. The IB and CB

were recomputed every iteration. The IB could be easily found using the `bwboundaries` function of the MATLAB® image processing toolbox, but the CB had to be computed using a custom function, `get_cur_boundary`, since `bwboundaries` is capable only of detecting internal, connected boundaries. Not only is the CB not internal, but if a cloud is located at the edge of the image, the CB is not connected.

This initial implementation showed that the algorithm worked and that the difference measures behaved as expected. Nevertheless, the execution time was unpractically slow even for a test-bench application. The complexity is $O(N^2)$, where $N = L \times M$ is the total number of pixels in the image, since the IB and CB are recomputed at each iteration, necessitating scanning the entire image twice for each pixel added. As will be discussed in section A.3, the region-growing algorithm has to be applied to each seed point in the image and, since the *global* maximum of the average contrast is used, each region has to be grown to the full image size so that the global maximum can be found. This means that

total execution time =
(time to grow region to image size) × (number of seed points).

To be able to test reasonable number of images, each containing a reasonable number of clouds, the algorithm had to be fast. Also, since it was a candidate for on-board implementation, effort spent improving the performance was thought to be worth while.

The first, obvious place where the initial implementation had been lacking was in requiring the re-calculation of the IB and CB at each iteration. Since only one pixel at a time is added to the region, it is possible to update the IB and CB instead of re-calculating them. For this realisation of the algorithm, two additional arrays are used:

IB array a $2 \times k$ array where k is the length of the IB,

CB array a $2 \times n$ array where n is the length of the CB.

Each array contained all the (x, y) coordinates of the pixels in the respective boundary. At each iteration of the region-growing algorithm the following steps must be taken to keep the IB and CB arrays updated:

1. search through the CB array for the brightest pixel, $f(i_b, j_b)$,
2. remove (i_b, j_b) from the CB array,
3. add (i_b, j_b) to the IB array,
4. set $b(i_b, j_b) = 1$,
5. consider $\{(i, j) | (i, j) \in$ neighbouring pixels of $f(i_b, j_b)\}$:

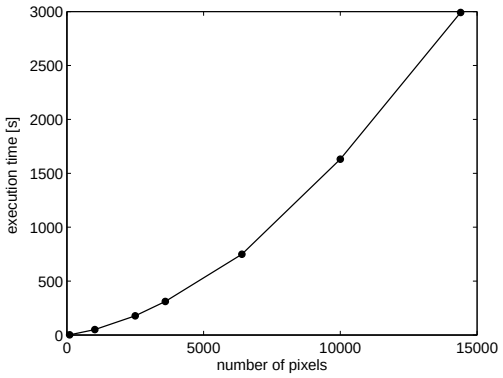


Figure A.1: Execution speed with boundary updating algorithm.

- IF $(b(i,j) = 0)$ AND $((i,j) \notin$ CB array) THEN (add (i,j) to CB array)
- IF $(b(i,j) = 1)$ AND $((i,j) \in$ IB array) AND ((i,j) has no neighbours (k,l) for which $b(k,l) = 0$) THEN (remove (i,j) from IB array).

Although this implementation was faster than the previous one, it was still exceedingly slow as shown in Figure A.1 for images of varying sizes. The shape of the execution time curve resembles a quadratic function and makes this realisation unsuitable for larger images.

The apparent $O(N^2)$ complexity was initially surprising here. However, running the MATLAB® profiler revealed a fundamental problem with the data structures used to store the boundaries. Most of the time was being spent traversing the two boundary arrays. Considering the above description, for each pixel added the CB and IB arrays are each searched multiple times: first when finding the brightest pixel in the CB and again each time ($(i,j) \notin$ CB array) or ($(i,j) \in$ IB array) are evaluated.

It was possible to eliminate these last two searches by using a different data structure. Instead of a mask array, IB array and CB array, an expanded mask array is used:

expanded mask array an $L \times M$ unsigned short integer array $e(x,y)$ where

$$e(i,j) = \begin{cases} 1 & \text{if } f(i,j) \text{ belongs to the current region but not to the IB,} \\ 2 & \text{if } f(i,j) \text{ belongs to the CB,} \\ 3 & \text{if } f(i,j) \text{ belongs to the IB and} \\ 0 & \text{otherwise.} \end{cases}$$

By using the extended mask, one can immediately determine the class of $f(i_b, j_b)$'s neighbouring pixels and update the CB and IB without searching

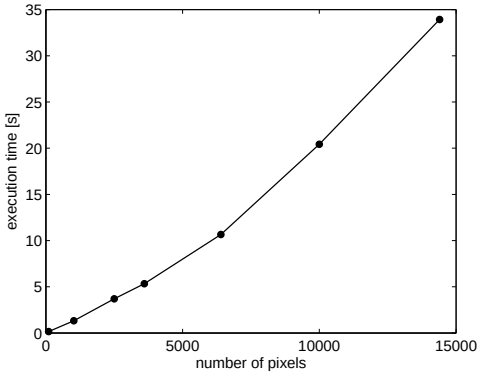


Figure A.2: Execution speed with extended mask array.

the boundaries. This represents a trade-off of memory for speed that would have to be reconsidered were it to be used in an on-board implementation. The resulting speed increase is substantial as seen in Figure A.2. However, the initial search for the brightest pixel in the CB at each iteration still remains. This means the complexity is $O(N \times \overline{n})$, where $\overline{n}$ is the average CB length. $\overline{n}$ is dependent on the shape of the region being grown and could in the worst case be a significant fraction of N, explaining the apparent quadratic shape of the execution time curve in Figure A.2.

At this point it was decided to move the prototyping environment from MATLAB® to Python [8]. This was done for two reasons:

- Python is faster at handling repeated element-wise operations on arrays. MATLAB® arrays and operations are optimised for matrices and matrix operations (it is the *mat*rix *lab*oratory). To work with specific elements of an array in an efficient manner in MATLAB®, logical indexing must be used as opposed to nested for loops. This is not possible in the context of the region-growing algorithm. The Python `numpy` module has different data types for arrays and matrices with operations optimised for each type.

- Python has more built in support for programming structures, for example a binary heap implementation (the use of which is described below).

Furthermore Python still has the memory management, visualisation tools and many of the image processing tools that made MATLAB® attractive. It is also free.

To improve the performance of the algorithm further, a *priority queue* abstract data type (ADT) [49] was used to store the CB, while the extended mask array was left unchanged. This ADT is used for storing a collection of prioritised elements and supports arbitrary insertion but removal in order of priority. The priority of each element is also called the key. Apart from utility

methods such as `size()`, `isEmpty()` and `peekMin()`, the two important access methods of the priority queue ADT are:

`insert(k,x)` insert entry with value x and key k into the queue,

`removeMin()` remove from the queue the entry with the smallest key.

When used to store the CB elements, brightness (more specifically its inverse) is used as the key and the (x, y) coordinate pair is the value for each entry.

Since the priority queue is an ADT a specific implementation has to be considered. It can be implemented as a list, either sorted or unsorted, in which case either `insert(k,x)` or `removeMin()` will take $O(n)$ and the other $O(1)$. This does not represent an improvement over using only the extended mask array. The other option is to use either a self balancing binary tree or a binary heap. When using a tree both access methods would have $O(\log(n))$ complexity. However, for the CB the element being added to the tree typically has a smaller intensity (greater key) than the all items in the tree, so the tree will have to be re-balanced often. In a vector based binary heap implementation both `insert(k,x)` and `removeMin()` also have worst case $O(\log(n))$ time, but amortised $O(1)$ time [49]. In this case the fact that items added typically have greater keys than those in the list means that execution time will lean towards the $O(1)$ limit. An existing module for Python, `heapq`, implements the priority queue in using a vector based binary heap.

This implementation finally had acceptable performance making it possible to test with multiple large images. Figure A.3 shows the result of tests on images of various sizes. Since elements can be removed from CB in $O(1)$ amortised the algorithm's performance is now $O(N \times 1) = O(N)$.

A.2 Stopping rule complications

To implement the stopping rule the last local maximum of the peripheral contrast before the global maximum of the average contrast must be found. One could argue that only the size of the cloud is important from a image quality perspective. If this were the case, one could consider the peripheral and average contrast graphs after execution and the index of the local maximum would be sufficient information, since it is equal to the size of the region to be segmented. However, when the region-growing algorithm is applied to multiple seed points in an image, the regions originating from different seed points could overlap. If one were to remember only the size of each region and sum these sizes, the total cloud cover could be over estimated. This necessitates that the shape of each region be stored. Taking the logical OR of the individual cloud masks then gives the correct total cloud cover.

This implies that output of the region-growing algorithm *must* be a segmented area. Whilst the same as described in [53], it results in complications

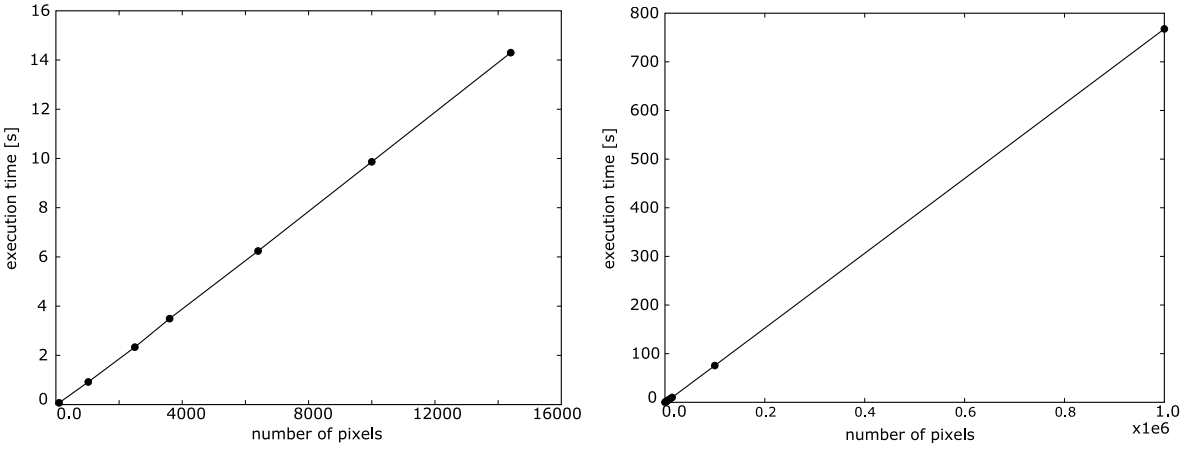


Figure A.3: Execution speed with extended mask array and priority queue. (a) is on the same scale as figures A.2 and A.1 for comparison, while (b) shows the linear performance extends to images of 1 megapixel.

not discussed there. The algorithm must return shape of the region, but the region must be grown to an upper limit size greater than its final shape to get the global average contrast maximum. Therefore, to return to a previous state of the segmented area one of three options is available:

1. Grow the region to its upper limit, determine the index of the segmentation point and re-grow the region to that size.

2. During region growing, store the last valid segmentation edge and fill the area inside the edge to get the area mask.

3. During region growing, store the last valid segmentation mask.

Option 1 is processor intensive and memory light, option 3 is processor light and memory intensive. It was decided to implement option 2 as it represents a good compromise. Options 2 and 3 imply keeping track of segmentation data corresponding to a recent local maximum of peripheral contrast during the growing process.

It is useful to consider the mathematical definition of a local maximum:

> a point x' is a local maximum of function f if there exists an $\varepsilon > 0$ such that $f(x') \geq f(x)$ for all $|x - x'| < \varepsilon$.

Figure A.4 shows some valid local maxima. Since the goal is to segment at the most recent local maximum of peripheral contrast prior to the current maximum average contrast, the boundaries corresponding to two maxima during the growing process must be stored:

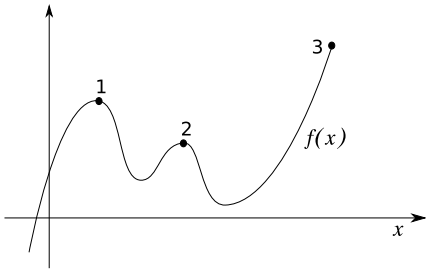


Figure A.4: Valid local maxima. In the context of peripheral contrast defined over a finite interval, point 3 is a valid local maximum.

1. the last local maximum of peripheral contrast prior to the current maximum of the average contrast

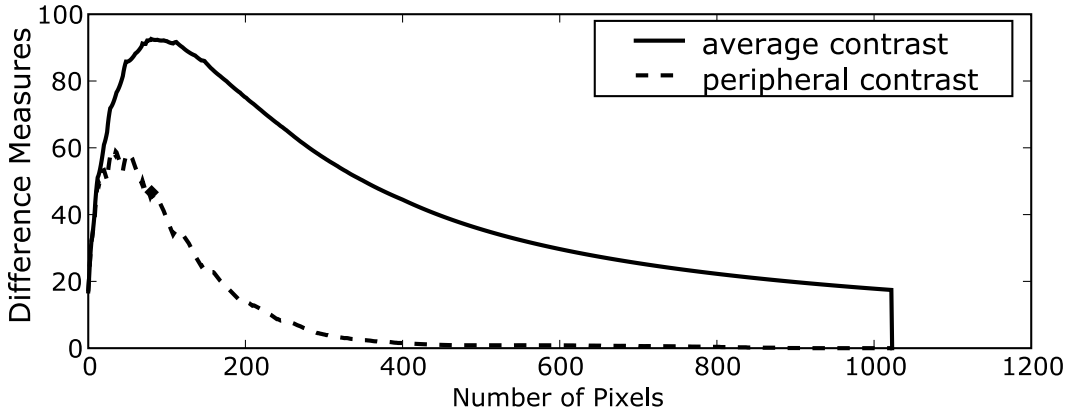
2. the most recent local maximum of peripheral contrast, given that it is greater than the maximum from 1.

The second maximum mentioned above is necessary because the average contrast might exceed its previous maximum, in which case the segmentation will boundary from 2 will be used instead of the one from 1. Since the peripheral contrast graph is 'growing' in the x-axis direction as more pixels are added, and since point 3 in Figure A.4 is a valid local maximum, the current point on the peripheral contrast graph is the most recent local maximum if it is greater than the previous ε points, where ε is now a discrete number.

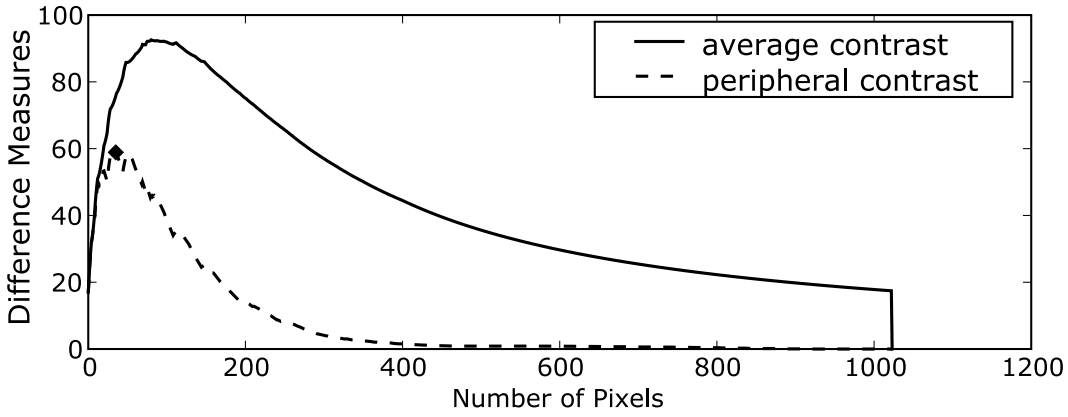
The size of ε determines the span of the local maximum. Choosing ε too small may result in detection of suboptimal local maxima caused by 'noise' in the peripheral contrast graph. Figure A.5, the difference measures for a 32×32 pixel Gaussian blob similar to 2.6(a) on page 23, shows an example of this. In Figure A.5(a) it is clear that the trend for the peripheral contrast curve is decreasing and that $\varepsilon = 1$ makes the algorithm over sensitive when detecting local maxima. Setting $\varepsilon = 20$ gave the desired local maximum in A.5(b).

What value of ε results in a meaningful local maximum is clearly relative to the size of the region being grown. However, since ε pixels have to be inspected every iteration, it is undesirable to have it too large (a significant fraction of N). Instead of increasing ε, it was found that spurious local maxima could be efficiently suppressed by allowing only integer values for peripheral contrast. This flattens very small increases which could erroneously be classified as local maxima.

After the segmentation boundary has been determined it is flood-filled to generate a cloud mask. A custom function, `fill_region.py`, was written to implement the filling. It is based on the concept of repeated morphological dilation as described in [47, pp. 535–536], but modified to improve performance speed. Instead of considering all the pixels in the image as candidates for dilation, only the pixels at the boundary of the expanding fill are considered.



(a)



(b)

Figure A.5: The effect of ε on the local maximum. The position of the final segmentation boundary is marked with a ⧫. In (a) $\varepsilon = 1$, in (b) $\varepsilon = 20$.

A.3 Using the algorithm for cloud detection

To detect all the clouds in image, the region-growing algorithm has to be applied at all the seed points – one for each cloud. Details on generating the seed points and handling multiple clouds are scarce in reference [54]. Also, since the PoSat GSI is 2000m minutiae that could result in spurious seed points have already been averaged out. The following algorithm was devised to apply the region-growing algorithm to cloud detection with a smaller GSI:

1. Threshold the image to identify bright areas, forming a rough cloud mask.

2. Identify and label the connected components in the rough mask.

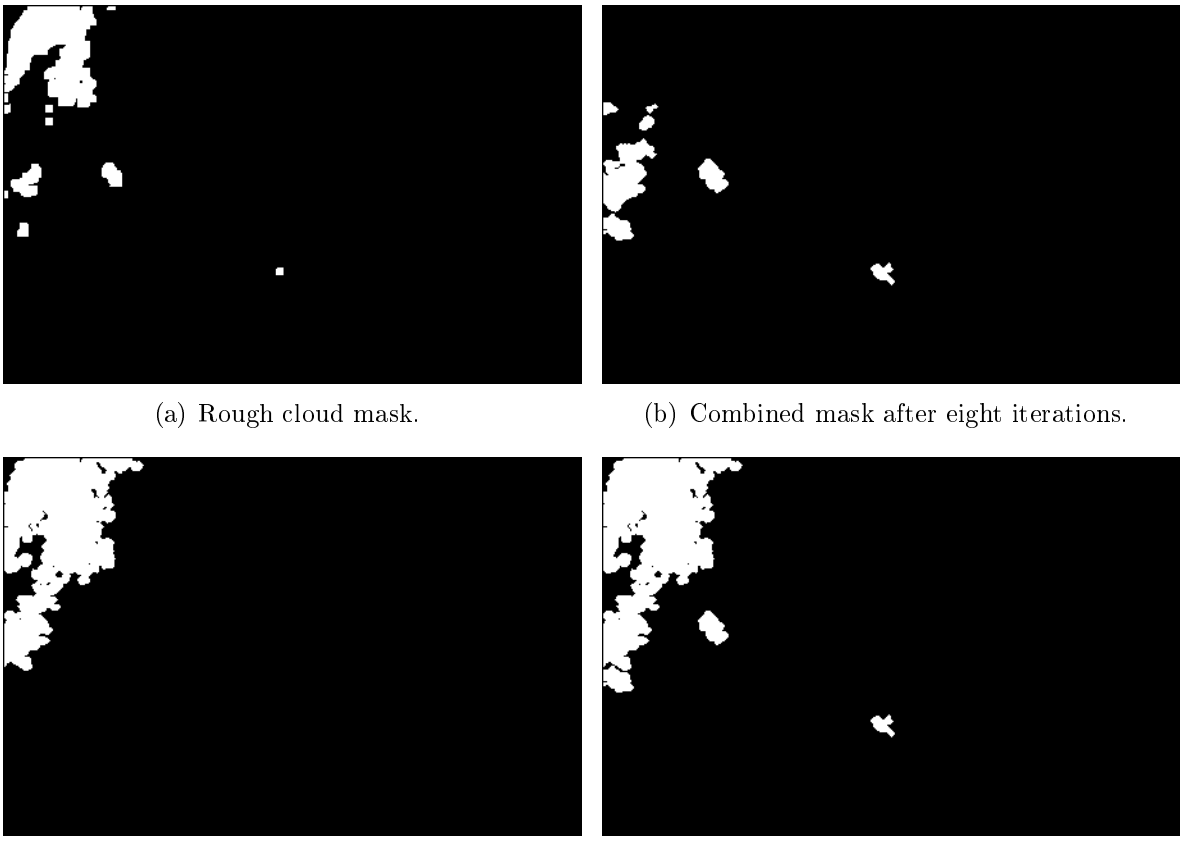
3. Find the centre and size of each component, use the centre as a seed point and a multiple of the size as an upper limit for the region-growing algorithm.

4. Take the logical OR of the masks generated from the individual seed points create to a single composite cloud mask.

The initial implementation of the connected component labelling was based on [47, pp. 536–538], but, while theoretically elegant, the solution proved very slow. The algorithm described in [95, p. 139] was used instead. Practically, the individual masks from each seed point need not be kept in memory while the others are grown; each can be ORed with the initially 0 composite mask as soon as it is completed and then discarded. Therefore a new cloud is added to the composite cloud mask after each execution of the region-growing algorithm.

As discussed in section 2.3.2, the region-growing algorithm is dependent on some upper limit. In point 4 above it was mentioned that a multiple of the rough cloud mask size is used as an upper limit. Since the algorithm is designed to be insensitive to the upper-limit, the exact value multiple should not be crucial; a factor of 2 was used. However, the upper limit does play a role when growing multiple seed points into regions. Since the upper limit is a multiple of the rough cloud mask size, larger areas have more 'growing room' than smaller areas. This can result in large areas consuming (or swamping) smaller, neighbouring areas during the region-growing process, i.e., the cloud mask generated from a seed point in the centre of a small area becomes a subset of the cloud mask generated from a seed point in the centre of a larger area. While this does not negatively affect the final composite cloud mask, it means processing time spent on growing the consumed region was wasted. This was easily avoided by checking if a seed point had already been consumed in the composite cloud mask before starting with region growing. However, this led to situations where a small region might just manage to consume the seed point of a neighbouring cloud and then reach the region-growing upper limit before being able to expand and consume the whole cloud. The final composite cloud mask would then be incorrect. This problem was solved by sorting the labelled connected regions in the rough cloud mask based on size. The seed points corresponding to larger areas are used first. These larger areas have enough 'growing room' to completely consume smaller clouds if they are close together. If they are not close together the stopping rule should keep the larger area from growing into the smaller one.

Figure A.6(a) shows a rough cloud mask for the image from figure 2.25. Morphological operations have been used to reduce the number of connected areas and therefore seed points. If no attention is paid to the order in which the regions are tackled, the cloud mask after eight iterations is depicted in Figure A.6(b). During the ninth iteration the largest region is grown (Figure A.6(c)) and consumes many of the previously grown regions, making that work redundant. By tackling the areas in order of size the total number of calls to the region-growing algorithm is reduced from nine to four with the same resulting combined cloud mask, A.6(d).



(a) Rough cloud mask.

(b) Combined mask after eight iterations.

(c) Region added during ninth iteration.

(d) Final combined mask.

Figure A.6: Regions consumed because of lack of ordering.

Appendix B

Critical evaluation of MATLAB neural network regularisation options

As mentioned in section 5.2.4 there are four common regularisation options available to prevent overfitting. To choose from these options a toy problem was used to evaluate the different methods. Early stopping was not evaluated, since it is the most basic and ineffective.

The data for the toy problem was a sinus curve evaluated every 0.2 in the range $\{0 - 4\pi\}$. The training data was corrupted by adding Gaussian noise with $\sigma_n = 0.2$. The validation data is the uncorrupted sinus curve evaluated at the same points. If overfitting occurs, the model will follow the training data too closely and model the unwanted noise. This will result in poor validation performance. On the other hand, if the regularisation method succeeds, the model should have adequate complexity to model the sinus without modelling the noise and the validation results should be good. Therefore, the toy problem has the advantage of being easier to visualise, containing accurate validation data and allowing more test runs than the collected experimental data. Figure B.1 shows the training and test data as well as two fits from models with different complexities.

As previously mentioned, training of neural networks is sensitive to the starting conditions. The recommendation in section 5.2.4 was followed whereby the model was trained 20 times for different random starting weights. The validation error was also computed 20 times and the results are presented as box-plots that show the distribution of the data across the 20 runs. Sufficient training epochs were used for the training error to stabilise and the error goal was set to zero, to ensure that early stopping does not occur.

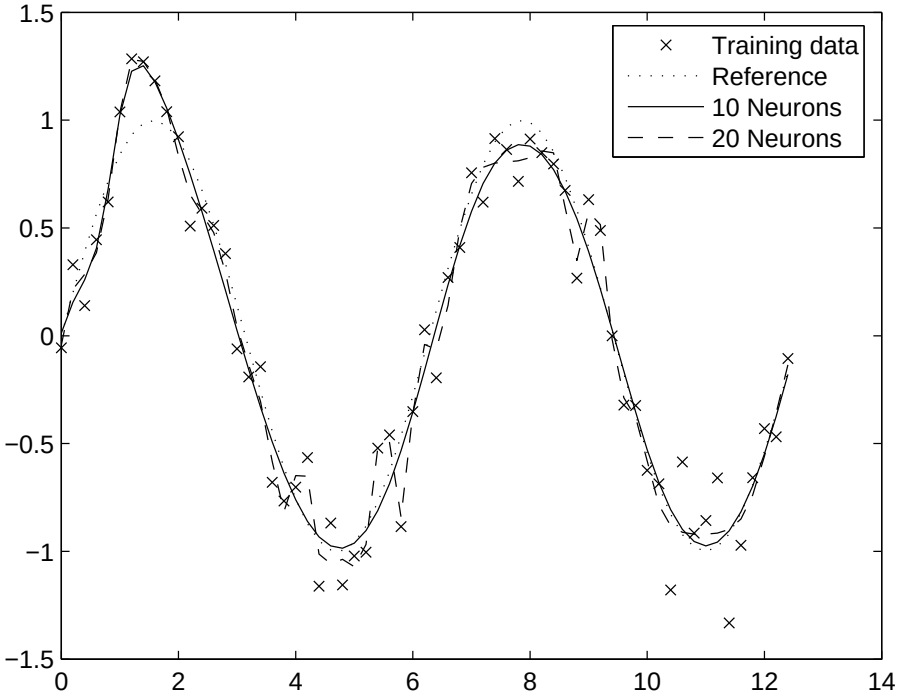
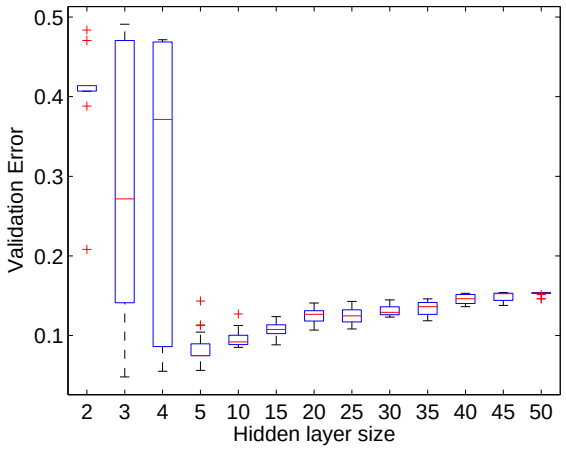


Figure B.1: The toy problem data and two example fits. The fits were generated by the automatic Bayesian regularisation method. Notice how the 20 hidden unit model overfits the data.

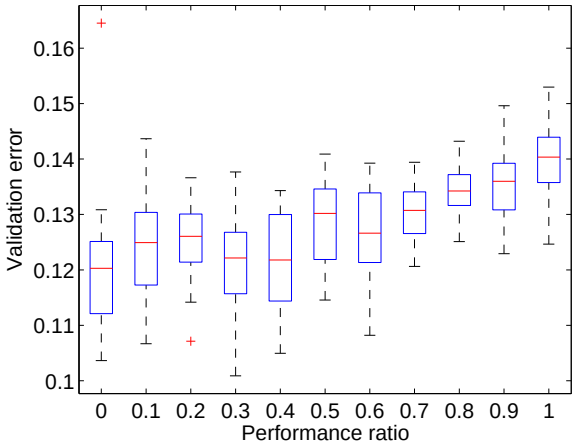
Manually varying the number of hidden nodes is algorithmically the simplest method. The validation error in Figure B.2(a) behaves as expected. It is initially large when the model does not have enough complexity to follow the sine, reaches an optimum at 5 hidden units and then increases again as the added complexity is used to model the noise.

To test the penalty term method (implemented in MATLAB® according to equation (5.2.4)), 20 hidden units were used and γ varied between 0 and 1. The effect of varying the performance ratio, γ, is visible in Figure B.2(b). It also behaves in an explainable manner, as γ increases, the penalty term loses its weight and overfitting worsens. It is strange that the model can be trained at $\gamma = 0$, it is possible that the MATLAB® implementation substitutes some finite minimum value. Although the penalty term does have an effect of the validation error, manually varying the number of units is more effective. The error here stays within the band of the 20 hidden units error from the previous test.

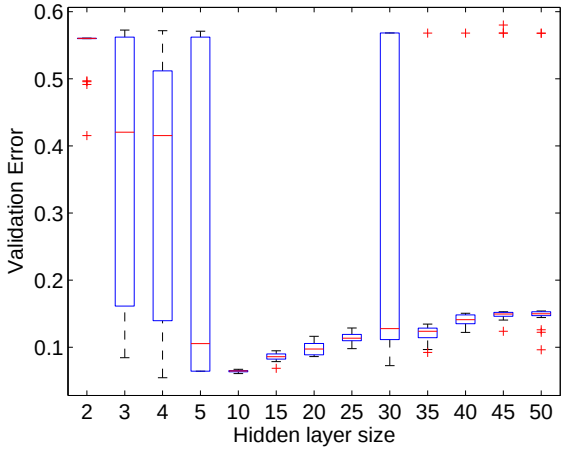
Lastly, MATLAB's automatic Bayesian regularisation was tested. It was evaluated across a selection of models with the number of hidden units varying similar to the manual method. If the method works as it is supposed to, the



(a) Manually varying number of hidden units.



(b) Using a performance ratio, γ.



(c) Using automatic Bayesian regularisation.

Figure B.2: A comparison between different regularisation options for neural networks.

validation error should remain constant once the optimum model complexity has been reached. Figure B.2(c) shows the results. At 5 hidden units, the method still limits the number of parameters in an unwanted manner, resulting in high validation error. At 10 hidden units, the method successfully limits the number of parameters so that the performance is similar to the 5 hidden unit network from the manual test. However, as the number of units are manually increased, so the validation error increases. Therefore, the method is not successful in completely regularising model complexity. At 30 to 50 hidden nodes there are also outliers caused by the method limiting the number of active parameters too severely. While the resulting median errors are smaller than similar sized models with no regularisation, the increase in validation error with model size shows that, for this toy problem, the method cannot regularise model complexity as well as optimal manual selection of model size.

Based on the results of this experiment, manual architecture selection was used since it proves adequate protection against overfitting and is the simplest method to implement.

Appendix C

Embedded implementation documentation

This documentation was adapted from the LaTeX documentation generated by Doxygen [3] to allow it to be incorporated as an appendix. Doxygen generates documentation by scanning source files for specially formatted comments.

C.1 Embedded implementation data structure documentation

C.1.1 ImageD struct reference

`#include <imaux.h>`

Data fields

- double ** **img**
- long **nRow**
- long **nCol**

Detailed description

Image structure containting memory for a double type image as well as information about the image.

Definition at line 24 of file imaux.h.

Field documentation

double ** **ImageD::img**
Pointer to two dimensional array.
Definition at line 25 of file imaux.h.
Referenced by allocateImageD(), angularAverage(), estimateNoise(), estimatePSF(), freeImageD(), getVarianceOrderStatistics(), printImageD(), and smoothPS().

long ImageD::nRow
Number of rows in image.
Definition at line 26 of file imaux.h.
Referenced by allocateImageD(), angularAverage(), freeImageD(), printImageD(), and smoothPS().

long ImageD::nCol
Number of columns in image.
Definition at line 26 of file imaux.h.
Referenced by allocateImageD(), angularAverage(), and printImageD().
The documentation for this struct was generated from the following file:

- **imaux.h**

C.1.2 ImageF struct reference

```
#include <imaux.h>
```

Data fields

- float ** **img**
- long **nRow**
- long **nCol**

Detailed description

Image structure containting memory for a float type image as well as information about the image.
Definition at line 34 of file imaux.h.

Field documentation

float ** **ImageF::img**
Pointer to two dimensional array.
Definition at line 35 of file imaux.h.
Referenced by allocateImageF(), and freeImageF().

long ImageF::nRow
Number of rows in image.
Definition at line 36 of file imaux.h.
Referenced by allocateImageF(), and freeImageF().

long ImageF::nCol
Number of columns in image.
Definition at line 36 of file imaux.h.
Referenced by allocateImageF().
The documentation for this struct was generated from the following file:

- **imaux.h**

C.1.3 ImageUC struct reference

```
#include <imaux.h>
```

Data fields

- unsigned char ** **img**
- long **nRow**
- long **nCol**

Detailed description

Image structure containting memory for an unsigned char type image as well as information about the image.
Definition at line 14 of file imaux.h.

Field documentation

unsigned char ImageUC::img**
Pointer to two dimensional array.
Definition at line 15 of file imaux.h.
Referenced by allocateImageUC(), downSampleImageUC(), estimatePSF(), freeImageUC(), getVarianceOrderStatistics(), loadbitmap(), and printImageUC().

long ImageUC::nRow
Number of rows in image.
Definition at line 16 of file imaux.h.
Referenced by allocateImageUC(), estimateNoise(), estimatePSF(), freeImageUC(), getVarianceOrderStatistics(), and printImageUC().

long ImageUC::nCol

Number of columns in image.

Definition at line 16 of file imaux.h.

Referenced by allocateImageUC(), estimateNoise(), estimatePSF(), getVarianceOrderStatistics(), and printImageUC().

The documentation for this struct was generated from the following file:

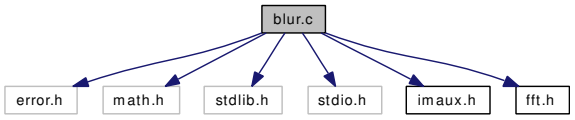
- **imaux.h**

C.2 Embedded implementation file documentation

C.2.1 blur.c File reference

```
#include <error.h>
  #include <math.h>
  #include <stdlib.h>
  #include <stdio.h>
  #include "imaux.h"
  #include "fft.h"
```

Include dependency graph for blur.c:



Defines

- #define **BLOCKSIZE** 128

Functions

- int **maxPos** (double *input, int size)
- float **getRelativeEnery** (double *input, int size, int peak)
- void **adaptiveCombFilter** (double *input, int size)
- void **angularAverage** (**ImageD** *inImage, double *avg, int avgSize)
- void **smoothPS** (**ImageD** *inImage)
- void **estimatePSF** (**ImageUC** *inImage, int *defocusExtent, float *relativeEnergy)

Detailed description

This file contains functions related to determining the defocus blur extent of an image.

Definition in file **blur.c**.

Define documentation

#define BLOCKSIZE 128

Image is divided into BLOCKSIZExBLOCKSIZE blocks during PSF estimation.

Definition at line 15 of file blur.c.

Referenced by estimatePSF().

Function documentation

void adaptiveCombFilter (double * *input*, int *size*)

An adaptive comb-like filter that amplifies peaks in input array which have harmonics and suppresses peaks which do not have harmonics. The filter is:

$$Out(r) = \frac{|In(r)|}{\sqrt{\frac{1}{M}\sum_{i\in A_r}(In(i))^2}}$$

$$for\ quefrency\ r,\ A_r = \{i | i > r_0\ and\ i \notin (kr-1, kr, kr+1), k = 0, 1, 2 \ldots\}.$$

A_r is the "disturbance set": the set of quefrencies where harmonics of r are not expected. This set resembles a comb-filter with 3-point stop bands. M is the total number of points in A_r. $r_0 = 3$ to avoid an A_r consisting only of stop bands, which would be an empty set. The output of the filter is therefore limited to values of $r > 3$. Input is replaced with filtered output.

Parameters:

input Input array.

size Size of input array.

Definition at line 99 of file blur.c.

Referenced by estimatePSF().

void angularAverage (ImageD * *inImage*, double * *avg*, int *avgSize*)

The input image is converted into polar coordinates using bilinear interpolation, and then averaged over all angles. This creates a 1D array that is a function of r, the distance from the origin in the input image.

Parameters:

inImage Pointer to the struct encapsulating the double image to be smoothed. The input array must be square.

avg Pointer to one dimensional double array large enough to contain the output.

avgSize The size of the avg array. Must be at least ceil(size/sqrt(2)), where inImage is (size x size). That is the output array must be large enough to contain values of r equal tot the radius in the corners of the image.

Definition at line 152 of file blur.c.

References allocateImageD(), freeImageD(), ImageD::img, M_PI, ImageD::nCol, and ImageD::nRow.

Referenced by estimatePSF(), and smoothPS().

Here is the call graph for this function:

void estimatePSF (ImageUC * *inImage*, int * *defocusExtent*, float * *relativeEnergy*)

Implements a variation on Cannon's point spread function estimation algorithm. ("Blind deconvolution of spatially invariant image blurs with phase", IEEE Transactions on Acoustics, speech and Signal Processing, v24, no1, Feb 1976).

- Divide image into 128x128 squares.
- Compute power spectrum of each square and average power spectrum over all squares.
- Reduce noise variance through angular smoothing using **smoothPS()** (p. 201).
- Subtract noise power.
- Take cepstrum.
- Do postprocessing with **adaptiveCombFilter()** (p. 198) to suppress spurious cepstral peaks.

Spectral subtraction and postprocessing are additions to the algorithm added by Fabian et.al. ("Robust Identification of motion and out-of-focus blur parameters from blurred noisy images:, CVGIP: Graphical models and Image Processing, v 53, no 5, Sept, 1991). Angular smoothing is a new addition

to the algorithm. The additions are aimed at increase the robustness of the method in presence of additive white gaussian noise.

Parameters:

inImage Struct encapsulating input image (8 bit greyscale).

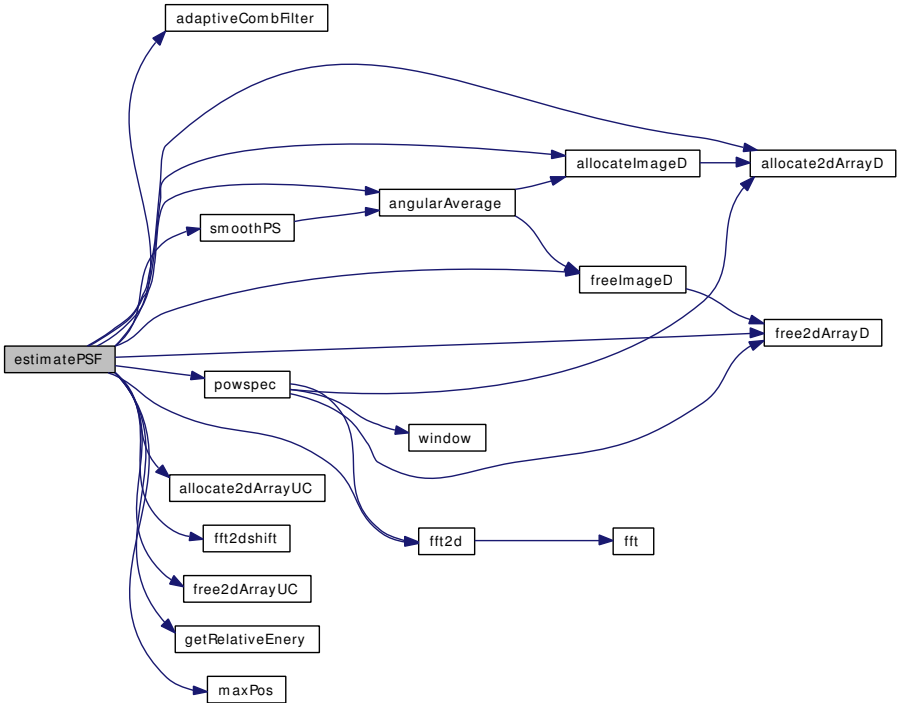
defocusExtent Pointer to value that will be modified to contain the estimated diameter of the defocus blur. If inImage is in focus this will contain zero;

relativeEnergy Pointer to value that will be modified to contain the relative energy in the peak. Can be used as a measure of certainty in the defocus extent estimation. If the inImage is classified as in focus this value is meaningless.

Definition at line 305 of file blur.c.

References adaptiveCombFilter(), allocate2dArrayD(), allocate2dArrayUC(), allocateImageD(), angularAverage(), BLOCKSIZE, fft2d(), fft2dshift(), free2dArrayD(), free2dArrayUC(), freeImageD(), getRelativeEnery(), HANN_WINDOW, ImageD::img, ImageUC::img, maxPos(), ImageUC::nCol, ImageUC::nRow, powspec(), and smoothPS().

Here is the call graph for this function:



float getRelativeEnery (double * *input*, int *size*, int *peak*)

Computes the energy in the peak of a 1D signal relative to the energy in the rest of the signal. The peak width is three indices.

Parameters:

input Pointer to the input signal array.

size The size of the input array.

peak The index of the peak in the input.

Returns:

The relative energy.

Definition at line 47 of file blur.c.
Referenced by estimatePSF().

int maxPos (double * *input*, int *size*)

Get the index of the greatest element in an array.

Parameters:

input The input array.

size Size of the input array.

Returns:

The index of the greatest element in the input array.

Definition at line 23 of file blur.c.
Referenced by estimatePSF().

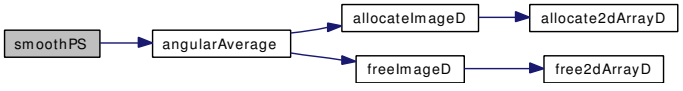
void smoothPS (ImageD * *inImage*)

Radially smoothes the power spectrum using **angularAverage()** (p. 198). This 1D function is then swept around the origin of the polar coordinate system (centre of the image) to create a surface of revolution in cartesian coordinates. Linear interpolation is used for this last step.

Parameters:

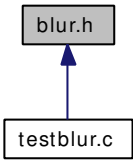
inImage Pointer to the struct encapsulating the double image to be smoothed. The input array must be square. Smoothed version replaces input.

Definition at line 232 of file blur.c.
References angularAverage(), ImageD::img, and ImageD::nRow.
Referenced by estimatePSF().
Here is the call graph for this function:



C.2.2 blur.h File reference

This graph shows which files directly or indirectly include this file:



Detailed description

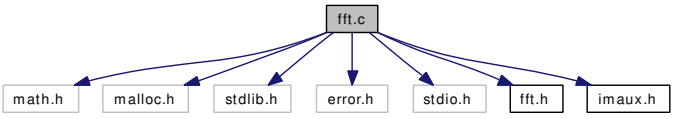
This file contains declarations for functions used to determine the defocus blur extent of an image.

Definition in file **blur.h**.

C.2.3 fft.c File reference

```
#include <math.h>
  #include <malloc.h>
  #include <stdlib.h>
  #include <error.h>
  #include <stdio.h>
  #include "fft.h"
  #include "imaux.h"
```

Include dependency graph for fft.c:



Defines

- #define **M_PI** 3.14159265358979323846

Functions

- void **window** (double **image, long nRow, long nCol, enum **windowOptions** windowOpt)
- void **fft** (int npoints, double *real, double *imag, int inv)
- void **fft2d** (double **imgReal, double **imgImag, long nRow, long nCol, short flag)
- void **powspec** (unsigned char **imgIn, double **imgOut, long nRow, long nCol, enum **windowOptions** windowOpt)
- void **fft2dshift** (double **inImg, double **outImg, long nRow, long nCol)

Detailed description

This file contains code for performing the Fourier transform and related tasks. The implementation has been adapted from code supplied with "Practical Algorithms for Image Analysis".

Definition in file **fft.c**.

Define documentation

#define M_PI 3.14159265358979323846

Pi.

Definition at line 17 of file fft.c.

Referenced by angularAverage(), fft(), and window().

Function documentation

void fft (int *npoints*, double * *real*, double * *imag*, int *inv*)

Uses time decomposition with input bit reversal. The Cooley/Tookey Fortran scheme for doing recursive odd/even decimation is used. The computation is done in place, so the output replaces the input. The contents of the arrays are changed from the input data to the FFT coefficients. (Adapted from Practical Algorithms for Image Analysis.)

Parameters:

npoints The number of points in the FFT. Must be a power of two.

real,imag Pointers to arrays of floats for input and output. Arrays must be allocated my caller.

inv 1 for inverse transform. -1 for forward transform.

Definition at line 41 of file fft.c.

References M_PI.

Referenced by fft2d().

void fft2d (double ** *imgReal*, double ** *imgImag*, long *nRow*, long *nCol*, short *flag*)

Performs two-dimensional FFT on square image. Places output in input real and imagenary image arrays. (Adapted from Practical Algorithms for Image Analysis).

Parameters:

imgReal,imgImag Pointer to real and imaginary arrays. Arrays must be allocated by caller.

nRow,nCol Number of rows and columns for real and img arrays. Must be a power of two.

flag -1 for forward transform, 1 for reverse transform.

Definition at line 114 of file fft.c.
References fft().
Referenced by estimatePSF(), and powspec().
Here is the call graph for this function:

void fft2dshift (double ** *inImg*, double ** *outImg*, long *nRow*, long *nCol*)

Shifts the two dimensional FFT output so that the origin (zero frequency) is in the centre of the image.

Parameters:

inImg The image array to be shifted.

outImg The shifted image array. Memory must be allocated by caller.

nRow,nCol The size of the image array.

Definition at line 293 of file fft.c.
Referenced by estimatePSF().

void powspec (unsigned char ** *imgIn*, double ** *imgOut*, long *nRow*, long *nCol*, enum windowOptions *windowOpt*)

Calculates the two-dimensional power spectrum of image, optionally applying a window function first.

Parameters:

imgIn Pointer to Image array - take note unsigned char implies 8bit depth. This might be changed to accommodate greater bitdepths.

imgOut Pointer to powerspectrum output array. Memory for arrays must be allocated by the caller.

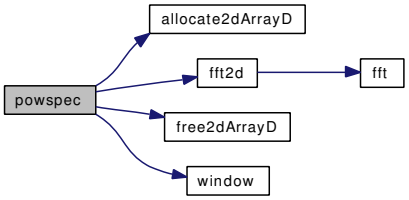
nRow,nCol Number of rows and columns for input and output arrays. Must be a power of two.

windowOpt can be any one of the following:

- NO_WINDOW
- HAMMING_WINDOW
- HANN_WINDOW

Definition at line 173 of file fft.c.
References allocate2dArrayD(), fft2d(), free2dArrayD(), log2, NO_WINDOW, and window().

Referenced by estimatePSF().
Here is the call graph for this function:



void window (double ** *image*, long *nRow*, long *nCol*, enum windowOptions *windowOpt*)
Multiplies input array by smoothing window.

Parameters:

image Two dimensional array to be windowed.

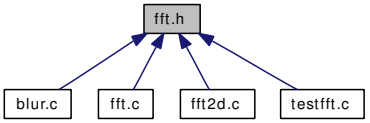
nRow,nCol Size of image array.

windowOpt Type of window.

Definition at line 236 of file fft.c.
References HAMMING_WINDOW, HANN_WINDOW, and M_PI.
Referenced by powspec().

C.2.4 fft.h File reference

This graph shows which files directly or indirectly include this file:



Enumerations

- enum **windowOptions** { **NO_WINDOW** = 0, **HAMMING_WINDOW**, **HANN_WINDOW** }

Detailed description

This file contains declarations of Fourier transform related functions. The implementation has been adapted from code supplied with "Practical Algorithms for Image Analysis".
Definition in file **fft.h**.

Enumeration type documentation

enum windowOptions

Arguments for **powspec()** (p. 204) and **window()** (p. 205) functions, used to determine spectral window type.

Enumerator:

NO_WINDOW Rectangular window.

HAMMING_WINDOW Hamming window

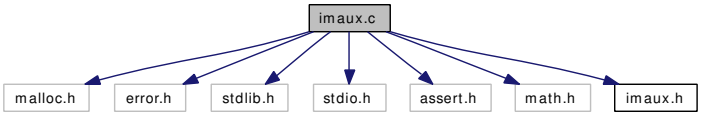
HANN_WINDOW Also known as raised cosine.

Definition at line 9 of file fft.h.

C.2.5 imaux.c File reference

```
#include <malloc.h>
  #include <error.h>
  #include <stdlib.h>
  #include <stdio.h>
  #include <assert.h>
  #include <math.h>
  #include "imaux.h"
```

Include dependency graph for imaux.c:



Functions

- double ** **allocate2dArrayD** (long nRow, long nCol)
- void **free2dArrayD** (long nRow, double **ArrayPtr)
- unsigned long long int ** **allocate2dArrayULLI** (long nRow, long nCol)
- void **free2dArrayULLI** (long nRow, unsigned long long int **ArrayPtr)
- float ** **allocate2dArrayF** (long nRow, long nCol)
- void **free2dArrayF** (long nRow, float **ArrayPtr)
- unsigned char ** **allocate2dArrayUC** (long nRow, long nCol)
- void **free2dArrayUC** (long nRow, unsigned char **ArrayPtr)
- **ImageUC** * **allocateImageUC** (long nRow, long nCol)
- **ImageD** * **allocateImageD** (long nRow, long nCol)
- **ImageF** * **allocateImageF** (long nRow, long nCol)
- void **freeImageUC** (**ImageUC** *in)

- void **freeImageD** (**ImageD** *in)
- void **freeImageF** (**ImageF** *in)
- **ImageUC** * **loadbitmap** (char *filename)
- void **printImageUC** (**ImageUC** *in, int flag)
- void **printImageD** (**ImageD** *in, int flag)
- void **downSampleImageUC** (**ImageUC** **input, int factor)

Detailed description

This file contains auxiliary image processing functions. These are memory allocation and input/output functions.

Definition in file **imaux.c**.

Function documentation

double ** **allocate2dArrayD (long *nRow*, long *nCol*)**

Allocates a two dimensional array containing doubles.

Parameters:

nRow,nCol The number of rows and columns in the array.

Definition at line 20 of file imaux.c.

Referenced by allocateImageD(), estimatePSF(), getVarianceOrderStatistics(), and powspec().

float ** **allocate2dArrayF (long *nRow*, long *nCol*)**

Allocates a two dimensional array containing floats.

Parameters:

nRow,nCol The number of rows and columns in the array.

Definition at line 86 of file imaux.c.

Referenced by allocateImageF().

unsigned char ** **allocate2dArrayUC (long *nRow*, long *nCol*)**

Allocates a two dimensional array containing unsigned chars.

Parameters:

nRow,nCol The number of rows and columns in the array.

Definition at line 120 of file imaux.c.

Referenced by allocateImageUC(), and estimatePSF().

unsigned long long int ** **allocate2dArrayULLI** (**long** ***nRow***, **long** ***nCol***)

Allocates a two dimensional array containing unsigned long long ints.

Parameters:

nRow,nCol The number of rows and columns in the array.

Definition at line 53 of file imaux.c.
Referenced by getVarianceOrderStatistics().

ImageD* **allocateImageD** (**long** ***nRow***, **long** ***nCol***)

Allocate the memory for an **ImageD** (p. 194) structure.

Parameters:

nRow,nCol Size of array contained within the **ImageD** (p. 194) structure.

Definition at line 172 of file imaux.c.
References allocate2dArrayD(), ImageD::img, ImageD::nCol, and ImageD::nRow.
Referenced by angularAverage(), estimatePSF(), and getVarianceOrderStatistics().
Here is the call graph for this function:

ImageF* **allocateImageF** (**long** ***nRow***, **long** ***nCol***)

Allocate the memory for an **ImageF** (p. 195) structure.

Parameters:

nRow,nCol Size of array contained within the **ImageF** (p. 195) structure.

Definition at line 192 of file imaux.c.
References allocate2dArrayF(), ImageF::img, ImageF::nCol, and ImageF::nRow.
Here is the call graph for this function:

ImageUC* **allocateImageUC** (**long** ***nRow***, **long** ***nCol***)

Allocate the memory for an **ImageUC** (p. 196) structure.

Parameters:

nRow,nCol Size of array contained within the **ImageUC** (p. 196) structure.

Definition at line 153 of file imaux.c.
References allocate2dArrayUC(), ImageUC::img, ImageUC::nCol, and ImageUC::nRow.
Referenced by downSampleImageUC(), and loadbitmap().
Here is the call graph for this function:

void downSampleImageUC (ImageUC ** *input*, int *factor*)

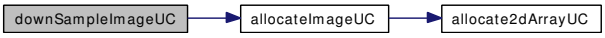
Downsamples the image by a constant integer factor. Memory footprint is also reduced. This is achieved by assigning new memory for the output image and freeing the old memory after subsampling is finished. To achieve this, double dereferenced pointer is necessary.

Parameters:

input The input image.

factor The amount to downsample by. E.g. if this is 3, every third pixel is preserved and image dimensions are approximately divided by three.

Definition at line 395 of file imaux.c.
References allocateImageUC(), and ImageUC::img.
Here is the call graph for this function:



void free2dArrayD (long *nRow*, double ** *ArrayPtr*)

Frees memory allocated with **allocate2dArrayD()** (p. 207).

Parameters:

nRow The number of rows in the array.

ArrayPtr Pointer to array to be freed.

Definition at line 39 of file imaux.c.
Referenced by estimatePSF(), freeImageD(), getVarianceOrderStatistics(), and powspec().

void free2dArrayF (long *nRow*, float ** *ArrayPtr*)

Frees memory allocated with **allocate2dArrayF()** (p. 207).

Parameters:

nRow The number of rows in the array.

ArrayPtr Pointer to array to be freed.

Definition at line 106 of file imaux.c.
Referenced by freeImageF().

void free2dArrayUC (long *nRow*, unsigned char ** *ArrayPtr*)

Frees memory allocated with **allocate2dArrayUC()** (p. 207).

Parameters:

nRow The number of rows in the array.

ArrayPtr Pointer to array to be freed.

Definition at line 139 of file imaux.c.
Referenced by estimatePSF(), and freeImageUC().

void free2dArrayULLI (long *nRow*, unsigned long long int ** *ArrayPtr*)

Frees memory allocated with **allocate2dArrayULLI()** (p. 208).

Parameters:

nRow The number of rows in the array.

ArrayPtr Pointer to array to be freed.

Definition at line 72 of file imaux.c.
Referenced by getVarianceOrderStatistics().

void freeImageD (ImageD * *in*)

Free all of the memory associated with an **ImageD** (p. 194) structure.
Definition at line 220 of file imaux.c.
References free2dArrayD(), ImageD::img, and ImageD::nRow.
Referenced by angularAverage(), estimateNoise(), and estimatePSF().
Here is the call graph for this function:

void freeImageF (ImageF * *in*)

Free all of the memory associated with an **ImageF** (p. 195) structure.
Definition at line 229 of file imaux.c.
References free2dArrayF(), ImageF::img, and ImageF::nRow.
Here is the call graph for this function:

void freeImageUC (ImageUC * *in*)

Free all of the memory associated with an **ImageUC** (p. 196) structure.
Definition at line 211 of file imaux.c.
References free2dArrayUC(), ImageUC::img, and ImageUC::nRow.
Here is the call graph for this function:

ImageUC* loadbitmap (char * *filename*)

Load a greyscale (8 bits per pixel) bitmap image form file.

Returns:

Pointer to a **ImageUC** (p. 196) struct containing the image. The caller must free the memory using **freeImageUC()** (p. 210) when it is no longer needed.

Definition at line 240 of file imaux.c.
References allocateImageUC(), and ImageUC::img.
Here is the call graph for this function:

void printImageD (ImageD * *in*, int *flag*)

Prints a **ImageD** (p. 194) to std-out or file depending on the value of flag.

Parameters:

in Pointer to **ImageD** (p. 194) structure to print.

flag 0: Print to std out. 1: Print to file 'ImageD.txt'.

Definition at line 359 of file imaux.c.
References ImageD::img, ImageD::nCol, and ImageD::nRow.

void printImageUC (ImageUC * *in*, int *flag*)

Prints a **ImageD** (p. 194) to std-out or file depending on the value of flag.

Parameters:

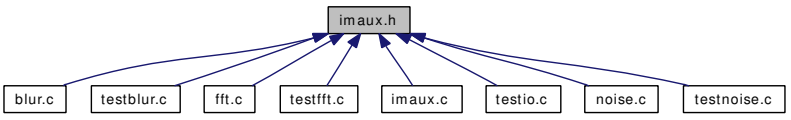
in Pointer to **ImageD** (p. 194) structure to print.

flag 0: Print to std out. 1: Print to file 'ImageD.txt'.

Definition at line 325 of file imaux.c.
References ImageUC::img, ImageUC::nCol, and ImageUC::nRow.

C.2.6 imaux.h File reference

This graph shows which files directly or indirectly include this file:



Data structures

- struct **ImageUC**
- struct **ImageD**
- struct **ImageF**

Defines

- #define **log2**(a) (log(a)/ 0.6931471805599)
- #define **exp2**(a) exp(a * 0.6931471805599)

Detailed description

This file contains declarations for auxiliary image processing functions. These are memory allocation and input/output functions. Structures for encapsulating two dimensional array information are also defined here. By including extra information in a struct the parameters passed during function calls can be reduced.

Definition in file **imaux.h**.

Define documentation

#define exp2(a) exp(a * 0.6931471805599)

Base two exponential. $2^x = e^{\ln(2^x)} = e^{x\ln(2)}$

Definition at line 41 of file imaux.h.

Referenced by estimateNoise().

#define log2(a) (log(a)/ 0.6931471805599)

Base two logarithm. $log2(x) = \frac{\ln(x)}{\ln(2)}$

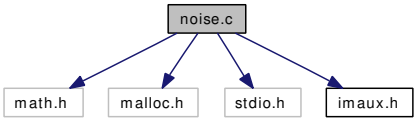
Definition at line 40 of file imaux.h.

Referenced by estimateNoise(), getVarianceOrderStatistics(), and powspec().

C.2.7 noise.c File reference

```
#include <math.h>
  #include <malloc.h>
  #include <stdio.h>
  #include "imaux.h"
```

Include dependency graph for noise.c:



Functions

- **ImageD** * **getVarianceOrderStatistics** (**ImageUC** *input)
- float **estimateNoise** (**ImageUC** *input)

Detailed description

Contains functions related to determining the amount of additive gaussian white noise present in an image.

Definition in file **noise.c**.

Function documentation

float estimateNoise (ImageUC * *input*)

Estimates the noise variance present in the image using the method of Meer et.al. ('A Fast Parallel Algorithm for Blind Estimation of Noise Variance', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 12. No 2. Feb 1990). The steps are:

- Sample variances are computed for square cells tessellating the image. Several tessellation levels are applied with the size of the cells increasing fourfold for consecutive tessellations. The four smallest variance values for each tessellation are retained. This step is done by **getVariance-OrderStatistics()** (p. 214).
- The variance values for each tessellation level are combined though an outlier analysis to yield a variance estimate sequence consisting of a single variance value for each level.
- The noise variance is estimated by interpolation of two variance values in this sequence.
- Which two variance values are best suited depends on the dichotomy between image and noise variance. Determining the dichotomy achieved by analysing the deviation sequence:

$$\alpha(l) = \frac{v(l-1)}{v(l)} - \beta(l), \qquad l = 3, 4, \ldots, n$$

where $v(l)$ is the variance estimate at tessellation level l and $\beta(l)$ is the lower variance ratio bounds obtained for a uniform image corrupted with additive gaussian noise. It is generated by the expression:

$$\beta(l) = 1 - (0.1)2^{-l+6}.$$

The exact interpolation expressions used depends on which at tessellation level the signal noise dichotomy occurs.

Parameters:

input The input image.

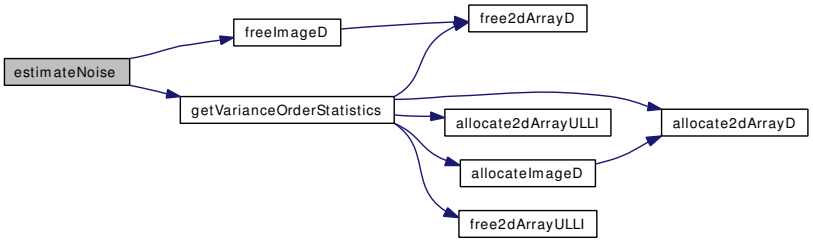
Returns:

The noise variance. Returns -1 if there was not a clear separation between signal and noise.

Definition at line 224 of file noise.c.

References exp2, freeImageD(), getVarianceOrderStatistics(), ImageD::img, log2, ImageUC::nCol, and ImageUC::nRow.

Here is the call graph for this function:



ImageD* getVarianceOrderStatistics (ImageUC * *input*)

Calculate the variance order statics for the image at each level:

- Divide the image into $2^l \times 2^l$ pixel blocks, where l is the current level.
- Calculate the variance for each block.
- Return the four smallest variances sorted in order.

Image pyramid levels range from $l = 1$ (2 x 2 pixel blocks) to $l = N$ where $2^N \times 2^N$ is the largest square block that can be contained in the image. This implementation prioritises speed over memory footprint. It stores the results of level n so they can be used at level n+1. If the input image is Q bytes (1 byte per pixel), this will require an additional Q(0.5*0.5 + 0.25*0.25)*(4+8) = 3.75Q bytes during execution.

Parameters:

input Pointer to struct encapsulating the input image (greyscale 8 bit depth image).

Returns:

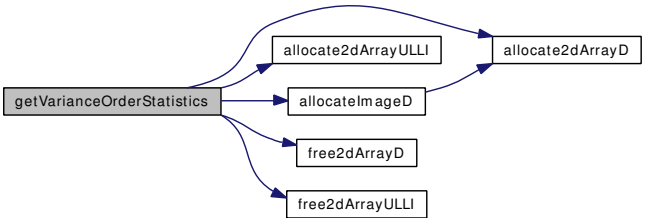
Pointer to **ImageD** (p. 194) struct encapsulating a 2D array that contains the variance order statistics. The rows indices correspond to levels in the image pyramid while the columns contain variances. For example the smallest variance at level "n" would be at output[n-1][0], while the fourth smallest would be at output[n-1][3]. This struct must be freed by the caller using **freeImageD()** (p. 210).

Definition at line 35 of file noise.c.

References allocate2dArrayD(), allocate2dArrayULLI(), allocateImageD(), free2dArrayD(), free2dArrayULLI(), ImageD::img, ImageUC::img, log2, ImageUC::nCol, and ImageUC::nRow.

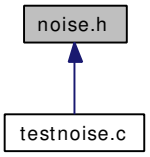
Referenced by estimateNoise().

Here is the call graph for this function:



C.2.8 noise.h File reference

This graph shows which files directly or indirectly include this file:



Detailed description

This file contains declarations of noise estimation related functions.

Definition in file **noise.h**.

Bibliography

[1] [Online]. Available: http://www.koffice.org/krita/ (Cited on page 24.)

[2] [Online]. Available: http://www.pythonware.com/products/pil/ (Cited on page 35.)

[3] [Online]. Available: http://www.doxygen.org (Cited on pages 71 and 194.)

[4] [Online]. Available: http://www.terraserver.com (Cited on page 134.)

[5] [Online]. Available: http://www.php.net (Cited on page 139.)

[6] [Online]. Available: http://developer.yahoo.com/yui/ (Cited on page 139.)

[7] [Online]. Available: http://www.mysql.com (Cited on page 139.)

[8] [Online]. Available: http://www.python.org (Cited on page 183.)

[9] "Final report from the video quality experts group on the validation of objective models of video quality assessment, Phase II," Aug. 2003. [Online]. Available: http://www.vqeg.org (Cited on pages 127, 139, and 157.)

[10] S. Andrefouet and J. Robinson, "The use of Space Shuttle images to improve cloud detection in mapping of tropical coral reef environments," *International Journal of Remote Sensing*, vol. 24, no. 1, pp. 143–149, 2003. [Online]. Available: http://dx.doi.org/10.1080/01431160305007 (Cited on page 12.)

[11] P. Atkinson, "On estimating measurement error in remotely-sensed images with the variogram," *International Journal of Remote Sensing*, vol. 18, no. 14, pp. 3075–3084, 1997. [Online]. Available: http://dx.doi.org/10.1080/014311697217224 (Cited on pages 58, 60, and 61.)

[12] L. Beaudoin, J. Nicolas, F. Tupin, and M. Huckel, "Introducing spatial information in k-means algorithm for clouds detection in optical satellite images," in *Proceedings of the SPIE - The International Society for Optical Engineering*, vol. 4168. SPIE, 2001, pp. 67–77. [Online]. Available: http://dx.doi.org/10.1117/12.413845 (Cited on page 18.)

[13] M. Ben-Ezra and S. K. Nayar, "Motion-based motion deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 689–698, Jun. 2004. (Cited on page 92.)

[14] J. Bendix, R. Rollenbeck, and E. Palacios, "Cloud detection in the tropics - A suitable tool for climate-ecological studies in the high mountains of Ecuador," *International Journal of Remote Sensing*, vol. 25, no. 21, pp. 4521–4540, 2004. [Online]. Available: http://dx.doi.org/10.1080/01431160410001709967 (Cited on pages 5 and 12.)

[15] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information Science and Statistics. Springer, 2006. (Cited on pages 15 and 17.)

[16] R. N. Bracewell, *Two-dimensional imaging*, ser. Prentice Hall Signal Processing. Prentice Hall, 1995. (Cited on pages 93 and 100.)

[17] M. Calvo, A. Manzanares, M. Chevalier, and V. Lakshminarayanan, "Edge image quality assessment: A new formulation for degraded edge imaging," *Image and Vision Computing*, vol. 16, no. 14, pp. 1003–1017, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0262-8856(98)00072-9 (Cited on page 89.)

[18] M. Cannon, "Blind deconvolution of spatially invariant image blurs with phase," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, no. 1, pp. 58–63, Feb. 1976. (Cited on pages 89, 92, 93, 95, 98, 110, and 119.)

[19] B. Chalmond, "PSF estimation for image deblurring," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 4, pp. 364–372, 1991. (Cited on page 91.)

[20] G. Chander and B. Markham, "Revised Landsat-5 TM radiometric calibration procedures and postcalibration dynamic ranges," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 11, pp. 2674–2677, November 2003. (Cited on page 24.)

[21] M. M. Chang, A. M. Tekalp, and A. T. Erdem, "Blur identification using the bispectrum," *IEEE Transactions on Signal Processing*, vol. 39, no. 10, pp. 2323–2325, Oct. 1991. (Cited on pages 92, 93, 95, and 96.)

[22] A. Chappell, J. Seaquist, and L. Eklundh, "Improving the estimation of noise from NOAA AVHRR NDVI for Africa using geostatistics," *International Journal of Remote Sensing*, vol. 22, no. 6, pp. 1067–1080, 2001. [Online]. Available: http://dx.doi.org/10.1080/01431160120633 (Cited on page 60.)

[23] P. Chen, R. Srinivasan, G. Fedosejevs, and B. Narasimhan, "An automated cloud detection method for daily NOAA-14 AVHRR data for Texas, USA," *International Journal of Remote Sensing*, vol. 23, no. 15, pp. 2939–2950, 2002. [Online]. Available: http://dx.doi.org/10.1080/01431160110075631 (Cited on pages 5, 12, and 14.)

[24] D. Childers, D. Skinner, and R. Kemerait, "The cepstrum: a guide to processing," *Proceedings of the IEEE*, vol. 65, no. 10, pp. 1428–1443, Oct. 1977. (Cited on page 93.)

[25] B. Corner, R. Narayanan, and S. Reichenbach, "Noise estimation in remote sensing imagery using data masking," *International Journal of Remote Sensing*, vol. 24, no. 4, pp. 689–702, 2003. [Online]. Available: http://dx.doi.org/10.1080/01431160210164271 (Cited on pages 59 and 63.)

[26] E. P. Crist and R. J. Kauth, "The tasseled cap de-mystified," *Photogrammetric engineering and Remote Sensing*, vol. 52, no. 1, pp. 81–86, January 1986. (Cited on page 14.)

[27] P. J. Curran, "Semivariogram in remote sensing: An introduction," *Remote Sensing of Environment*, vol. 24, no. 3, pp. 493–507, 1988. [Online]. Available: http://dx.doi.org/10.1016/0034-4257(88)90021-1 (Cited on page 60.)

[28] P. J. Curran and J. L. Dungan, "Estimation of signal-to-noise: A new procedure applied to AVIRIS data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 27, no. 5, pp. 620–628, 1989. [Online]. Available: http://dx.doi.org/10.1109/TGRS.1989.35945 (Cited on pages 58, 59, 60, 61, and 69.)

[29] N. Damera-Venkata, T. Kite, W. Geisler, B. Evans, and A. Bovik, "Image quality assessment based on a degradation model," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 636–650, Apr. 2000. (Cited on pages 59 and 122.)

[30] P. A. Devijver and J. Kittler, *Pattern recognition: a statistical approach*. Prentice-Hall International, 1982, p. 15. (Cited on page 124.)

[31] L. Di Girolamo and R. Davies, "Image navigation cloud mask for the Multiangle Imaging SpectroRadiometer (MISR)," *Journal of Atmospheric and Oceanic Technology*, vol. 12, no. 6, pp. 1215–1228, 1995. [Online]. Available: http://dx.doi.org/10.1175/1520-0426(1995)012<1215:TINCMF>2.0.CO;2 (Cited on pages 13, 26, and 37.)

[32] D. J. Diner, J. C. Beckert, G. W. Bothwell, and J. I. Rodriguez, "Performance of the MISR instrument during its first 20 months in earth orbit," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 7, pp. 1449–1466, July 2002. [Online]. Available: http://dx.doi.org/10.1109/TGRS.2002.801584 (Cited on page 2.)

[33] D. J. Diner, L. D. Girolamo, and E. E. Clothiaux, "Multi-angle imaging spectro-radiometer: Level 1 cloud detection algorithm theoretical basis," Desember 1999, accessed October 2008. [Online]. Available: http://eospso.gsfc.nasa.gov/eos_homepage/for_scientists/atbd/docs/MISR/atbd-misr-06.pdf (Cited on page 177.)

[34] J. A. Du Preez, "Efficient high-order hidden markov modelling," Ph.D. dissertation, University of Stellenbosch, 1998. (Cited on page 27.)

[35] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, Inc., 2001, ch. 6, pp. 282–335. (Cited on page 130.)

[36] Eduard Kriegler, "An image compression system for LEO satellites," Master's thesis, University of Stellenbosch, 2003. (Cited on page 2.)

[37] J. H. Elder and S. W. Zucker, "Local scale control for edge detection and blur estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 699–716, 1998. (Cited on page 91.)

[38] C. F. England and G. E. Hunt, "Bispectral method for the automatic determination of parameters for use in imaging satellite cloud retrievals," *International Journal of Remote Sensing*, vol. 6, no. 9, pp. 1545–1553, 1985. (Cited on pages 9 and 11.)

[39] B. Escalante-Ramirez, J.-B. Martens, and H. de Ridder, "Multidimensional characterization of the perceptual quality of noise-reduced computed tomography images," *Journal of Visual Communication and Image Representation*, vol. 6, no. 4, pp. 317–334, 1995. [Online]. Available: http://dx.doi.org/10.1006/jvci.1995.1027 (Cited on pages 124 and 125.)

[40] R. Fabian and D. Malah, "Robust identification of motion and out-of-focus blur parameters from blurred and noisy images," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 5, pp. 403–412, Sep. 1991. (Cited on pages 94, 95, 96, 98, and 112.)

[41] J. Flusser and T. Suk, "Degraded image analysis: An invariant approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 590–603, 1998. [Online]. Available: http://dx.doi.org/10.1109/34.683773 (Cited on page 91.)

[42] B.-C. Gao, "Operational method for estimating signal to noise ratios from data acquired with imaging spectrometers," *Remote Sensing of Environment*, vol. 43, no. 1, pp. 23–33, 1993. [Online]. Available: http://dx.doi.org/10.1016/0034-4257(93)90061-2 (Cited on pages 59, 62, 69, 71, 73, and 86.)

[43] P. Gastaldo, R. Zunino, I. Heynderickx, and E. Vicario, "Objective quality assessment of displayed images by using neural networks," *Signal Processing: Image Communication*, vol. 20, no. 7, pp. 643–661, 2005. (Cited on pages 123, 124, 127, and 134.)

[44] G. B. Giannakis and R. W. J. Heath, "Blind identification of multichannel fir blurs and perfect image restoration," *IEEE Transactions on Image Processing*, vol. 9, no. 11, pp. 1877–1896, Nov. 2000. (Cited on page 91.)

[45] L. Gillick and S. J. Cox, "Some statistical issues in the comparison of speech recognition algorithms," in *IEEE International Conference on Accoustics, Speech, and Signal Processing*, vol. 1, 1989, pp. 532–535. (Cited on page 27.)

[46] G. Ginesu, F. Massidda, and D. D. Giusto, "A multi-factors approach for image quality assessment based on a human visual system model," *Signal Processing: Image Communication*, vol. 21, no. 4, pp. 316 –333, Apr. 2006. (Cited on pages 109, 122, 124, and 127.)

[47] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Prentice Hall, 2002. (Cited on pages 2, 4, 6, 8, 9, 18, 31, 58, 60, 86, 88, 89, 109, 186, and 188.)

[48] A. H. Goodman and A. Henderson-Sellers, "Cloud detection and analysis: A review of recent progress," *Atmospheric Research*, vol. 21, no. 2, pp. 203–228, 1988. (Cited on pages 24 and 54.)

[49] M. T. Goodrich and R. Tamissa, *Data structures and algorithms in Java*, 3rd ed. John Wiley & Sons, Inc., 2004, pp. 305–338. (Cited on pages 183 and 184.)

[50] F. Gunst, Richard and R. L. Mason, *How to construct fractional factorial experiments*, ser. Basic References in Quality Control: Statistical Techniques. American Society for Quality Control Press, 1991, vol. 14. (Cited on pages 128, 136, 138, 161, 163, and 164.)

[51] G. G. Gutman, "Satellite daytime image classification for global studies of earth's surface parameters from polar orbiters," *International Journal of Remote Sensing*, vol. 13, no. 2, pp. 209–234, 1992. (Cited on pages 5, 6, 11, and 12.)

[52] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer Series in Statistics. Springer, 2001. (Cited on pages 16, 125, 127, 128, 129, 130, 132, and 133.)

[53] S. Hojjatoleslami and J. Kittler, "Region growing: A new approach," *IEEE Transactions on Image Processing*, vol. 7, no. 7, pp. 1079–1084, 1998. [Online]. Available: http://dx.doi.org/10.1109/83.701170 (Cited on pages 21 and 184.)

[54] P. Hou, M. Petrou, C. Underwood, and A. Hojjatoleslami, "Improving JPEG performance in conjunction with cloud editing for remote sensing applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 1, pp. 515–524, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/36.823946 (Cited on pages 2, 19, 20, 21, 28, 56, 57, and 187.)

[55] M.-F. Huang, S.-H. Liu, L. Li, and Q.-J. Zhu, "Study on data models of image quality assessment for the Chinese-Brazil earth resources satellite," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 6, 2004, pp. 3949–3952. (Cited on page 2.)

[56] B. Jahne, *Practical handbook on image processing for scientific technical applications*. CRC Press, 2004, p. 293. (Cited on page 3.)

[57] J.-D. Jang, A. A. Viau, F. Anctil, and E. Bartholome, "Neural network application for cloud detection in SPOT vegetation images," *International Journal of Remote Sensing*, vol. 27, no. 4, pp. 719–736, 2006. [Online]. Available: http://dx.doi.org/10.1080/01431160500106892 (Cited on page 19.)

[58] G. Jedlovec, S. Haines, and F. LaFontaine, "Spatial and temporal varying thresholds for cloud detection in GOES imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 6, pp. 1705–1717, June 2008. [Online]. Available:

http://dx.doi.org/10.1109/TGRS.2008.916208 (Cited on pages 8 and 177.)

[59] J. Kittler and D. Pairman, "Contextual pattern recognition applied to cloud detection and identification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. GE-23, no. 6, pp. 855–863, 1985. (Cited on pages 17, 19, and 57.)

[60] K. Krause, "Radiometric use of Quickbird imagery," DigitalGlobe, Longmont, Colorado, USA, Tech. Rep., 2005. (Cited on page 24.)

[61] N. Kumar and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition," *Speech Communication*, vol. 26, no. 4, pp. 283–297, 1998. [Online]. Available: http://dx.doi.org/10.1016/S0167-6393(98)00061-2 (Cited on pages 16 and 17.)

[62] D. Kundur and D. Hatzinakos, "Blind image deconvolution," *IEEE Signal Processing Magazine*, vol. 13, no. 3, pp. 43–64, May 1996. (Cited on pages 91, 92, and 95.)

[63] ——, "Blind image deconvolution revisited," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 61–63, Nov. 1996. (Cited on page 91.)

[64] J. Lee and K. Hoppel, "Noise modeling and estimation of remotely-sensed images," *Digest - International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2, pp. 1005–1008, 1989. (Cited on page 70.)

[65] S. U. Lee, S. Y. Chung, and R. H. Park, "A comparative performance study of several global thresholding techniques for segmentation," *Computer Vision, Graphics, and Image Processing*, vol. 52, pp. 171–190, 1990. (Cited on page 8.)

[66] X. Li, "Blind image quality assessment," in *Proceedings 2002 International Conference on Image Processing*, vol. 1, 2002, pp. 449–452. (Cited on pages 90, 123, and 124.)

[67] J. Lim, "Image restoration by short space spectral subtraction," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-28, no. 2, pp. 198–204, Apr. 1980. (Cited on page 94.)

[68] E. Lorenz, W. Barwald, K. Briess, H. Kayal, M. Schneller, and H. Wusten, "Resumes of the BIRD mission," in *Proceedings of the 4S Symposium: Small Satellites, Systems and Services*, no. 571, 2004, pp. 249–259. (Cited on page 2.)

[69] D. J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992. (Cited on page 133.)

[70] P. S. Mann, *Introductory Statistics*, 5th ed. John Wiley & Sons, Inc., 2004. (Cited on pages 27, 157, and 161.)

[71] I. v. Z. Marais, J. A. du Preez, and W. H. Steyn, "An optimal image transform for threshold-based cloud detection using heteroscedastic discriminant analysis," *International Journal of Remote Sensing*, accepted pending minor changes Dec. 2008. (Cited on page 56.)

[72] I. v. Z. Marais and W. H. Steyn, "Robust defocus blur identification in the context of blind image quality assessment," *Signal Processing: Image Communication*, vol. 22, no. 7, pp. 833–844, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.image.2007.06.003 (Cited on page 120.)

[73] I. v. Z. Marais, W. H. Steyn, and J. A. du Preez, "Construction of an image quality assessment model for use on board an LEO satellite," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 2. IEEE, 7 – 11July 2008, pp. 1068 – 1071. [Online]. Available: http://dx.doi.org/10.1109/IGARSS.2008.4779183 (Cited on page 174.)

[74] J. A. Marchant, "Testing a measure of image quality for acquisition control," *Image and Vision Computing*, vol. 20, no. 7, pp. 449–458, 2002. [Online]. Available: http://dx.doi.org/10.1016/S0262-8856(01)00088-9 (Cited on pages 123, 124, and 125.)

[75] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi, "Perceptual blur and ringing metrics: Application to JPEG2000," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 163–172, Feb. 2004. (Cited on pages 90 and 123.)

[76] P. Meer, J.-M. Jolion, and A. Rosenfeld, "Fast parallel algorithm for blind estimation of noise variance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 216–223, 1990. [Online]. Available: http://dx.doi.org/10.1109/34.44408 (Cited on pages 63, 65, 82, and 86.)

[77] C. Nikias and M. Raghuveer, "Bispectrum estimation: A digital signal processing framework," *Proceedings of the IEEE*, vol. 75, no. 7, pp. 869–891, Jul. 1987. (Cited on page 93.)

[78] S. Olsen, "Estimation of noise in images: An evaluation," *CVGIP: Graphical Models and Image Processing*, vol. 55, no. 4, pp. 319–323, Jul. 1993. (Cited on pages 70 and 81.)

[79] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Prentice Hall, 1975, ch. 11, pp. 543–548. (Cited on pages 102 and 105.)

[80] K. Panchapakesan, D. Sheppard, M. Marcellin, and B. Hunt, "Blur identification from vector quantizer encoder distortion," *IEEE Transactions on Image Processing*, vol. 10, no. 3, pp. 465–470, Mar. 2001. (Cited on page 91.)

[81] C. H. Park and H. Park, "A comparison of generalized linear discriminant analysis algorithms," *Pattern Recognition*, vol. 41, no. 3, pp. 1083–1097, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2007.07.022 (Cited on page 16.)

[82] P. Z. Peebles, *Probability, Random Variables, and Random Signal Principles*, 4th ed., ser. McGraw-Hill Series in Electrical and Computer Engineering. McGraw-Hill, 2001, pp. 227–230. (Cited on page 93.)

[83] W. Potter and J. Gasch, "A photo album of earth: Scheduling Landsat 7 mission daily activities," in *Proceedings of the International Symposium on Space Mission Operations and Ground Data Systems*, 1998. [Online]. Available: http://isd.gsfc.nasa.gov/Papers/DOC/WPms2b010.pdf (Cited on page 178.)

[84] J. G. Proakis and D. G. Manolakis, *Digital signal processing: principles, algorithms, and applications*, 3rd ed. Prentice-Hall, 1996. (Cited on pages 93, 103, and 110.)

[85] R. Richter, "Spatially adaptive fast atmospheric correction algorithm," *International Journal of Remote Sensing*, vol. 17, no. 6, pp. 1201–1214, 1996. (Cited on pages 13 and 14.)

[86] ——, "Atmospheric/topographic correction for satellite imagery: Atcor-2/3 user guide," 2008, accessed March 2008. [Online]. Available: www.rese.ch/pdf/atcor23_manual.pdf (Cited on pages 13 and 14.)

[87] W. Rossow, "Measuring cloud properties from space: A review," *Journal of Climate*, vol. 2, pp. 201–213, 1989. (Cited on pages 9 and 13.)

[88] P. K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen, "Survey of thresholding techniques," *Computer Vision, Graphics, and Image Processing*, vol. 41, no. 2, pp. 233–260, 1988. (Cited on page 8.)

[89] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2. IEEE, 11 – 15 July 2000, pp. 1129–1132. (Cited on page 16.)

[90] R. W. Saunders, "Automated scheme for the removal of cloud contamination from AVHRR radiances over Western Europe," *International Journal of Remote Sensing*, vol. 7, no. 7, pp. 867–886, 1986. (Cited on pages 5, 10, 12, and 13.)

[91] R. W. Saunders and K. T. Kriebel, "Improved method for detecting clear sky and cloudy radiances from AVHRR data," *International Journal of Remote Sensing*, vol. 9, no. 1, pp. 123–150, 1988. (Cited on pages 5, 11, and 12.)

[92] A. Savakis and J. Easton, R.L., "Blur identification based on higher order spectral nulls," in *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 2302, 1994, pp. 168–177. (Cited on pages 92, 94, 95, 96, and 98.)

[93] A. A. Sawchuk, "Space-variat image motion degradation and restoration," *Proceedings of the IEEE*, vol. 60, no. 7, pp. 854–861, Jul. 1972. (Cited on page 92.)

[94] R. A. Schowengerdt, *Remote Sensing, Models and Methods for Image Processing*, 2nd ed. Academic Press, 1997. (Cited on pages 3, 13, 14, 31, 58, 59, 60, and 88.)

[95] M. Seul, L. O'Gorman, and M. J. Sammon, *Practical Algorithms for Image Analysis*. Cambridge University Press, 2000. (Cited on pages 123 and 188.)

[96] H. R. Sheikh and A. C. Bovik, "Image information and visual quality," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 430–444, Feb. 2006. (Cited on pages 122 and 128.)

[97] H. Sheikh, M. Sabir, and A. Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, Nov. 2006. (Cited on pages 59, 122, 127, 133, 134, 136, 138, 139, 141, 142, 157, 171, and 172.)

[98] D. Shin, J. Pollard, and J.-P. Muller, "Cloud detection from thermal infrared images using a segmentation technique," *International Journal of Remote Sensing*, vol. 17, no. 14, pp. 2845–2856, 1996. (Cited on page 12.)

[99] X. Song, Z. Liu, and Y. Zhao, "Cloud detection and analysis of MODIS image," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 4, 2004, pp. 2764–2767. (Cited on page 19.)

[100] C. Stubenrauch, W. Rossow, F. Cheruy, A. Chedin, and N. Scott, "Clouds as seen by satellite sounders (3I) and imagers (ISCCP). Part I: evaluation of cloud parameters," *Journal of Climate*, vol. 12, no. 8 I, pp. 2189–2213, 1999. (Cited on page 5.)

[101] K.-C. Tan, H. Lim, and B. Tan, "Windowing techniques for image restoration," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 5, pp. 491–500, 1991. (Cited on page 93.)

[102] B. Tian, M. R. Azimi-Sadjadi, M. A. Shaikh, and T. Vonder-Haar, "FFT-based algorithm for computation of gabor transform with its application to cloud detection/classification," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 2, 1996, pp. 1108–1110. (Cited on page 19.)

[103] H. J. Trussell and B. R. Hunt, "Image restoration of space-variant blurs by sectioned methods," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-26, no. 6, pp. 608–609, 1978. [Online]. Available: http://dx.doi.org/10.1109/TASSP.1978.1163161 (Cited on page 92.)

[104] D. S. Turaga, Y. Chen, and J. Caviedes, "No reference PSNR estimation for compressed pictures," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 173–184, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.image.2003.09.001 (Cited on page 123.)

[105] D. Van der Weken, M. Nachtegael, and E. Kerre, "Combining neighbourhood-based and histogram similarity measures for the design of image quality measures," *Image and Vision Computing*, vol. 25, no. 2, pp. 184–195, 2007. (Cited on pages 122 and 127.)

[106] P. Walder and I. MacLaren, "Neural network based methods for cloud classification on AVHRR images," *International Journal of Remote Sensing*, vol. 21, no. 8, pp. 1693–1708, 2000. [Online]. Available: http://dx.doi.org/110.1080/014311600209977 (Cited on page 19.)

[107] H. Wang, T.-Z. Shen, and Z.-H. Xie, "Blind image quality assessment based on hybrid fuzzy-genetic technique," *Journal of Beijing Institute of Technology (English Edition)*, vol. 12, no. 4, pp. 395–398, 2003. (Cited on pages 123 and 127.)

[108] D. Welch, "The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-15, no. 2, pp. 70–73, Jun. 1967. (Cited on page 93.)

[109] R. Welch, K. Kuo, and S. Sengupta, "Textural characteristics of cloud- and ice-covered surfaces in polar regions," in *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium*, vol. 5, 1989, pp. 2773–2776. (Cited on pages 18 and 19.)

[110] R. M. Welch, K.-S. Kuo, and S. K. Sengupta, "Cloud and surface textural features in polar regions," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 520–528, 1990. [Online]. Available: http://dx.doi.org/10.1109/TGRS.1990.572939 (Cited on page 19.)

[111] J. Wertz and W. Larson, *Space Mission Analysis and Design*, 3rd ed. Kluwer Academic Publishers, 1999, p. 6. (Cited on page 1.)

[112] M. Wettle, V. E. Brando, and A. G. Dekker, "A methodology for retrieval of environmental noise equivalent spectra applied to four hyperion scenes of the same tropical coral reef," *Remote Sensing of Environment*, vol. 93, no. 1-2, pp. 188 –197, 2004. [Online]. Available: http://dx.doi.org/10.1016/j.rse.2004.07.014 (Cited on pages 69 and 86.)

[113] B. A. Wielicki and L. Parker, "On the determination of cloud cover from satellite sensors: The effect of sensor spatial resolution," *Journal of Geophysical Research*, vol. 97, no. D12, pp. 12 799–12 823, August 1992. (Cited on page 54.)

[114] Y. Yang, L. Di Girolamo, and D. Mazzoni, "Selection of the automated thresholding algorithm for the multi-angle imaging spectroradiometer radiometric camera-by-camera cloud mask over land," *Remote Sensing of Environment*, vol. 107, no. 1-2, pp. 159–171, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.rse.2006.05.020 (Cited on pages 8, 10, 13, 25, and 177.)

[115] P. Yap and P. Raveendran, "Image focus measure based on Chebyshev moments," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 151, no. 2, pp. 128–136, 2004. [Online]. Available: http://dx.doi.org/10.1049/ip-vis:20040395 (Cited on page 88.)

[116] Y. Yitzhaky and N. Kopeika, "Identification of blur parameters from motion blurred images," *Graphical Models and Image Processing*, vol. 59, no. 5, pp. 310–320, 1997. (Cited on page 91.)

[117] Y. Yitzhaky, I. Mor, A. Lantzman, and N. Kopeika, "Direct method for restoration of motion-blurred images," *Journal of the Optical Society of America A: Optics and Image Science, and Vision*, vol. 15, no. 6, pp. 1512–1519, 1998. (Cited on page 91.)

[118] Y. Yitzhaky, R. Milberg, S. Yohaev, and N. S. Kopeika, "Comparison of direct blind deconvolution methods for motion-blurred images," *Applied Optics*, vol. 38, no. 20, pp. 4325–4332, 1999. (Cited on pages 91 and 95.)

[119] Y.-L. You and M. Kaveh, "Regularization approach to joint blur identification and image restoration," *IEEE Transactions on Image Processing*, vol. 5, no. 3, pp. 416–428, 1996. [Online]. Available: http://dx.doi.org/10.1109/83.491316 (Cited on page 92.)

[120] Y. Zhang, B. Guindon, and J. Cihlar, "An image transform to characterize and compensate for spatial variations in thin cloud contamination of landsat images," *Remote Sensing of Environment*, vol. 82, no. 2, pp. 173–187, 2002. (Cited on pages 14 and 39.)

[121] Y. Zhang, Y. Zhang, and C. Wen, "New focus measure method using moments," *Image and Vision Computing*, vol. 18, no. 12, pp. 959–965, 2000. [Online]. Available: http://dx.doi.org/10.1016/S0262-8856(00)00038-X (Cited on page 88.)