CHARACTERISATION OF RICE GRAINS USING DIGITAL IMAGING TECHNIQUES

A Thesis Submitted to the University of Kent
For the Degree of Doctor of Philosophy
In Electronic Engineering

By

David Mark Hobson BSc(Honours)

October 2009

Characterisation of Rice Grains Using Digital Imaging Techniques

David Mark Hobson, PhD 2009

Abstract

Visual characteristics of rice grains can be quantified through image acquisition and processing. In this research image capture hardware and dedicated image processing algorithms are developed as part of a system for rice grain characterisation. A series of experimental work was undertaken to test the effectiveness of the image processing algorithms. The imaging technique is non-intrusive and non-destructive of sample grains, and can be implemented at a low cost compared to other more established techniques.

In this thesis a review of techniques for the characterisation of rice grains is presented, with the main focus upon digital imaging approaches to understand the role it can play in isolation and in conjunction with other techniques. Following the review a physical rice grain image capture setup is designed, implemented and tested extensively with a range of cameras. A number of established image processing techniques are used with the custom built image capture setup to complete a novel system for the characterisation of rice grains. Extraction of features from the processed images is undertaken in order to test the features as being suitable to return descriptive characteristics of individual rice grains. In the course of these activity novel features, methodologies and feature analysis are created and implemented with wider potential applications beyond rice imaging. Results of these tests are given within this thesis. These tests take several forms in multiple sets of rice grain images, which were created, processed, and subject to different kinds of analysis to find different features.

Findings show that digital imaging can return a range of valid rice characteristics which are connectable to fraud and grading issues, identification of rice types, and assisting the physical measurement of rice grain properties. The performance of the system is discussed and suggestions given for the future development of the methodology.

Acknowledgements

I would first like to thank my supervisor Professor Yong Yan and the staff and students of the Department of Electronics whose advice and contributions made it possible for me to complete this work.
Dr Robert Carter was always most supportive, particularly in reading the endless drafts of papers and thesis chapters and offering near endless help.

Thanks to Karl Harmer who helped me develop the idea for the measurement of edge detection as a form of texture analysis. The staff and students of the Instrumentation, Control and Embedded Systems Research Group in the Department of Electronics, particularly Ken Roberts for his support with cameras and optics, Lingjun Gao (Sally) for assisting in feature value calculations, Uche Nnolim for lots of help with image processing and Fourier transforms, and Farzin Deravi for his help with tangent angle curvature functions.

Special thanks to the friends I've known here who kept me going, James Weston, Jason Shao, Jan Krabicka, Gary Lu, Eric Geng, Neil Xie, Peter Lee and his students, Sarah Spurgeon, Sally and Danny Sun, Omer Cliric, Julian, Harvey and the workshop staff, Tristan Linnel, Hannah Greer, Johan Helmer, Claire Pollard, Tim Freeburn, Yiquing Liang, Serkawt Koala, Guillermo Guillabert, Eric Geng, Neil, Kostas, Michael Wisdom, the list goes on.

And thanks to my family for their unwavering care and support, and my brother for giving me the idea of getting into academia in the first place.

Contents

Chapter 1

Requirements for Imaging Based Characterisation of Rice Grains

1.1 Introduction

The measurement of visual characteristics of rice grains has useful implications to agricultural and commercial sectors. Imaging provides low cost measurement solutions to analyse rice and other food grains. Hardware and methodologies are of interest to a wide range of organisations and present practical research opportunities involving applied image processing. In the present work focus is narrowed to a small set of varieties of different rice grains which can be examined with digital imaging. By digital imaging, this refers to development of imaging setups and image analysis techniques in order to perform measurements of grain characteristics for various purposes. These include practical standards of measurement for identification, grading and authentication. Certain visible features present in the shape of a grain,can also be detected and measured. These include unique features specific to milled rice grains, such as the curvature at an end of the milled grain. Novel image processing methodologies are needed to solve real problems.

Imaging based measurement of food grains has been of interest to the Instrumentation, Control and Embedded Systems Group at the University of Kent in recent years. The present work builds on a number of useful imaging methods established by Carter & Yan (2005), and considers requirements for this subject defined by other relevant organisations such as the International Rice Research Institute (IRRI) and British Retail Consortium (BRC). These organisations have set a number of standards for the quality of rice which can be derived from grain appearance. In addition, other researchers such as Paliwal (2003) and Senkai (1996) have over an extended time specialised in the measurement of food grain characteristics through imaging approaches indicating this is a wide field of research with many overlapping problems and potential solutions. This is compounded by related work undertaken during the course of this research for the evaluation of coal, non-ore gangue, and biomass material. Such work resulted in a separate publication (Hobson, Carter & Yan 2007) although the focus of this thesis is the assessment of rice grains only.

During the course of this research it has been intended to implement approaches for the characterisation of grains solely using simple low cost imaging systems. Given the constant reduction in cost and common availability of digital cameras and computer systems and their potential as a non-invasive form of controlled instrumentation, there is commercial viability for many analytical tasks. Simple static imaging setups have been widely employed for the imaging of objects to such an extent that specialist literature has been found covering the subject of shape and textural analysis from resultant two-dimensional images. The present work focuses on the study of the effectiveness of various camera systems of different cost and quality, as well as image processing methods and shape features for rice grain analysis and assessments.

1.2 Importance of Grain Characterisation

Rice is a vital worldwide commodity. The united states department of agriculture records extensive statistics on world and regional rice production (www.fas.usda.gov/). Production in 2007/2008 totalled over 430 million tons, a figure that increased close to 450 in 2009. Major producers are China and India, with Bangladesh and Indonesia close behind. Little rice is grown in Europe compared to the United states and other regions, which in turn have their production dwarfed by the aformentioned asian countries. However rice imports to these regions are considerable. 1.1 million tons of rice were imported into the European Union between 2005/2006 according to one defra report (www.defra.gov.uk/). Similarly, according to UK Government statistics (www.statistics.gov.uk), imports of milled rice in the uk in 2007 were valued at over 90 million pounds sterling, with over 216,000 tons imported into the UK. Milled rice at this time had a typical price of 71 pence per kilogram.

A critical issue is that these standards have two limitations. First, they are not international standards. These are generally the standards of specific research agencies and national bodies. Second, these are not legislative, and can only be taken as guidelines by national agencies. This constitutes a significant factor in the adulteration of rice grains, as there are no legal obligation to conform to any standards This presents problems particularly for Basmati Rice has up to four times the commercial price of conventional long grains .

Varieties of rice grains are subtly different in size, colour, shape and texture. Fraudulent labelling of one variety as another is a major concern in the food industry. The International Rice Research Institute (IRRI) (http://www.irri.org) and other organizations including the British Retail Consortium (http://www.brc.org.uk/) discuss the appearance of rice as being critical to consumers. These organisations have established standards and regulations for rice grain varieties. These standards also grant special status to specific rice varieties such as Basmati due to their high market value. Certain physical characteristics such as minimum average length and aspect ratio are specified explicitly in the standards (more in chapter 2).

Imaging of food grains and other similar granular or particulate materials of various scales reveals important properties useful for scientific and commercial purposes. Many particle types can be identified based solely on their physical appearance, and the behaviour of particles can be attributed to visible characteristics. Quality measures of granular materials impact many industries. Food grains such as rice are often the subject of fraud and such concerns are frequently reported in popular media (Keating 2006, Ravilious 2006, Chakravarty 2007) and by organisations such as British Retail Consortium (2004). Such reports have indicated some visible characteristics can be influential in their identification. For many grains their shape, texture, size and other properties may offer useful measurements for purposes varying from identification to grading. Such features are important as many materials can be clearly identified by such factors alone. Measuring these properties can also be indicative of the characteristics and behaviour of materials. In addition these imaging features of particulate and granular objects can be used in conjunction with a range of other measurement methods such as chemical and spectroscopic analysis.

1.3 Major Challenges

The challenge in this research is to address the following problem. Given that there is interest in standards for grain measurement for quality and adulteration detection, can a low-cost imaging system process images of and retrieve useful characteristics of rice grains. This problem leads to questions:

- Other techniques exist, how can imaging specifically offer useful measurements?
- Given the low-cost nature and ease of use of many camera systems, can this be made simple and inexpensive, and how might this be useful?
- What features of use could possibly be derived using imaging, and how useful could they be?
- To that end, how can image processing techniques be employed and improved in this research for this task, and similar image processing research at large?

These challenges can be broken down further to several specific challenges, both scientific and technical in nature. The imaging of granular foods is not a new field, but there are still many issues to be resolved. The main challenges that will impact research activities are described briefly below and will be elaborated upon in the literature review. It is of our aim to investigate the solutions to these issues in particular addressing the real world problems that previous research has shown that can occur.

1.3.1 Passive Low Cost Measurement

In many areas of the supply chain, passive assessment of food grains is highly desirable in order to assess quality and grade products. This is often associated with grain sorting machinery and many devices exist in the market for this purpose. Non-destructive analysis is potentially versatile for all stages of the supply and distribution chain as it preserves the grains examined. Grains may also be fragile, prone to cracking and breaking and require a sensible level of interaction. One potential field of application is a low cost practical device may find a wide range of commercial applications in the food supply chain, such as a system to passively measure rice that is being sorted, or use with existing systems described in the literature review. Cameras can be used in monitoring setups (Carter & Yan 2005). Imaging techniques can be employed with existing more specialist forms of evaluation as a supportive measurement medium. A low cost imaging system is also useful in this role as a simpler and cheaper alternative to many more complex methods.

The primary challenge in this sense was the need to produce a low cost system that was reliable enough for effective measurement to be performed. As the approach adopted was digital imaging, this requires dealing with different systems with a range of physical drawbacks, including digitisation due to low resolution. Image compression was often integral in the cameras used. Physical problems that impact the area of imaging, in terms of size of region of interest, distortion, light sensitivity and automatically adjusting contrast also needed to be addressed . All these issues need to be considered when implementing an imaging based technique for a measurement task.

1.3.2 Imaging Problems

This type of research offers the opportunity to develop solutions for a number of existing imaging problems that effect static and flowing imaging situations. One such problem is the apperance of physically touching grains within an image. One solution to this problem is segmentation. Relevant existing work on segmentation of contacting particles, food grains and other objects in a sample of images can be extended for specific problems in rice imaging. Previous work has attempted to avoid the problem, and only a few solutions of limited scope exist. While it is possible to reduce the risk of this occurring in a flowing system, the present work will show segmentation of touching objects is a viable alternative. This has been addressed often in separate avenues in research yet it may be particularly applicable to the rice grains and other particle types. Such research by others is reviewed for this purpose.

Segmentation tests on real rice grains were proformed to meet this challenge. This required research and development of effective segmentation functions, as well as physical tests on grain images. Rice grain images requiring segmentation are illustrated in Figure 1.1.

In this specific task it was important to address a question: What constitutes a good quality of segmentation? The task specifically is to segment grains so they can be measured individually, so the answer to this question must be in this context. The nature of segmentation is highly subjective and must be carefully considered.

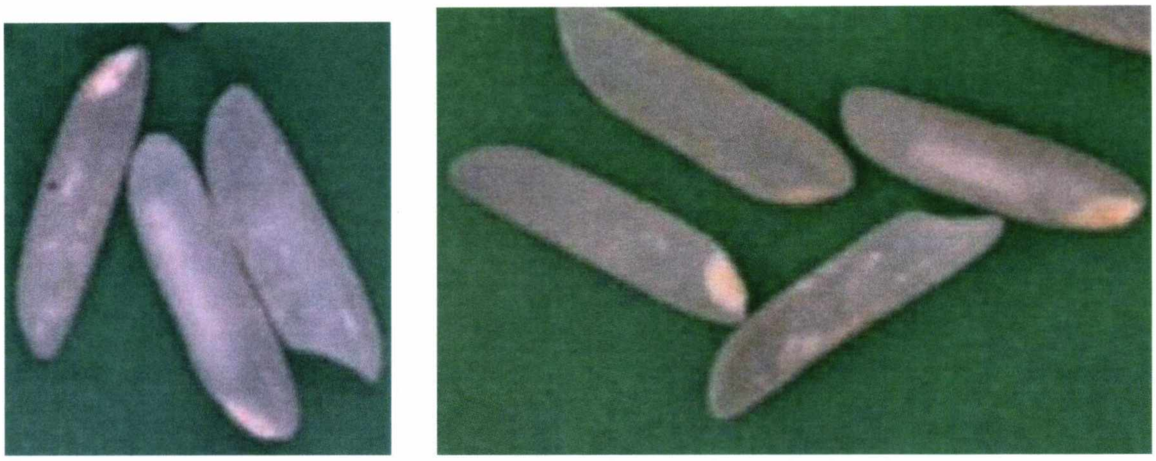


Figure 1.1. Examples of touching grains in an image requiring segmentation.

1.3.3 Identification of Effective Features

For the purpose of classification between different types of rice grains, a target particle needs distinct criteria for measurement. These can include standardised metrics such as length, width, area, or more invariant general descriptors such as shape and texture. Retrieval of useful features from images is considered to be a goal of this research, along with the actual image acquisition and processing. Quality of features will be discussed throughout this thesis. The Classification of useful features required extensive research, although some has been identified in the past by the present research group (Carter & Yan 2005) and elsewhere (Sakai, Yonekawa & Matsuzaki 1996; Brosnan & Sun 2002; Cox & Budhu 2008).

This task involves testing for the evaluation of different features in a range of different ways. Early work focussed on simple shape description techniques such as shape features which were extenstively investigated to see if one or several in conjunction could be applied towards classification of different types of rice grains. The effectiveness of these features are described and discussed in relation to the classification performance. An example simple feature would be aspect ratio of a shape. Further to this, detection of special features requires more complex shape description. An example feature of interest was a curvature based approach of detecting the 'eye' socket of the rice grain and the testing of such features on images describing this part are also extensively tested. Texture of milled and brown grains was also considered as a measurement target. Texture measurements were developed and were used in conjunction with extended work during the course of this research in the analysis of coal samples. A feature was also created that was believed to be novel as no existing examples had been found by time of publication. This feature was essentially a simple textural measure using edge detection, which appeared to effectively differentiate grains. Able features for the distinction or grading of grains are sought in this research. Although this will also depend on research to identify what properties of rice may be of use, as result there are two elements to this particular challenge. First is review of literature to find what features may be of use, and second analysis of the features returned from processed images of rice. These will be described in later chapters

1.3.4 Development of Image Processing Techniques

There are a number of issues needed to be described that impact image quality and objects to be measured. Appropriate image manipulation can mitigate these issuesand. For instance, the shape, size and texture of rice grains are themselves significant properties in an image. If grain characteristics are known to be of a certain type, and if in an image an object is found with highly irregular parameters, such an object can be isolated for further processing. In the earlier segmentation example an object like this may be not one single object, but two distinct objects in contact and may be distinguished through further image processing. Although it should be noted that work investigating the impact of some forms of segmentation and surveying segmentation functions has been carried out in the past (Zhang 1995, 1996). In such work the impact of segmentation on shape properties was quantified, which has some direct bearing on this research. However these types of segmentation approaches related more to specific threshold segmentation type functions, such as Carter's adaptive threshold (Carter & Yan 2005). Other general image properties can assist in detection of several types of image distortion and assist in calibration of cameras and an imaging set-up. Physical changes in the imaging set-up are also considered as an application of such research. For example, incorporating colour imaging permits more approaches to thresholding and illumination than ways considered in previous research.

1.4 Grain parameters

From a 2D image of grains *size*, *shape* and *texture* properties are the general features that can be retrieved. However these three words only summarise large established areas of measurement and analysis. Existing research such as particle sizing is relevant to grain sizing as a long established field already used as a reliable form of measurement. Particle shape is also heavily used and is relevant to grain shape, and the wider scope of shape analysis techniques can be imported from general areas of mathematics, image processing and other particle imaging for this task.

1.4.1 Size

Grain sizing is derived using calibration images, and is useful characteristic for distinguishing specific grain types. Methods are used based on past work by Carter & Yan (2005) for particle sizing, which was shown to perform similarly to laser diffraction methods. A relatively negligible +|- 10% of sizing accuracy difference between these techniques supports this position. Typical sizing is used particularly for grain lengths and is described in later chapters.

1.4.2 Shape

Shape measurement has been long considered and its forms vary from simple single number descriptors to complex transforms. Research led to survey papers (Loncaric 1997; Loncaric 1998; Kindratenko 2003) and books (Costa & Caesar 2000) on the subject, applied to 2D imaging in a wide range of fields. This included grain imaging of which much existing research was found (often using the terms 'grain' and 'particle' ambiguously, applied in different fields such as geology and agriculture). For analysis different single parameters, lengths, perimeters, areas, can be used in functions to describe the shape. Past work has used perimeters for Fourier and wavelet transformation for shape measurement. Such techniques offer other descriptive power than features such as surface area, which alone indicates reactivity and other properties. They can identify significant properties of the shape such as distinct shape boundary features. The investigation of a wider range of shape features than previous work is a significant part of this thesis. The literature review will discuss extensively efforts to classify shape description methods and different useful shape description techniques. A number of these methods have been applied in practice with the images obtained through captured images and software developed in the course of research. This investigation covers methods related to grain shape and the broader field of shape analysis in order to retrieve meaningful characteristics.

1.4.3 Texture

Texture is a long established and recognised quantifiable characteristic derivable from imaging and is used commonly in colour, greyscale, electron microscopy, infrared and near infrared methods. Its use is ongoing as a powerful classifying characteristic in similar static imaging research (Choudhary, Paliwal & Jayas, 2008) in combination with shape, intensity and sizing properties. The texture in this instance refers to the surface of materials in a 2D image, which is not to be confused to the boundary of a shape as it is occasionally described as in similar literature. In the case of images of target objects, the surface area within the boundary is representative of texture, The difference in surface roughness between white and brown grains is one target of textural measurement and a visible characteristic of such grains. The texture of individual objects is also of interest for measuring the properties of each grain. This research is not an attempt to survey texture analysis methods that are appropriate for this task. Such methods would likely be rotation invariant (Zhang & Tan 2002), and taken under consistent illumination.

1.5 Research Programme

This research programme targets design and implementation of appropriate hardware and image processing methods for the analysis of grain materials. This combines some development of physical imaging systems with computer controlled image acquisition and processing. The computer controlled aspect includes the development of different software tools for image acquisition and image processing is also part of this work for the assessment of features from images. Some of this software was readily available at the

University of Kent, and other parts require extensive research and development as part of this research program.

In order to capture images in a straightforward scientific and repeatable manner, appropriate hardware is obtained and assembled for the capture of images. This takes the form of a small imaging setup specially developed in the course of research to take images. Extensive testing is conducted using this apparatus by obtaining images of different rice samples, as well as calibration images. Calibration testing included accounting lens distortion and image scaling. Sample tests included organised collections of images of the different materials in various ways. An adaptation in the top of the box permitted the use of a range of different cameras for the different tests. The ultimate results of this system are top down 2D images of various target grains and calibration grids which are eligible for further processing.

Past work in the instrumentation, control and embedded system lab only considered sizing and limited shape analysis as there was an emphasis on the larger problem of on-line flow monitoring. Most time was allocated to building specialist equipment and conducting industrial trials. Very little static imaging analysis could be carried out aside from some calibrations and performance tests for a small set of features. In the present work the static imaging is expanded considerably in both testing and applied imaging of particulate materials.

The main parameters to be measured will be size, shape, texture, and some unusual features specific to certain types of grains. In the case of rice, this includes the eye curve feature. The literature review will also cover other interesting rice issues that these imaging based features are relevant to, and experimentation will be undertaken to implement as many situations as possible where the right research materials are found.

1.6 Objectives of the Research Programme

Having defined the general scope of the research program several significant objectives were defined within this project:

- To review existing work in image acquisition and processing research related to food grains and any other relevant materials. Also to consider alternatives to imaging for characterisation and identification tasks.
- To research and obtain appropriate equipment for imaging, including appropriate samples of materials, imaging hardware, computing equipment and software.
- To construct setups of appropriate imaging equipment and utilise it for the non-invasive, non-destructive acquisition of images of target particles and materials.
- To develop new and expand upon existing image processing methods to ensure valid measurements of rice grains can be made. New methods will be discussed and contrasted with existing approaches if possible.
- To characterise materials using appropriate feature extraction and analysis to obtain descriptive parameters. Features will be evaluated in terms of their effectiveness and the types of traits they can retrieve.
- To improve the image acquisition, processing and feature analysis developed in previous research with novel development and extended use of other existing techniques.

1.7 Thesis Outline

The elements of work in this research are defined here for each chapter of this thesis:

Chapter 1. An introduction to the general areas of research including imaging of rice, research goals, and some of the image processing of interest with regard to the research program objectives.

Chapter 2. A review of rice analysis particularly relating to hardware and software relevant to this work. This includes alternatives to imaging for the analysis of materials, and an overview of issues and advantages of imaging. Methods to ensure quality imaging are also discussed, and some of the image processing techniques considered are described. The types of material properties considered in imaging are discussed extensively. Previous work in applied imaging and image processing is also discussed to ensure novelties of the research programme.

Chapter 3. Here methodology taken to acquire images is discussed, first though construction of the imaging setup. A range of camera devices are tested in order to find optimal devices for this research. Calibration tests are performed for these devices and experimental results provided in order to give a measure of the image quality expected from the camera, relevant to image processing methods such as threshold functions and distortion correction.

Chapter 4. In this chapter the image processing that can be employed on images recorded with the physical setup is described, with the goal of improving images . This leads to the overall system software architecture and the chosen low-level image processing methods used are discussed in full detail. These methods include a watershed (Vincent & Soille 1991) based segmentation function used in contrast to a concavity approach described in chapter 5. Texture analysis from the processed images is also described with use of a edge detection based feature.

Chapter 5. This chapter discusses post processing of images for the obtaining of the desired features for measuring objects. These include shape, textural features and sizing. This chapter begins with the measurement of these features from the image, and describes a more sophisticated image process that depends on some shape analysis to function. This also includes a final element of further image processing. Such processing materialises as a rule-based concavity segmentation function that itself depends on several shape properties to function.

Chapter 6. This chapter can be broken down into two sections. First, a general overview of shape descriptors, which was initially investigated for further potential in wider fields before the focus upon rice grains. Next are the results of image capture, processing, and feature analysis tests using the methodology discussed in the other chapters are provided. The purpose of this chapter is to discuss how effective efforts have been to identify and characterise different grains, as well as the effectiveness of steps of image processing. Appropriate results and figures inspired by past similar research are used to express findings, which are discussed with some summary conclusions drawn.

Chapter 7. The conclusions are expressed and a discussion of any possible future expansion on rice research, the imaging systems, image processing, and applied experimentation is carried out.

Chapter 2

A Review of Techniques for Characterisation and Measurement of Rice Grains

2.1 Introduction

In the previous chapter the focus of research was directed towards that of measurement of rice grains. Here chapter existing relevant research is discussed in order to expand upon this. Existing imaging techniques that are relevant to the characterisation of rice are extensively discussed. Rice is discussed in some further detail, notably significant properties that may be derived using imaging techniques. Rice processing and distribution is discussed as well as existing companies and devices that perform rice imaging already. Areas where new measurements based on rice characteristics considered in this research could be useful are described. Alternative rice analysis methods will be discussed, then imaging methods including the chosen imaging approach. This review will then lead into methods actually employed for image capture and processing in the next chapter and the further work beyond.

The analysis of rice grains with imaging has been covered partially in past work at the University of Kent. Much relevant literature by other research groups has been found in relation to food grains and in further relevant literature in wider domains. Techniques for the analysis of food grains are summarised, including some of the many other alternative techniques to imaging systems as well as a detailed discussion of applied food imaging techniques. In succession the review of Digital imaging is expanded on including previously developed imaging systems, sensors and processing used in relevant existing research. The properties of images obtained (illumination quality, lens distortion, resolution, impact of any image compression, impact of physical situational effects) are also described as this knowledge is also relevant to research. Finally, the relevant Image processing literature investigated will be discussed. The general overview of key literature is now given.

2.2 Relevant Research in Food

In a broader literature review general food analysis was investigated through survey papers (Sun 2000, Brosnan & Sun 2002, 2004). These were surveys on the specific application of computer vision solutions in a range of food grading problems. These also contained some mention of rice research such as capture systems by Liu (1998) described later in this chapter. In general, the imaging approaches taken in many of the different tasks were highly similar. From the review of literature it became apparent that improvements in available hardware, notably of lower prices yet improving performance, has led to an enormous diversity of consumer food research based on camera systems. This is often included in agricultural and biotech journals such as Biosystems Engineering (Paliwal, Visen et al. 2003). One particularly elementary argument reinforced within the surveys (Brosnan & Sun 2002) stated that the unique way cameras systems can be implemented is much the same as a system that would otherwise be undertaken by a manual worker. The task of grading is a tedious addition to what is already a highly repetitive production line labourer role, and the implementation of vision systems can assist in automation and reliability of grading. This is applicable in many areas of distribution and packaging of different foodstuffs. In addition it offers unique ways to record and analyse food images with the full potential of image processing. Particularly with regard to rice and similar grains, the Survey papers describe several examples of existing research of major relevance, many of which will be discussed in this chapter.

2.3 Rice Grading and Classification

This section lists a number of critical issues and properties of rice relevant to this research, and what type of measurements are desirable to rice and food grain related activity. These include fraud concerns relating to specific grains, classifications and grading of grains, unique features and concerns relating to rice quality that can be visibly identified in rice. Several key organisations who express interest in these properties are referenced in this section. Reports from these organisations were used as critical literature in justifying this research.

Rice is a vital worldwide agricultural product. Total annual production figures are in hundreds of millions of tonnes (http://www.rice-trade.com/) with 40,000 varieties known to exist (http://www.riceassociation.org.uk/). A vast industry spans from the source agriculture to commercial distribution and retail. Rice is a staple food for billions with demand expecting to increase (Juliano 1993) and is a heavily scrutinised subject of bioengineering research.

Uses of rice extend from cooking to brewing of wines such as sake, and the chemical properties of distinct rice varieties are influential on its taste and market role as well as the brewing properties of aforementioned sake (Atkinson 1881). It is sometimes a subject of serious controversy as genetic modification research occurs with goals of increasing rice's versatility as a staple food. Furthermore concerns over rice 'fraud' are addressed later in this chapter.

2.3.1 Rice Processing in Industry

To understand the role imaging could play in industry, understanding of commercial rice process stages is relevant. Several stages are employed in the processing of rice starting with hulled grains from the paddy and ending with brown or milled grains. Grains are found in husks as the seed of the rice plant. There is the panicle containing the husk, containing the grain. This is extensively processed producing brown grains of further milled for white. Milled rice is more commercially desirable as it has a longer shelf life, although it is frequently reported that vitamin, mineral and fibre content of brown rice is noted as of potential benefit to health. Manufacturers of rice analysis machinery include the Buhler group (http://www.buhlergroup.com/), and Kett (http://www.kett.com/) who produce a wide range of grain measurement and sorting devices to a range of stages in the agricultural supply chain. These include the following stages:

- Cleaning of Rice Grains.
- Hulling to remove the grains' protective hull.

- Whitening and Polishing: Outer 'pericap' or Brown rice layer is uniformly removed by abrasive processes leaving only the white grain.

- Grading: Broken and discoloured grains are detected and removed. Grains are sorted into fractions according to length – small/medium/ and large broken grains. Sieving is usually employed although image processing may be applicable.

- Optical sorting: Discoloured grains and chalky kernels are often deemed of poor quality and can be detected and separated by imaging based sorting systems. In larger commercial devices colour and infrared sorting can be achieved at a high efficiency, in the order of several tons per hour. It is this stage at which imaging is particularly relevant and measurements likely to occur.

- Final Blending and Packaging of grain varieties for retail.

Further to grading, Imaging research in the grading of milled rice by Yadav & Jindal (2001) considers the features Head Rice Yield (HRY) and Characteristic Dimension Ratio (CDR) as means to measure milling quality and the percentage of broken kernels of rice. Proportion of grains of specific ratio ranges expressed in percentages of grain samples are a common approach here for benchmarking and rice grading.

2.3.2 Rice Standards

Several measures regarding grading and identification are mentioned under the documentation of the many standards organisations. IRRI and BRC classifications for rice grains are measures for features like shape and surface chalkiness. A great many of these features can be determined from appearance. One example parameter, physical grain length in millimetres, falls under three categories for short, medium and long grains. As mentioned by these organizations and identified in previous work by Carter & Yan (2005) and Sakai (1996). According to IRRI (Graham 2002), the grading of rice depends on several factors: proportion of broken grains, chalkiness, length and shape. These properties are important for several reasons:

- Broken grains denote a poorer grade of rice, and are more prevalent in economy rice or rice delegated for rice flour or sake brewing.
- Chalkiness as mentioned earlier denotes poorer grades of rice.
- Length denotes rice variety – different types of rice are of different size.
- Shape also denotes variety – different type of rice are also of different shape.

In order to illustrate the classifications, Tables 2.1 to 2.3 list the properties of rice varieties for key features (Graham 2002).

Table 2.1 Size Classifications

Scale	Size Category	Length (mm)
1	Very long	>7.5
3	Long	6.61-7.5
5	Medium or Intermediate	5.51-6.60
7	Short	<5.5

Table 2.2 Shape Classifications

Scale	Shape Category	Length/Width	Width/Length
1	Slender	>3.0	<0.333
5	Medium	2.1-3.0	0.33 to 0.476
9	Bold	<=2.0	>=0.5

Standards in IRRI are given as Length/Width, although past utilisation of aspect ratio features (Carter & Yan 2005) use the opposite to determine scale, so the Width/Length is also provided in Table 2.2. Chalkiness in 2.3 represents the reduction in translucency of the grain described earlier.

Table 2.3 Standards for grain chalkiness

Scale	% Area with Chalkiness
0	None
1	Less than 10%
5	10-20%
9	More than 20%

Finally whitening is also defined in the IRRI Knowledge Base (www.knowledgebase.irri.org) as being 'a combination of varietal physical characteristics and the degree of milling. In milling, the whitening and polishing greatly affect the whiteness of the grain. During whitening, the silver skin and the bran layer of the brown rice is removed. Polishing after whitening is carried out to improve the appearance of the white rice. During polishing some of the bran particles stick to the surface of the rice which polishes and gives a shinier appearance'. This forms an intermittent coating of rice powder particles on some grains. This is caused by the milling process and is different from chalkiness which can be indicative of further physical problems.

2.3.3 Physical Rice Properties and Disease Concerns

A number of physical properties distinct to rice can be measured through the appearance of the grain. This section lists several significant examples which emerge in literature. The common recorders of these traits are organisations such as IRRI, which maintains a large knowledge bank of these problems.

This section focuses on the following problems: The detection of the grain eye and embryo, the detection of whitening and chalkiness, and problems with grain disease. Many diseases can be visibly detected and potentially measured, and new problems are continuing to be documented such as 'shrunk grain' rice disease (Zhang, et al 2008). However this was discovered late in research and such analysis is beyond the scope of this thesis. Samples or images of these types of grains could not be obtained. The focus has switched primarily to features that may be influential in characterisation and identification of different types of grains and their significant features.

2.3.3.1 Grain Eye and Embryo Features

Some reference also exists to a presence of the rice grain 'eye' feature. One IRRI report describes (Graham 2002) that 'In some varieties, the grain tends to break more frequently at the "eye" or pit left by the embryo when it is milled. Rice grains with damaged eyes have a poor appearance and low market value'. In addition this report announces a causative correlation between grain chalkiness and breakage in milling. This suggests metrics of grain breakages and the presence of grain embryos or cleanly milled grains with a 'curved eye socket' where the grain has been cleanly milled and the embryo removed are relevant. The detection of these features are of relevance to rice grain quality measures and attempts to detect the presence or absence of rice grain embryos are not widely reported in existing literature. Efforts to describe it from information available from IRRI sources have led to a classification illustrated in Figure 2.4, with an annotated image of rice grains captured with the camera system developed as part of this research. Here can be seen a clear instance of the presence of the grain embryo, its absence after being physically removed, and an unclear case without an embryo that might not be so easily classified based solely on the shape of the grain. The socket like feature commonly occurs in milled rice. While cases exist where shape only may be sufficient to determine embryo absence, texture and intensity properties are also relevant in order to say confidently that the characteristics of the grain are with or without the presence of an embryo.

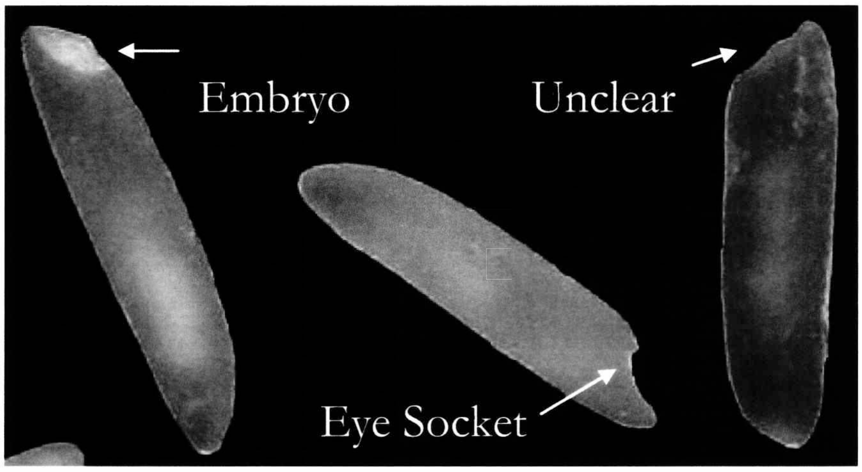


Figure 2.4. Grains with an embryo, eye socket, or a potentially unclear edge.

2.3.3.2 Whitening and chalkiness

Whitening of a grain relates to the polishing process that produces milled grains. This process can also reveal chalkiness in grains. Concerns over 'chalk' rice grains have existed for a long time, and contributing factors to its cause include exposure of paddys to higher temperatures, improper accelerated rice development and some genetic factors

(Braidotti 2007). This is also likely due to rice grown in a different location, such as a different region or country. Rice that does not properly mature in time has an endosperm that is less translucent. This rice has a cloudy 'chalky' quality that is distinctly visible. This rice is lower quality due to production under stressed circumstances, limiting its value as food. Research is ongoing to selectively breed rice that can survive higher temperatures with less chalk occurring. Imaging solutions can assist in detection. In imaging, this can be derived approximately as the percentage of visible surface area that constitute as a visibly chalky region. The higher the percentage area of chalkiness within the endosperm, the high the resultant scale of chalkiness in that grain. Proportions of Chalkiness are also provided by IRRI and are given in Table 2.3, with an example of a translucent and chalky grain given in Figure 2.5. Standards given by IRRI are illustrated in an example from the IRRI website in Figure 2.6. Proportion of chalkiness by area is a potential method to determine the percentage of chalkiness, however training sets need to be properly determined in order to more precisely quantify levels of chalkiness.

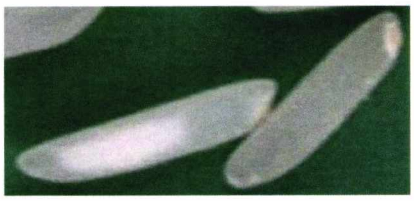


Figure 2.5. Partially chalky (left) and translucent grain (right).

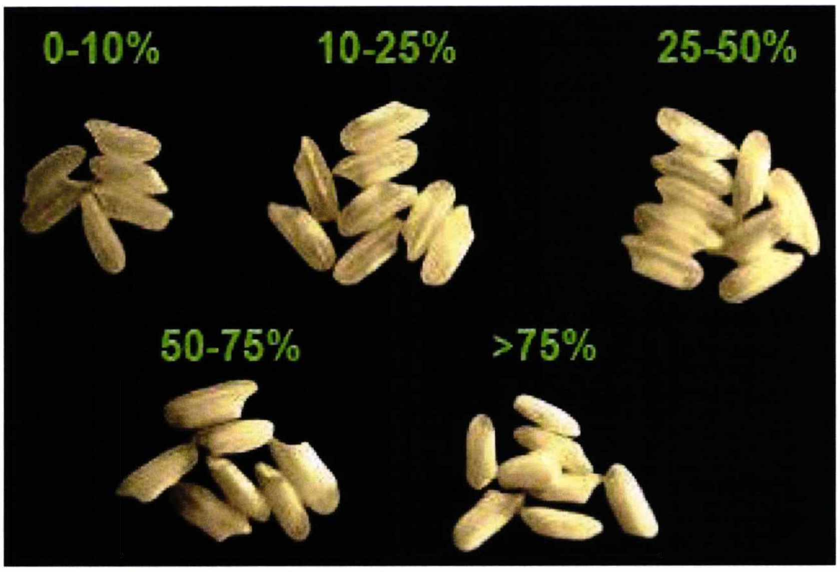


Figure 2.6. IRRI example grains by percentage chalkiness. (www.knowledgebank.irri.org)

2.3.4 Authentication and Adulteration of Rice

There has been increasing concern reported among popular media and investigated by international food agencies regarding 'food fraud'. While, in the case of rice there have not been cases as dangerous or high profile as the widely reported melamine scare in foods, the passing off of a cheaper variety of rice as one more specialist is prevalent and is reported in the media, as referenced extensively in chapter 1. Rice is of some cultural significance in parts of the world and in future certain rice growing regions may change with the climate. Tracking and identifying rice types is critical for all commercial rice in relation to these issues. This is particularly common with higher price aromatic rice varieties, which are often partially supplemented or even entirely replaced with a separate cheaper grain. This is particularly true of Basmati aromatic rice varieties.

With regard to the concerns of Basmati Rice fraud a range of agencies, including the International Rice Research Institute (www.irri.org), The UK Food Standards Agency and Retail (http://www.food.gov.uk/)have issued standards of what constitutes a specific rice variety for this purpose. The code of practice for Basmati Rice devised by the UK food standards agency include the following requirements:

- Milled Grain length/breadth ratio >3.5 (or < 0.285 breadth/length)

- Minimum average pre-cooked length of 6.5mm

- Less than 10% 'broken' grains by volume. Broken denotes ¼ of the grain missing. Samples with 10-20% broken grains denote an economy brand. Samples with >20% broken denote a 'Basmati with broken grains' label requirement indicative of this fact in the brand name.

- 97% of a specific variety of grain should be classifiable as that variety of Basmati Rice if a specific variety is named. This also applies to labelling of country of origin (97% of grains should come from any country mentioned in brand).

- Products described as 'Basmati Rice' should not have any more than 7% other grain types in their volume.

Ongoing investigations by the UK Food Standards agency have revealed that most supermarket rices at least partially fail to meet these standards. The issue of 97% of grains being a specific variety and 7% being another likely relate to genetic factors that judge the rice, so measures of particular interest are the <10% brokens, as that rice type, length and aspect ratio properties. Efforts have been made to return results to contrast with these standards in this thesis.

2.4 Existing Techniques for Rice Measurement and Characterisation

In this section common measurement techniques will be addressed for the specific analysis of food grains, particularly rice. There will be some mention of relevant techniques from other food imaging, and wider image analysis and related fields. The following subsections describe methods for measurement of rice grains, first general methodologies for characterising and identifying specific types of rice. The focus will however be mainly on the imaging approaches particularly as to why it is optimal for the work in this thesis in comparison to other methods. This list is not exhaustative, and a number of other measures exist. Methods chosen to be described here were done so as they particularly represented the range of powerful analysis methods that are commonly employed in examining rice. Many of these methods are often more complex and expensive than imaging, although this is not the rule. One simple example is physically sieving grains, which is frequently mentioned as a highly effective means to isolate grains of specific size ranges and separate broken fragments from whole grains. However this method can be dismissed here as limited in scope and more sophisticated forms of evaluation return greater ranges of characteristics.

2.4.1 DNA Analysis

Polymerase Chain Reaction (PCR) is used for the analysis of DNA from a sample rice graincontaining the embryo of the rice plant. This process requires the destruction of the sample in a laboratory environment using a complex process. While this process is highly accurate when used, it involves a procedure of 20-40 repeated cycles of temperature variation steps. These individual steps can last several minutes, resulting in the process time being exceptionally long per sample. The process requires expertise and expense even to this day and is widely used in biological sciences for rice studies (Woolfe and Primrose 2004). However the genetic basis for many rice grain varieties exists and genetic factors have been tied into shape and sizing properties for several distinct rice varieties. Work that combines genetic knowledge with physical properties has been investigated (Tan, Xing, Li, Yu, Xu and Zhang 2000). These include image based properties such as length, aspect ratio and chalkiness. These suggest that DNA analysis results can be combined with measured features derived from imaging or other approaches to goals relevant to grain characteristics.

2.4.2 Chemical Analysis

The electronic nose is a chemical gas sensor for detecting qualities distinct in odour. Given the aromatic nature of Basmati rice a considerable variation in chemical properties is detectable from conventional long grain rice varieties. An example commercial sensor capable of fulfilling this role would be the ARTINOS multi-scan 510 portable unit (http://www.specs.com/) shown in Figure 2.8. A sample vial containing an air sample from processed grains is connected by tube to the sensor in this example. Existing research has established 90% reliability in the identification of discernable 'waxy' and 'non-waxy' rice varieties (from significantly different countries, and representing rices of different ethnic regions). Separately Graham (2002) acknowledges the use of Gas Chromatograph mass-spectrometer for recording the aroma from Basmati and Jasmine rice varieties returning different measurements reflective of their different constructing compounds. However the precise nature of the chemical sensing process is also destructive.

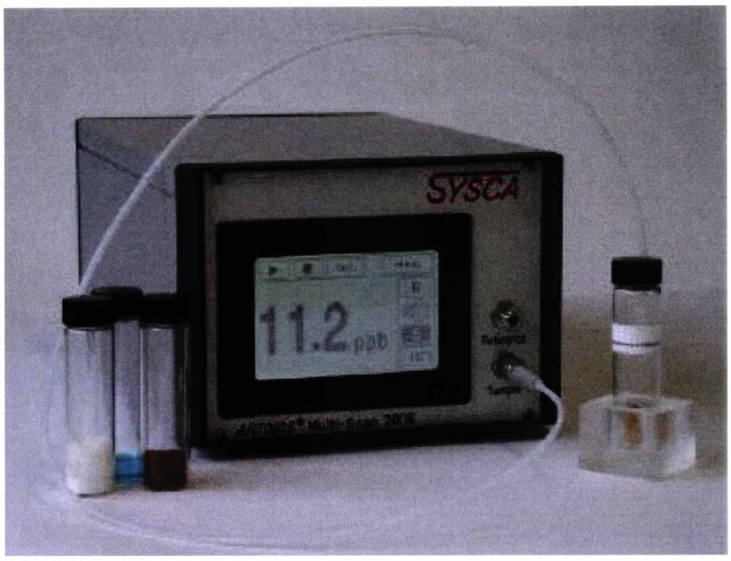


Figure 2.8. Electronic Nose System.

While the process of testing gas emission from rice is less complicated than PCR, it requires the use of and preparation of samples for the electronic nose sensor. While the sensor's utilisation is more straightforward and generally quicker than that of a PCR test, the sample's denature/destruction through essentially 'cooking' is required to release the chemical odour. Past use of Electronic Nose for food grain analysis (Magan and Evans 2000) involved heating of samples to 50°C for several test cycles lasting several minutes, in order for air released from the sample to be fed through a sensing 'headspace' with pumped purified air. While in that example, the finding were of relevance to identifying distinct compounds to denote infection progress in spoiled grains, the approach is ultimately denaturing of the sample and is a different type of application from non-invasive, non-destructive ones considered here. It is also more specialist than imaging and consequently of higher minimum expense.

2.4.3 Spectroscopic methods

Several more sophisticated imaging systems have also been employed in the measurement of rice and food in the past (Theanjumpol et al 2005). For the imaging of near-infrared of rice grain samples, spectrophotometers have been employed for a series of scans of rice grain samples. This non-destructive method can quantify a range of different organic molecules according to the vibration absorption profile resultant from the spectrometer.

This is essentially a single wide scan of a grain sample and is indicative of its composition. In this example, work was carried out on authentication of aromatic Basmati rice using NIR. Acquisition of spectra required sample preparation in a spinning module. Multiple sample runs were obtained and the average spectra determined for each sample type. The data was also subject to further analysis including PCA in order to identify distinctions between rice varieties. In general this alone was enough to produce a clear distinction between several specific Thai rice varieties, yet calibration was needed for reliable identification. The NIR system still requires a specialist setup and an understanding of the interpretation of an NIR scan for this task.

The use of NIR analysis was separately considered (Delwiche & Webb 1996) for deriving quality characteristics. These included amylose content, protein, viscosity and appearance of grains. A range of short, medium and long grain classes were subject to scanning. A number of the characteristics were separately measured with high reliability with benefit for rice breeding selection purposes for farmers. However this process involved the analysis of slurry of ground rice turned into a rice flour paste, and is destructive of the sample.

The critical advantage of many of these methods over the DNA/Chemical approaches is they are not destructive and invasive of the sample grains. Spectroscopic methods, such as x-rays can be employed passively on a rice sample (Hideshi et al 2005) and with some useful measurements in terms of grain composition. Although X-ray generators are not practical in everyday use near people as regular operators require protection, the entirely passive nature of imaging permits application to rice that is to enter commercial use without loss. However, some imaging systems are expensive or impractical, and system limitations must be addressed.

2.4.4 Imaging Methods

The usefulness of image analysis of rice grains has been investigated in a range of different capacities. Several different forms of imaging apply to the analysis of rice, using different types of imaging sensors and assessing a wide range of properties. Each of these will be now evaluated in turn. By imaging, the analysis of rice involves the use of a scanning sensor which sweeps an area containing rice grains, or an area of the surface of rice grains themselves. Some sensors can penetrate the surface of the grain revealing detailed properties of the grain interior. All of these methods require a sensor capable of recording the incident light/waves from the imaging target, be it visible, infrared or other frequencies.

The effectiveness of conventional digital imaging for rice grains and other varieties has been quantified in the past in many cases (Liu et al 1998, Yadav and Jindal 2001, Senkai 1996). The focus here will be on CCD and CMOS scanning systems. Visible light imaging is the primary focus of this research, although the potential benefits of imaging of other spectrums are worthy of notice. Special cases also occur where valid alternatives to using just visible light also exist. One method of using infrared is illustrated In Figure 2.4. Here an example use of ultraviolet detection with a camera system for an imaging solution to detecting a breakage line in some types of grain is given (www.knowledgebank.irri.org). Imaging a grain under ultraviolet light can reveal stress fractures on the interior of the grain. This method reveals locations where brakes in the grains are likely to occur. This suggests some application for special illumination in some cases of otherwise conventional image capture scenarios.

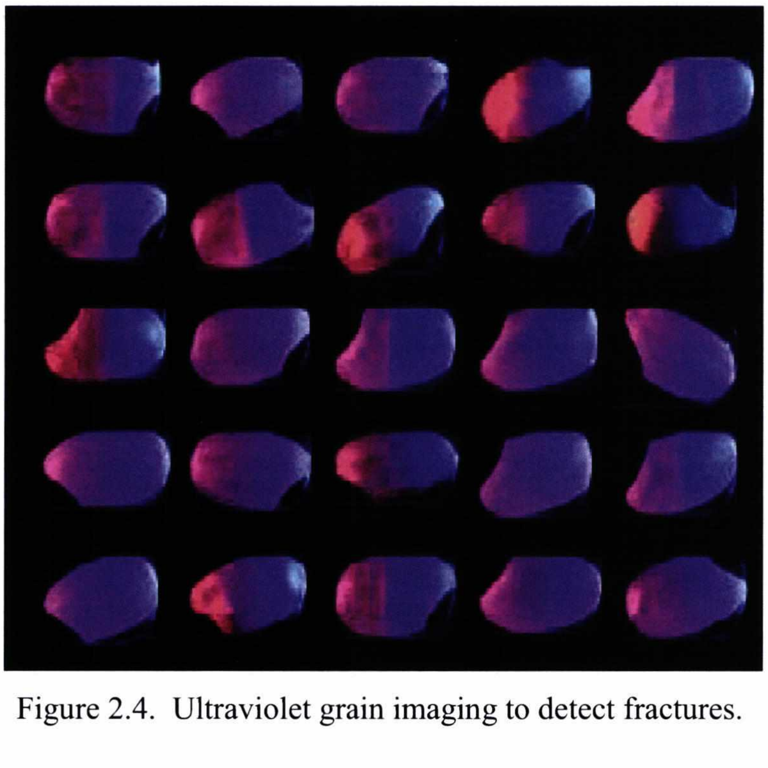


Figure 2.4. Ultraviolet grain imaging to detect fractures.

The imaging of grains in the visible spectrum can be carried out on a range of scales, typically through the use of optics, Charged-couple devices, and under controlled illumination conditions. This type of imaging exploits a number of physical effects that occur as result of illumination. This essentially is the application of optical theory. Waves of light from a source will be selectively absorbed, refracted and reflected by different materials when photons interact with them. This results in the two common forms of visible imaging, front-lit and back-lit imaging.

Front lit imaging has the image target in the foreground against some contrasting background. This results in the order of the imaging setup being camera -> lighting -> target -> background. Incident light from the camera bounces off the surface of the imaging target and background, resulting in detailed surface information being retrievable from the imaging scene. This requires a contrast between the target and the background in order to clearly distinguish the imaging target. Such systems have been used in the past (Carter and Yan 2005) by many others (Visen and Paliwal 2001, Yadav and Jindal 2001) and one case is illustrated by Figure 2.9. Placing the target on a contrasting background and employing diffuse uniform lighting reduces shadows and eases image processing. Although in practice, a ring light approach like in Figure 2.9 may not be employed in the exact same manner for a conveyor system.

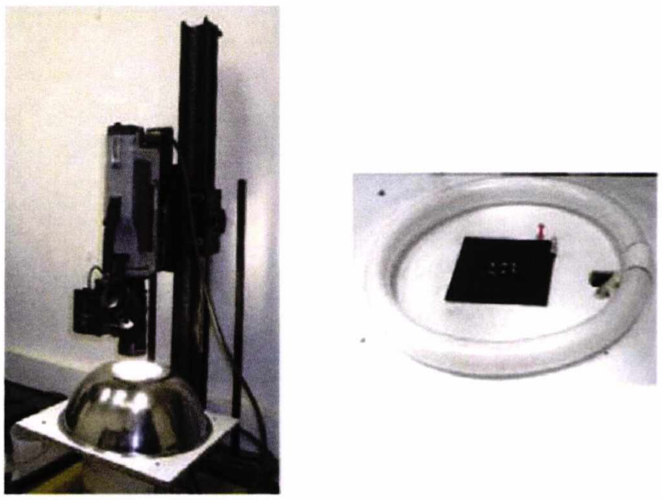


Figure 2.9. Visen's image acquisition system (left) with focus on system background and illumination ring (right).

Backlighting has illumination behind the target, and consequently no surface detail aside from the physical boundaries and regions of targets can be seen. Dependent on imaging conditions, this method can return highly visible 'silhouettes' that can be effectively processed returning boundary information, yet return no surface information of the target. In summary, front-lit imaging records the incident light from the object, and back-lit imaging absorbs the absence of the incident light. Figure 2.10 illustrates a back-lit imaging system for measuring scallops on a conveyor system (Tomanovich 2006). This is one of many examples of a 2D-binary image system on the Vision Systems industry specialist website (http://www.vision-systems.com).

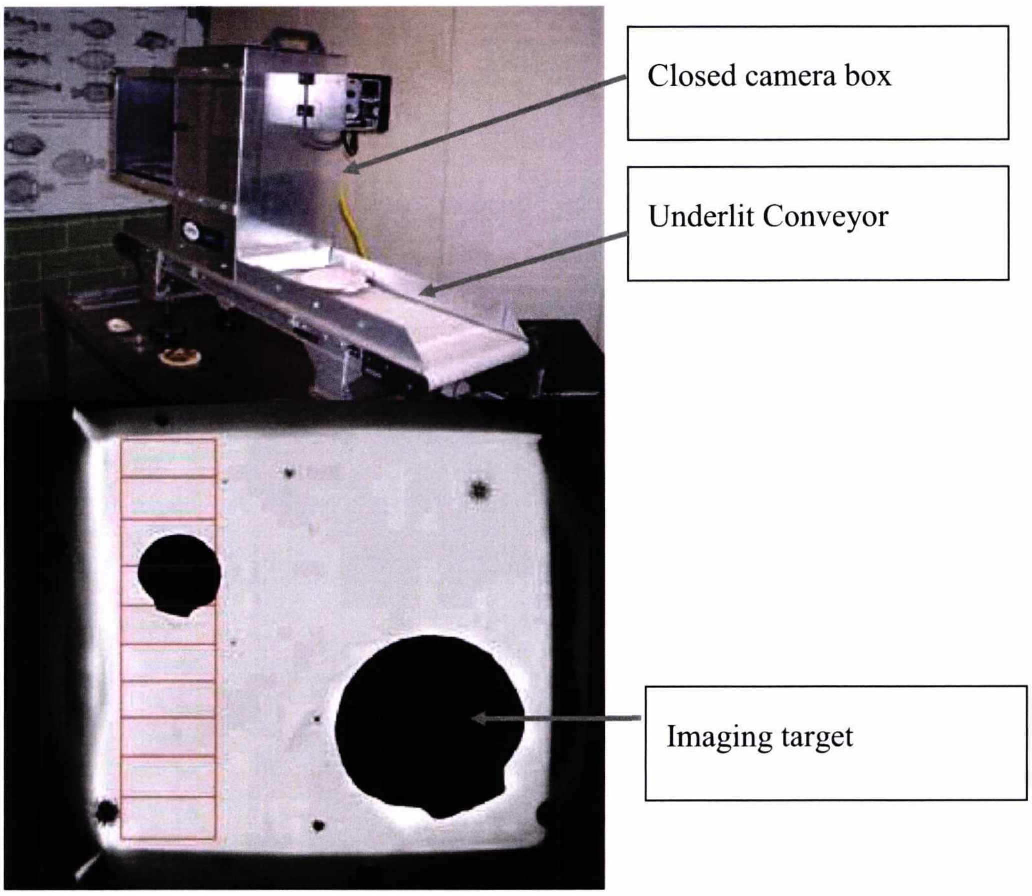


Figure 2.10. Conveyor imaging system with example captured image.

Each approach requires slightly different imaging hardware. The lighting system and physical background varies accordingly. There is more variety with the hardware involved in different front lit systems than back-lit. However back-lit is also made more complicated by the need to pass a light through a suitable medium, such as glass. This can be an impractical choice for many reasons, particularly with regard to the presence of debris in an image interfering with measurement. An advantage many front lit systems possess is that under certain circumstances some objects can become very difficult to distinguish from the background, meaning only specific objects can be seen. Small noise and debris bits can be difficult to pick out against a coloured background or one of similar intensity, where larger objects with more mass will cast shadows or demonstrate visible elevation.

Either imaging approach utilises an imaging sensor and a lens. Dependent on the choice of lenses and sensors, different measurements are possible. Variable magnification and macro/micro photography is one example. However, varying sensors have different optical measurement properties such as monochrome or colour imaging, as well as different imaging resolutions. Colour imaging can return a larger volume of image information, although illumination is critical to the effectiveness of the imaging system dependent on receiving incident light from the target. Alternatively, mono cameras can be used under different coloured illumination to record intensity under different colour lights, then use that information to represent that colour spectrum. This method also has the advantage of permitting the use of a low-cost greyscale sensor. Taking several images under the different lighting provides three or four intensity images for red/green/blue and an ultraviolet channel, an approach employed in scientific imaging and even on spacecraft where low power imaging is essential (Soon Sam Kim 2006).

A number of valid imaging approaches can be taken for this research. These primarily fall under static or conveyed imaging systems. The simplest to construct is the Static imaging setup, yet a conveyed system is more commercially applicable and capable of rapidly assessing a much larger sample. Static imaging provides an excellent testing ground for functions applicable to conveyed systems and comparisons can be drawn between the two systems. This is particularly so for relatively low flow and feed rates of some belt conveyed systems. For static imaging, a range of solutions exist in capturing images. A camera can be pointed at the imaging target, or the target can be conveyed past the sensor. Precise means of conveyance vary from a physical conveyor system illustrated earlier in figure 2.10, to gravity fed systems employed in the Sortex Z+ (Group 2007). The results of this system are situation specific, but it is clear from the example image that relevant information could be extracted for a specific set of problems.

Where as the conveyor system illustrates a typical backlighting approach, a front-lit system is adopted by many researchers also interested in surface properties of grains as well as shape. This was adopted in past research (Carter and Yan 2005, Visen 2001). These examples use static setups where grains are imaged against a contrasting background, usually a dark/matte background in contrast to the intense illuminated grains. As well as shape this approach has the advantage as illustrated in similar past research of permitting analysis of surface textures of many different materials. This imaging approach uses a top down lighting approach which is physically simpler than the back lit system. Dependent on the material front and back lit imaging may or may not be appropriate. In the case of rice, back lighting has the potential to retrieve shape and size much as the system employed in Figure 2.10.

Elsewhere another system devised in the past (Liu 1998) for express measurement of individual grains is illustrated in Figure 2.12. This roller-driven system permits digital image acquisition of the profile of individual grains. This design is of relevance to this area of work and suggests individual grain measurements are also valid. In addition, Liu directly compared imaging results with a conventional chemical analysis approach normally used for this type of evaluation. A reassuring correlation coefficient of 0.9819 was recorded in this comparison. However this roller approach likely has some practical drawbacks due to its complexity, has slow speed of measurement compared to commercial grain sorting devices, and problems handling grains of variable size.

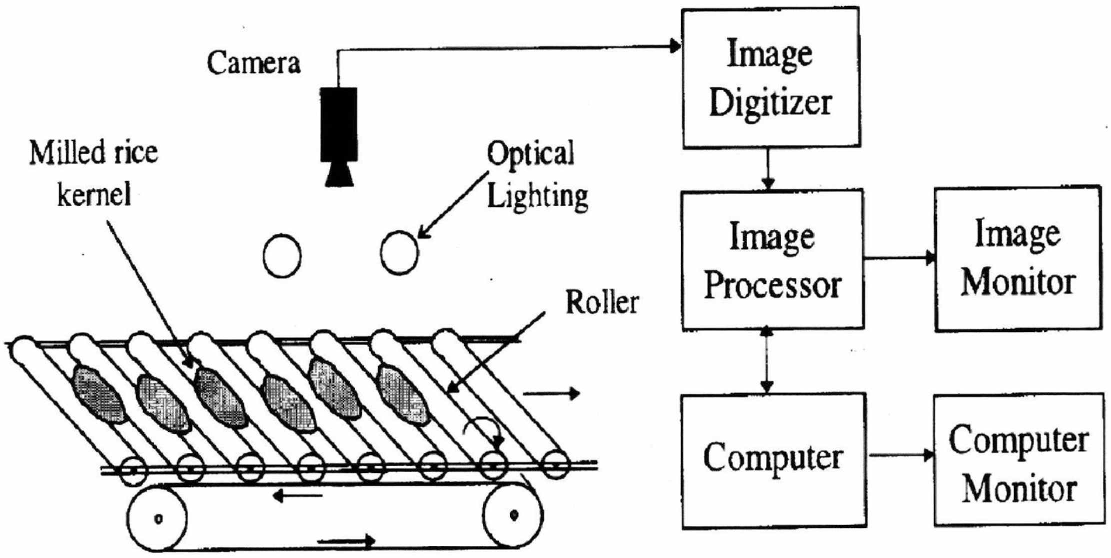


Figure 2.12. Rice roller conveyor for gauging milling quality (Liu et al 1998).

Static imaging is still important for the evaluation of food grains, and much work still uses a less physically complex static imaging setup. Paliwal (2003) used static imaging setups for evaluation of non-contacting grains of different types of cereals, while varied in shape and size, possess extremely similar in rheological properties to rice grains and can be subject to similar processing and transport.(Paliwal and Visen 2003) Static imaging was employed with a difuse illumination chamber with a good standard of illumination provided by a specialist ring light, an approach used elsewhere by Carter (2005) and Visen (2004). In Visen's setup, A colour CCD (DXC-3000A Sony) camera saved three 640×480 tiff images for red green and blue signals with a lens with a focal length of range 10-120mm. This was in conjunction with a direct computer control from a PC with a video card and specialist software.

In addition to the camera systems, there are other possible imaging devices. For the purpose of research, flatbed scanners are a cost effective method of retrieving high quality static images without significant noise or distortion. A colour flatbed scanner can provide a high quality image of a static sample, and has been considered for use in past research (Graham 2002). While the development of a dedicated flatbed imaging system for scanning conveyed and moving samples is more involved than the development of a similar camera setup, commercial flatbed scanners are adequate for static imaging tasks without modification. However, problems arise with their limited physical versatility. The samples must be positioned on the rice scanner, and constructing a specialist scanner is more difficult than constructing a camera system, that can be just as easily adapted for many types of moving scene as well as a stationary one. This limits the potential application in cases as illustrated for imaging in figures 2.11 and 2.12.

Grain imaging research is still an active field with several research groups focussing almost exclusively in it. Overlapping research fields include particle imaging, ore imaging, segmentation research, food imaging and comparative studies with other methods such as DNA and so on. The use of digital images of grains can reveal a great deal of measurable properties for individual grains and some descriptive information of a complete sample. This in turn relies on a number of image processing techniques which can and have been developed and used not only in this branch of research but in other imaging fields. The processing and analysis of images are covered in the following sections.

2.5 Image Processing

The processing of an image is a stage to transform the acquired image to a state from which valid measurements can be derived. Many of the measurements derived can in turn be fed back into a system for further image processing. Figure 3.3 illustrates the stages proposed by Sun (2000) for an image processing system that reuses the knowledge derived by the system to improve every stage of the task. Figure 3.3 begins with the capture of an image from the problem domain, then subsequent image processes are applied based on the properties defined in the knowledge base. This can be a supervised or unsupervised approach, either with defined parameters or some degree of automation by the system. Processes can be managed by a separate computer system or incorporated into an embedded device. The ultimate goal of image processing is to return an image to a state where useful properties can be returned.

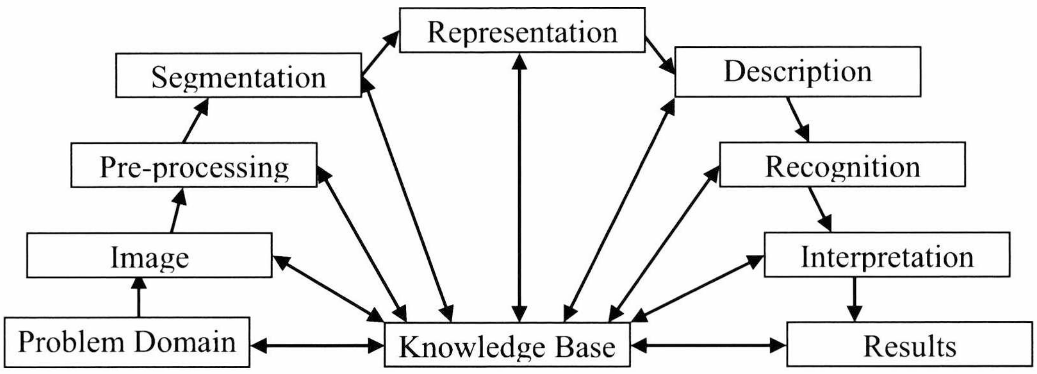
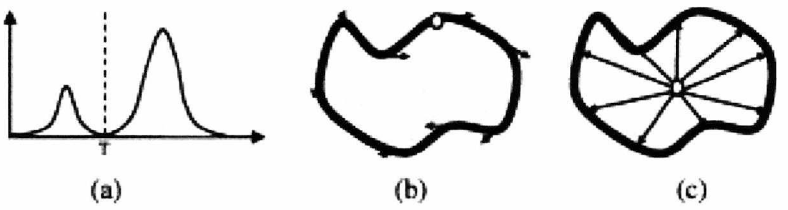


Figure 3.3. Stages of a complete Image Process incorporating segmentation. (Sun 2000)

Literature in image processing and computer vision is extensive with an enormous number of journals and books. Of specific interest here are relevant techniques to transform the image into a useful state and eliminate measurement problems that can occur within the imaging. Work by Wang (Wang 2006) and other relevant methods (Wang 1998, et al 2006, Carvalho et al 2007) detects and segments contacting objects that may occur during image in real world scenarios. This led to an extensive investigation into different types of segmentation function for the separation of physically touching grains within an image. A range of elaborate methods exist, such as the employment of Hough Transforms (Zhang et al 2005), Watershed Methods (Vincent and Soille 1991), and Concavity based approaches.

The most common approaches are the segmentation of an image, as defined by Sun (2000) and illustrated in Figure 3.4. Of these, thresholding of the image is an extensively studied (Mehmet and Bulent 2004) and extremely common form of segmentation between an irrelevant removable set of data and the targets of interest. Figure 3.5 illustrates the result of an adaptive threshold function (Carter and Yan 2005) based on image intensity, however this case is a flawed example where a mistake has occurred. Of the two grains in the image, only the one on the right can be argued to be effectively segmented by the image threshold. Poor contrast and dynamic range of illumination can result in more difficult image measurements of individual objects. Please note also that the square pattern in the binary image was result of a morphological operation in addition to the threshold. This grain can be considered to be poorly thresholded, and the other grain properly thresholded. In addition, a loss of sharpness also becomes visible towards the image edge, likely partially due to the impact of lens distortion and being outside of a region of uniform illumination. In practice with low-cost systems these problems are likely to occur and need to be carefully managed. Work by Carter and Yan has addressed some issues with lens distortion and produced an effective adaptive threshold function for work in particle and material imaging under controlled illumination conditions. Thresholds are also abundant in image processing research, notably the Otsu threshold (Otsu, 1972) descriptions of which can be found in popular books (Gonzales and Woods 2002). Careful application of these techniques has been an important part of this research.



(a) Threshold. (b) Boundary. (c) Regional.

Figure 3.4. Segmentation approaches (Sun 2000).

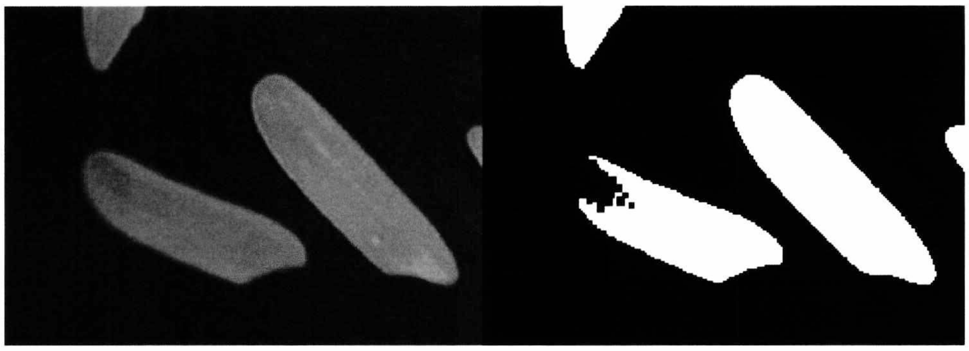


Figure 3.5. Grain image and result of applied segmentation and morphological operations.

Other forms of Segmentation are different from the image threshold function, and are useful in different ways, or can be used in conjunction with a threshold as part of an extended image process. In figure 3.4 is shown the boundary and region based segmentation strategies. Boundary based segmentation requires the boundary information of different sets of objects. Examples are concavity based segmentation functions (Wang 1998). Region based segmentation involves the entirety of a set representing a specific region. 'Watershed' Functions are an example of this (Gonzales and Woods 2002), and region based methods have been employed for segmentation of contacting grain kernels in the past (Paliwal and Visen 2001). Simultaneous to this research, a more recent study (Levner and Zhang 2007) resulted in a sophisticated watershed approach using markers to resolve this issue. This appeared to improved on many of the issues with Paliwal and Visen's research, although this was only discovered later in the course of research, and was not extensively investigated due to its utilisation in very complex scenes of multiple overlapping objects.

Many other segmentation functions are valid for this task and have undergone many iterations of research. Several are described now. The Mumford-Shah method (1989) is a level set method (Lie, Lysaker and Tai 2006), which can be interpreted as a more elaborate take on the adaptive threshold function implemented by carter. A more recent example of this is the use of a level set analysis for the detection of white blood cells or leukocytes (Mukherjee and Ray 2004). This process involves the application of multi level thresholding and complex analysis of the resultant contours at every intensity level, to determine the most likely contour shape to segment significant regions. This powerful process is complex and is a poor choice for a system that should be as simple as possible. In this case environmental condition control can simplify the situation. Such segmentation functions are becoming increasingly able to differentiate noise in the interior of objects and fill in gaps. However Carter's method has previously proved adequate under controlled conditions, and a number of changes to the imaging environment and the employment of simple morphological operations can be implemented to improve performance. In addition, this was still a learning process for the student and many potentially valid methods were not found until late in the process of research. Methods such as Carter's and Otsu's would probably be classed as 'standalone' methods by Mezarsis (V. Mezaris 2003) as opposed to 'relative' methods that would be based on some provided 'truth' such as a reference 'ground truth' image. Methods such as the level set analysis require a ground truth image, which is not even an option for consideration for a system where a constant supply of unknown images may be used, even if this source is extensively controlled. Ground truth images, for example as demonstrated by an example image database maintained on the University of Florida Department of Computer Science and Engineering's website (http://marathon.csee.usf.edu/range/seg-comp/SegComp.html) did however indicate that such comparisons are useful for testing and development of algorithms.

In the course of research many resources were regularly investigated to find existing solutions supportive for the prototyping of image analysis and image processing solutions. Several websites were frequently examined for useful code and content, including the codeproject website (www.codeproject.com), the open source Cimg library project (cimg.sourceforge.net), and the Matlab file exchange (www.mathworks.com). All are sources of freely available solutions including image acquisition and processing code. These open sources distributions were generally encoded in C, C#, Java and Matlab languages. Understanding of these languages and environments was also important and a wide range of popular textbooks and websites were used for general reference. This led to a range of different shape description types even from only a two-dimensional binary image of a shape boundary. Through dialog with others in the Embedded Systems lab and other research groups in the department of electronics, Texture also became of interest. This emerged particularly when reading of other image processing research involving static imaging setups in a diverse range of subjects (Pons and Plagineux 2005, Pydipati 2006) as well as past publications in particle texture and shape analysis (Xu, Luxmoore et al. 1997, 1998) and led to experimentation in applied image processing during research.

2.6 Sizing, Shape and texture features from processed images

Expanding on work by Carter and Yan (2005), shape was considered as a means to describe imaging objects. A focus was made on survey literature on the subject of shape representation (Loncaric 1997, 1998, Kindratenko 2003, Zhang and Lu 2004) in order to provide a summary of the subject. Further to this reliable approaches for shape feature extraction from binary images are listed by several authored textbooks (Davies 1990, Whelan and Molloy 2001, Forsyth and Ponce 2003, Pratt and Adams Jr 2007) and a range of existing coding strategies could be employed to develop new algorithms and complete analysis solutions.

Past research mentioned earlier has shown that use of shape features can quantify and distinguish rice varieties using pure samples of whole grains, however more special cases exist in this regard which are worthy of further investigation. In practice the image processing required to return these features may need to be a quick process that could be adapted for work on-line or in real time. Techniques developed in this line of research also have the benefit of being broadly applicable in other fields of imaging research (Carter and Yan 2005). Likewise, while Carter's use of a Shape factor feature was novel in coal and rice imaging, past Research by many others (Metzler 1999, Schäefer 2002, Senkai 1996) suggest that other shape features such as the compactness ratio and similar simple measures are also useful measures of irregularities in shape as well as the measurement of fuzziness and digitisation error that can occur and impact shape measurement (Sladoje Nyström and Punam 2005, Ziedan et al 2006).

Particle sizing research (Carter and Yan 2005) through imaging applies directly to the imaging of food grains, as similar methods can be applied and extended. Per pixel area can be determined through methods normally used for calculating lens distortion; consequently this can be transformed to represent real object size with accuracy comparable to laser diffraction mechanisms. However this is extremely variable dependent on imaging conditions. A poor contrast within the image environment will result in a weaker dynamic range of values and poorer image processing. The inevitable knock-on effect is considerably weaker image thresholding. Weak and poorly defined thresholds lead to less accurate representations of target objects. Gradient based thresholds are vulnerable to this, although a strong contrast and stable imaging environment has been shown to mitigate problems. Further attempts to reduce this were taken in the course of research, in the hope of returning a good standard of sizing and shape measurement and reduce image processing problems as illustrated in figure 3.5.

2.7 Analysis of features from processed images

Other research groups investigating food grain analysis using imaging take a statistical analysis approach evaluating a large number of different features, using classification models to determine which features or sets of features give the greatest descriptive power (Choudhary, Paliwal and Jayas, 2008). These have suggested features such as wavelets are useful in conjunction with other shape features, with some support from fourier, intensity, texture, and simple shape or morphological features. Such a combination of features has not been considered in past work by Carter and Yan, and such methodologies should have some bearing on the type of characterising features that can be drawn on rice. Previous work can be extended by successfully considering surface texture descriptions as well as other more specific and elaborate shape-descriptive measures, particularly relating to cases unique to rice grains.

Experienced grain imaging researchers such as Paliwal (2003) and other publishers to the Journal of Biosystems Engineering (Paliwal and Visen 2004) and others working on geological particle imaging research (Carter and Yan 2005) have a range of ways to analyse features. Paliwal extracts large numbers of features from grain images, and have built an increasing body of research through the investigation of feature after feature, and analyses the features with statistical software packages which rank the features by 'usefulness'. Visen tested a grain feature classification system based on shape and textural features from different cereal grains. This approach tested different neural network architectures to find reliable classifiers for the different grain varieties. Carter employed unsupervised clustering based on mean feature values for the different grain samples, highlighting the respective difference and similarities between varieties. Common imaging approaches often incorporate several types of feature for characterising particles, foodstuffs, grains and other similar types of objects (regardless of actual scale, objects in images in these different situations may appear similar and techniques are transferable).

These include simple morphological features such as simple shape descriptors. Another is intensity and textural properties of the shape region. Yet more research has incorporated more elaborate and newer methods such as Fourier descriptors and wavelet transformations. Investigations have also considered colour imaging and a range of image transformations to different colour spaces, such as Hue/Saturation/Intensity, RGB (intensity per colour) and LAB (light intensity and dimensions a-b) (Choudary 2008). In addition, the features themselves are assessed with statistical classifier methods. The use of statistical packages such as SAS permit the derivation of lists of 'top features'. These rankings identify the individual features which offer the most descriptive power in a given classification task. In the case of grains, this is the classification and differentiation of different grain varieties, broken and intact grains, and so on.

In this thesis, it was also important to find ways to express recorded feature results. Graphical and statistical approaches were employed previously (Carter and Yan 2005). This is reinforced by simultaneous research by others, such as imaging geological granular materials by Cox (2008) using similar means to express results.

Textural features are extensively covered like shape in literature with surveys (Zhang 2001, Zhang and Tan 2002) and book chapters such as given in The Handbook of Pattern Recognition (Tuceryan and Jain 1998), Digital image processing by Gonzalez, Woods and Eddins (2004) and many others. Although it is important to note the impact of illumination on texture. To this end the direction of this research has moved to imaging of materials from a top-down setup employing diffuse illumination. What is absolutely critical in texture analysis is the realisation of the consistency of illumination. Highly variable illumination conditions in terms of intensity and illumination direction will have serious impact on texture. However, this may be advantageous in some roles. Chanter (1995) discusses this in particular with reference to classical image processing test image sets used in research, such as (Brodatz 1966, Chantler 1995) (commonly used in image processing and freely available on various internet sites), indicating illumination direction effects the validity of measurement from these types of test images. However in this research it is critical that illumination direction is accounted in texture, as it will not be constant under different conditions. Diffuse top-down illumination needs to be correctly achieved for appropriate hardware setups.

A concluding point follows regarding existing research observed during the literature review. It is common for only part of the problem is addressed in singular publications. Some will focus more on the critical image processing issues (Kumar et al 2006, Wang 2006), others only on the quantification of features (Cox and Budhu 2008). Often this literature will omit critical issues. In the case of Cox's work, segmentation can be seen to be applied but no mention of the segmentation functionality is provided. While it is effective to focus on each of these aspects separately, it is absolutely critical in this research to ensure the image processing and shape/texture analysis functions integrate correctly to produce a working system capable of returning valid characteristics of individual grains.

2.8 Summary

This chapter began by explaining past research into the evaluation of food grains, and now concludes by reiterating major points. There are many existing methods available for rice identification, including DNA testing or 'electronic nose' analysis. Though reliable these methods are expensive, tedious, laborious, and destructive of the sample material. Digital image analysis offers a low-cost, efficient, non-invasive and non-destructive alternative solution. With the advent of imaging hardware and computing software, digital image analysis is becoming increasingly attractive to system developers and many different industrial end-users involved in the quantitative characterization of granular food products. A number of existing situations have been explained where imaging derived measurements are useful in a range of systems. It has been established that a number of methods are already being implemented in this capacity and expansion for a number of novel applications is worthwhile. The shortcomings of several past imaging systems and the methods employed in the past have been shown, and room for further research exists. Several of these areas will be elaborated upon in this thesis with proven examples of complete study. In the following chapters the use of imaging in static setups and the comparison with flowing systems will be investigated, as well as methods of image processing to assist analysis.

Chapter 3

Rice Image Acquisition Hardware

3.1 Introduction

Controllable conditions are essential in order to achieve repeatable and reliable environments of image acquisition of target grains. In this chapter relevant methods for capture of rice images obtained are discussed. A range of factors influential the quality of the images. are addressed. The chapter starts with an overview of the imaging setup that has been developed in this research, following a discussion of the hardware used with this system. Descriptions of camera calibration tests employed to test system accuracy and gain a standard for measurement from the images are given. Lens distortion is detected with digitally added lines compared to calibration images. In addition, illumination quality is tested through detection of a reasonable level of uniformity to ensure reliability of imaging. All this is proven necessary as the captured images must be of sufficient quality that image processing can be successfully applied in order to extract characterisation and identification properties of grains.

3.2 Image Acquisition Environment

All imaging was performed under enclosed conditions under controlled illumination. A range of different cameras were considered for research under the same conditions in order to consider the range of low to moderate cost sensors investigated. To this end the development of a new system that could work in a desktop environment and support a range of different cameras was crucial. First is the physical imaging environment where image capture occurs. This includes illumination conditions and other variables. Next is the choice of imaging sensor. Once images are acquired with the sensor, they can be recorded and processed. This leads to the control system that handles these tasks. This can be an integrated system built into a camera, or a separate dedicated computer the camera can be attached to. This finally leads to the output, which can take the form of an analysis of the image contents of some use to a system operator. Ultimately the output takes the form of general properties of grains recorded from the processed images. Figure 3.1 outlines the entire structure of the type of imaging system assembled. This involves capture and processing elements. This typical setup is similar to methods described in the literature review. Specifically this follows previous standards implemented by Carter and is reinforced by other surveys (Malamas et al 2002) which show this setup is also widely applicable for a range of research activity.

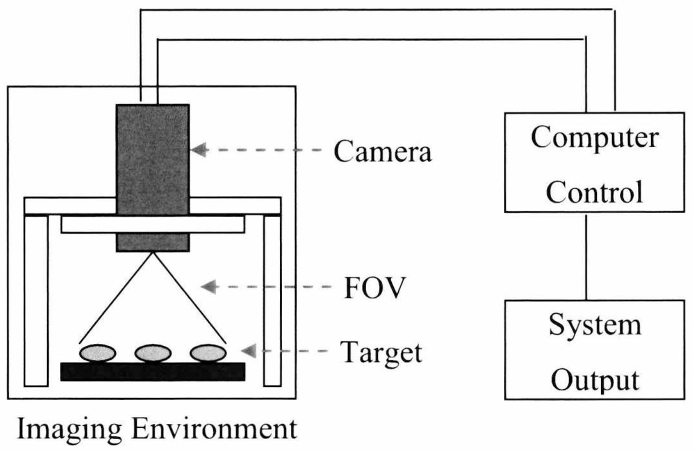


Figure 3.1. Type of imaging system produced.

The imaging environment requires careful evaluation as to wherever anything in operation can affect the imaging task. In the following sections in this chapter, the choice of camera, its lens and the properties of the sensor are considered. In the course of research different cameras were used. Initially a separate digital camera was used with integrated storage capable of some image processing, although this required physical transfer to a computer in order for further image processing. Later in research a system was developed with direct computer control through connectivity between the camera and a controlling PC. The precise layout of the acquisition environment has also changed over time with cameras employed, and slight changes are described here for the various experiments conducted.

The static imaging in this research is achieved with a camera in a similar setup, positioned directly above or beneath the target objects, given that top-down or bottom-up 2D profiles are considered adequate for measurements. This approach is also compatible with a number of the conveyor approaches described in Chapter 2, providing the rate of flow is of a rate realistically measureable by the camera employed.

With the situation narrowed to static and slow conveyor imaging, the next issue is which type of sensors to be employed. These now fall into three classes, digital cameras, progressive scan cameras, and imaging scanners. Digital cameras in this context refers to specific types of CCD sensors chosen for research. Progressive scan cameras refers to old models of CCDs that retrieve images in horizontal lines scanned one at a time from the sensor. Other scanners are flatbed and reader scanners that are either static and record moving objects, or mobile, such as the laser within a commercial flatbed scanner.

The focus here will be differentiation on image types – colour, and greyscale. Colour images offer a wider range of image information, yet greyscale image quality can be higher in a smaller space and at marginally lower cost. The ubiquitous nature of such cameras has resulted in extensive mass production and exceptionally low prices for many mono cameras with optics. This is particularly true for S-video cameras, although digital conversion is also possible with low-cost converters such as S-video to USB. Entirely digital sensors are also highly popular in the commercial camera market yet these models still often correlate performance with cost and are they are sold to target markets and integrated into complete commercial camera units.

Different image processes can be implemented with greyscale and colour imaging. Work by Carter (2005) involved greyscale only, which permitted only greyscale image operations. This generally limited image thresholding to contrast based approaches. While these methods were effective overall, issues occurred where variation in contrast began to occur. Using thresholds became unreliable in regions where variation in image intensity was significant, particularly around the edges of images. A method less impacted by this is through colour imaging through the use of a chroma key. Alternatively, mono cameras can be used under different coloured illumination to record intensity under different colour lights, then use that information to represent that colour spectrum. This method also has the advantage of permitting the use of a low-cost greyscale sensor. However it is still possible to develop imaging setups using greyscale sensors with variable coloured lighting in order to capture intensity for different colour channels.

Many commercial digital sensors, such as webcams often incorporate a rapid image compression in order to speed transmission and to better support streaming video. This has resulted in a wide range of commercial webcams optimised for use with USB connections and for streaming across the internet. The format of this video can be automatically saved in many cases as a JPEG or MPEG for efficient storage. However, these formats are highly lossy due to the compression and actually result in poorer image quality than progressive scan cameras in many cases. Regardless, this highlights the capabilities of on-chip image processing, and is standard among most commercial digital cameras on the market.

For research purposes connection to a PC for direct image processing desirable as this offers flexibility. A dedicated PC with image processing software encompasses the entire versatility of that software. For instance, the MATLAB environment contains extensive image acquisition and processing toolboxes, and retrieved images can be subject to further analysis with a wide range of processing. Image acquisition is also possible with assistance of a range of tools such as the Windows GDI, WIA and DirectShow. This permits prototyping of software based solutions, as well as using computer graphics hardware for rapid processing. This was the case in research by Carter, who used hardware graphics for rapid processing of images. The abundance of desktop computers makes development on dedicated PCs still desirable, as a standard desktop machine or even a portable laptop can facilitate a USB camera system, from image capture to processing and the important feature extraction.

3.3 Factors Influential to Digital Image Quality

Image quality factors include a range of physical effects that can be controlled and mitigated to achieve the best possible image quality for extracting useful measurements. Another factor is cost that limits the choice of sensor for use with in inexpensive test system. Determining where to draw the line between cost and performance is important in this type of application. In terms of performance of the setup and the chosen cameras. A number of important factors are described in the following subsections.

3.3.1 Environment and Illumination Factors

The control of environmental conditions relates to physical obstructions to imaging quality. This encompasses everything from beyond the lens and the imaging target. This includes Lighting issues and debris

Lighting issues, include poor contrast between background and target. This can be caused by position, intensity and orientation of lighting. The consequences include the presence of shadowing and loss of illumination uniformity. These can be mitigated with controlled conditions and testing can gauge the impact of illumination .

Debris and artefacts from the rice itself can be a problem, including rice grain flour and fragments (See Figure 2.11). Also a poor quality choice of background material, that is scratched/worn or dirty can affect imaging results. In practice disturbances in the background material due to both of these reasons may be inevitable. Testing of image capture systems within the Department of Electronics at the University of Kent has shown this can occur. In this research, the focus was on assessing whole relatively intact grains, generally within the appropriate size range for that type of rice grain. Visibly obvious rice grain fragments are not assessed. The imaging approach employed here is a front-lit system. This requires a clear contrast between background and targets. In the case of monochrome imaging, a contrast of intensity values is an expedient option employed previously by Carter and Yan (2005). In colour imaging, chroma key and background subtraction are also feasible. In reality, this background in a flowing imaging system will likely suffer contamination, so a method that is resistant to this is ideal. Too strong a contrast will result in debris being highly visible, yet too weak will result in poor measurement of grain boundaries.

3.3.2 Optical Factors

Optical distortion is possible in this type of macro photography in two forms: A poor focus due to objects exceeding the depth of field of the camera, and the distortion of the image by the curvature of the lens. These phenomena are situation and hardware specific, and can be negated through calibration and distortion correction algorithms as needed.

Calibrations involve the imaging of a real world calibration target of known geometry, such as a grid. If the scale of the grid is known, it is also possible to derive real world scaling from the imaging in this situation, assuming the grid is equidistant to the depth of field. A manual comparison of images can be captured then compared to digitally added lines on the image. Deviation between the grid and the line can indicate the proportion of distortion resultant in the image. This distortion is typically a result of projection and curving in the lens used in imaging. Incident rays normally expected to hit one element of a sensor will in fact hit another due to the change in trajectory from passing through the curved lens. This distortion is usually minimal, and can be corrected with existing methods.

Typical lens distortion takes the form of a pincushion distortion effect, although this can be negligible with common lenses.

3.3.3 Sensor Types

Many types of sensor exist for research. However of interest particularly in this work are lower cost sensors which can be adapted easily to simple imaging systems. While dedicated research cameras can be obtained with a firewire or ethernet based data transfer, these are vastly more expensive systems, often not including any optics with their purchase. Low cost alternatives include older composite video systems, which are widely mass produced and consequently cheap and additionally can be directly input into via conversion hardware.

For the purpose of research webcams are cheaper valid option as they include a complete optics package and data transfer capacity, which can be adequate for macro photography dependent on the camera. However in practice different manufacturers release a wide range of different sensors integral to the camera device. Older composite video devices and higher performing research models tend to employ *charge-couple-device* (CCD) rather than *complementary-metal-oxide-semiconductor* (CMOS) sensors (frequently in webcams), which despite low cost have a poor dynamic range (Litwiller 2001) and sensitivity compared to most CCDs on the market. In the case of composite CCDs, image quality can be good dependent on the situation, although traditional CCDs have relatively low frame rates (Carter and Yan 2005). However for the purpose of static imaging this is irrelevant. In Webcam CCDs (Rutler 2001) often the lower cost sensors cameras can only produce images of scale in 640 by 480 even if they possess 1.3 megapixel sensors, and all resultant images are downscaled or compressed. Additionally many such cameras incorporate compression on chip due to their use of low-speed USB 1.0 connectivity. While newer sensors can use higher speed USB, many still incorporate the internal on-chip compression. In addition, many sensors will inefficiently write the compressed data into an uncompressed storage medium. Fortunately, alternative sensors exist such as now cheaper mass-produced composite CCTV board cameras, and specialist purpose CCDs for new markets. However many of these systems come without optical packages, or with other problems. This will be discussed further in section 3.5 as these systems were chosen over webcams for further investigation.

Different types of CCDs exist, such as three-colour CCDs exist that use a beam splitting prism to split the light into three entirely separate dedicated CCDs, although the focus here will be on simpler and cheaper standard colour CCDs. These use a single arrangement of cells of red green and blue pixels arranged in a Bayer (1976) pattern, filtered to absorb from the respective visible spectrum. This gives a slight bias to the number of one cell type (typically green) over the others. While the luminance intensity information is the same are recorded by a greyscale image, the colour resolution is limited by the spread of receptive pixels in the Bayer pattern and consequently lower colour sensitivity than a 3CCD system. Monochrome sensors also form the lowest cost single CCD systems.

3.3.4 Image Format and Storage

Existing image standards were used in the course research including Bitmap, GIF, and JPEG/MPEG compressed images and video streams. Recognition of physical issues with these images is important in this research. JPEG and MPEG are prevalent image and video streaming formats in digital photography particularly for USB camera devices which are using an older USB standard. These compressed format permits greater storage density and quick transfer from camera systems with slower connection standards such as USB 1.0. This permits a transfer rate of around a megabyte per second for older USB standards, although newer USB 2.0 offers transmission in orders of magnitude faster. However, this standard is prevalent in many low-cost camera systems. Use of image compression is avoided where possible given that images may be only used briefly and long term storage may be unnecessary in practice.

3.4 Imaging Approach Developed

In order to obtain good quality images in an easy to control environment, an imaging box was created. The box itself was a simple 180x180x100mm Aluminium work box with a large removable lid. A hole with a diameter of 30mm was cut precisely at the top centre of the removable box lid. This box provided complete darkness for implementing controllable light conditions. This was achieved by the installation of two relatively uniformly illuminating Perspex light panels provided by the Department of Electronics at the University of Kent. These 3.3 volt 220mAmpere panels contain three bright white Light Emitting Diodes within a strip at one end of the panel. Light is diffused through an intricate arrangement of spherical features within the glass of the panels, resulting in a spread of light almost evenly throughout the entire length and width of the panels, and results in directed diffuse light oriented forwards from the breadth of the panel. This is highly localised, and needs to be near To power these panels a 9 volt power supply was used, rigged to the box via a drilled out adapter plug in the side and then regulated though a simple regulator and a potentiometer. This was secured to the inside of the box on a small isolated circuit, and mounted via a metal screw. The screw was fastened to the box through the regulator to act as a heat sink to ensure its stable functioning and secure it to the box. This complete setup is illustrated in Figures 3.2 to 3.5.

In practice during the imaging research conducted the imaging box went through several stages of development during use. Initial work used the box with a canon camera and a matte contrasting background placed on the base of the box. The Camera was positioned over samples through a large circular hole in the box lid, and illumination panels were fastened to a plastic base on which the matte background rested. The camera was initially positioned on a small set of adhesive stilts approximately 5mm high in order to ensure the camera was orthogonal to the target. Further development produced a more robust camera mount capable of holding several different types of cameras that was securely bolted to the box of approximately the same distance. This mount was created with a 35mm diameter hole, permitting the resting of mounted cameras onto the box within the mount, leaving the cameras perfectly perpendicular to the box. A plastic screw was tightened through a hole in the side of the mount to ensure the position of new cameras used remained fixed.

Positioning of the LED panels was important for stable illumination. Initially these were held in place with a weak adhesive and small plastic panel on one end of the panels, holding them together. Later improvements produced an elevated camera mount was produced to position a range of cameras into the box. This guaranteed the stable positioning of the light panels, the camera, and also permitted use of a range of cameras and easier sample placement with a tray, and maintaining the approximate imaging distance for each sensor. The sample tray was created that also acted as a holster for the LED Panels. This tray was positioned in a trough made out of a machine cut nylon block. This holster for the lights also had concavities for the positioning of the light panels on either side of the tray and was fastened to the box base with a weak adhesive. Its dimensions were 100x60x20mm to hold the panels along its length close to a 'tray' area indented by 2mm. This indentation was hold removable sample trays that would keep target materials within reasonable depth of field for macro and high magnification photography. The LED panels provide soft ambient illumination in order to minimise shadows against the contrasting background and provide a reasonable standard of consistent illumination within the imaging area. The tray is shown in Figure 3.4 with a removable background, which can be replaced with various sheets of paper, plastic, glass or other materials. Paper or plastic was generally used, either black for a matte contrast or colour for employing a chroma key as background subtraction.

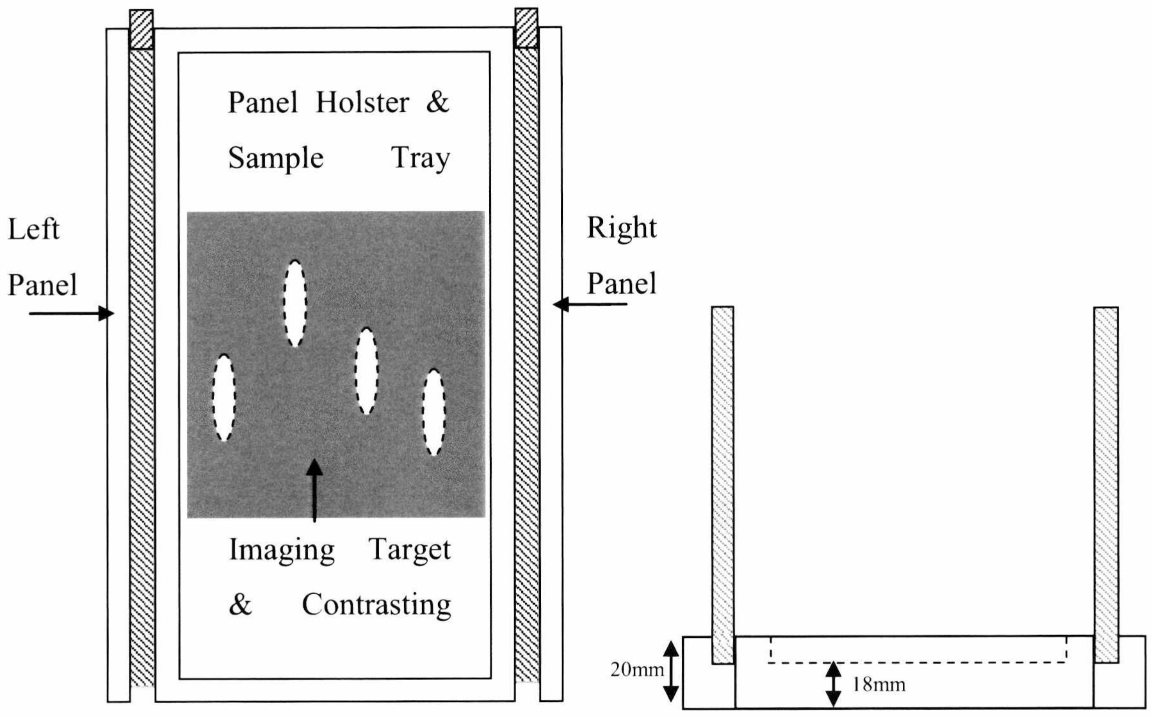


Figure 3.2. Top down and side view of sample trough and light panel holster.

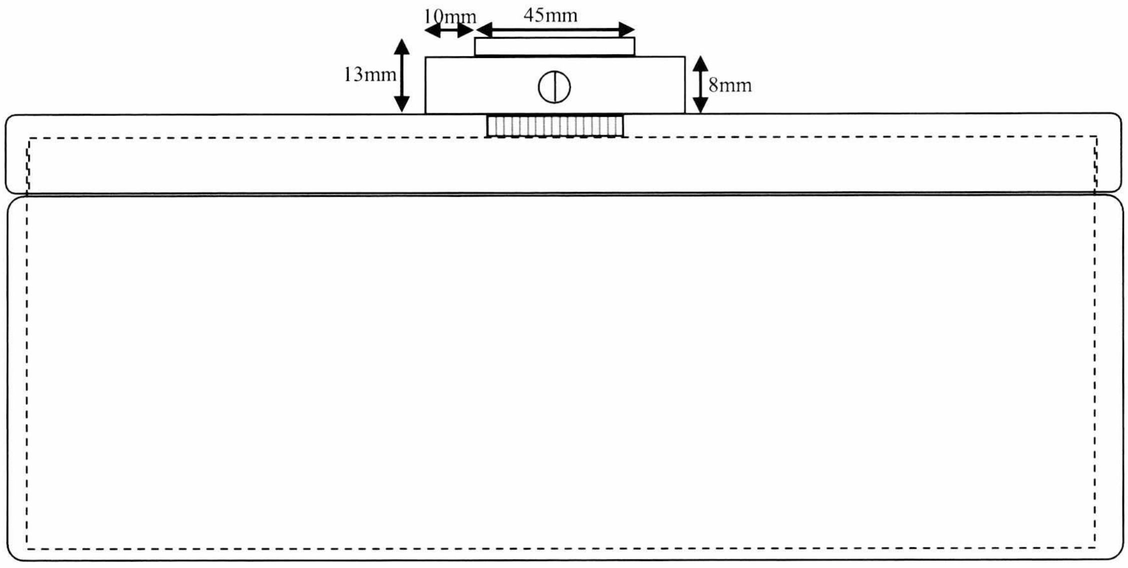


Figure 3.3. Side view of imaging box interior.

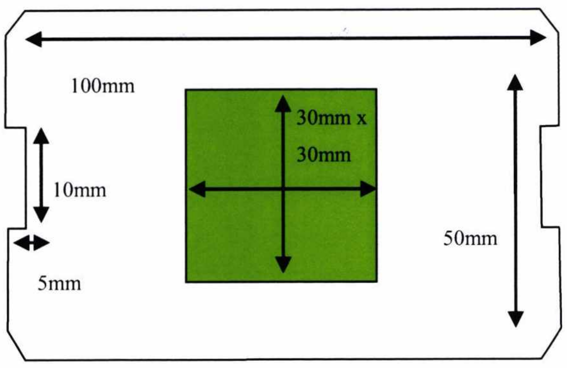


Figure 3.4. Removable sample tray.

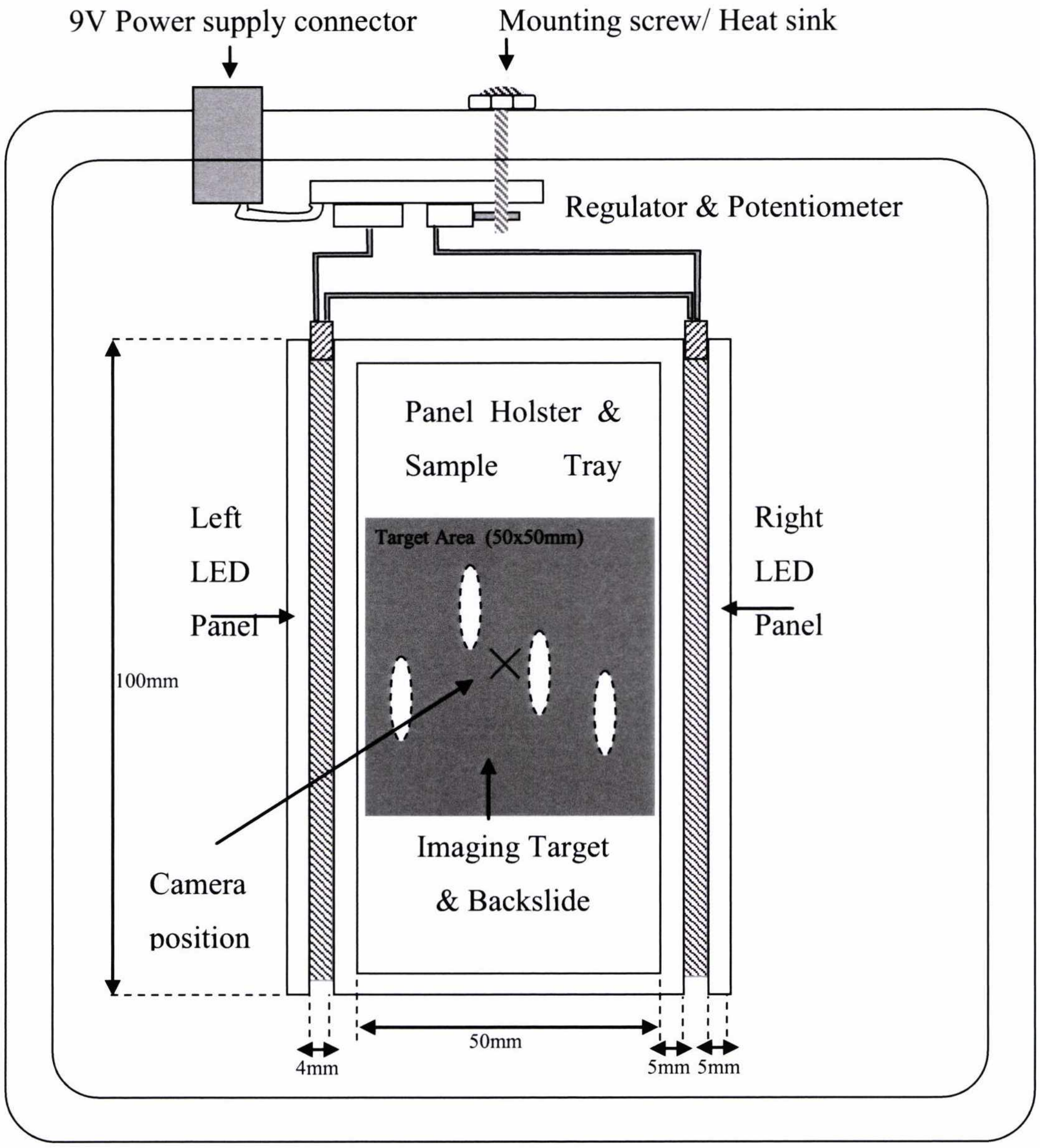

9V Power supply connector
Mounting screw/ Heat sink
Regulator & Potentiometer
Panel Holster & Sample Tray
Left LED Panel
Right LED Panel
Target Area (50x50mm)
100mm
Imaging Target & Backslide
Camera position
50mm
4mm
5mm 5mm

Figure 3.5. Top down view of imaging box interior.

3.5 Imaging of Grain Samples

A series of steps needed to be performed to prepare the set up and acquire images of samples of grains for research. The finalised order of preparation is now given for samples with the complete imaging box and a dedicated CCD sensor with a customised lens mount. This mount permits the use of a range of sensors and lenses. The final process of preparation for the completed system with the Celestron and Pecan sensors is as follows, relative to figure 3.6:

(1) Appropriate camera positioned in slot.
(2) Secure screw tightened to hold camera in place.
(3) Target or Samples positioned in tray.
(4) Lid positioned to seal box and hold camera.

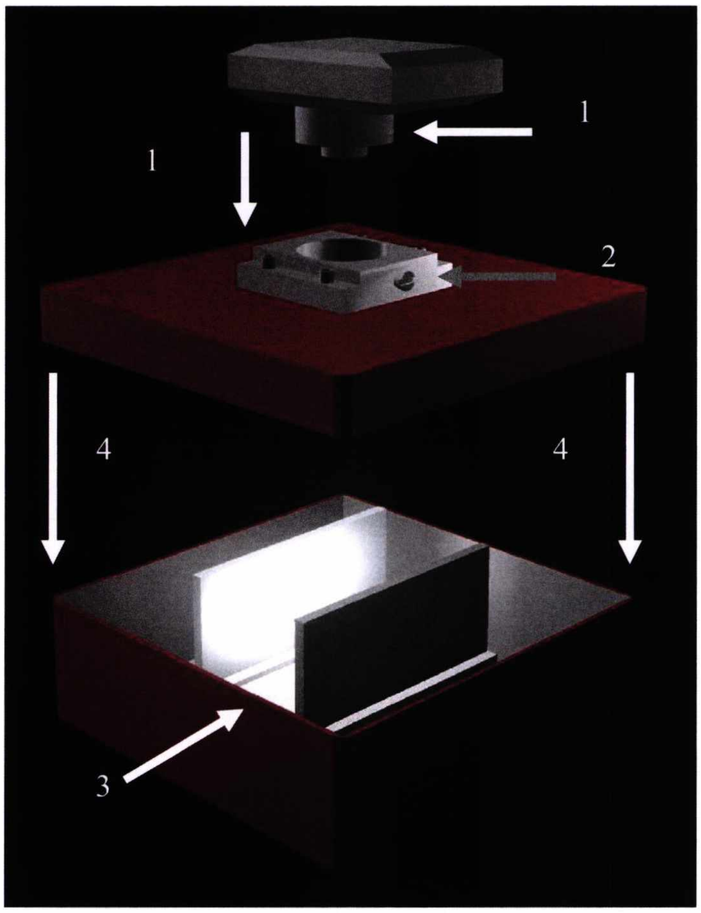


Figure 3.6. Order for positioning of final set of imaging box parts.

The first two steps occur in general preparation, and can be immediately followed with calibration testing instead of sample imaging. The third and forth steps are repeated for each target image acquisition. More specifically, step one relates to the preparation of the camera. Specifically some of the cameras employed have been developed in a way to fit precisely into the box, notably a Celestron Neximage CCD with a Lens described in the next section. These cameras have been fitted with custom-cast lens mounts to fit into a metal mounting on the top of the imaging box. Once in place, a small plastic screw is positioned in a hole in the side of the mount and tightened securing the camera firmly by its custom lens mount to the box mount. Power and data cables can be run directly from the camera to a computer and/or power supply as needed. Following this the samples are positioned within the imaging box. This is achieved using a plastic sample tray that can be removed from the box. This tray is holed and can have a contrasting background in the form of a thin paper or plastic sheet attached underneath with a weak adhesive. Sample grains can be positioned in the tray on the contrasting background then placed in the target area between the light panels. The desired intensity of illumination can be controlled by adjusting the potentiometer..

Finally the lid can be placed on the box, and an image of the sample can be obtained with relative ease. A wide range of cameras are capable of performing capture, although an additional preview feature in cameras employed can assist as the researcher can make a reasonable estimation of image quality. This results in digital cameras with a preview screen or a USB connection to a PC with monitor being important for this task.

3.6 Choices of Digital Camera

A number of different digital cameras were used in the course of this research. CCD sensors were the chosen devices as they were both directly compatible with the imaging box and low in price. This section evaluates each camera considered during research. The different cameras employed are described individually in complete tables 3.7 to 3.14. These tables list the make of camera, its specific sensors, any optics that were included or employed with the camera, applicable Pricing information and any relevant software included with the camera. The optics is significant to pricing as separate optics may be expensive. Relevant software is important with regard to the camera's ease of use, and how adaptable the camera is with a control device. The PECAN sensor was the final sensor obtained, and was not extensively used in the course of research, however is a standard C-mount board camera. Its testing was to ensure the compatibility of this common camera mount with the imaging box, and to give an assessment of its performance as the cheapest of all the sensors obtained. Data sets in research were obtained with the Canon and Celestron Cameras, with reference images using an old system with a CCIR camera. Preview images for each camera are provided in Figure 3.8 to 3.13.

Light sensitivity is critical and is defined with the LUX standard unit illumination (luminous emittance from a surface). All rice grains to be considered are generally reflective and will be highly visible even under low lighting conditions approaching 1 lux. Light panels employed in the imaging box are capable of producing localised illumination in a small region within this range, well within the parameters of the sensors described in Tables 3.7 through 3.14.

Table 3.7. Specifications of USB Celestron NexImager CCD used during research.

USB Celeston NexImager CCD	Known Specifications
Sensor	CCD Only ¼'' HAD, Colour Sensor 640x480 VGA RGB resolution Light sensitive to <1 lux
Optics	Techspec megapixel Finite conjugate μ-Video Miniature Board Lens (edmundoptics.com ref 58204) 100mm Focal length 90-150mm Field Of Vision 0.068X-0.040X Magnification Range
Price	£50(~$80) Lowest commercial price found for sensor, not including optics. Optics ~£52 (~$80). Total price <£110.
Software	Supports Directshow, TWAIN, WIA

The Celestron camera was chosen as it was a good balance of customisation and cost. It was a colour camera which could be modified easily to support a variety of optics. This gave the sensor some flexibility in implementing it with the setup. It is also the sensor that is most optimally positioned within the setup. Figure 3.8 is an example image.

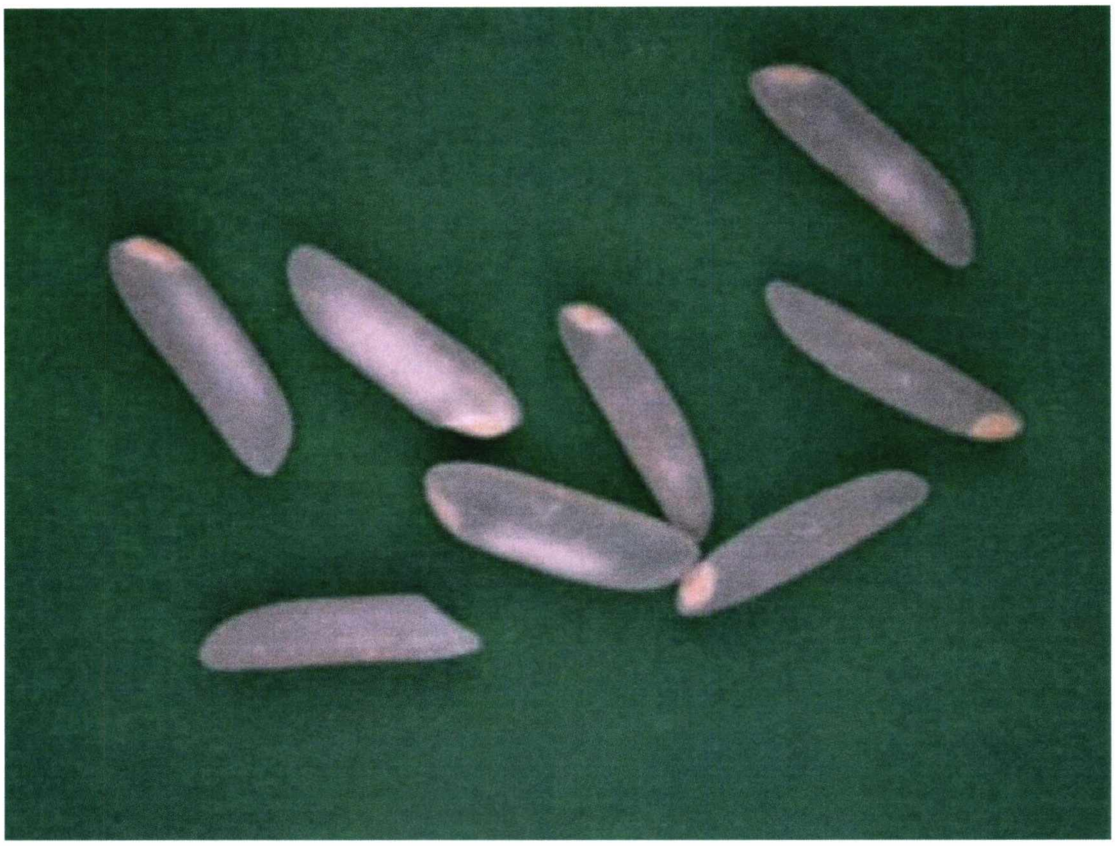


Figure 3.8. Typical Rice image from Celestron NexImage Camera.

The canon camera was extensively used during research due to its high resolution and sharp image quality. Its high resolution capacity was however limited by the physical design of the camera making positioning of the camera requiring manual effort. This was however aided by a good quality macro mode and many automatically adjusting settings. However this camera saved images to an internal memory card in JPEG format, requiring transfer after imaging rather than the live USB link from the Celestron Sensor. As the Canon camera was used in early work, it was used when the system went through several iterations of development. Initially the camera was positioned on top of the box before the final camera mount was positioned, with the samples placed in the base of the box rather than the holster for the final system. The difference between this and the new system is negligible as the distance between samples and targets was approximately the same within the order of millimetres. Different background materials were also used for different tests, as shown in Figure 3.10 and 3.11. Early work on separating rice varieties used a matte background, with later work employing colour imaging to improve the thresholding.

Table 3.9. Specifications of Canon Digital Ixus Camera used during research.

Canon Digital Ixus Camera (commercial camera)	Known Specifications
Sensor	5-megapixel RGB CCD capable of producing 2560*1920 Images Intergrated image processing and enhancement: Filtering, Colour balancing, Mono capable Controllable exposure time Automatic JPEG2000 image compression with several standards for image quality. No ability to disable image compression.
Optics	On board (commercial digital camera) No Specifications disclosed, minimum focal length ~100mm.
Price	£130 (original price ~2005) Rechargeable battery, cable and memory card included.
Software	Supports windows TWAIN and WIA drivers, although no direct feed from camera possible (save to memory card first)

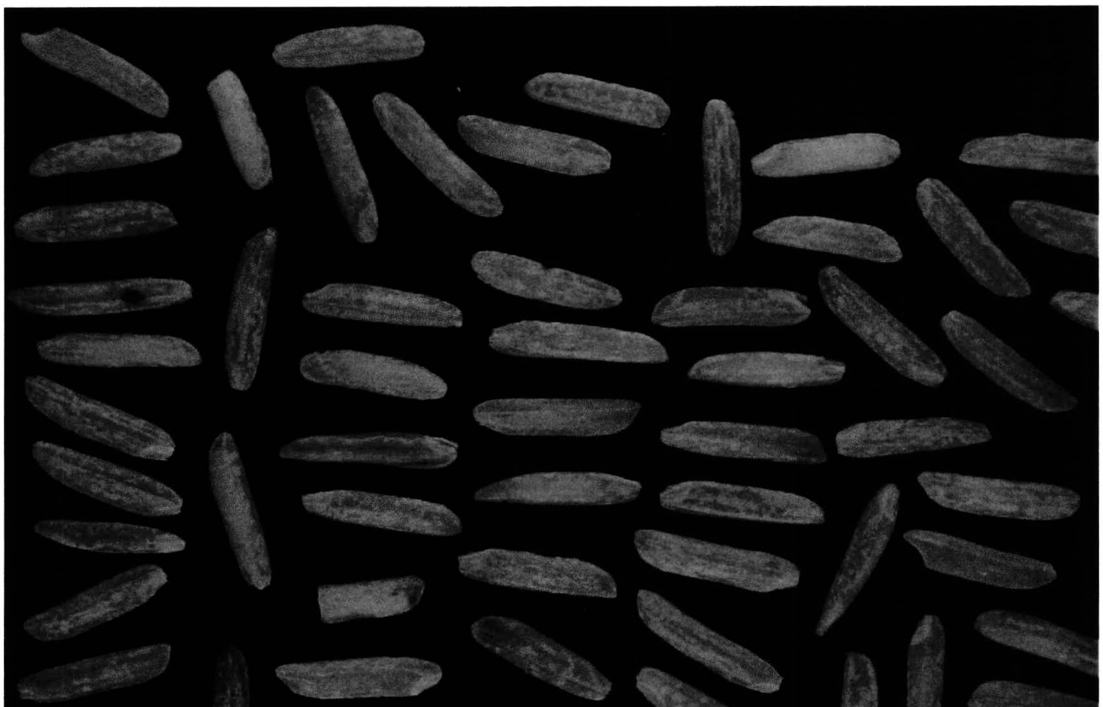


Figure 3.10. Region of interest from the first iteration of development of the Imaging system with the SD20 Camera.

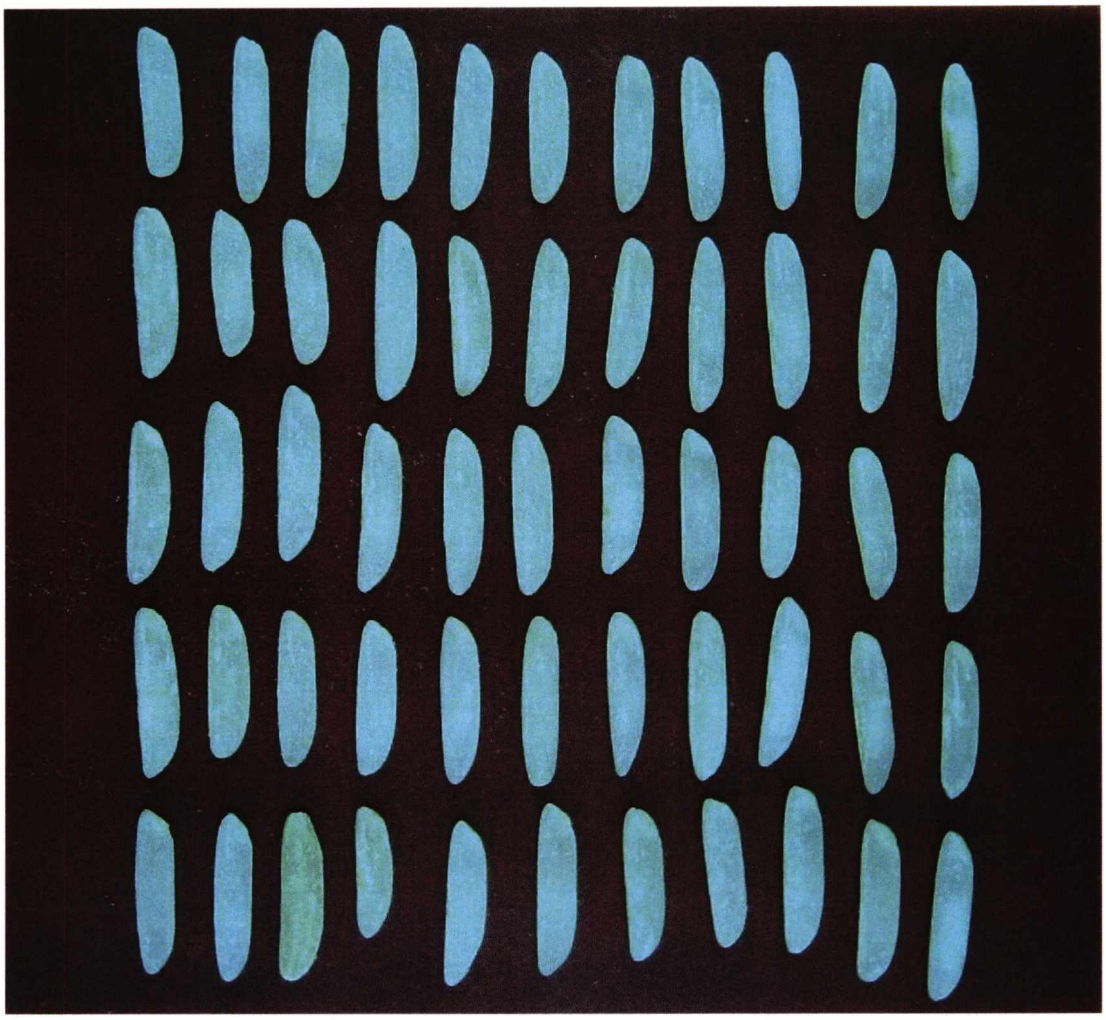


Figure 3.11. Region of interest from the second iteration of development of the imaging system with the SD20 Camera.

In comparision The CCIR Camera described in Table 3.12 was used with a previous setup developed by Carter as a comparative camera. An example image shows reasonable quality although distortion and a poor contrast present problems in Figure 3.13

Table 3.12. Specifications of Sony Monochrome CCIR Camera used during research.

Sony Monochrome CCIR	Known Specifications
Sensor	Analogue 1/3'' Sensor S-Video output combined with a USB Convertor Effective Resolution of 768 x 576, Monochrome only Minimum Light sensitivity of 0.25 LUX
Optics	M12 x 0.5 miniature board lens
Price	£40 original price for board camera between 2003-2005. (not including lens or USB Frame Grabber).
Software	Supports windows TWAIN/WIA drivers (Via USB Frame Grabber)

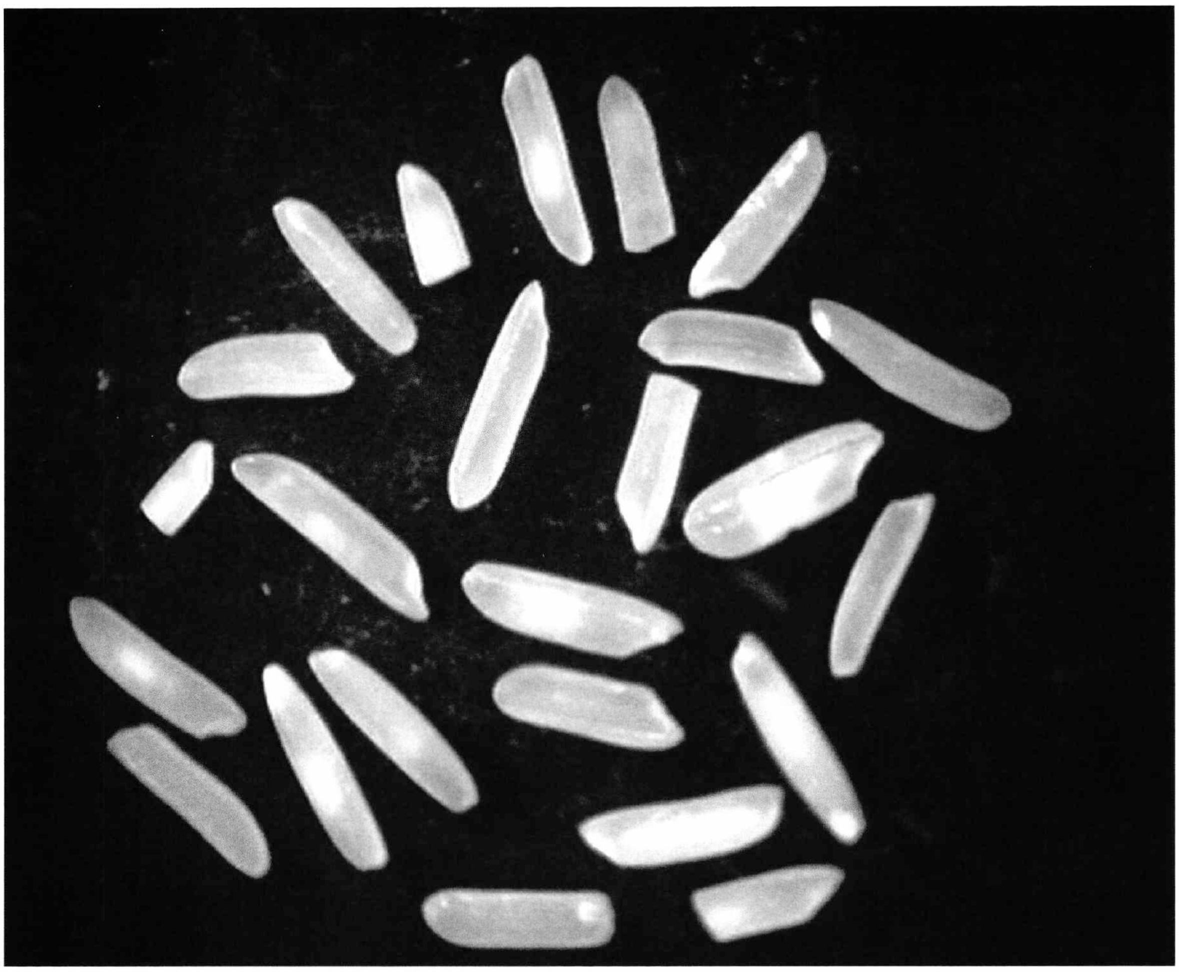


Figure 3.13. Image from Sony CCIR Camera using DR Carter's imaging system.

Table 3.14. Specifications of PECAN CCTV sensor used during research.

PECAN CCTV Sensor	Known Specifications
Sensor	Analogue 1/3'' Sensor S-Video output combined with a USB Convertor Effective Resolution of 720 x 576 Monochrome only. Minimum Light sensitivity of 0.25 LUX.
Optics	16.0MM C-mount Lens included with camera. Although lack of polarising filters or iris results in high light sensitivity. Very low light conditions essential. Working distance of 100mm acceptable.
Price	£6 original price for board camera (purchased 2008) Includes 16.0 MM lens immediately usable for macro imaging. USB 2.0 Frame grabber obtained separately to connect S-Video to PC. Commercially available for £10.
Software	Supports windows TWAIN/WIA drivers (Via USB frame grabber)

The Low cost PECAN camera was able to be integrated successfully in the box, however its image quality was not sufficient for this research. However, it is extremely likely given the past effectiveness of Carter's use of a CCIR camera that similar models exist of the same board type, and such equipment as illustrated in figure 3.15 is compatible with the imaging setup. The Celestron and custom sensor is shown in figure 3.16 as contrast.

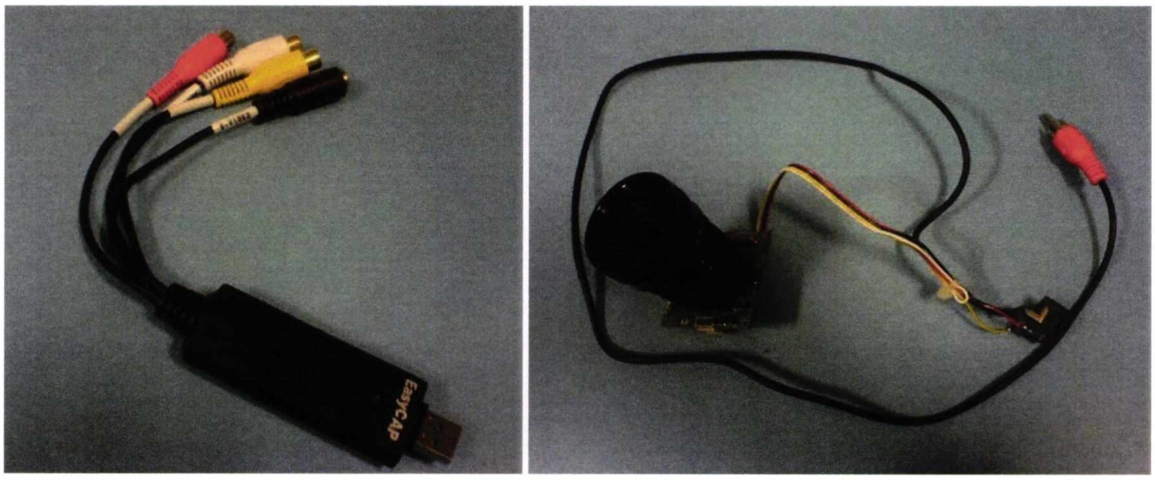


Figure 3.15. EasyCAP USB 2.0 Composite/S-Video input, and PECAN CCTV CCD with bundled lens and 12 Volt Power adapter.

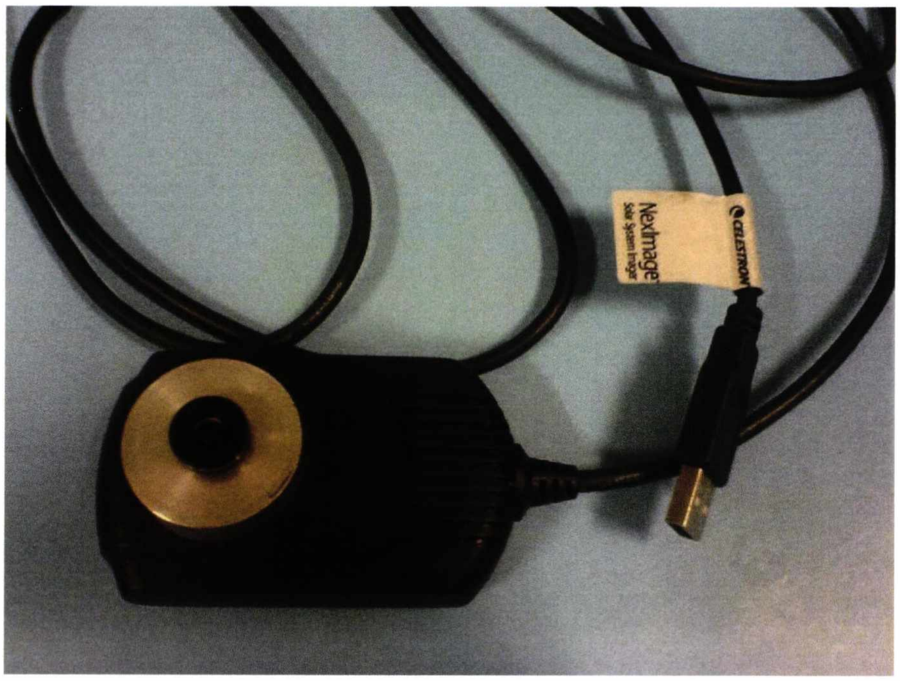


Figure 3.16. Celestron NexImage Dedicated CCD with custom lens mount.

3.7 Controlling Conditions in the Imaging Box

Several conditions needed to be met to image effectively using the box. Control of lighting, control of camera focus and lens distortion, and dealing with a limited depth of field for macro photography must be accounted.

Lighting is driven by diffuse panels that provide a consistent spread of illumination across their surface area spanning out from their direction of orientation. To ensure the illumination was sufficiently nominal, spatial dependency tests (Carter 2005) were performed for each sensor, and results are given in the subsections next.

3.7.1 Depth of Field, Lens Distortion and Image Scale

In order to properly assess the effectiveness of the imaging setup with different cameras, Focus and lens distortion were accounted through the use of a calibration images. This was an image taken with a calibration object of sufficiently accurate scale. Two types of image were taken in the course of research. Earlier work used the Canon camera primarily. This wider angle camera permitted a broader calibration with an image of graph paper as used in previous research (Carter 2005) and illustrated with an example image in Figure 3.17. The image of the graph paper can be compared to artificial digital straight lines in order to determine the presence of lens distortion, such as a pincushion or trapezoidal effect. While functions exist to detect and distortion through the use of image processing (Carter 2005), it can be more efficient to minimise its impact. Orthogonal positioning of cameras in the setup ment no significant trapezoidal distortion could occur, and pincushion distortion is reduced by focussing on a smaller region of interest. This distortion was particularly apparent with the Canon in a macro mode, notably towards the edges of the image. This was reduced by only focussing on a region central to the image where calibration lines from early test images were relatively straight. Later work using dedicated board cameras and CCD sensors were less impacted by this type of distortion after similar improved calibration tests (Hobson, Carter and Yan 2009). These tests also permitted derivations of sizing for Camera imaging targets. Correction for the lens distortion was performed in Figure 3.18.

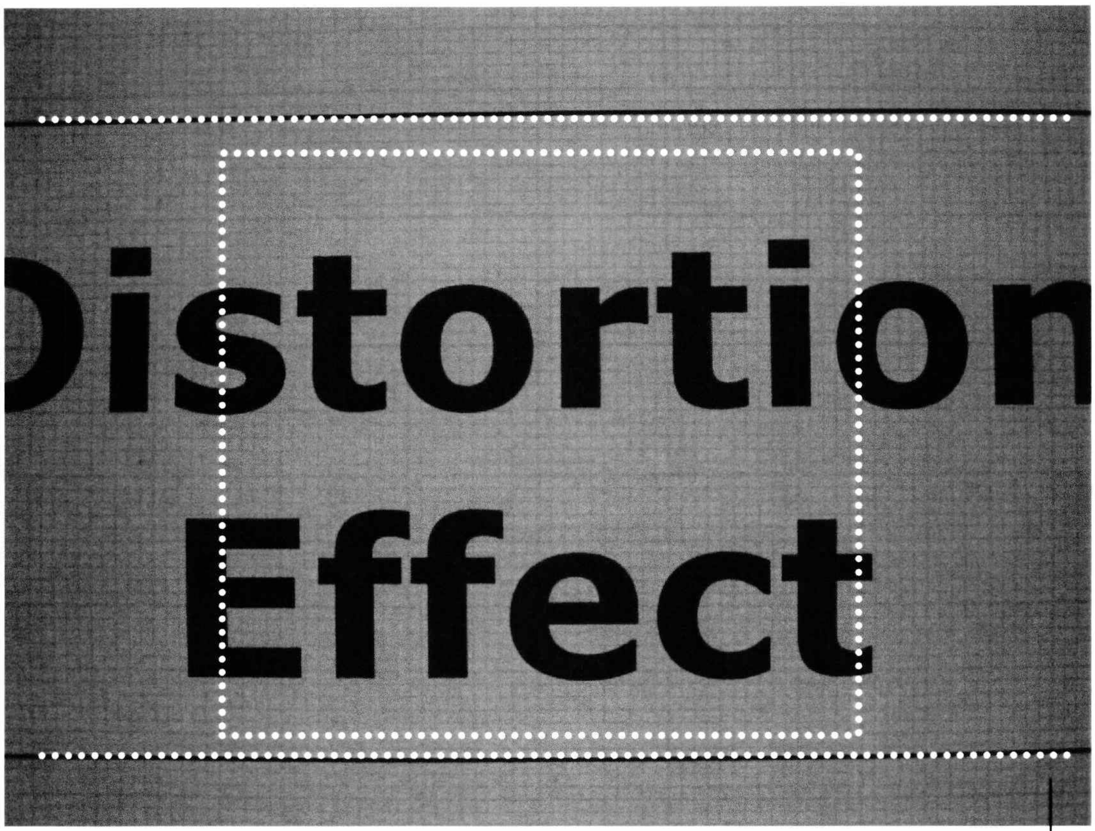


Figure 3.17. Canon camera calibration image and impact of distortion.

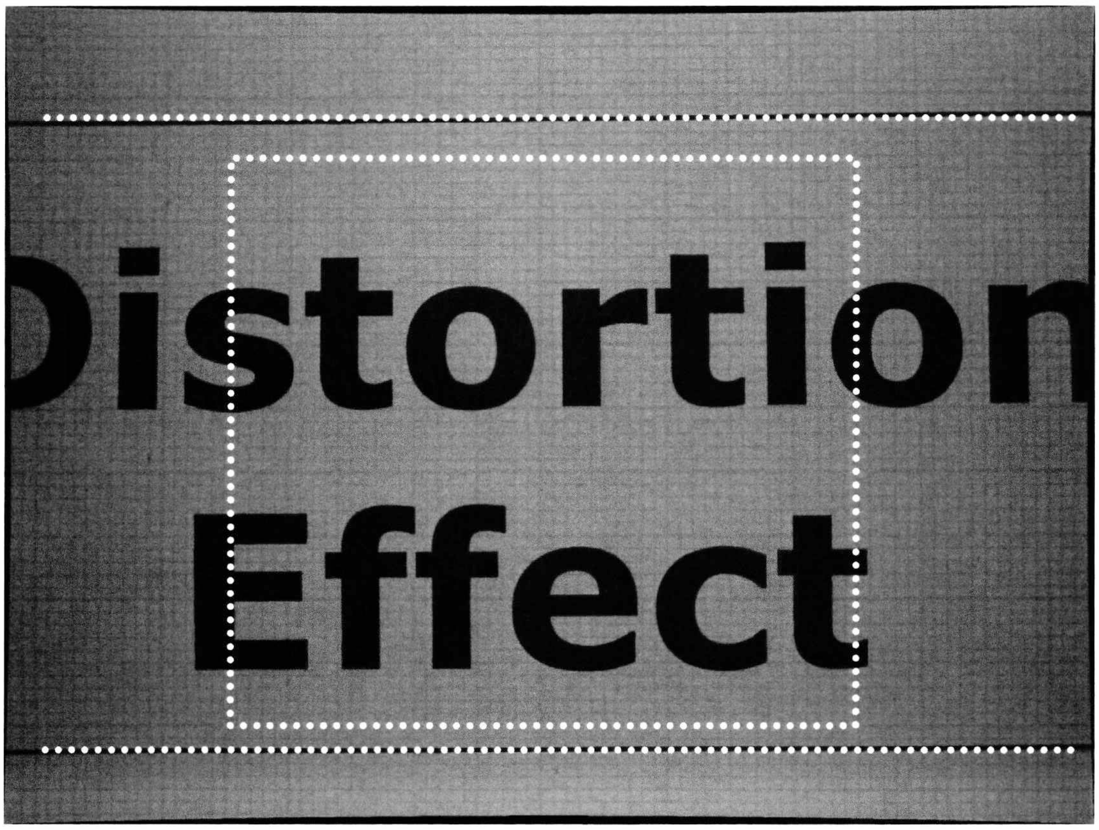
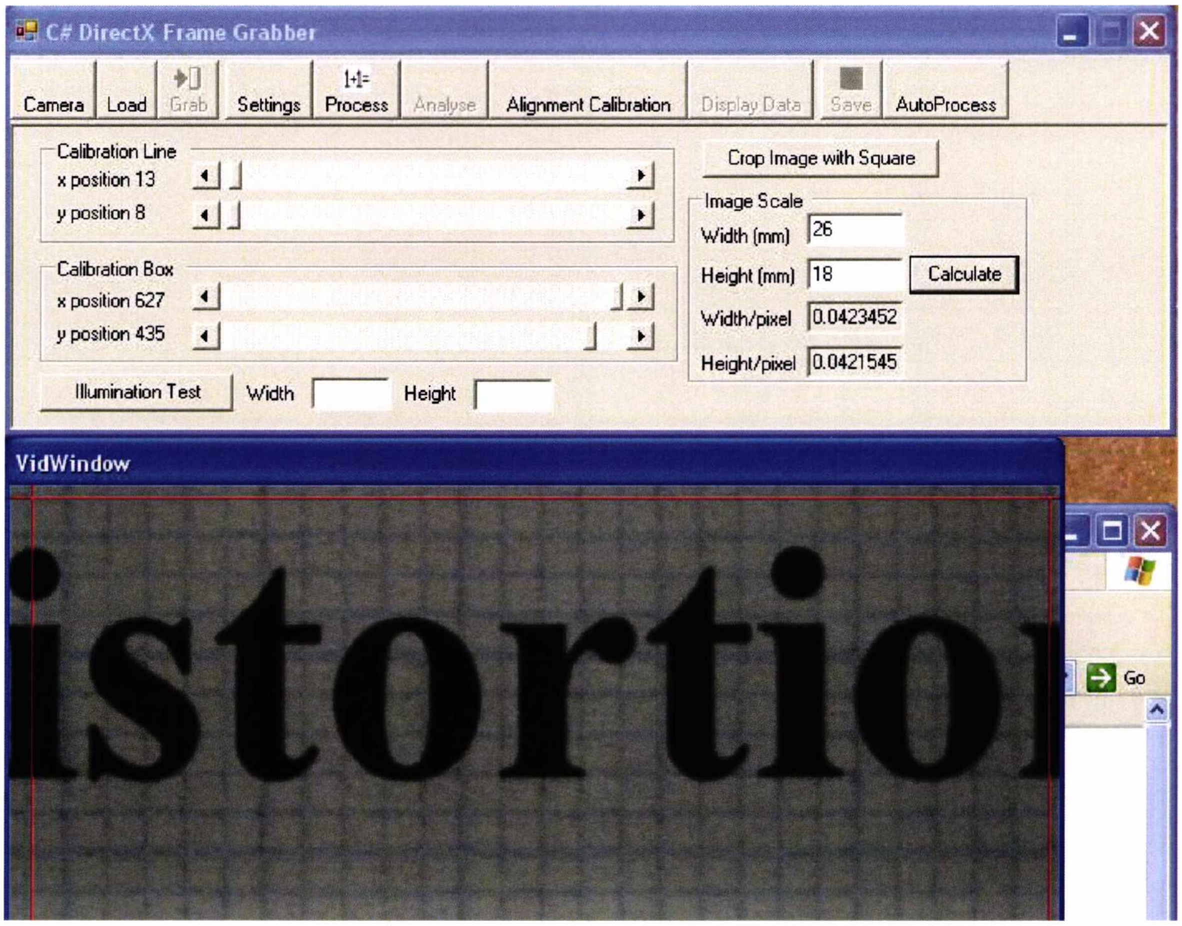


Figure 3.18. Canon camera calibration image following distortion correction.

Similar work was undertaken for the Celestron camera. Imaging of the distortion target was performed to detect distortion. This appeared negligible and digitally added lines were not significantly distant from the graph paper. By this stage special software had been written to use in conjunction with the USB frame grabber that would permit drawing a set of aligned calibration lines. From these lines it was possible to determine both the impact of lens distortion and the size of the image relative to the true scale. It is clear that the deviation between the grid lines and the pixel wide calibration grid in Figure 3.19 is negligible, indicating that the loss of accuracy due to lens distortion is approximate to a sub-pixel scale. In addition these targets permit a good basis for the derivation of per-pixel scale in the image, which can be integral in using imaging as a sizing method. In Figure 3.19 the capture application is used to draw the calibration lines, and specify the dimensions of the target area in millimetres. This can easily be used to calculate per pixel dimensions.



Calibration Line
x position 13
y position 8

Calibration Box
x position 627
y position 435

Image Scale
Width (mm) 26
Height (mm) 18
Width/pixel 0.0423452
Height/pixel 0.0421545

Width
Height

Figure 3.19. Software interface with digitally added lines for calibration image.

Further to the derivation of per-pixel area, the dimensions of the cameras and summary of lens distortion is given in Table 3.20. The cameras operate on similar scales although difference is significant enough to justify calibration per camera for sizing measurement in images. Lens distortion is also not a significant problem with any camera considered. The most significant problem is a slight 'squashing' of the vertical image of the Pecan board camera that occurred, resulting in an imbalance between the width and height per pixel. This causes a slight measurement inaccuracy that is rotational invariant. However the difference between the dimensions is 0.002mm or a relative difference of around 10%. This raises concerns over the reliability of this device, and consequently was not as trusted as other devices.

Table 3.20. Dimensions of camera images from calibrations.

Camera/Lens	Per Pixel Lengths	Lens Distortion
Celestron NexImage Custom Lens mount	0.04mm	Negligible (not detectable)
Canon Powershot Camera Integrated Lens	0.03mm	Slight (detectable), at edges of images. Negligible towards center. Cropping and distortion correction negates effect.
PECAN Board Camera (16.0mm C-mount Lens)	Height 0.02mm	Negligible (not detectable) Slight vertical squash

3.7.2 Illumination Uniformity

Spatial dependency tests were used by Carter to determine the general uniformity of illumination dependent on the positioning of the light source. This was performed to ensure the choice of positioning for the light panels within the imaging box was adequate for stable and repeatable imaging. This test originally imaged a printed sheet of circles of an arbitrary size against a contrasting background. This image was subject to a low level automatic threshold using carter's function, and the mean intensity of each circle determined in order to calculate the deviation of intensity between regions within the image. Carter's work indicated gave results from the test as a deviation approaching +4 and -6% towards image edges. While on the surface this appears to be a negligible impact on intensity, this can result in an error in sizing for the particles when applying the automatic threshold. Carter calculated this as approximately $\underline{+}2\%$. This was performed using a 6x4 grid of white discs on a contrasting matte background, with each s possessing a 2mm radius. To test the cameras used in this research a larger imaging area was considered and consequently a larger space required a larger map. As the precise imaging area for some of the cameras was smaller than Carter's setup, grids of 1mm squares were used initially for the high resolution commercial camera, and reused for other camera systems. From the stability of the illumination testing, this became the region of interest. Figure 3.21 shows an image of this 20x20 region taken with the SD20 Camera. Distortion correction was applied and the test was performed. Figures 3.22 and 3.23 show the test images obtained with the Celstron and PECAN cameras of relative size. The smaller scale and closer focus of these images resulted in a narrower field of vision limited to the center of the region of interest. The difference in geometry has a negligible impact on intensity values, and distortion correction was not necessary for those cameras.

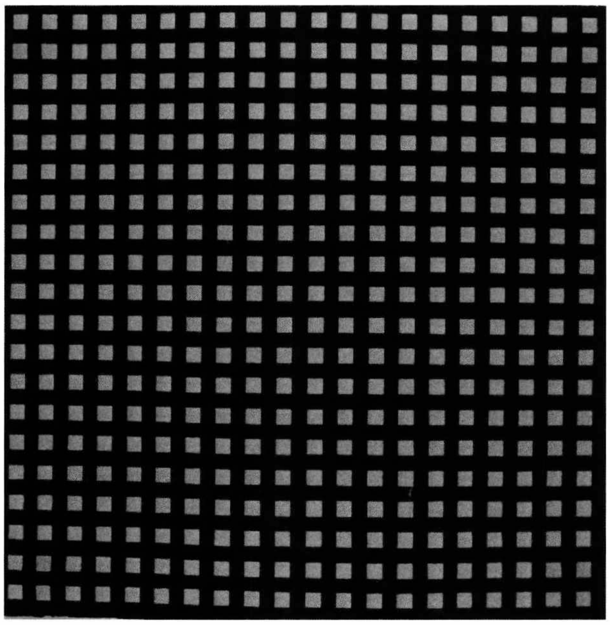


Figure 3.21. Spatial dependency mask captured with SD20 camera.

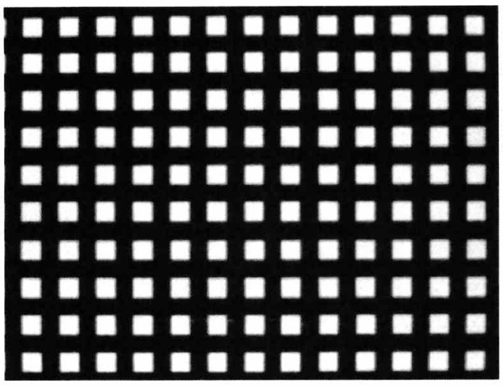


Figure 3.22. Spatial dependency mask (13x10) captured with Celestron camera.

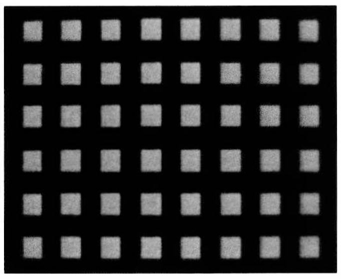


Figure 3.23. Spatial dependency mask (8x6) captured with PECAN camera.

Figures 3.24 and 3.25 gives the spatial dependency of the mask from Figure 3.21 as a topographic map. This was produced by applying an adaptive grey level image threshold function on the target in figure 3.20, then comparing the resultant image with the original greyscale image data using connected components and image arithmetic. Both mean intensity and area of each target are recorded and plotted separately for the entire mask.

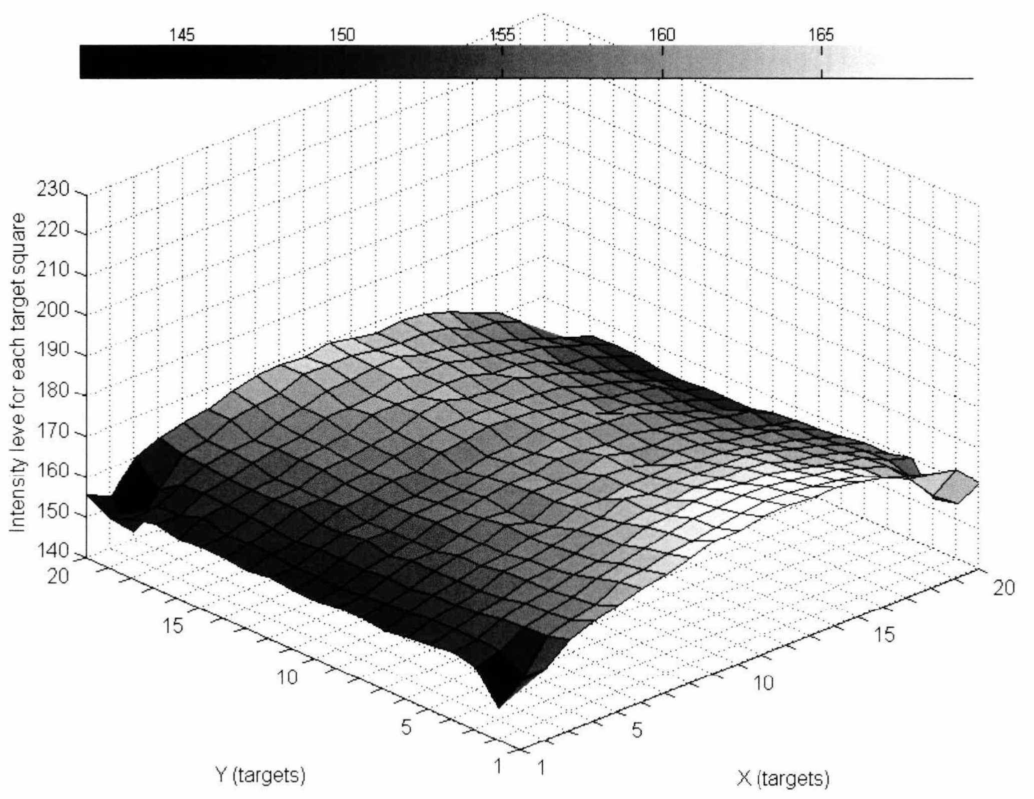


Figure 3.24. Measured target illumination intensity for Canon SD20 camera.

This calibration test suggested some minor issues with illumination uniformity towards the image edges, of a similar scale to that observed by Carter with his system. Correlating size variation with intensity reveals there was no significant correlation in size variation between intensity and area, as can be seen when comparing figures 3.24 and 3.25. Size variation is significantly higher than intensity variation however, likely due to the impact of a combination of intensity variations in the printed paper target and the application of the adaptive threshold function. However, even the maximum variation of area of these small targets is within a range of 200 pixels In both cases, slight variation occurs along the x-axis, suggesting targets are slightly brighter towards the central space between the panels, although precise vertical distance from the panels is less influential in this setup. The impact of variation is negligible within the target area.

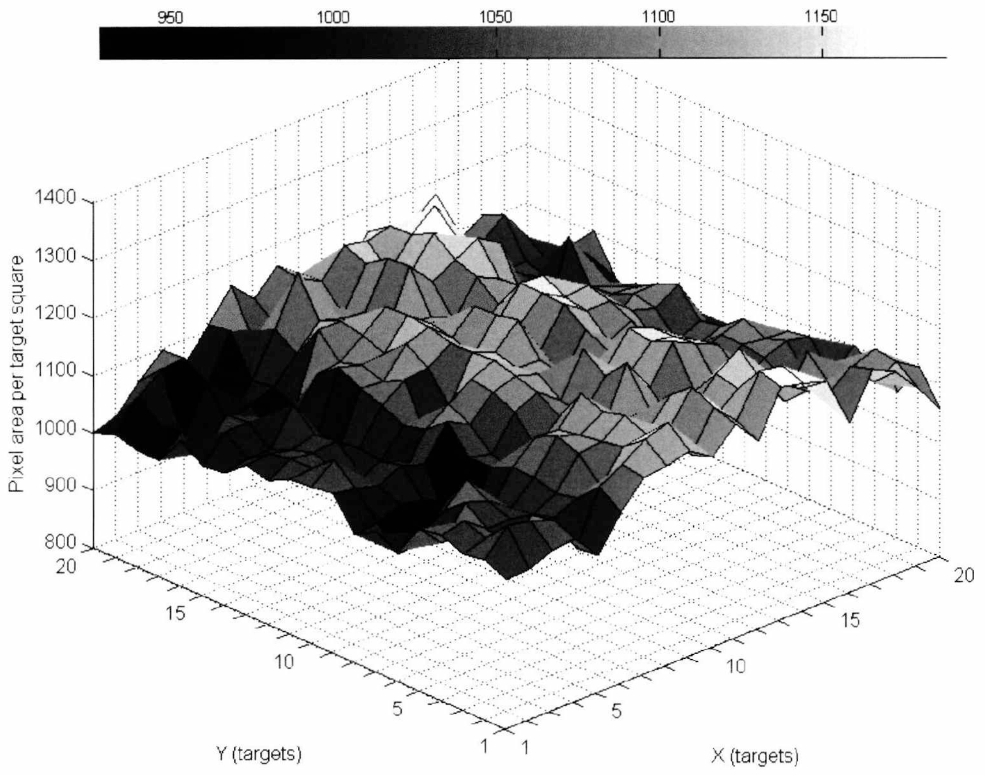


Figure 3.25. Measured target illumination area for Canon SD20 camera.

Similar calibration tests were performed for the other camera systems and results are quantified for each camera in order to assess and contrast their performance. It is also possible to gain some indication of the difference in uniformity of illumination between the sensors, and get a general impression of the camera's dynamic range and illumination sensitivity. This is particularly the case of the PECAN CCTV camera, which had such high light sensitivity that the potentiometer of the lighting power supply had to be adjusted to reduce the intensity of illumination in the box in order to detect any image whatsoever. Despite a lower quality reading than that of the Canon camera, the circles were all detected with a close range of sizes and intensity ranges, although with a larger proportional deviation compared to the Canon camera. The smaller imaging area resulted in only 4x3 circles within the camera field of vision, shown in Figures 3.26 and 3.27

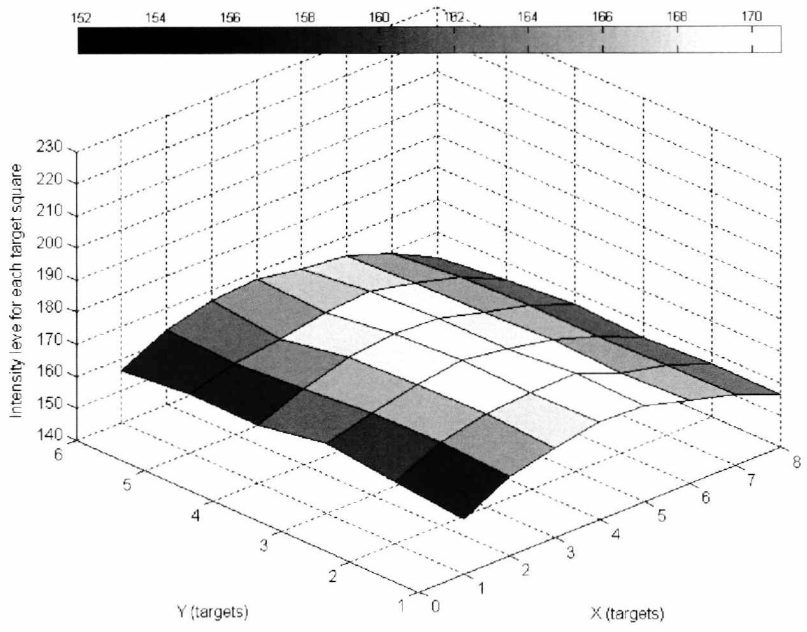


Figure 3.26. Measured target illumination intensity for PECAN camera.

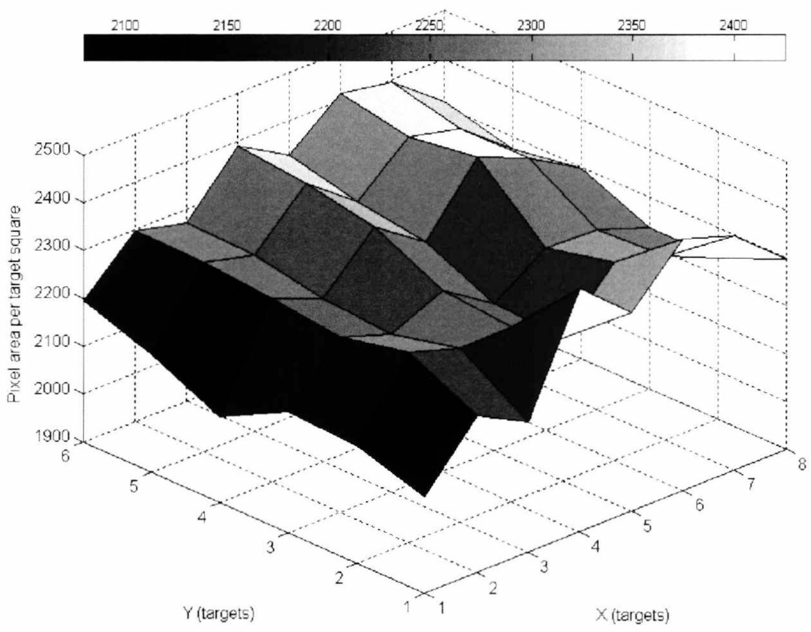


Figure 3.27. Measured target illumination area for PECAN camera.

While the low cost PECAN camera gives similar ranges of spatial illumination quality it suffers as result of its smaller target area due to its optics, and also required modification to the illumination level of the box in order to function at all. The main issues with this camera were practical ones rather than the potential image quality that could be obtained. The high light sensitivity of this sensor and difficulty in adjusting the lens within the box resulted in this camera being the most difficult to calibrate in this work. The Celestron is however rather different. While the lens required manual adjustment the camera could automatically adjust its intensity sensitivity due to its onboard image processing.. Its larger imaging area gives a better indication of a range of intensity values in a more consistent and brighter range of illumination values. Intensity varies in a relative range of about 5% from the mean. Results are given in Figures 3.28 and 3.29.

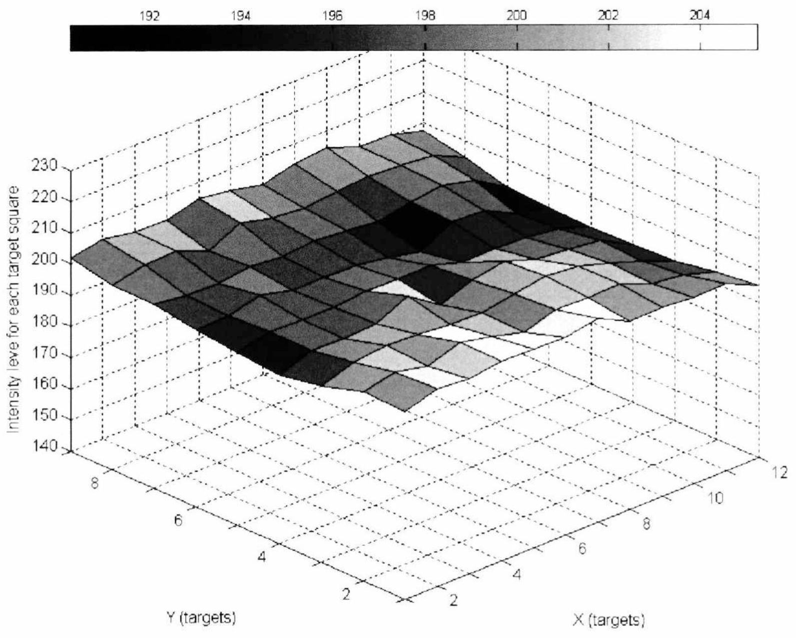


Figure 3.28. Measured target illumination intensity for Celestron camera.

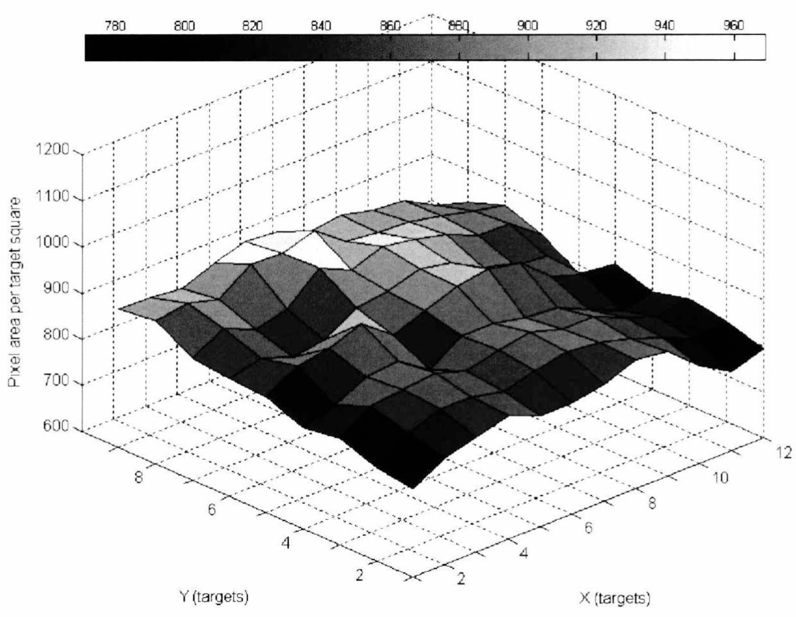


Figure 3.29. Measured target illumination area for Celestron camera.

3.8 Summary

In this chapter suitable cameras for the capture of grain images have been found and tested in order to perform the capture of rice grain images for the processing and analysis in the forthcoming chapters. A range of different cameras were compared in order to determine an optimal one. Despite lens distortion issues, the best camera in terms of image quality performance is the Canon camera. However, the versatility of the direct PC connecting and lower cost of the Celestron sensor suggests it is a better compromise in practice, and can be implemented much more simply without the need for lens distortion correction. Its colour imaging gives it similar versatility to the Canon sensor despite lower image resolution.

The physical setup that has been used for image capture in this research has been described. This setup, being essential for the capture of grain images, requires sufficient image quality for effective application of the processing techniques described in the next chapter. These will incorporate low level processing to return a good quality binarised image of grains and their textural information. Although the application and results of calibration tests have shown sufficient standards for reliable imaging in this chapter, the functionality of image processing required within these calibration tests is described in later chapters. It is through the development of the imaging setup and its successful employment to produce images of sufficient quality for grain measurement. These are then recorded for further analysis. Further research into the specific characterising features and advanced image processing such as segmentation can be now investigated using real grain images obtained with the system.

Chapter 4

Image Processing of Rice Grain Images

4.1 Introduction

In this chapter the image processing aspects of the system is discussed. This is integral to the finished frame grabbing software on PCs attached to the imaging system. However the algorithms for this processing were required to be developed specifically for this task. Several Matlab functions written using known image processing techniques and integrated with common image acquisition routines are described in this chapter. These were used to obtain images from which the relevant properties of grains could be obtained. In addition, similar C++ codes developed in the course of research for this purpose and depreciated in later work in favour of the Matlab functions are given in Appendix A, and Matlab code in Appendix B.

When developing methods for processing images of rice grains imaging approaches adopted by other researchers are compared. Hardware and image processing must be considered, and also what features are to be extracted and how they are to be used. This chapter describes methods to transform the image from its initial captured state to a state from which features can be derived. This occurs as image processing in order to isolate grain properties from images. Next in this chapter texture analysis based on image or region properties from processed images is described. This leads to shape analysis of the grains based on assessment of their digital shape boundaries and regions, addressed in chapter 5.

4.2 System Software Architecture

In this section the software architecture of the imaging system is illustrated. This begins from the point of the frame grabber, which can be provided by various sources such as a dedicated framegrabber and a direct connection between PC and camera, or from an image file saved on disk. Beyond this point is the control and processing system which will perform a variety of image processing functions dependent on user preference. In Figure 4.1 the entire architecture is summarised in a brief diagram, although some stages can be omitted in practice for more specific processing tasks.

The software system is divided into two sections – *Acquisition and Processing* and *Post Processing*. The first stage relates to the frame grabbing process and the low level image processing to remove redundant information. These include stages such as the application of a threshold to simplify the image to binary. Morphological operations are applied to make various image corrections. Extending from this it is possible to implement some low level segmentation functions on the binary image. Included in the morphological operations are arithmetic operations, which can be extended to use with the edge detection function for the first type of analysis. Outputs from this stage are the original captured image, the binary image, and the results of edge detection correlating with processed image regions. All of these can be in image form and saved in random access memory (RAM) for further evaluation. Original and processed images were also saved to disc.

The second stage relates particularly to the analysis of the binary image for the purpose of shape description and extraction of relevant features for analysis. However these features can in turn be used for more detailed processing of the image in the case of a second more sophisticated segmentation function. Finally features can be recorded in memory or to disk. These can then be subject to analysis relevant to the classification and identification of target grains with a wide range of different analytical techniques. Analysed features can be output and recorded.

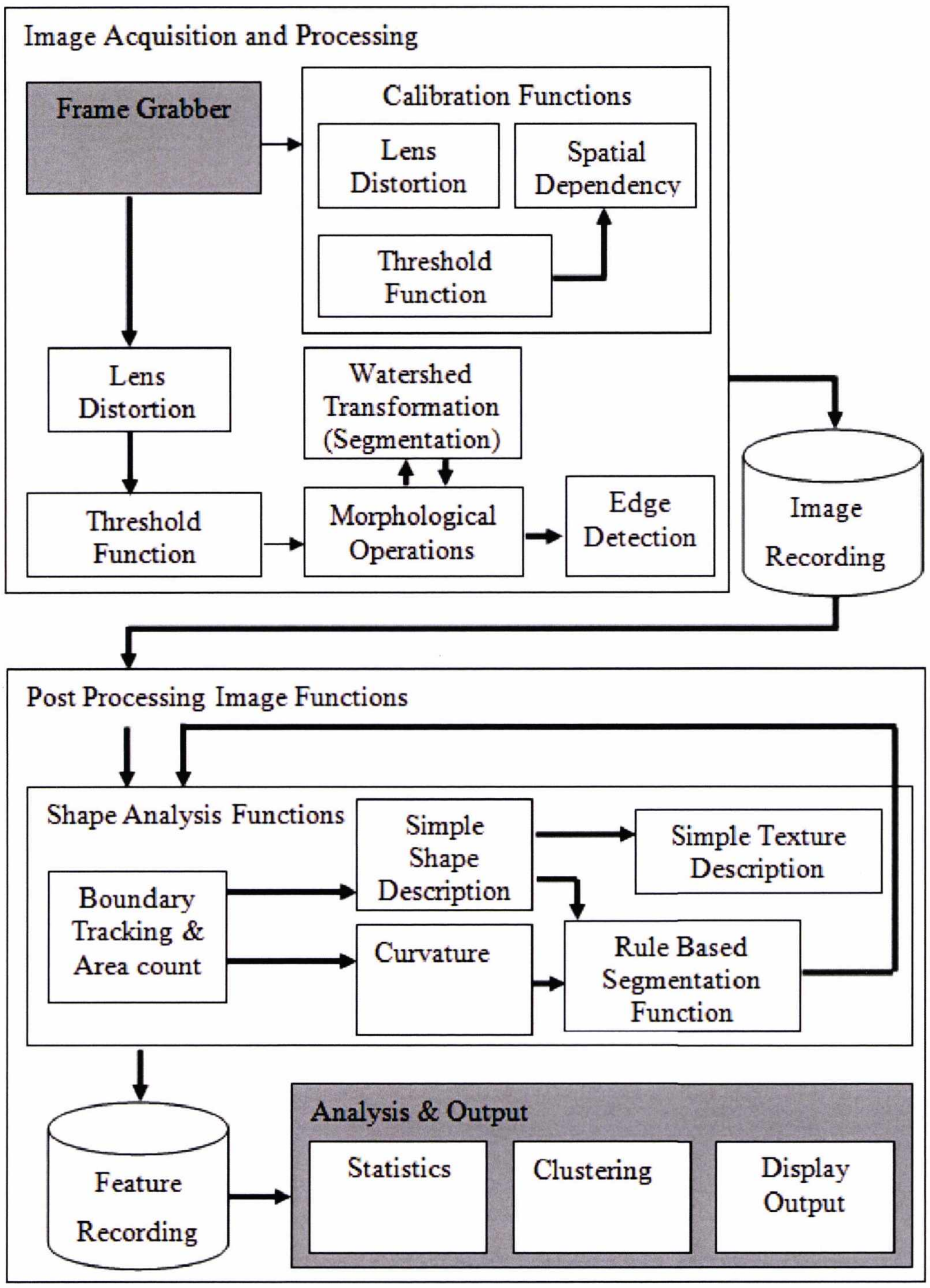


Figure 4.1. Imaging system software architecture.

The low level steps relating to acquisition and processing are described next in this chapter, and the higher level steps are discussed in the next chapter. The software architecture is intended to be a flexible design to support a range of input camera devices and a reasonable level of alterations to the physical imaging setup. These include camera properties such as:

- Use of a colour or greyscale sensor.
- Use of sensors of different resolutions.
- Changes in optics.
- Scenario changes, such as static or conveyed imaging systems.

4.3 Retrieval of grains from images

A number of low level image processing techniques are employed in the system for the detection of the boundary and region of grains within the images. These are implemented on either an attached computer that retrieves the images via a frame grabber, or can be built into an on-chip image processing system with dedicated hardware. Given the low cost of many complete PCs, a stand alone computer dedicated for this task is highly practical. In the laboratory environment a range of older and newer PCs and laptops with limited software installed are adequate for this task. A standard operating system such as Windows XP, a supporting software kit such as the Matlab Compile Runtime, and a stand alone application written in languages such as C++ or C# can be taken together to run complete image analysis. The steps for image processing were given previously in figure 4.1 under the software architecture.

Critical to this stage is the distinction of target grains from irrelevant background information, ideally with minimal impact from noise, debris and so on. While controlled conditions can eliminate the majority of problems and make this process easier, a threshold function is required to return a good standard of measurable grain from the image.

The type of imaging used in the described research relates to two-dimensional imaging of the profiles of target grains (Paliwal and Visen 2003, Senkai 1996), particles (Bowman, Soga and Drumond 2000, Xu, Luxmoore and Deravi. 1997; 1998; Hobson, Carter and Yan 2007)), and so on. Images obtained are defined in this research as discrete arrays of values recorded by the Sensor. Generally this sensor type is a CCD. These are standardised as a two-dimensional array, or three dimensional in the case of a colour image with three distinct colour channels. Image *I* is defined as:

$$I = \{(x, y, z) | 1 \leq x \leq N, 1 \leq y \leq M, 1 \leq z \leq L\} \tag{4.1}$$

Where x and y represent pixel coordinates in the image of width and height N and M, z represents colour bands and L is 1 for a greyscale image or 3 for colour. Many ways of representing this discrete form of of image on machine or *system-on-chip* (SOC) exist. For example embedded systems can store images in memory using languages such as C and VHDL. Past work has shown special devices are indeed optimal for this type of work (McBader and Lee 2002), such as Xilinx Field Programmable Gate Arrays. These massively parallel devices are progressively increasing in capacity and becoming more ubiquitous. However it is not necessary to implement this research on such technology until the algorithmic aspects are proven effective so imaging will be discussed in common languages such as set theory and relatively simple Matlab code in this thesis. Implications will be further discussed in future work in chapter 7.

In the system, an image is typically interpreted as an array structure stores intensity values in a single array. Several image properties are stored with this array – Width, Height, and bits per pixel. The width and height indexes are used to express the dimensions of the image. The bits per pixel express the values stored in each index of the array. In early work, an image object class to represent bitmaps read as files was devised in a C code structure that can be found in the appendix. Later work switched to a combination of MATLAB integer array classes and C# Image and Bitmap classes, as these formats were optimal for these environments. Development in Matlab and C# was preferential in later research due to ease of use of Matlab's function library and the versatility of Windows API for C#.

4.3.1 Lens Distortion Function

To correct any potential lens distortion, an automatic lens distortion filtration function was developed in Matlab based on existing methods previously developed by Carter and Yan (2005). This function, given here, was devised to take the following key parameters required for its implementation:

- The source Image, parameter *InputImg*.

- The Working Distance in millimeters – the distance between the focal point and the imaging target. This is parameter *distance*.
- The approximate width of the source image in millimetres, *imwidth*.
- A scaling coefficient in order to determine the scale of curvature of the lens. This was determined manually through trial and error as parameter $C_{coefficient}$.
- A resizeing variable to crop the edges of the image by a proportional number of pixels. This crops the sides of the image in order to remove the subset of pixels left blank by the distortion correction. If these pixels remain, they may have unusual effects on any further segmentation functions.

Carter's algorithm was adapted into a routine that only needs to determine the curvature in a single quadrant of an image, then can apply the same transformation on respective pixels in other quarters of the image. The function involves the following steps:

Convert the input image to a greyscale image, *InputImg*.

Calculate several constants. First, millimetres per pixel (m_p, equation 1) based on scene width in millimetres *imwidth* and width X and height in pixels *Y*. This is used in conjunction with a constant *h* (2) based on the diagional image width to calculate distance Z (3) between the image corner point and the lens :

$$m_p = (\frac{X}{imwidth}) \tag{4.2}$$

$$h = \sqrt{X^2 + Y^2} \tag{4.3}$$

Which are used to calculate Z

$$Z = \sqrt{(\frac{m_p \times h}{2})^2 + {m_d}^2} \tag{4.4}$$

Which can then be used for the mapping of pixels.

Create a second empty (all pixels=0) image *OutputImg* of same dimension as *InputImg* to receive the mapped pixels.

For each (x,y) pixel in the range in OutputImg(X/2,Y/2), calculate the following parameters:

$$m_x = ((X/2) - x) \times m_p \tag{4.5}$$

$$m_y = ((Y/2) - y) \times m_p \tag{4.6}$$

Which represent the real distance x and y in millimetres between the image center and the pixel in question. The following parameters are used to calculate angle and distance for that pixel from the lens:

$$qh = \sqrt{(m_x^2 + m_y^2)} \tag{4.7}$$

$$z = \sqrt{(qh^2 + m_d^2)} \tag{4.8}$$

$$z_p = Z - z \tag{4.9}$$

$$\alpha = \cos^{-1}(\frac{d}{z}) \tag{4.10}$$

$$\theta = \tan(\alpha) \times z_p \tag{4.11}$$

Which is immediately used to calculate the parameters required to calculate the location of the new mapped pixel.

A final parameter in practice needs caution handling division by zero. If $mm_x = 0$, the value beta in equation (12) can be assumed to be zero and doesn't need to be determined:

$$\beta = \tan^{-1}(\frac{mm_y}{mm_x}) \tag{4.12}$$

Giving values which can be used to calculate mappings for specific distance to the location of the point x and y in millimeters. However in practice this must account for the coordinate x when x=X/2. In this case, the value in equation 13 used to determine the x distance can be replaced with zero. 13 and 14 are used to determine x and y distances.

$$x_{n'mm} = (\theta + qh)\cos(\beta) \tag{4.13}$$

$$y_{n'mm} = \sqrt{(\theta + qh)^2 - x_{n'mm}^{\;2}} \tag{4.14}$$

As this first materialises as actual distance in millimetres, it must be converted to the approximate pixel to be transferable to the discrete image:

$$x_{n'} = \frac{x_{n'mm}}{m_p} \quad (4.15)$$

$$y_{n'} = \frac{y_{n'mm}}{m_p} \quad (4.16)$$

And a final transform returns the true coordinates within the image origin space. Although this is from the top left corner of the image, and only in that quadrant of the image.

$$x_n = ((x-(X/2)-x_{n'}) \times C_{coefficient}) + x \quad (4.17)$$

$$y_n = ((y-(X/2)-y_{n'}) \times C_{coefficient}) + y \quad (4.18)$$

From these coordinates the output image pixel can be set four times for each quadrant, modifying the precise coordinate written to by incorporating the image width and height as needed, shown in the following vectors. O_{img} represents the destination image and its respective pixel, and I_{img} the source image.

$$O_{img}(x, y) = I_{img}(x_{n'}, y_{n'}) \quad (4.19)$$

$$O_{img}((X-x), y) = I_{img}((X-x_{n'}), y_{n'}) \quad (4.20)$$

$$O_{img}((x), Y-y) = I_{img}(x_{n'}, (Y-y_{n'})) \quad (4.21)$$

$$O_{img}((X-x), (Y-y)) = I_{img}((X-x_{n'}), (Y-y_{n'})) \quad (4.22)$$

The distortion correction function returns the cropped image following the application of correction. This is used in calibration testing described later in this chapter.

4.3.2 Segmentation Overview

The use of the term 'segmentation' in image processing is ambiguous and specifically in this research refers to imaging situations where sought objects are physically contacting in an acquired image. Specialist processing is required to separate the objects in order to measure them independently. This particularly relates to 2D binary image segmentation techniques, although there is potential application and expansion involving intensity values from greyscale and colour images. To this end, the types of segmentation techniques involve the properties that can be extracted solely from a binary image of shapes. This information alone is sufficient for established segmentation techniques such as the watershed segmentation and concavity based approaches.

However first segmentation will be expressed in relation to image thresholding. In this sense, an image threshold is designed to produce a binary image where different regions are denoted by their position and Boolean value. Many image threshold functions exist and perform as segmentation functions in this sense. Particularly common are *Region-growing* methods, which appear in this thesis as morphological operations such as distance transformations, image thresholds and watershed transformations.

A further segmentation, applied to the resultant binary image produced by the first segmentation, is the watershed segmentation. This is described later following the description of its component morphological operations. It can be performed on a binary without any situation specific knowledge as a regional segmentation approach.

4.3.3 Image Threshold Function

The objective of the threshold is a segmentation of target rice grains and contrasting as shown in image capture in chapter three. A wide range of Image thresholding techniques exist that are appropriate for this task, such as the Otsu threshold (Otsu 1979), static thresholds manually derived, and special 'chroma key' utilisation of colour in images. In addition more elaborate functions are possible involving properties derived from the image scene. A method using actual shape features has been implemented, and will be described after the descriptors themselves in chapter five.

Previous research in the embedded System Laboratory produced a highly effective adaptive threshold for static imaging of particles against a contrasting background (Carter and Yan 2005), and this method was reused in this research due to its effectiveness in similar tasks, although some modifications were added for colour utilisation. This approach involves the following steps:

(1) For every intensity level of the image, apply a threshold at that level on the image and produce a new binary image. Record the number of separate objects in the image at that threshold level using connected component analysis (Gonzalez, Woods and Eddins 2004).
(2) Storing the recorded objects per level in an array, determine the local gradient of the array using its +/-3 neighbour points and store in a new array.
(3) Find the highest peak in the data, and follow the decreasing gradient in the direction of increasing intensity level, until the gradient begins to increase again.
(4) Use the index of increasing gradient as the threshold intensity value.

However this approach was limited in a number of ways. In terms of speed, the function was relatively slow due to its repeated use segmentation and connected component analysis. This process is highly iterative and given the large size of images in this research could take a considerable length of time to execute. It also is only effective for the imaging where density of objects against a contrasting background is lower than that of the background. The contrast between objects and background must also be absolutely clear through a wide difference in intensity values. This requires good standards of imaging, particularly the illumination quality and dynamic range of values recorded by the imaging sensor. As shown by Carter, this method is particularly effective for greyscale imaging and was extensively employed when using the Canon camera for research. However, A computationally simpler alternative is possible when employing colour imaging.

The careful calibration of the imaging environment and the performance of illumination tests ensures maximum stability in to produce good standards of image in which to apply segmentation. Where conditions are poor, for instance, on the edge of the illuminated region of interest, objects can be poorly segmented. This may result in large holes on the interior and/or the shape boundary. This is further managed through a process illustrated in Figure 4.3, showing the image processing steps required to obtain textural information from an image. An input image I is subject is stored twice in memory, once as the colour original and again as a converted to greyscale image. This can be efficiently stored as the original colour image array with an additional 4th dimension on the Z axis representing intensity. The colour image is subject to the removal of one colour channel and the averaging of the others to produce a greyscale image without the chroma key. This proved to be an effective way of ensuring a high volume of intensity data survived intact and the contrasting background could be removed. The high volume of green cells in the typical bayer pattern (1976) of a CCD sensor results in a high proportion of edge detail of the image being represented by edge pixels in the image. The resultant image can be thresholded with a low value static threshold (intensity value of 1). Correlating pixels within the resultant binary image I1 and the intensity image I2 reveals the textural properties of individual grains. This is somewhat computationally more efficient than the use of the adaptive threshold and proved effective in later research during this project. Otsu and the Adaptive threshold can however be employed for calibration in order to find an optimal threshold value, which can be reused for several tests within the imaging environment. Morphological operations can also be employed to eliminate any small irregularities in object boundaries or interiors. Figure 4.4 shows an original image and image I1. Figure 4.5 shows the final image produced combining I1 and I2.

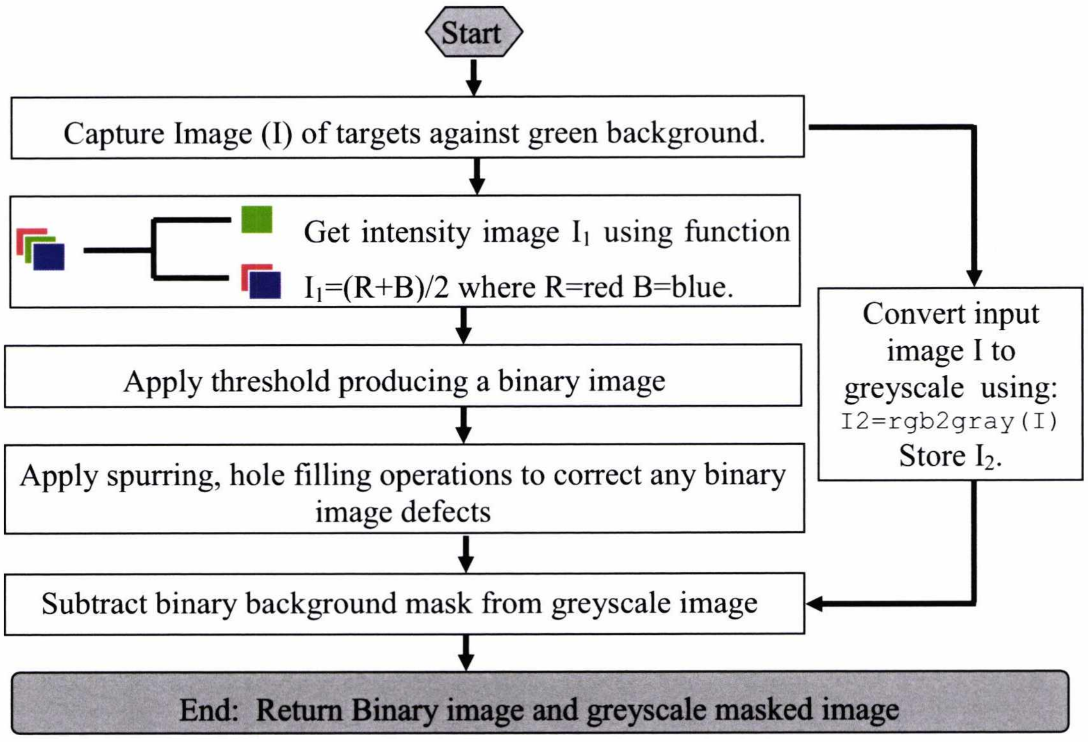

Start
Capture Image (I) of targets against green background.
Get intensity image I_1 using function I_1=(R+B)/2 where R=red B=blue.
Apply threshold producing a binary image
Apply spurring, hole filling operations to correct any binary image defects
Subtract binary background mask from greyscale image
End: Return Binary image and greyscale masked image
Convert input image I to greyscale using: I2=rgb2gray(I) Store I_2.

Figure 4.3 Image Process for creating images.

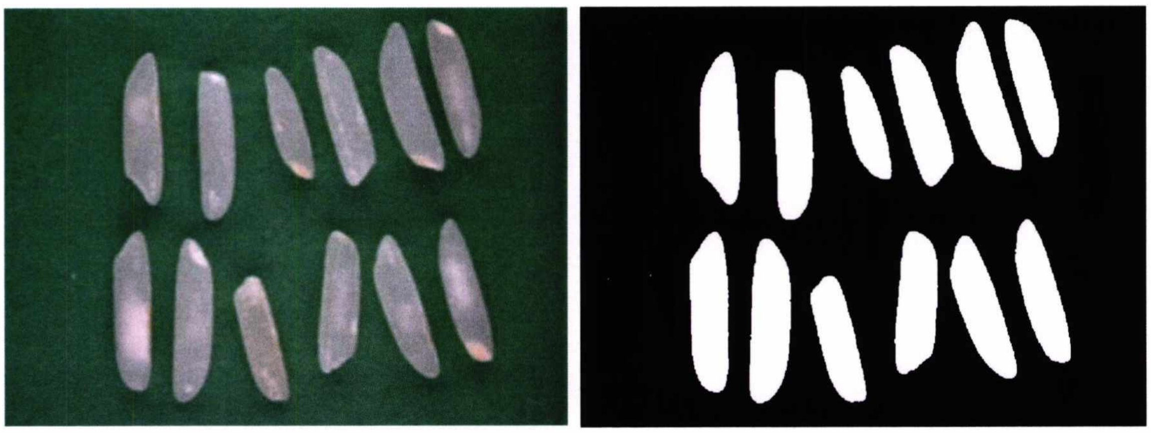


Figure 4.4. Original Image and Mask.

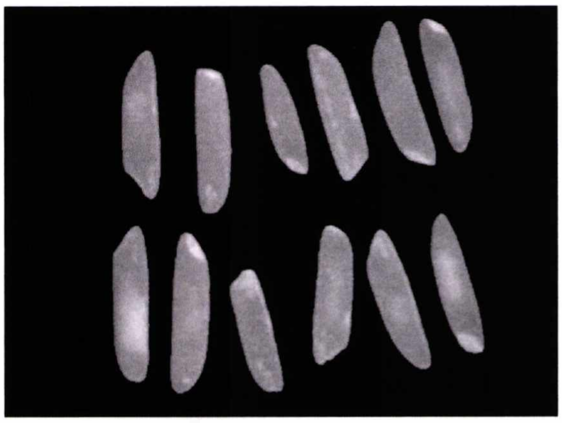


Figure 4.5. Area correlating between I_1 and I_2.

4.3.4 Edge Detection

Canny edge detection was employed in research on rice grain images. Edge detection is a complex underlying image process and requires expansion. To describe how this process works, a brief summary of the principles of edge detection are given leading into how the Sobel method and its offshoot the Canny edge process works. Edge detection is a powerful method for detecting discontinuity in an image. This is particularly apparent in regions where there are significant changes in gradient, for instance, in a rough surface of a texture, or in an 'edge'. Literature describes (Gonzalez, Woods and Eddins 2004) this process completely which will be outlined here. The implementation of this function was carried out in Matlab, yet an understanding of this function is critical for implementation in hardware or in a stand alone image processing application. To this end, the sobel and canny edge detectors will be described.

4.3.4.1 Sobel Edge Detector

Edges are detected in a discrete two-dimensional image through first and second order derivatives, or gradient. The gradient is determined through a monadic process – each pixel (cell) within an image (vector) is examined with a local mask (kernel), that permits the application of a function with that pixel and its NxN neighbours, dependent on the mask size. Typical edge detection operations can apply two of these masks on the local 3x3 neighbourhood, which are in turn used for convolution operations. In the case of the frequently described (Gonzales and Woods 2004) and widely used sobel operation, which is simple similar function to the canny operation, two masks (23, 24) are used for each pixel considering x and y coordinates:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \tag{4.23}$$

$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{4.24}$$

Where A is the source image. Similar techniques such as the Prewitt edge detector use the same approach below but with different masks. The objective here is to use the values of Gx and Gy to produce a gradient magnitude, which can be provided by :

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{4.25}$$

Giving G as the gradient. For faster computation, the following is also acceptable (26):

$$|G| = |Gx| + |Gy| \tag{4.26}$$

The values of Gx and Gy are derived from the two masks applied to the image, (4.25) for x and (25) for y. The choices for each neighbour pixel, P, in the operator are given as in figure 4.6:

$$\begin{bmatrix} P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 \\ P_7 & P_8 & P_9 \end{bmatrix}$$

Figure 4.6. Local mask for Pixel set P

Which can be applied directly using the mask in figure 4.6 expanded in (29) to implement the sobel edge detection operation. This uses equations (23,24) and either of (25,26) with the kernel in 4.6 as (27, 28):

$$|G| = \sqrt{[G_x^2 + G_y^2]} \tag{4.27}$$

$$|G| = \sqrt{|(P_1 + 2P_2 + P_3) - (P_7 + 2P_8 + P_9)|^2 + |(P_3 + 2P_6 + P_9) - (P_1 + 2P_4 + P_7)|^2} \tag{4.28}$$

This gives the gradient for horizontal and vertical texture, combined, giving the complete texture for an image. Members of G can also be vetted by a threshold in order to look for edges of specific intensity. Simple versions can omit the threshold, however this makes the approach extremely insensitive. This coupled with the 3x3 neighbourhood of the sobel operation makes it very sensitive to single points in the image, and noise. However this will not be a significant issue in this case due to further processing described later in this chapter.

4.3.4.2 Canny Edge Detector

The canny edge detection operation extends sobel by incorporating additional steps before and after the derivation of gradient from the masks. First, a convolution operation is applied to the entire image to smooth it. This removes individual points and leaves only the most varied regions as distinct, rather than single pixels. Next the gradient G is determined, but also the gradient direction for each point. The normal sobel operation produces one output image of gradient magnitude. This stage for canny produces two, one for the magnitude and one for the edge direction. Edge direction (angle of maximum rate of change, α) is determined as follows:

$$G_{\alpha} = \tan^{-1}(\frac{G_x}{G_y}) \tag{4.29}$$

Using the sum of Gx and Gy. The angle is approximated given the discrete nature of the image and the 3x3 neighbourhood of the mask down to 45 degree intervals.

The next step is to use the magnitude and direction images together to determine if neighbouring points in line with the angle are of a similar magnitude. If this is the case, this point and its neighbours will likely be edge pixels and stored in the output image as such. This process is performed with two thresholds – a higher and lower threshold, either specified at the start of the canny operation, or determined automatically. The automatic threshold of matlab's integral canny edge function was used for this task. The higher threshold is used for detecting the initial point in the gradient data, and the lower threshold is for comparision with its neighbour in the given gradient direction. Assuming both points pass the threshold, the point is considered an edge pixel. This process is known as *nonmaximal suppression.* The implementation of the canny edge detector found in Matlab can automatically determine the higher threshold point T1 and allocate the second point T2=T1*0.4 Points that are greater than the first threshold are 'strong' points. Points that are adjacent to strong points –and- are greater than the second threshold are 'weak' points. Both of these types of points are accepted as edge pixels by the edge detection operation. The result is they are represented as binary *true* in an output binary image. An example is shown later, with additional processing in section 4.5.

Implementation of edge detection in research used the canny operation as its implementation during experimentation returned a significant contrast between the brown and white grain varieties. The use of different edge detection routines was investigated further in related work (Hobson, Carter and Yan 2007), and results from rice imaging with the texture feature are given in chapter 6.

4.3.5 Morphological Operations

A range of additional processes are required to complete the processing of the rice grain images. This refers to mathematical morphology which is particularly applicable in discrete vectors and is excellent for image processing operations. The morphological operations in this chapter are well defined in various image processing sources, (Davies 1990, Gonzales and Woods 2002). Application of these methods on binary images can correct minor defects in poorly thresholded shapes, detect interior holes within connected components, and remove small elements of noise from images. Several morphological techniques were employed in various capacities for these reasons and a number of additional reasons described later on in relation to more advanced image processing. The following processes were employed successfully in order to return relevant results for research:

- Erosion (debris removal and boundary adjustment)
- Spurring (irregular boundary correction)
- Connected Component Analysis (object detection)
- Filling Holes (image arithmetic)
- Image Arithmetic operations
- Distance Transformations
- Segmentation based on Distance Transformations (watershed transform)

These approaches have variants which can be considered binary or greyscale morphology. Some applications are for slight adjustment of small image defects and others for use in extraction of key information in the image space for a specific task. Experimentation in image processing led to the use of only these features in applied research, although a wider range of morphological techniques exist which are applicable in similar image processing tasks.

The basics of morphological operations extend from Set Theory, applied to the set of integers denoted Z. A two dimensional digital image is composed of members of Z and is represented by a function f(x,y) for its dimensions. For a specific image a set of pixels (an image area) can be represented as A, and a single pixels (x,y) as w:

$$w \in A \text{ OR } w \notin A \tag{4.30}$$

Denoting w as a member or not a member of A dependent on the criteria for a pixel to be part of a region.

4.3.5.1 Erosion

In a binary image, erosion (and similarly dilation) operations refer to an intersection between member pixels of a certain type. This is between the regions of interest in the source image (A), and structuring elements through which the operation is applied (B). In the case of erosion, this is only applied when the member pixels of the structuring element intersect with the image area A. This denotes the following pixels:

$$A \& B = \{w \mid w \in A, w \in B\} \tag{4.31}$$

For the purpose here all that is needed is an understanding that the central point of the structuring element must be a member of the set of pixels to be eroded. For this to be true, at least ONE pixel must be a member of the same set as the structuring element. This is shown in Figure 3.27, and denoted in set theory as:

$$A \oplus B = \{z \mid (B)_z \cap A != \emptyset\} \tag{4.32}$$

The erosion operation requires that the structuring element to be able to physically fit into a specific pixel within the shape. If that pixel and its respective neighbours are the same value as the structuring element, then that pixel is marked to be kept in a new output image. This output image is initialised blank, then populated with the valid pixels as an algorithm carrying out this analysis is run. Figure 4.6 shows a simple binary image and a structuring element on the left of the image. In the structuring element the marked central pixel is the 'origin'. This is the point in the original image that can be preserved in the output if the structuring element entirely fits in the image at that location. In the image on the left, the structuring element applies to the speckled pixels in the lower image as members of the set of pixels which can fit the structuring element based on their neighbour pixels. The result of erosion is shown next on the right, with only the pixels that are valid 'origin' points according to the chosen structuring element kept. On the far right, the result of applying erosion again on the resultant image with the same structuring element is given. In this case only two pixels can fit the structuring element, and all others are discarded as they are not members of the erosion set. This can provide a highly expedient means to remove small noisy objects and artefacts from an image, or boundary detection when used in conjunction with image arithmetic. The effect of multiple applications of the small structuring element can also be repeated to an extent with a modified larger element.

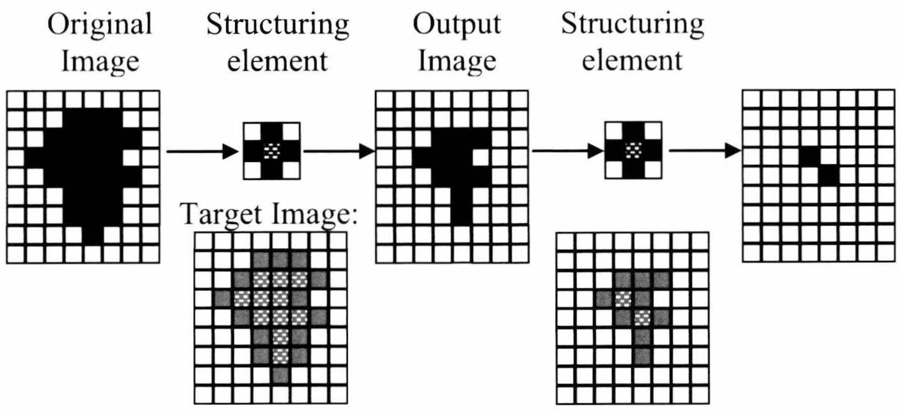


Figure 4.6. Example shape and structuring element for erosion operation.

4.3.5.2 Spurring

The spurring operation is performed on a specific subset of foreground pixels with a distinct local neighbourhood illustrated in Figure 4.7. This applies to any pixel with less than two immediate neighbours.

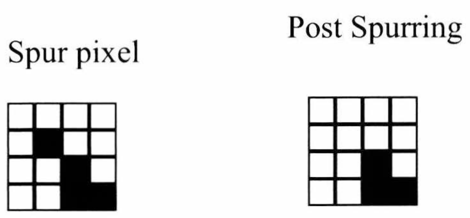


Figure 4.7. Spurring of a binary image.

This can be applied quickly with an algorithm that counts the number of foreground pixels in the 3x3 neighbourhood of the current foreground pixel. If a pixel has less than two neighbours it is a spur or a single speckle of noise. Such pixels are eliminated from the image in one pass. An example within the Appendix illustrates the coding for this function which will scan the image an additional time after spur detection in case some elongated spurs exist. The function can be set to repeatedly apply as long as spur pixels exist, or conclude after repeatedly evaluation the image after a set number of iterations.

The presence of spurs can also cause errors in shape measurement due to impact on boundary length measurement. Essentially spurs are isolated pixels on the boundary of the shape that are sufficiently irregular to offset the shape boundary measurement. It is possible these could be noisy pixels or small debris in contact with the shape boundary. While erosion can eliminate these pixels, the general application of the erosion operation can be excessively damaging to the shape boundary. Separate application of spurring 'cleans' the shape boundary in advance of other analysis such as boundary measurements of shape. By eliminating these points the shape is simplified with a minimal loss of shape information.

4.3.5.3 Connected Component Analysis

Connected Component Analysis (CCA) reflects a method to detect individual objects within an image. This approach involves counting the number of member pixels in a connected region with a simple counting algorithm in two passes. The process begins with the scanning of an image pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel foreground regions (binary 1). In practice Connected component algorithms can also function on grey level images and different measures of connectivity can be implemented, however, for the following we assume binary input images and 8-connectivity. The algorithm functions through scanning the image by moving along a row until it comes to a point p (where p denotes the pixel to be labelled at any stage in the scanning process) where $p>0$. At this point the four neighbours of p which have already been encountered in the scan that are members of the masks in figure 3.30. These are scanned starting at $x1$ and in order to $x4$. Based on this information, the labelling of p occurs as follows:

(1). If all neighbour pixels of $p=0$

```
        Increment the counter and assign to p.
(2). If one neighbour pixel in the subset correlating with mask 1 of p>0
        Assign its label to p.
(3). If one or more neighbour pixel in the subset correlating with mask 1 of p>0:
        Assign new label to p and Add it and all neighbours to a new list of
        'equivalence class' lists e. If any neighbours are already in one of the list of
        lists, add the points to that sub list.
(4). Upon reaching the last pixel, repeat steps 2 and 3 working backwards with mask 2.
```

Finally a second iteration of the algorithm looks at each list of points in each unique equivalence class. Through scanning each pixel row by row and column by column, each separately labelled pixel of each equivalence class is assigned the same value. The result of this is each shape being individually labelled with a low number relative to the order in which the object was scanned and the total number of objects.

A unique number then represents each connected component. This permits individual measurement of each object within the image with its respective connected component number. An algorithm was implemented in earlier work in C++ following similar principals, varying only in that the equivalence labelling stage was replaced with a simpler setting of all values in the same connected region to the value of the first point of that region to be scanned. This does result in the labelling possessing higher values than with the equivalence classes, yet each connected region will be identified with a unique integer. Code for this function can be found in the appendix. Figure 4.9 Illustrates a source binary image where -1 represents the set of pixels of interest. The same region of an image Is shown three times, at three stages in the algorithm. The initial state is state 1. The first iteration applies using mask 1, comparing neighbour pixels in the order stated, and always beginning with a pixel with a neighbourhood like Pixel 1 and producing an image as shown in state 2. The second iteration takes the result from state 2 and operates in reverse with mask 2, reallocating the high number pixels to the value of their lower neighbours. The final output is shown as state 3.

1. Start

0	0	0	0	0	0	0	0	0	0
0	1	-1	-1	0	-1	-1	-1	-1	0
0	-1	-1	0	0	0	-1	-1	-1	0
0	-1	0	0	-1	0	0	0	-1	0
0	0	-1	-1	0	-1	0	-1	0	0
0	0	0	0	0	0	0	0	0	0

2. First Iteration

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	2	2	2	2	0
0	1	1	0	0	0	2	2	2	0
0	1	0	0	4	0	0	0	5	0
0	0	1	1	0	4	0	5	0	0
0	0	0	0	0	0	0	0	0	0

Pixel 1

0	0	0
0	1	-
0	-	-

Mask 1

x3	x2	x1
x4	p	x8
x5	x6	x7

Mask 2

x7	x6	x5
x8	p	x4
x1	x2	x3

3. Second Iteration

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	2	2	2	2	0
0	1	1	0	0	0	2	2	2	0
0	1	0	0	1	0	0	0	2	0
0	0	1	1	0	1	0	2	0	0
0	0	0	0	0	0	0	0	0	0

Figure 4.9. Examples for connected component algorithm.

4.3.5.4 Filling Holes with image inversions

Filling holes is important as it a way to distinguish background n a binary image can come in two forms. As a local morphological operation it can be employed to fill in single pixel holes with a 3x3 mask, similar to the process for spurring. The code example given in Figure N can achieve this function. Alternatively a combination of connected component analysis, inversion, and image arithmetic can be employed. In the type of imaging scenario considered here, the limited number of objects in the image expected to appear results in the largest connected component region in the image being the image background. Such regions can be ignored by ignoring any region that is in contact with the edge pixels of the image. This can be expressed as a member pixel *w* in a region *A*, where one of the *x* or *y* coordinates of *w* are zero, or the width-1 and height-1 of the image respectively. The filling of holes larger than a single pixel can be conducted by applying arithmetic operations upon the image following the detection of the background components. This is illustrated in part in Figure 4.10. The source image, designated *I1* is duplicated producing image *I2* and subjected to a hole filling image processing operation and an image inversion. The hole filling operation itself involves the further application of both connected component analysis and an inversion operation. This process identifies the large background of the image *I2* and sets any other separate components equal to the inverted set of objects, erasing them, then inverting the resultant image. This is finally added to Image I2 and the resultant image is inverted again. Matlab Code for the separate Filling, Inversion and Addition processes can be found within the appendix.

In Figure 4.10 the functions can be employed to retrieve all the holes within objects within an image. Part of this function uses image arithmetic which is elaborated upon in the next section. By applying a function in which CCA is applied followed by a threshold to retrieve a single connected component, it is possible to find the holes for individual components. These can be individually measured in the same capacity as the entire shape.

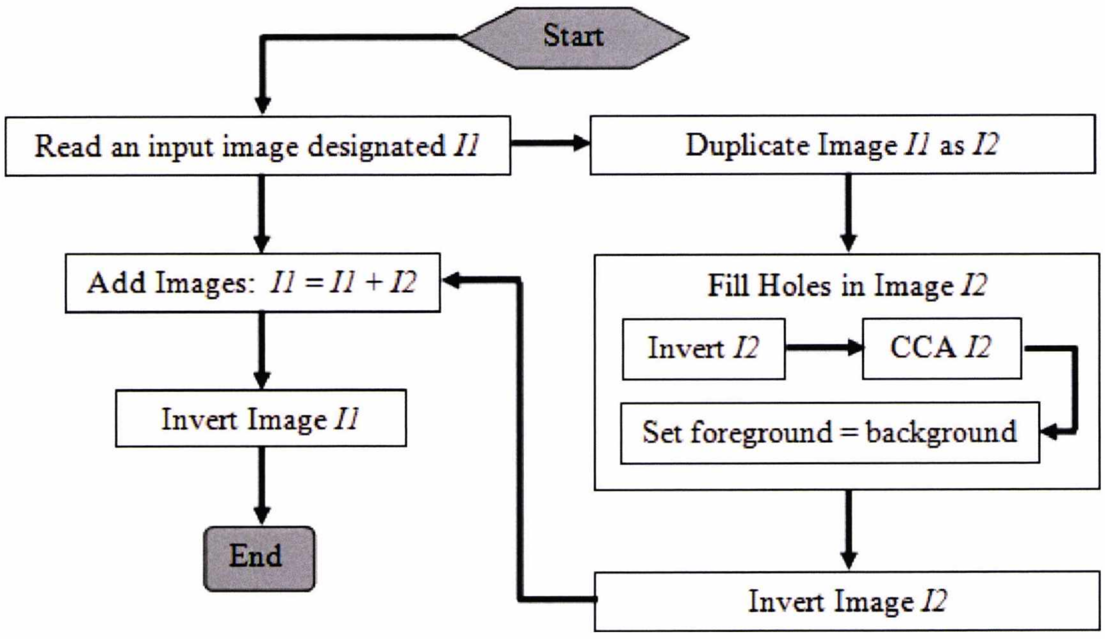


Figure 4.10 Stage of process for returning an image of the interior holes of an object.

4.3.5.5 Image Arithmetic

Image processing employed can involve some manner of arithmetic operations, such as masking the image to focus solely on specific regions of interest. This is usually in the form of a logical operation involving processed binary images and source images from hardware. In terms of mathematics, this can be expressed as an arithmetic matrix operation between two matrices. In this sense two images of the same dimension, can added, subtracted, multiplied and so on. Logical images of thresholded objects in an image could be used as a mask for representing the textural information for a specific object within a greyscale or colour image. An example of this is given in Figure 4.10 as the *Fill Holes* process. It is also illustrated in in 4.11 In which image (left) is added to the inverted hole-filled image (middle) producing an image containing the hole as the foreground object (right). This creates a completely filled image with the exception of the holes within objects in an image. Image arithmetic can be employed in a similar capacity with other operators such as subtraction. The image is inverted, then connected component analysis independently labels background and internal regions. The background is identified due to its contact with the image edge and can be subtracted, leaving only target regions which can then be subject to shape analysis.

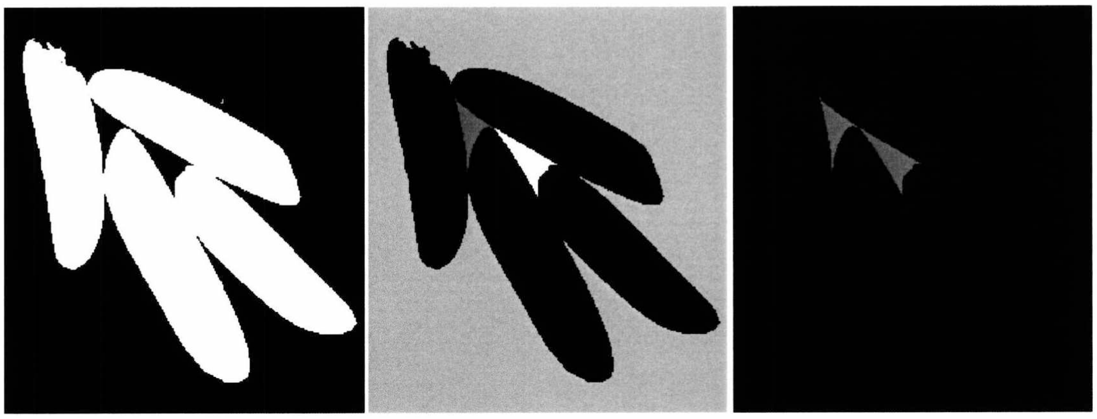


Figure 4.11. Arithmetic operation to detect inner holes.

4.3.5.6 Distance Transformation for use in Segmentation

A binary distance transformation is useful in image segmentation functions. Examples of applying two different types of distance transformation on a binary image are given in Figure 4.12. The distance transform is shown as a topographic map of the original image, where intensity increases proportional to the value of each pixel. The distribution and range of distance values is dependent on the choice of distance function employed. Several possible functions for this purpose include *Manhattan*, *Euclidean* and *Queen's* distance (which will be addressed in the next chapter when covering distance functions). However a simple implementation can also be implemented as shown on the right in figure

4.12, with code provided in the appendix. The code for the matlab implementation given in the middle is closed source, although a C++ implementation is freely available (Felzenszwalb and Huttenlocher 2004). This transformation works by converting the foreground pixels of the binary image into a numerically increasing set of pixels dependent on the distance of the pixel in question from the background set. Pixels that are adjacent to the background set, either 4-or 8-connected varying according to connection approach are set to equal 1 or sqrt(2) dependent on distance function employed. Proportionally pixels that are on the interior of shape regions and are not on the shape boundary are set to equal their distance to the shape boundary. In Figure 4.12, matlab's double precision (diagonal distances = sqrt(2)) distance function is compared to simpler integer (all distances =1) only version as functions A and B respectively. The legend beneath each set represents the distance value as a topographic map, with higher distances being represented by brighter pixels. Here decisions can be over wherever to compromise accuracy for ease and speed. For the purposes of research the matlab function was used, although this alternative was considered sufficient for the task.

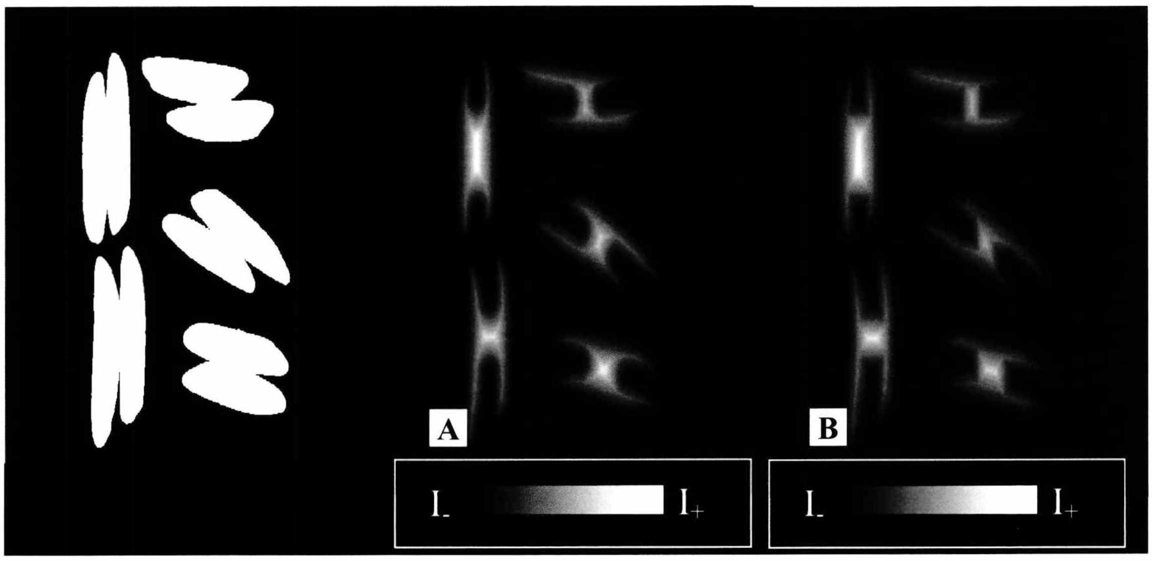


Figure 4.12. Binary Image (left) and Distance functions A (middle) and B (right)

The criteria for the distance function algorithm in the appendix considers a pixel to be an edge pixel if it has a pixel equal to zero in its 3x3 neighbourhood. If this is true a respective output image 'map' which is initialised as blank has the same pixel location incremented to the current layer value. The pixel is removed from the 'out' image. Once all pixels in the 'input' image are assessed the 'input' is set to the updated image 'output', and the layer counter is increased. The outer layers of the input image are stripped away for n iterations, where n is the maximum distance in pixels between background and foreground pixels. This process is directly applicable in the watershed segmentation function.

4.3.5.7 Watershed Segmentation

In this research, watershed is one of two methods considered for the segmentation of physically touching grains. This is different from the previous threshold function which separated irrelevant background information from the objects of interest. In this section is presented the simple morphological watershed transformation adapted based on existing research. The second method implemented requires an analysis of shape curvature and convexity, consequently that approach is described in chapter 5 due to its requirements of features discussed in that chapter. The watershed approach is based upon the mass of separate connected components and their respective centres of gravity determined through a distance transformation.

This segmentation is applied by filling the 'shallow' regions of a distance transformed image (the regions of that image equal to zero) and gradually stripping away separate pixels until only a skeleton remains. This appears similar to a diagram of a two-dimensional Voronoi tessellation (a space of discrete sets based on distance between chosen points), where the skeleton represents the boundary separating the different points. The process of generating this new image of segmenting lines is as follows. Imagining that the most distant points are the 'deepest' pits, holes, or valleys, these points are all simultaneously filled with 'water'. This is performed by creating a new image by thresholding the inverse of the lowest level of distance (0) in the image of the scene, and performing connected component analysis to label each resultant region. Each resultant component is interpreted to represent a 'separate body of water'. In this image all points of the same depth are filled with water one level at a time, until the point at which separate bodies of water converge. Pixels neighbouring to two separate connected components are preserved and never overflowed. This results in a set of pixels in the output image becoming a border separating the two bodies of water, or a morphological skeleton/medial axis. This set can be subtracted from the original image, resulting in segmentation of the source scene.

The typical watershed segmentation function approach has already been shown to be ineffective by past research, although it has been successfully improved and extended through the use of internal markers. Figure 4.13 shows binary images of rice grains with segmentation applied using different settings for the derivation of internal markers. The internal markers are determined through the application of statistical and image arithmetic functions on the morphological distance transform and binary image combined with the watershed image. A new image, dubbed an 'internal marker' image is derived from the distance function. This is a new binary image representing a set of points of specified distance. What matters is precisely how this set of points is derived. A simple method would be all points of equal or greater value than the mean value of of points greater than zero. Effectively, the mean distance. This can be modified with variations on this value, mean +/- standard deviation or an arbitrary distance value determined elsewhere. Four variations of the distance are given in Figure 4.14. In all cases, no ideal segmentation is found.

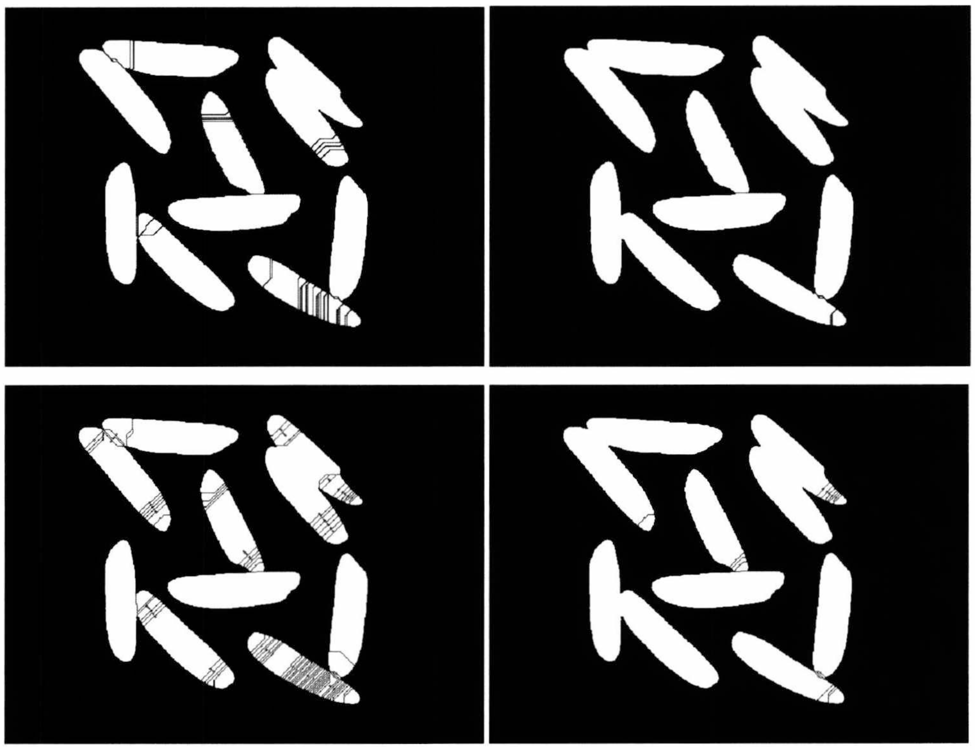


Figure 4.13. Variations of watershed segmentation with different internal markers.

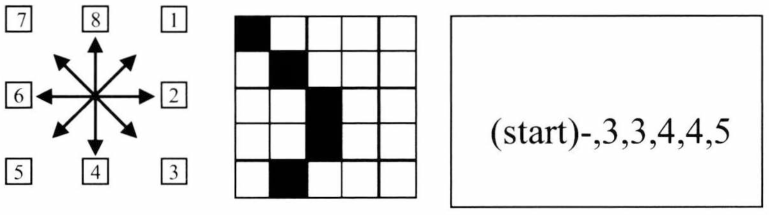
In Figure 413, the black lines represents where segmentation has occurred. Notice that the lines penetrate the interior of the grain extensively regardless of parameter setting. The grain shape is too elongated and irregular for this approach in its present state. In some cases, there is undersegmentation where the influence of the internal markers causes a failure of segmentation. However in most cases there is oversegmentation where the internal markers are not effective at preventing invalid segmentations.

4.4 Boundary Tracking with Chain Coding

The next stage upon identifying each individual segmented grain object is to identify one of its most descriptive properties – the boundary of the grain. This region can be recorded and used within shape analysis in a number of ways described in the next chapter. This approach is employed to record the boundary of a shape in terms of connectivity. This provides a complete record of the shape boundary in two possible formats. First, a single coordinate coupled with a 'code' of neighbour pixels, which can provide efficient storage. Second, a complete list of boundary coordinates (x,y) for each boundary pixel.

In practice, this was achieved in two ways. In later work using the support of matlab its function library was adopted due to convenience. A matlab function, *regionprops*, returns area, boundary, perimeter length and a wide range of other useful shape properties automatically from all individual connected components within a processed binary image. However In early work a dedicated C++ algorithm was developed in order to record the shape boundary coordinates in a structure representing each point in the shape. This function has the potential to be employed in a rapid processing manner and is now described as it may be of use in future work and applicable in a dedicated system. Boundary pixels were individually noted with a bit indicating true/false wherever they constituted an edge pixel. The number of boundary pixels could then be determined and a preallocated pointer could be created for storing all boundary pixel coordinates.

Such an algorithm is employed to iteratively map adjacent pixels, either through their coordinates or direction of connectivity. The 'code' denotes the orientation travelled from the current pixel to the next pixel, in line with the example provided in Figure 4.14 (note this is an example of diagonal connective chain coding, 4-way coding also exists). This can also permit reducing the size of information required to store the list of boundary pixels in memory limited situations, from a 2D Array of size N, where N is is the length of the shape boundary, to a 1D array of N and the starting coordinate. Using the code wheel in N.a, a boundary such as illustrated by the dark squares in N.b can be tracked from the top left most pixel to become the pixel of interest. This will always be the first pixel read in any boundary when an image is scanned row by column. On reaching this pixel, its coordinates are recorded. The algorithm looks at the adjacent pixel (starting using code 2) and looks at each neighbour in turn until it finds a new pixel. On finding a pixel, the code or boundary pixels are recorded (dependent on algorithm), and the code is reset to the opposite side of the code wheel. The scanned pixel then becomes the pixel of interest and the process repeats until there are no more pixels or the scanned pixel is the same as the first pixel scanned.



(a) (b) (c)

Figure 4.14. Chain coding. (a) Coding wheel. (b) Example perimeter. (c) Encoding.

Chain coding is an appealing approach due to its *O(n)* computational simplicity. The process can be used to record the coordinates of each boundary pixel in a list in a concise order. It is frequently used to determine coordinate lists for shape boundaries. Example implementation can be found in C++ in the appendix. The Matlab environment with the 'regionprops' function under the image processing toolbox also functions in a similar capacity, calculating the distance between adjoining pairs of pixels. No significant difference was found between Matlab's function and the custom C++ approach for sizing and in terms of pixel lengths used to gauge perimeter length (assumed either 1 or $\sqrt{2}$ for diagonal connectivity). However, these functions are both vulnerable to discontinuous boundaries. Notably if there is a single-width spur on the boundary of the shape, boundary pixels can be recorded adequately, but this information can be slightly misleading in gauging perimeter length. This is fortunately not a significant a problem with larger spherical shapes like grains, even when several grains are in contact. In addition prior application of morphological spurring will suppress such pixels.

4.5 Digitisation and aliasing of boundaries

The Digitisation of an captured image creates problems and opportunities for measurement and is particularly influential on the boundary of the binary representation of shape in this research. This is known in computer graphics as *aliasing* or the presence of 'jaggies' in shape or region boundaries. The discrete nature of a digital image as result of the finite number of pixels in the sensor imposes a physical limit which while making digital images easier to measure imposes a loss of information. This has a number of advantages which makes the domain of image processing possible, permitting discrete functions to be written which can be processed very quickly. However limited data also means limited measurements. Means to determine the impact of digitisation are very important to material imaging, as it can be indicative of the minimum size of an object for a valid shape measurement. The impact of aliasing on the power of simple shape descriptors on simple shapes is addressed with experimentation described in chapter 5.

Methods to manage the impact of aliasing include digitisation error functions that approximates the aliasing per pixel (Ziedan 2006). Schaefer's work on particle sizing error (Schaefer 2002) suggested a number of measures to impact the effect of digitisation on particle imaging, although a number of simple generalisations are applicable and effective. A common strategy is to recognise pixel connectivity in the boundary of a shape in a digital image. Straight 4-connection between two pixels can be represented by a distance of 1, yet diagonal connectivity can be represented by an approximation of the Euclidean, which is the square root of 2. The use of these values together results in an accurate representation of boundary proportional to the resolution of the image. Examples of aliasing can be found in chapter 5.

4.6 Texture features from Grain Images

Beyond the use of the boundary of the shape is the surface properties of imaged grains and particles in an image - the texture. Texture can be useful for a range of classification and identification purposes. In the case of food grains, surface texture can betray unique identifying characteristics. Simple textural descriptors can be created for specific applications. Several types of textural analysis can be considered for the assessment of these objects. A widely used texture analysis method is the evaluation of co-occurrence matrices (CMs). These matrices build up a table defining the relationship between pixels of different intensities. Different surface types produce different CM relationships. Intensity is an obvious variable in texture, yet CM's can also pick up other properties such as variations in smoothness and surface roughness. Another approach considered is edge detection and its use to identify significant intensity gradient variation as a texture feature.

Such a method believed to be novel by the author at time of writing was incorporated into the analysis of images. This method involved the determination of a ratio of edge pixels and area pixels within a shape. Edge pixels denote the sum of pixels that are identified as edges by an edge detection operation upon the greyscale image that also fall within the same area of a mask of an established shape from the mask image. Area pixels denote the area of that mask image shape. This gives a function of Ratio of *edge to area*:

$$R_e = \frac{E}{A} \tag{4.33}$$

From the mask, the area of the binary shape is derived as A, and the edge pixels within the same region in the edge detection image is E. Both E and A can be derived from individual connected components through an image arithmetic operation. A represents area which can be determined sum of each connected component. E is the edge pixels for the respective connected component. Dependent on the type of edge detection operation employed, the precise nature of E will vary and could be subject to other analysis such as through use of co-occurance matrices. Several edge detection techniques were considered for this work, however Canny was ultimately selected due to its automation and retrieval of adequate results in testing with image sets. The process for applying the edge detection routine in conjunction with obtaining the relevant mask for the relevant shape in the image is described in Figure 4.15. This requires two input images, such as the greyscale image and the Mask Image shown in Figures 4.4 and 4.5. Edge detection is applied on the greyscale image. This is then associated with each individual shape in the mask image as part of the measurement of that shape.

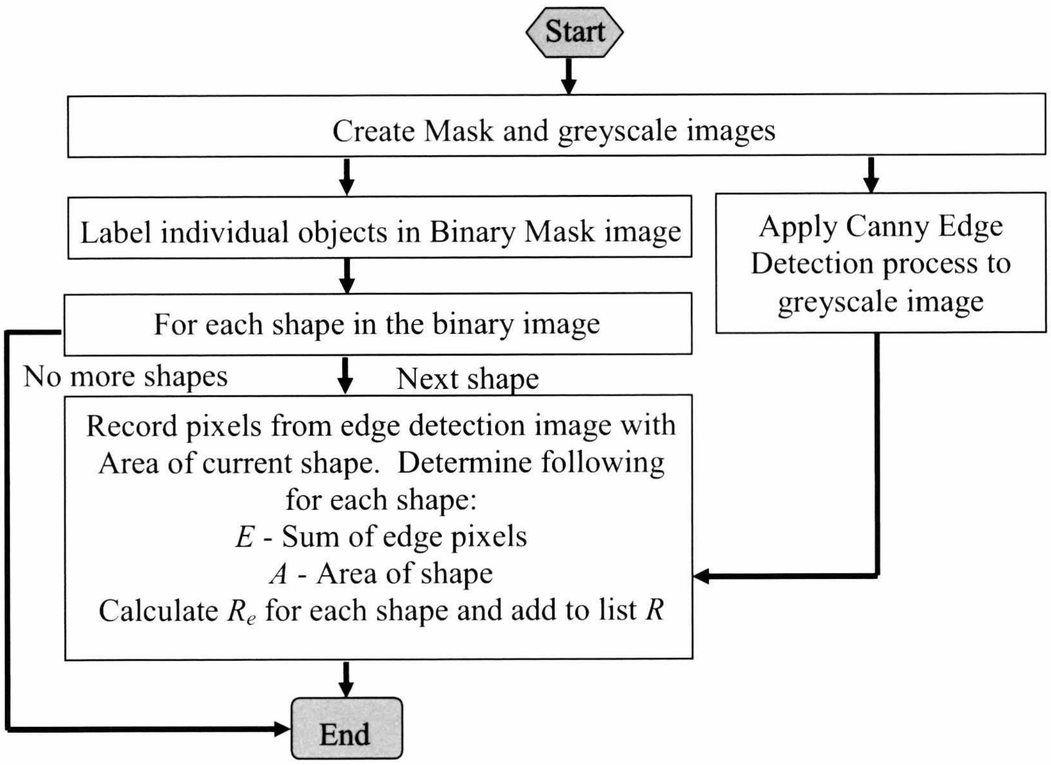

Start
Create Mask and greyscale images
Label individual objects in Binary Mask image
Apply Canny Edge Detection process to greyscale image
For each shape in the binary image
No more shapes
Next shape
Record pixels from edge detection image with Area of current shape. Determine following for each shape:
E - Sum of edge pixels
A - Area of shape
Calculate R_e for each shape and add to list R
End

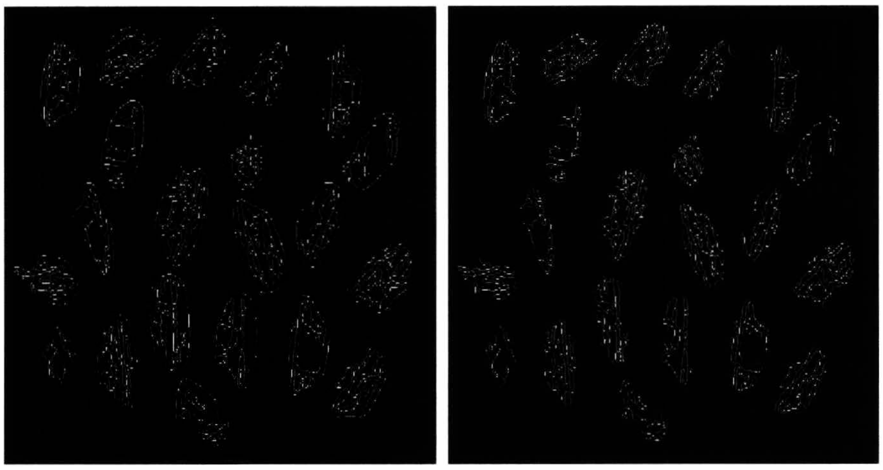
Figure 4.15. Process for determining *Re* for each shape in an image.

During this process, ratio (33) is counted using these values. For each shape processed from the binary image mask, the ratio is added to a list of values R for further analysis. In Matlab, this can be added to a structure containing other descriptive properties for each shape, such as area, mean intensity and shape properties.

Eventually as testing of the edge area ratio function were implemented, significant factors in the images that impacted results became apparent. Two significant points arose, although the precise choice of edge detection function is a third separate matter. However the focus on this section is on two other matters. Some considerations were given to the choice of edge detection function in published work (Hobson, Carter and Yan 2007). The significant points are:

- The impact of the outer edges of the grain shapes on edge ratio results.
- The automatic selection of a threshold by the canny edge detection operation.

The first issue is illustrated in Figure 4.16. This shows the result of edge detection on an image of paella rice grains. The image on the left is an outright combination of the mask image and the edge detection image. The image on the right is the same, with an additional morphological 'thinning' operation applied to the mask, shrinking it. The difference is that the 'ring' of edge pixels surrounding the image is removed. This is significant as that information is not particularly useful to measurements – likely every grain imaged with have a partial boundary at that location due to the edge of the grain shape. This is true of grains with less distinct texture, which could be resultantly harder to distinguish due to the presence of these boundary texture points. Eliminating these points altogether with a morphological erosion operation produced the result shown in Figure 4.16, with a comparison between an image without the operation on the left and one with it on the right.



(a) Original Image (b) Processed Image

Figure 4.16. Result of applying eroded mask on source canny edge Image.

This operation effectively removes the grain edge pixels, leaving only the interior edge pixels that compose the surface texture of the grain. The second issue is the decision regarding the automatic or manual selection of a threshold point for edges. There are a number of concerns that impact this process. First of all, many automatic functions can be self-contained, with no explanation of how the automatic edge detection is decided. This is particularly true of Matlab's canny edge operation. Second, the automatic function might vary in performance due to a wide range of situational variables.

One possible solution to this is to specify the threshold for the canny operation. However certain feature values will be required for this to work. Matlab's canny operation permits specifying two variables for controlling canny operations – the higher threshold value, and the standard deviation of the convolution filter applied on the image. The lower threshold value is derived as 0.4 times the higher threshold. Therefore, by creating a training set of different rice grain types, it is possible to find a threshold that can effectively distinguish brown and white rice grains more effectively. A simple way to implement this is to repeatedly apply the process in figure N3, and each time record the difference in the Edge ratio between the white and brown grains at different threshold values T. Repeating 100 iterations for threshold values 0.1 to 1 would give a reasonable indicator of a good threshold point for differentiation of the grain varieties. In addition this was used in further work beyond the scope of this thesis during the course of research.

4.7 Summary

In this chapter the image processing functions that have been developed for use in the rice grain imaging system are described as well as the overall system architecture. Using captured images with the physical setup discussed in chapter 3, Images were obtained for the development of the image processing algorithms described in this chapter. This gives processed images from which key properties of the target grains can now be retrieved. The unique way in which textural information is extracted has been described. Using the processed images generated by routines in this chapter, means to apply shape description can be implemented on the area and boundary of the grains. Scaling determined from combining methods in chapters 3 and 4 can be used for sizing. The segmentation of grains is also shown to be a challenge, with a watershed method not entirely suitable for this task. This will be discussed more in the next chapter, when shape description features derived from the processed image will be used to create a more powerful segmentation function. To achieve these goals, a wide range of image processing techniques have been employed within a system. The entire architecture of this type of system was shown first in order to provide an overview of the different processing elements. This chapter permits the retrieval of individual grains from images with segmentation, particularly through the process of thresholding. A more elaborate segmentation function to segment different types of grains was shown to be difficult to transfer to this field or research. Consequently, a more elaborate segmentation routine is described using concepts addressed in chapter 5. The dependency on the images obtained from the equipment in chapter three has however been shown to deliver images of sufficient quality for this further processing. This was achieved through adaptive thresholds and Chroma key incorporating the colour sensor and use of background removal. The first feature, the textural feature is now obtained, leaving only shape properties to be retrieved from the boundary and regional knowledge of each grain. These have been corrected with morphological operations as a means to further process minor anomalies in the grain shape and eliminate some irrelevant image information from some of the different types of analysis. The next chapter can now address the evaluation of grain shape from the recorded properties.

Chapter 5

Shape Features from Processed Grain Images

5.1 Introduction

The previous chapters addressed capture and processing of binary images representing object shapes, leaving this chapter to address analysis of the processed binary images for shape features. This chapter addresses a wide range of features derived from the boundary and regional properties of the processed grain images. Texture analysis has already been conducted through image arithmetic, conjoining source images with the binary shape images. General shape description is described here based on regional and boundary properties. Using several of the feature extraction processes described here, a final image process is also described in this chapter. The segmentation function employed to separate contacting shapes uses the curvature function which forms the basis of powerful shape description techniques. Effectively the very shape of objects that is measured is itself used to enhance image processing. This chapter leads through a description of the different features by type. Then the segmentation function is described. Next methods of describing results of feature analysis are described. Simple measurement features are considered first as these are the most immediately derivable shape features. Next General features based on measured shape properties are discussed. Finally analyses of shape signatures are considered, leading to the advanced segmentation function that is dependent on several shape features to function

5.2 Shape Features from Grain Images

First is an explanation of the context for the use of the term 'shape' in this chapter. Shape is defined by Costa and Caesar (2000) as any connected set of points. The context by which shape is defined here is from the 2D binary silhouette representing significant objects captured and extrapolated from a digital image, grains in this case. Shapes are represented by a two-dimensional discrete matrix. This gives a generalised discrete representation of a shape, which is accurate in proportion to the complexity and the number of points (image pixels) that compose the shape. In this instance these are images of grains and particle-like objects. We therefore consider approaches in shape analysis from relevant research into particle shape (Bowman, Soga and Drumond 2000) as well as grains and irregular objects (Carter and Yan 2005) among others. The source of shape properties from these objects come from the two sources mention in chapter 2 section 5 – regional and boundary properties of two dimensional shapes, which were shown there to be used in segmentation. Further research using these properties is expanded in this chapter.

Research implementing simple and complex particle shape features for measurement tasks is applicable to rice grains and features such as compactness among others are used frequently by organisations such as the Buhler group in optical rice measurement. The field is subject to continued publication of relevant survey papers influential in this research (Hentschel and Page 2003). Over time, more sophisticated shape features have been derived from general shape properties, such as the utilisation of Fourier methods which are described later as well as wavelet features applied for relevant grain measurement (Choudary 2008). In this and the next chapter, many features are applied to simple geometric shapes and to rice grains as a means to investigate their potential usefulness for shape description.

Similar classifications of shape features have been considered in this work. Past researchers have previously created classifications of shape features. These include many survey and review papers of shape features (Kindratenko 1997, 2003, Loncaric, 1997, 1998, Zhang and Lu 2004). Other researchers have attempted classification based on functions, yet the divide between functional and set theory approaches suggested by Kindratenko are similar to region and boundary (or contour) based approaches discussed earlier. Some industry standards also incorporate shape features for compression. Using reconstruction techniques based on certain shape descriptors, it is possible to reconstruct a representation of a shape, at a cost of detail dependent on the level of compression. Figure 5.1 shows a breakdown of different types of classifications according to a review of shape description techniques (Zhang and Lu 2004). It is not necessary to exhaustively review all shape description techniques in this research, although research into relevant features was a critical area of activity. It is important to clarify the precise origin of each feature type. For instance the following would apply if in research a stated goal was to quantify curvature. Without putting the definition of curvature in the correct context its purpose becomes unclear, and measurements have no meaning. The precise features used are described in the following sections relative to Figure 5.1's shape features and the classifications of other researchers.

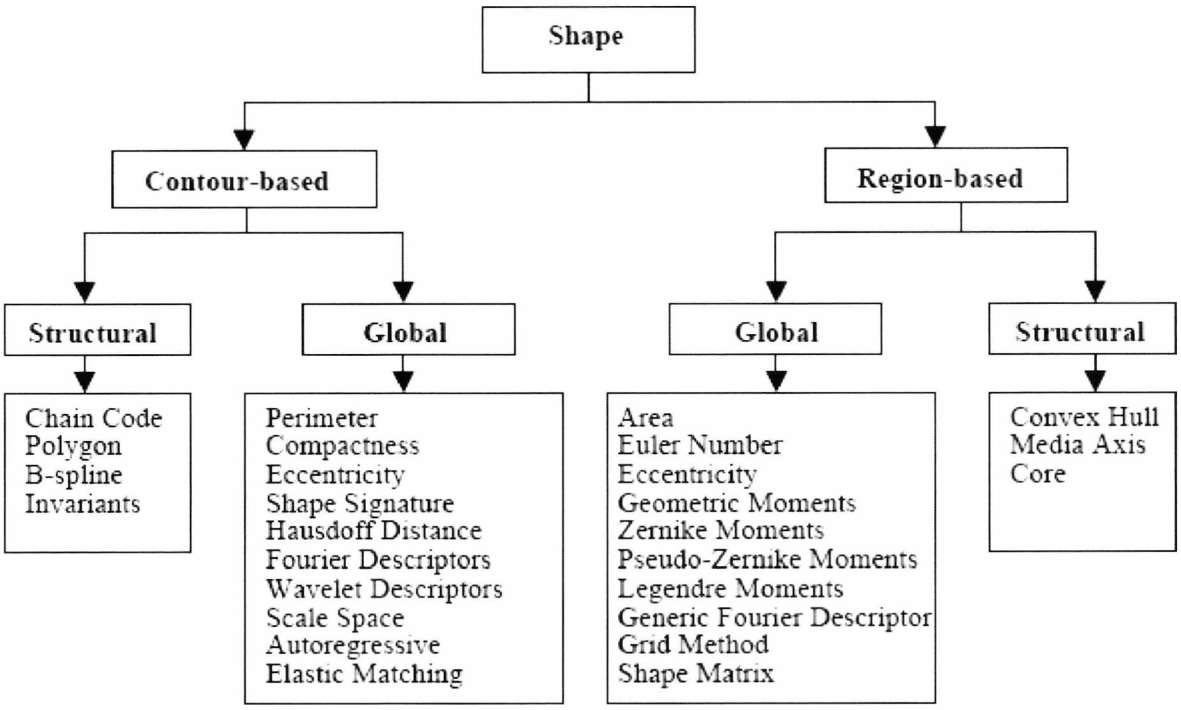


Figure 5.1. Shape feature classification (Zhang and Lu 2004)

The following features and techniques are of particular relevance to this research. *Contour* based techniques describe methods that are primarily impacted by variations in shape boundary. The chain code for example is a tracking of the boundary of the shape. *Region* based techniques relate to a region in which the shape or a part of the shape is contained. For instance, Zernike moments relate to a circular area which encapsulates part or all of the shape. This must be a circular area regardless of the shape considered, so generally this can be a circle wider than the widest point of the shape centred on the shape's centre of gravity. *Structural contour description technique*: represent ways to represent key structural elements of a shape, such as the shape boundary in a format that permits retrieval of the significant properties of the shape. Polygonal approximation is another valid method (Wang 1998) as it encapsulates a simpler representation of a complete shape. *Global contour description techniques* include single features and transforms of a shape.

Rather than storing structural information, a measurement can be directly derived from the shape. For instance the perimeter would be a quantity to represent the boundary with a quantified value rather than store it in its entirety. This is practical for efficient statistical measurements and application in instrumentation. This also extends to signatures and fourier harmonics, of which only a small subset of the elements of each may be of use in specific measurements. *Region based structural description techniques* are description processes that are actually further processing of images applied for descriptive purposes. *Region based global description techniques* list features that are used for complete shape reconstruction and offers much descriptive power. Several of the most significant types of shape descriptors will now be investigated.

5.3 Shape Descriptors

To describe the actual process of shape description first the techniques are described for retrieving shape properties from a binary image. This begins with low level structural contour and global region elements, and leads into simple shape descriptors which incorporate these descriptive shape elements to generate quantifiers. Although it should be noted as well as shape the incorporation of textural analysis is considered by Xu in conjunction with shape, and new features can also be considered from both areas. This section describes the existing simple shape features that have been considered for the assessment of rice grains, and are also used within more sophisticated descriptors:

1. *Area* is interpreted as the sum of pixels that compose a shape in a digital image. However this can refer to the complete area of a shape including the filled area of any internal holes. It is possible to retrieve the area rapidly through connected component analysis and the summation of all pixels of a particular value or set.

2. *Perimeter* refers to perimeter length. This can be derived from tracing of a shape's boundary pixels with methods like chain coding and examining their connectivity. Digitisation, particularly with regard to imaging of small shapes such as digital particles (Schaefer 2002) raises issues relevant to the digital imaging of rice grains. Accurate perimeter length of such shapes is critical, and a single method is used consistently for accurate length representation. Solutions for some of the issues

with shape boundary digitisation have been considered earlier in chapter 4. The discrete nature of pixels within an image results in aliasing, causing the digital perimeter length to not truely be equal in length to the real shape boundary of the real object. In cases where pixels are diagonally connected, the distance between these pixels is not 1. This distance is more accurately the Euclidean $\sqrt{2}$, and diagonal connectivity is counted this way to increase accuracy of perimeter length representation. This can be done through chain coding by counting the even codes as 1 and odd codes as $\sqrt{2}$. Total perimeter length is the sum of these values.

3. *Centroid* or centre of gravity of a shape is a point central dependent on the mass of the shape. This point is useful as a frame of reference for other points, such as the shape boundary, for shape length measurements. Plotting distances between the centroid and the boundary is also one of the methods to derive a shape signature discussed later. Calculation of the 2D Centroid is an x and y coordinate that is the average of the vertical distribution of the set of pixels that compromise the object for x and horizontal for y. A code example can be found in the appendix

.

5.3.1 Shape Lengths

A range of length measurements relevant to this task exist. A large number were considered, such as equivalent circular diameter, and eigenvector approaches. The chosen shape lengths were internal measurements of shape, which were ultimately chosen from the selection of methods described in this chapter. Lengths can be derived in several ways and some approaches for selecting lengths are described in figure 5.2. Shape properties can be used to measure shapes in a number of simple ways:

- The shape boundary can be derived as a list P_{List}. In this case P_{List} is the coordinates of this list of pixels are represented by x_p and y_p where length $(P_{List})=N$.
- Using x_p, y_p and the coordinate of the *centroid*, *Radius* can be derived.
- Using only x_p, and y_p diameter can be derived using the distance of $P_{List}/2$ down the list.
- Using area A of the shape or size of equivalent shapes such as a bounding box

derived from the orientation of the shape or circle of equivalent area can be derived.

The final approach (c) considers the longest and shortest external measurements. This feature is different as dependent on the precise method used it is highly rotationally variant. The methods illustrated in (a) and (b) are different from (c) in the regard they are non-perpendicular. However, perpendicular measurement can be made by a slight change in approach. For approach (a), finding the longest or shortest point as Pi, then the shortest or longest radii at location Pi-(n/4) or Pi+(n/4). For approach (b), the same is true, only considering the length between both Pi-(n/4) and Pi+(n/4). These are only a few of the methods considered in this work. A non-exhaustive list is presented in table 5.3.

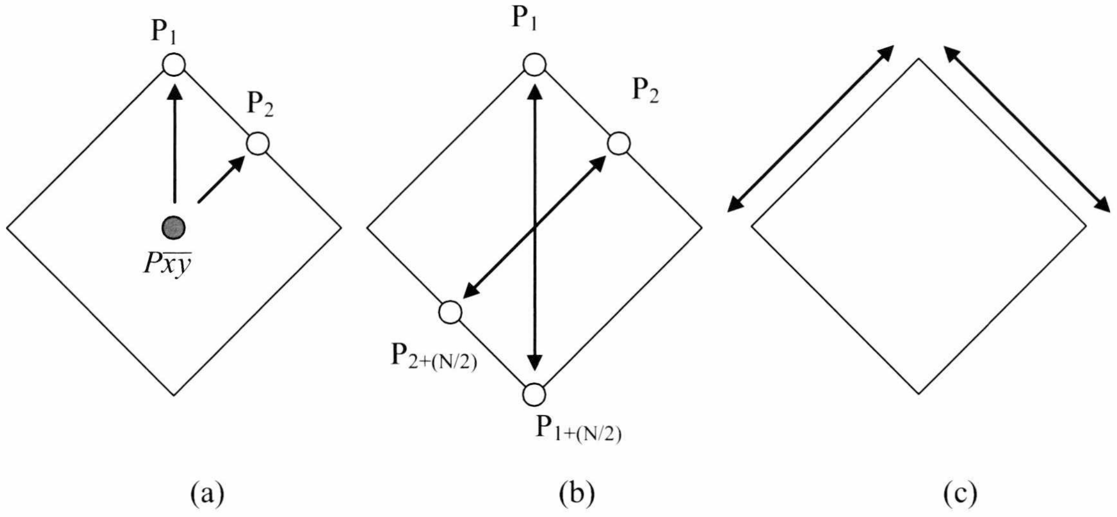


Figure 5.2. Measurement approaches. (a) Radii. (b) Diameters. (c) External.

Table 5.3 Shape length measurement approaches considered.

Measurement approach	Advantages/Disadvantages
Internal Radii, Perpendicular or Non-Perpendicular	Easy to compute, yet dependent on Centroidal location
Internal Diameter, Perpendicular or Non-Perpendicular	Easy to compute, yet reliability decreases as shape perimeters irregularities increase
Internal Longest and perpendicular Eigen Axes	Indicator of longest dimension by mass – subjectively may not be the true longest dimension
External - Equivelent circle diameter	Using Area as a global indicator of size, then assuming circularity of the shape. Inaccurate for more irregular elongated shapes.
External – rotated according to longest eigen axes, then dimensions of bounding box used.	As above, yet new rotated shape permits simple inspection by an operator for verification

Carter (Carter and Yan 2005) differentiates 'calliper' external from internal measurements based on three criteria – Intuitiveness, ease of processing, and variance management. Acknowledging these points is important for choice of measurements to employ for practical reasons in a task.

- Intuitiveness reflects simplicity of use. Simpler features are less complex and easier to evaluate for instance. Radius would be an example of a simple approach; a sophisticated external length measurement using eigenvectors to determine precise orientation to draw a bounding box is more complex.

- Ease of processing relates to requirements to obtain a feature. This is important when executing a process within a simple low-cost device of limited capability or for a potential real time application.

- There may exist a need for variance management due to small variations in a shape boundary dependent on its length which could influence measurement based on the shape interior. Variances could be wrongly assumed to be significant features. The diameter approach might suffer in cases with shapes with highly complex boundaries, and would be likely ineffective.

5.3.2 Distance Functions

Further to the forms of shape measurement are the ways of physically gauging distance in a discrete image of a shape. Several methods exist for this purpose each returning variable results which need to be accounted. Logically the same distance function should be consistently used for the same task to recover compatible results. We consider the notation for each method as follows: the distance, *c*, is determined between two sets of *x* and *y* coordinates of points *a* and *b*.

- *Euclidean* distance C_e is the first method considered:

$$Ce = \sqrt{|x_a - x_b|^2 + |y_a - y_b|^2} \tag{5.1}$$

  This gives a precise distance between two points on a two-dimensional plane. While this method of distance is accurate, computationally simpler methods exist for high speed applications.

- *Manhattan* distance C_m avoids square root and multiplication functions as a direct summation of the *x* and *y* distances between two points:

$$Cm = |a_x - b_x| + |a_y - b_y| \tag{5.2}$$

  This is a simplified generalisation of a distance function, and can dramatically overestimate the real distance.

- The *Queen*'s distance function C_q accounts diagonal distances between individual pixels. This permits a balanced approximation when movement between pixels occurs in eight directions rather than four. Between two points in a discrete 8-

directional space, the axial distance ($\delta x = | a_x - b_x |$ or $\delta y = | a_y - b_y |$) can be interpreted as the one with highest diagonal connectivity:

$$Cq = | \delta x - \delta y | + \sqrt{2} \min(\delta x, \delta y) \tag{5.3}$$

Determining the longest and shortest axis between points *a* and *b*, we reinterpret the value of the short axis distance as the square root of 2 rather than 1. Adding the two axis together produces the queen's distance *Q*. This function gives a closer although still overestimating approximation than Manhattan distance. These functions are contrasted in figure 5.4.

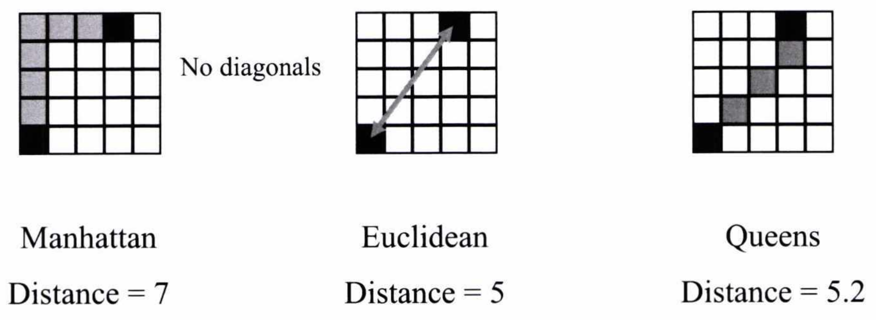


Figure 5.4 Examples of use of distance functions in a discrete plane.

The Manhattan function is computationally simple yet the accuracy of its length measurement is of limited precision. Queens's distance offers a balanced approach that is accurate proportional to the discrete nature of the data. The trigonometric Euclidean distance is the most direct distance for a Cartesian space, although it is of a slightly increased complexity in its use of powers and square roots. This can be simplified by removing the square root and using the sum of the distance instead.

Alternate approaches include the use of eigenvectors and Eigen values to determine Eigen axes of shapes. Orientation of the shape denotes the longest axis, from which centroidal and diameter distances can be derived.

5.3.3 Shape Signatures

A shape signature is a transformed discrete plot of the perimeter of the shape similar to what would be expected with polar coordinates and a continuous plot. This is a one-dimensional representation of distance between perimeter pixels and frames of reference using one of the distance functions in the earlier section. This can either be the radius to Centroid, the approximate diameter between perimeter points, or an extrapolation of shape curvature. Figure 5.6 through 5.9 represent a number of different signature plots for a star, square and circle illustrated in Figure 5.5. Dependent on the choice of signature, the square and star fluctuate consistently with their sides, while the circle is relatively constant with slight variation is due to digitisation. Euclidean was used in all approaches as it the most accurate method for deriving distance in a Cartesian space between two points. The boundary length (size) of these three shapes in individual pixels (not pixel lengths) is given in the figures for each boundary for each shape. This figure is halved for the diameter case.

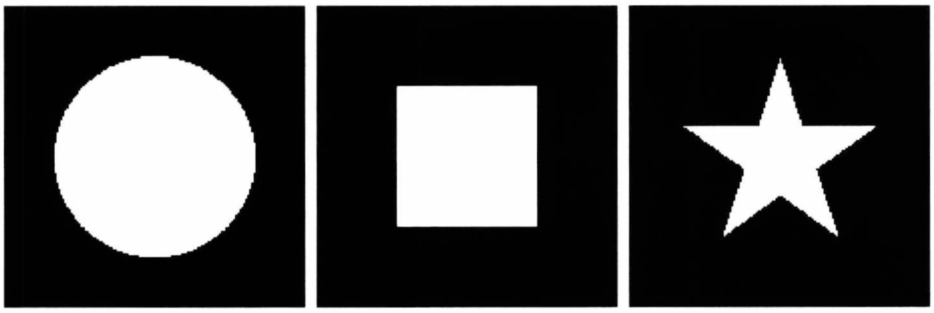
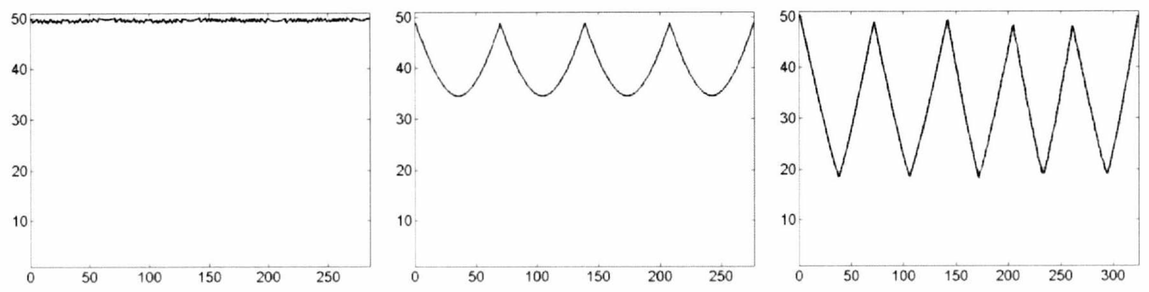


Figure 5.5. Shapes used for signature representation.

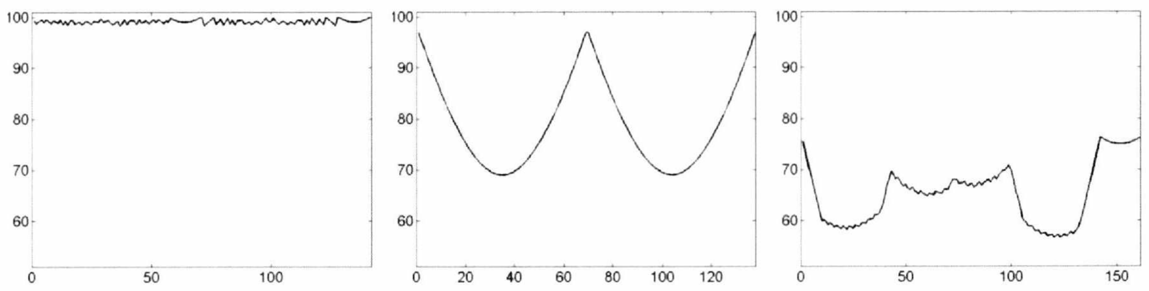
Past sources (Zhang and Lu 2004, Loncaric 1997-1998, Kindraenko 2003, Xu and Luxmoore 1997-1998) suggest that signature of a shape can be created based on functional methods. These include forms of curvature, and different types of lengths. The nature of these plots depends upon features used. Three types of signature are considered in this work. Radius, diameter and curvature derived signatures. To store *Radius* as a signature, all that is required is to record a list of all radii for each boundary pixel to the centroid. Plots for radius have been given in Figure 5.6.



(a) Circle (b) Square (c) 5-point Star

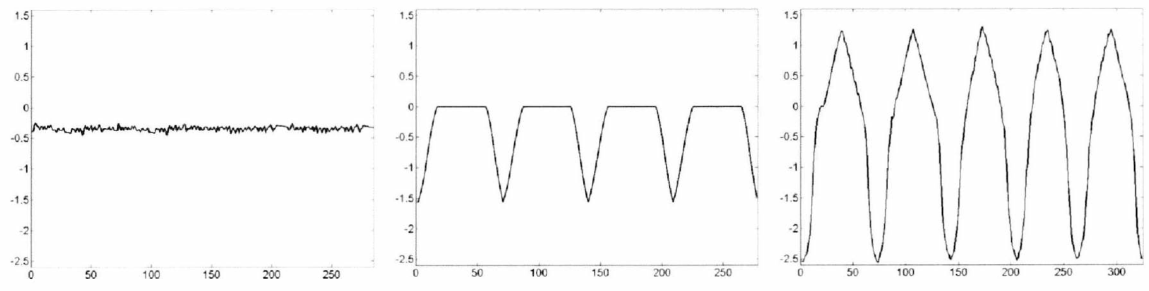
Figure 5.6. Shape radii signatures with Euclidean distance.

Diameters require the same type of approach with the difference being the choice of the frame of reference as the point roughly on the other side of the shape from the source point, and the number of points considered being halved. This is illustrated in Figure 5.7 and can be directly compared to 5.6. All plots are boundary pixel vs. distance to centroid. The remaining plots in Figures 5.8 and 5.9 address *curvature* based signature which are explained in the next section. Figure 5.8 shows the curvature plots for the same shapes, with boundary pixel plotted against angle of curvature in radians. Finally FKZ signature is given in Figure 5.9, with concaveness plotted for each boundary pixel.



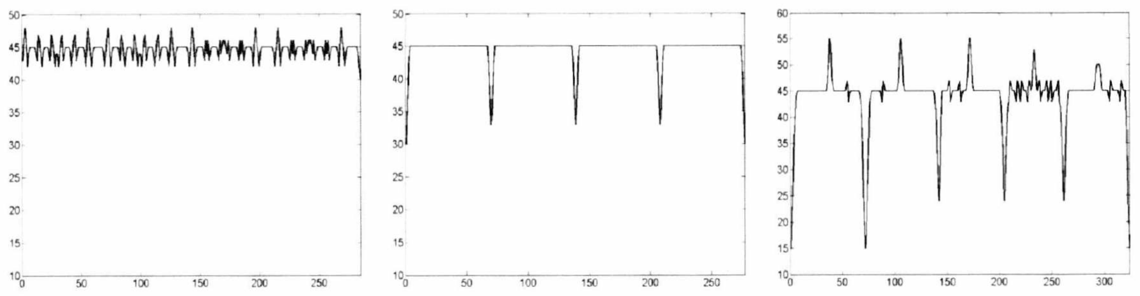
(a) Circle (b) Square (c) 5-point Star

Figure 5.7. Shape diameter signatures with Euclidean distance.



(a) Circle (b) Square (c)5-point Star

Figure 5.8. *m*-distant neighbour tangent angle curvature signatures.



(a) Circle (b) Square (c)5-point Star

Figure 5.9. FKZ curvature signatures.

In the resultant distance plots two qualities can be derived. There is a sense of type of signature created by each method and also an indication of the impact of aliasing (discussed in chapter 4.5) upon the discrete 'digital' boundary of the image. The latter is particularly visible in the circle where its regularity results in all cases the signature nearing a constant value. Slight variation is indicative of the aliasing of the circle due to the loosely approximated boundary pixels.

The key difference between the three signature types is their exact values are dependent on the structural properties of the shape. All approaches are generally effected by irregularities in the shape perimeter, however only radius is dependent on the location of the centroid. Diameter is different from radius for this reason, and the fact that the longest and shortest dimensions can potentially be different from the radii. Diameter repeats trends for simple shapes as radius, but complex shapes can return entirely different types of plot. This limits diameter's effectiveness as a form of descriptive shape signature compared to other methods, yet it is still effective enough to apply to simple shapes. Before describing curvature it must be discussed in full.

5.4 Simple Descriptors of Shape

Shape description can offer methods to characterisation of rice grains in images. In this section the functions for the simpler shape features are listed and described. Using the various properties listed, such as area, perimeter, and the shape signatures, true descriptors based on significant statistical moments. We begin with *Aspect Ratio* (R_a). This is defined as the ratio between the dimensions of a shape. However the precise specification of these dimensions is dependent on the means the shape is measured. A longest d_{max} and shortest d_{max} distance measurement can be determined and used directly in the aspect ratio function. The precise choice of this distance function depends on the method of length measurement shown in table 5.10.

Table 5.10. Choices of shape distance metrics based on method of length measurement.

Length measurement method	Source
Internal - radii	Distance between perimeter and Centroid
Internal - diameter	Distance between P_i and $P_{i+(N/2)}$ where $N/2 > i > 0$
Internal – most distant points	Calculate which two boundary pixels are most distant. Perpendicular shortest distance can be derived as the distance between the two points from around the perimeter of the shape
External	Longest and shortest Eigen axes -OR- Dimensions of bounding box of original shape, rotated to fit angle of orientation

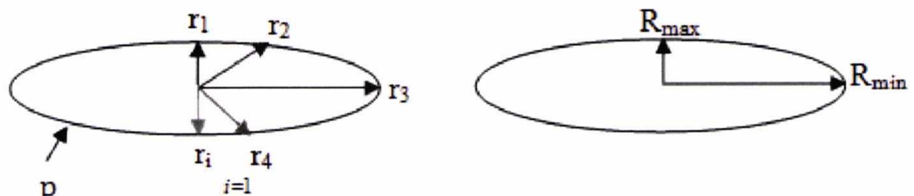
Dependent on the choice of feature slight variation occurs, which is discussed in application to theoretical shapes in the results chapter. The complexity of analysis is also dependent on the chosen feature, with functional approaches such as the eigen vectors requiring more expansion than the simpler internal approaches. Such shape features derived from simple internal length properties are now individually described.

- The *Aspect ratio* R_a functions are:

$$R_a = \frac{d_{\min}}{d_{\max}} \tag{5.4}$$

$$R_a = \frac{r_{\min}}{r_{\max}} \tag{5.5}$$

For *radius r* and *diameter d* of shapes. This gives a value between 0 and 1 for shapes of matched to imbalanced dimensions. Complexity and sensitivity to detail varies slightly with each approach. Rather than only evaluate half the shape and use double double-length radii (Carter, 2005) all possible radii are considered from the discrete boundary. Features here are still comparable to that work as feature values are highly similar. In addition, the mean diameter is integrated previous software developed for a rice analysis system at the University of Kent. This mean diameter is given in 5.11 in conjunction with the radius function, in order to illustrate the difference between aspect ratio and mean diameter approaches. Use of this will arise in a comparative study in chapter 6.



$$D_{mean} = \frac{\sum_{i=1}^{p/2} r_i}{p/2} \qquad R_a = \frac{R_{\max}}{R_{\min}}$$

Figure 5.11. Difference between Mean Diameter (D_{mean}) and Aspect Ratio (R_a) methods.

Aspect ratio is limited in descriptive ability. Use of complementary features can supplement the limitations of considering only two dimensions of a shape.

- The *Shape Factor* (S_f) is a value that generally ranges between 0 and 1, although not exclusively, defined as:

$$S_f = \frac{d_{rms.d}}{d_{mean}} \tag{5.6}$$

where distance d_{mean} of the grain is used with the root-mean-square deviation $d_{rms.d}$:

$$d_{rms.d} = \sqrt{\frac{(d_{max} - d_{mean})^2 + (d_{mean} - d_{min})^2}{2}} \tag{5.7}$$

And substituting *d* for *r* gives the radius version. The incorporation of the RMS function results in the shape factor considering the mean properties of the shape over the aspect ratio, adding a new dimension to shape analysis with a simple easy to compute feature. These features can only be easily derived from some of the approaches illustrated in table 5.10.

- *Compactness Ratio* (R_c) returns values from 0 to 1 for two-dimensional shapes that are elongated to perfectly compact (disk):

$$R_c = 4\pi \frac{A}{P^2} \tag{5.8}$$

Where *A* and *P* represent the *area* and *perimeter* length of the grain.

- *Form Factor* (F_f) defined by Horton for measurement of drainage basins of rivers (Horton 1932) uses area *A* and a measure of distance *d* to devise a metric:

$$F_f = \frac{A}{d^2} \tag{5.9}$$

Where distance *d* can be provided from a range of sources such as:

  - Mean, minimum or maximum internal diameter d_{mean}, d_{min}, d_{max}
  - Radius equivalents r_{mean}, r_{min}, $rmax$
  - External length *L*.
  - Radius of the circle of equivalent area.

- *Elongation* (Costa and Caesar 2000) is valid for internal and external length measurements of distance:

$$Elongation = \frac{d_{max}}{P} \tag{5.10}$$

Which contrasts distance d_{max} with the length of the shape perimeter *P*. These functions are dependent on the choice of distance function. Radius, Diameter or others can be used for this role.

These descriptors have been considered due to their ease of processing and implementation. Shape boundary analysis is the evaluation of a list of coordinates and measurement can be conducted in a single evaluation of the shape boundary. This suggests such shape can processed rapidly, although this can be accelerated further If necessary. There are two ways to manage this issue. First, is to reduce the rate of sampling of each such as by one sampling half the shape (Carter 2005). This has some cost of accuracy. Using a simple feature reduce the number of calculations further yet descriptive power is weakened. What is required here is a contrast of the effectiveness of these different simple parameters in order to derive useful measurements from them. The quality of shape description is evaluated in chapter 6 through application of descriptors on theoretical shapes.

5.5 Curvature analysis

The analysis of curvature used here is a compromise between what is practical given the discrete nature of the shape and a true analysis of the shape perimeter as a new space. Curvature of shape can be plotted like a signature although it involves more extrapolation than determining distance between points. Two approaches were elaborated upon in the course of research. The simplest is Fernandez's FKZ algorithm for deriving concaveness (designated C_v), resembling:

$$C_v(j) = \sum_{i=j-1}^{j+1} \sum_{x=1}^{5} \sum_{y=1}^{5} M_j \wedge B \geq threshold \qquad (5.11)$$

Where M is a mask for the ix5x5 area for each point *j* in the shape boundary and B is the source binary image of the target shape. The resultant concaveness will be higher for pixels with more neighbours and peaks in the resultant plot of C_v are vetted with a threshold. This can be set as high as the desired concaveness sort. The value of *i* used is typically 3, although this is flexible. Value *i* should not be set to a high value in the case of small shapes highly irregular shapes as this may falsely attribute background pixels on the other side of the shape as concavities. Figure 5.9 shows FKZ plots for a star, square and circle. These plot are also effected by aliasing, and vary according to regions of significant concavity. Five peaks for the star exist around the 50 point, suggesting a threshold for a star of that size and concavity will identify the 5 inner points of the star, and a low threshold less than 25 will identify the five tips of the star. 4 low points are visible on the plot for the square denoting the four convex corners, and the circle is relatively constant like in other plots.

5.5.1 Curvature with Archtangent

Tangent angle curvature functions have previously been used for particle assessment (Xu, Luxmoore and Deravi 1997, 1998) and are discussed elsewhere for similar 2D shape image analysis tasks. This method uses the chain coded list of pixels to create a new plot of shape curvature. This list of points θ is N pixels long, the length of the chain coded list of perimeter pixels P_{List}, where the coordinates in the list are given as x and y. Each point θ is defined by the following functions:

$$\theta_{j-m} = \tan^{-1} \frac{(y_j - y_{j-m})}{(x_j - x_{j-m})} \tag{5.12}$$

$$\theta_{j+m} = \tan^{-1} \frac{(y_{j+m} - y_j)}{(x_{j+m} - x_j)} \tag{5.13}$$

These two functions represent tangent-angles θ between points θj, $\theta j\text{-}m$ and $\theta j\text{+}m$, where j is the current entry in vectors x and y. Note, when deployed as a function on a machine as a discrete array representation this function will point value m to the other end of the array if m>N or m<0. This represents a smoothing factor critical to this function. An arbitrary number of pixels distance is needed to be specified for sought curves. Different values of m will return different results. Ultimately, the value of each point θ is determined as:

$$\theta_j = \theta_{j+m} - \theta_{j-m} \tag{5.14}$$

This Gives the *jth* point of θ to be the difference in curvature between the two points m places up and down the list. At present, we examine resultant plots of curvature. Plotting θ reveals curvature along the shape perimeter with respect to m, and is indicative of significant curvature. The relationship between these functions and the initial boundary points of the shape is illustrated in figure 5.12. This shows a marked boundary point isolated on a shape and the two reference points. The coordinates of these points are determined, and using equations (44) and (45), the arctangent angle of curvature between the three points is determined, and can be plotted for the entire shape boundary.

The smoothing factor m is required as a constant in function. Choosing a value to represent m requires careful consideration. As an example of determining *m*, we give as an approximation to round twenty-degree distance round the shape perimeter. This is *m=N/scale* where *N* = *length(P).* It is important to note that this approach is a generalisation and will be affected by local irregularity in the shape boundary. Figure 5.13 shows curvature plots for different shapes using different values for the scale of curvature, essentially the higher scale denotes a smaller angle. 72 represents a 5-degree local angle around the shape boundary, 36 a 10-degree angle, and 18 a 20-degree angle. An approach was considered involving determining the angle from the centroid to each boundary point and plotting for every *Z* degrees, yet this was ruled out in practice due to the complexities of this operation.

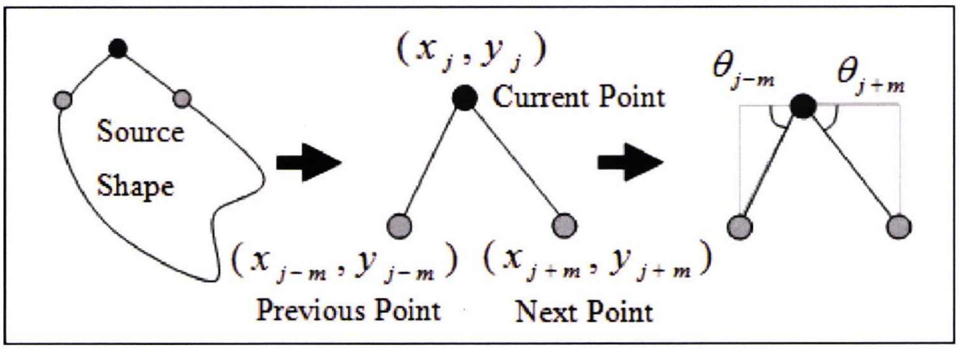
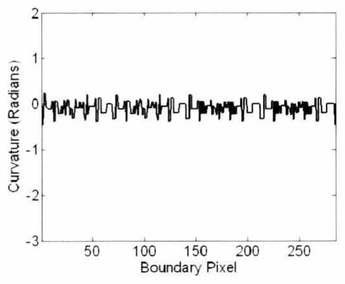
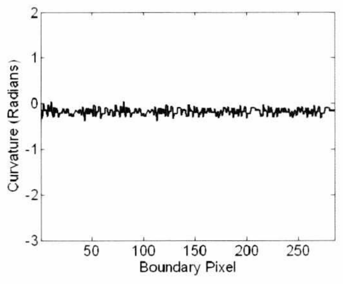


Figure 5.12. Relationship between curvature points on a shape perimeter.

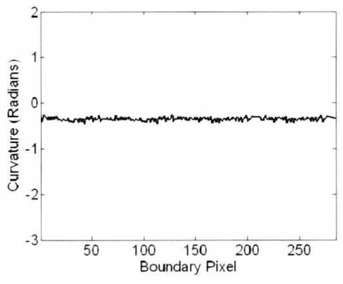
In Figure 5.13, it is clear that with increased angles a Gaussian smoothing effect occurs, although to a point this does not have an impact on the magnitude of the larger angles in the shape. Also too low an angle and the noise of aliasing of the digital boundary dominates the curvature plot. Seeking an optimal scale of curvature is advisable dependent on the application. 18 was used as the value of *scale* in most applications as it was sufficiently high a distance to eliminate most aliasing, yet was preserving of some fine detail in the shape boundary. A fixed distance boundary was also used this is described later in relation to its application in a segmentation function. The flexible code used for the implementation of this function is found in the appendix.



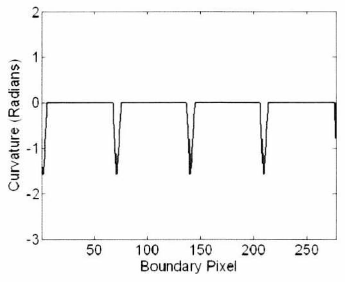
(a) Circle, m=N/72.



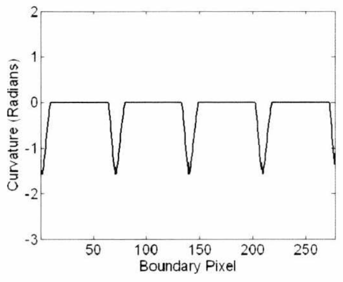
(b) Circle, m=N/36.



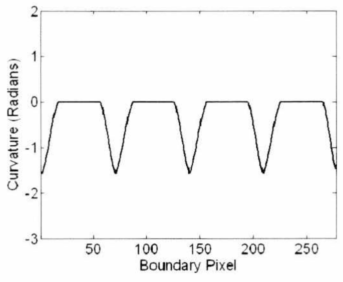
(c) Circle, m=N/18.



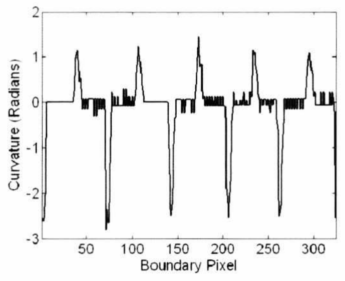
(d) Square, m=N/72.



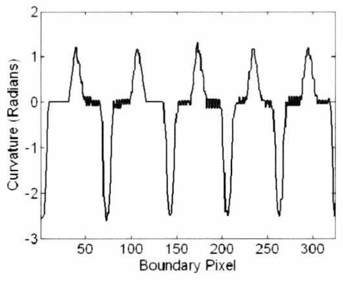
(e) Square, m=N/36.



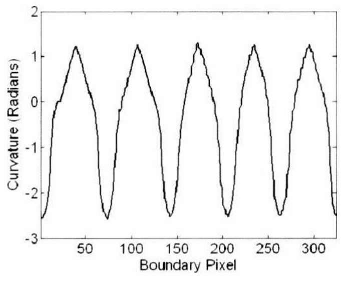
(f) Square, m=N/18.



(g) Star, m=N/72.



(h) Star, m=N/36.



(i) Star, m=N/18.

Figure 5.13. Curvature for Star, Square and Circle with different smoothing factors *m*.

Several methods exist to analyse these types of curvature signatures. The plot can be analysed with statistical approaches, deriving high and low points, means and standard deviations. Further filtering can be applied to smooth or alter the data keeping only high or low pass values. Moments and second order moments (gradients) can be derived. The signature can also be completely transformed by methods such as Fourier, Wavelet or Fractal analysis.

5.6 Fourier description of signatures

The Fourier description described in this section relates to that of the curvature signature of the shape contour P_{List}. In the following examples P_{List} is given with the angular approach using *N/18*. This comes in both the form of a signature based method and analysis of the coordinates of boundary curvature. Fourier analysis permits representing this signal as a *trigonometric* series through transformation. This is a series of coefficients composed of *kernel* functions to represent the shape. The function designated f_k is now taken to represent the curvature derived with figure N, and is assumed to be of length *N*. This can be described as a *periodic* function (given that the recorded data is discrete). This is ideal for the discrete signatures of shapes which can be be described as:

$$f_k = a_0 + \sum_{n-1}^{N}[a_n \cos(\frac{n\pi t}{L}) + b_n \sin(\frac{n\pi t}{L})], \quad n = 0...N \tag{5.15}$$

In order to get Fourier descriptors from f_k this is expanded to a series of coefficients:

$$A_o = \frac{1}{N}\sum_{k-0}^{N-1} f_k \tag{5.16}$$

$$A_n = \frac{2}{N}\sum_{k-0}^{N-1} f_k \cos(\frac{2\pi jkn}{N}) \tag{5.17}$$

$$B_n = \frac{2}{N}\sum_{k-0}^{N-1} f_k \sin(\frac{2\pi jkn}{N}) \tag{5.18}$$

The following applies to these equations. The lowercase *n* represents the number of the harmonic. The lower *k* represents the current coefficients considered in the list of up to *N-1* coefficients. Up to *N-1* harmonics *n* are permissible to describe a signal of length *N*. In this example, *N* originates from the length of the periodic function *f*. To clarify, a periodic function is defined as one which repeats itself after a definite period. Interpretations of contour of a shape such as signatures meet this criterion. A one-dimensional boundary Fourier descriptor using signature plots of radius or diameter through is considered through a MATLAB function found in the appendix.

5.6.1 Curvature specific Fourier Transformations

Fourier descriptors traditionally used in signal processing are applicable to a one-dimensional signature of a shape. As implemented upon shape curvature in work by Xu (1998) and Bowman (2000), Fourier descriptors return a series of harmonics based upon the original shape perimeter. This results in a quantifiable approach to describing the characteristics of shapes. This type of approach is a discrete Fourier transform on a one-dimensional signal of length N. It is possible to create 'harmonics' or coefficients which can be used to recreate the original signal. The past research on this subject mentioned earlier considered the use of Fourier analysis with different number of harmonics. An increased number of descriptors permit a greater level of reconstructive detail. The greater the number of harmonics considered, the finer the detail can be recovered. This includes fine details of shape, particularly small irregularities. A large number of small irregular features are reported to be recoverable with 64 descriptors (Bowman 2000). With a smaller number of descriptors, general reconstruction is still possible, with quantities such as 24 or 8 descriptors. This also has lead to the harmonics produced through fourier transformation to be used as powerful shape descriptors. The use of 8 harmonics by Xu effectively supports the distinction of shapes using Fourier coefficients for a 2D shape classification task.

These coefficients are created with the use of sine and cosine kernel functions. From application of the coefficients (17, 18) upon a curvature plot, it is possible to create a new set of coefficients we will call harmonics to quantify the shape:

$$C_n = \sqrt{A_n^2 + B_n^2} \tag{5.19}$$

This function is repeated for the *n* harmonics. The types of harmonics produced by different shape is illustrated in figure 5.14. This plots the first 30 Harmonics for the three basic shape types using *Cn* where *0<=n<=30*. Combining the results of the two functions *A* and *B* gives the final harmonic *C* retrieved using trigonometry. Xu employed eight harmonics *n* are effective for a classification problem; however this involved shapes of significant geometrical variance. This differs for rice grains and other shapes. In figure 5.14, the shapes in question differ dramatically in very specific harmonics, with the exception of circles which are nearly all consistently close to 0. The more complex the shape, the more subtle the features and the more descriptors are needed to represent specific elements of the composition of the shape boundary.

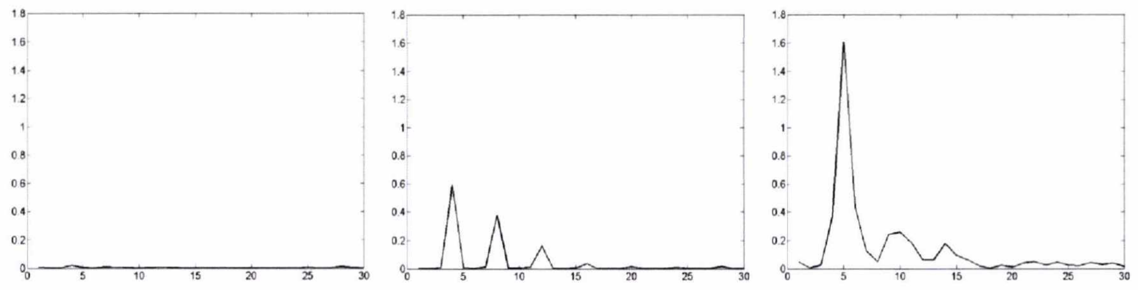


Figure 5.14. Harmonics 1:30 of curvature (m=N/18) for circle. square and star.

5.7 Gradient Analysis of Signatures

The gradient *G* can now be derived from each member of a signature *P*:

$$G_i = (P_{i-m} - P_{i+m})/2 \tag{5.20}$$

Where different values of *m* will return different gradients. Existing research (Xu, Luxmoore and Deravi, 1997, 1998) uses value 3 for parameter *m*.

Aliasing in the discrete boundary can cause problems if there are low values m employed. A solution considered to smooth this data was the use of interpolation through the application of a spline. A spline function in the Matlab environment was employed for this process. This permitted rescaling the plot to a fixed length. The example in Figure 5.15 interpolates the curvature plots from Figure 5.8 to a function of curvature over 100 points. This approach significantly reduces the spiking in the circle plot, yet causes some curvature to appear in the other angular plots.

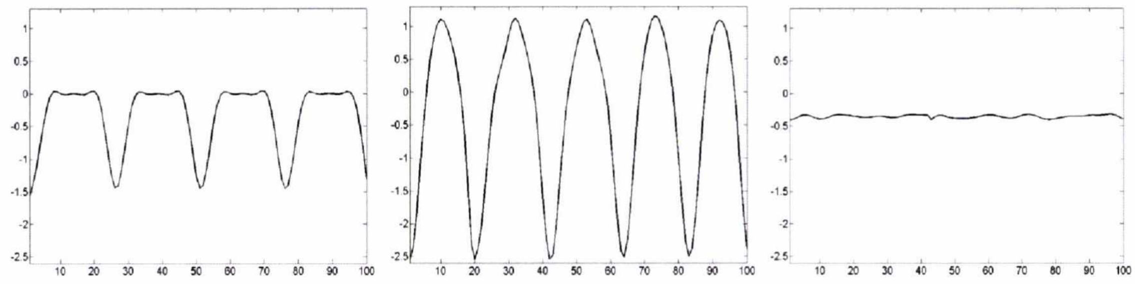


Figure 5.15. Plots of splines of curvature plots (m=N/18) for Square, Star and Circle.

Further evaluation is possible with the analysis by deriving the second order moment of *G* as a function *F*:

$$F_i = (G_{i-m} - G_{i+m}) / 2 \tag{5.21}$$

Where this function results in the values of *F* being absolute (>0).

In order to develop this function for an ideal theoretical shape similar to a grain with and without an eye, first of all, a binary image of an ellipse was created of dimensions 191 by 71 pixels. A second ellipse of the same dimensions was also created with a proportion cut by a smaller elliptical object, creating a 'fake' eye curve grain for developing detection of this curve. These are shown in Figure 5.16.

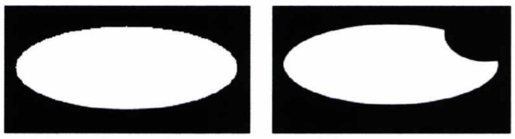


Figure 5.16. Ellipse and Curved Grain.

Example plots of splined curvature for the boundary of an Ellipse shape are given in Figure 5.17, where degree of curvature is plotted for 100 interpolated spline points of a shape curvature signature. In this data, the gradient is plotted by the mid tone grey, and the function of gradient is plotted by the light tone grey. Significant peaks in both data have been highlighted with bars by using a threshold criterion on both G and F. Where there is a positive or negative peak in G, and the same position in F=>0.01, that point is a significant peak This threshold is arbitrary and was determined for an experiment described further in the results chapter. This function can assist in detecting significant regions of curvature in a plot of data, which can be then denoted by the number of significant peaks. A boundary of an elipse with a hollowed out 'eye' feature similar to that of the subject rice grains is shown in figure 5.18.

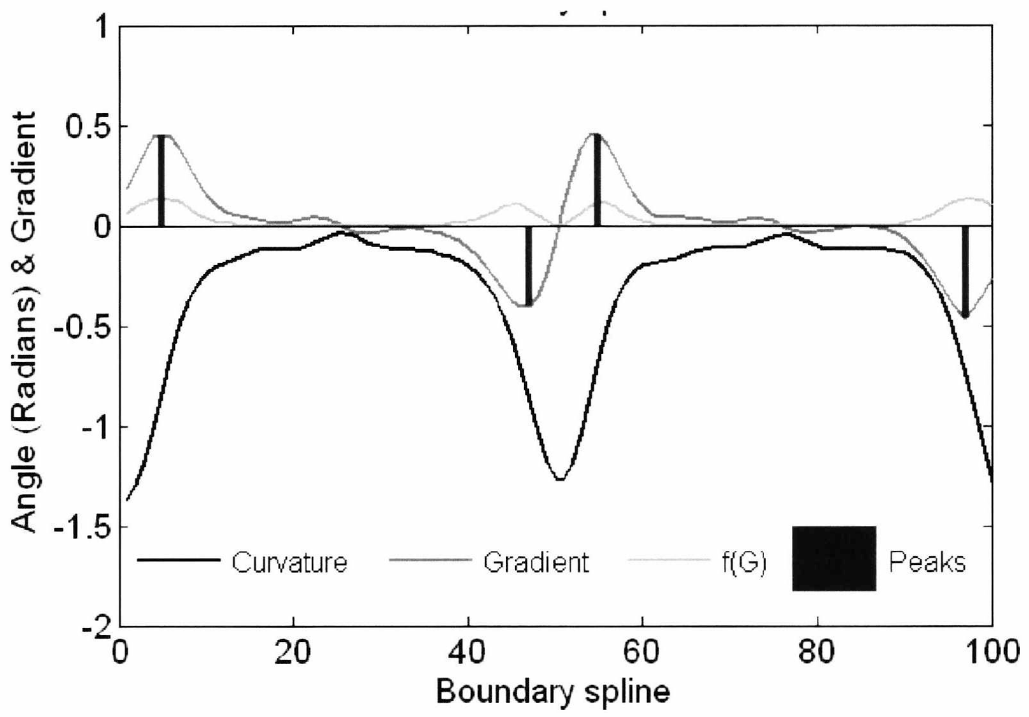


Figure 5.17. Boundary spline functions of curvature for the boundary of an ellipse.

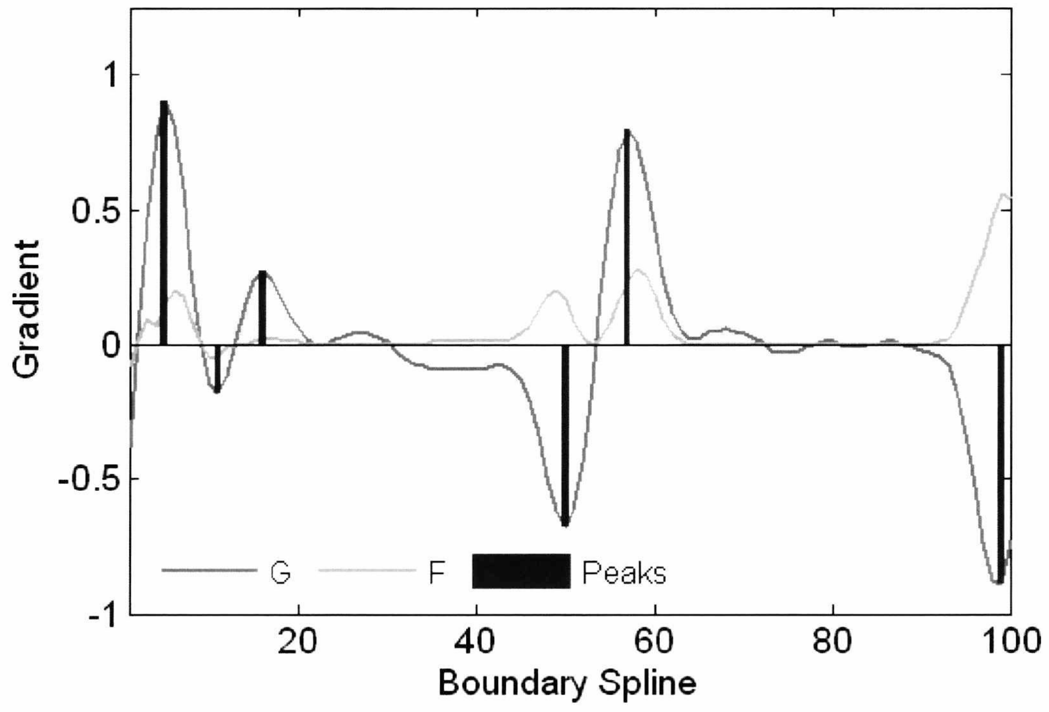


Figure 5.18. Gradient, Function and Peaks for a rice grain with an eye feature.

As well as gradient peaks, zero crossings will occur and provide a simple point of interest to evaluate. Simply counting the number of zero crossings of the gradient gives some useful measurement of unusual occurrences distinct for that shape. Although clearly some zero crossings of low magnitude can distort results, like the minor dips present in Figures N and N2. Rather than using the function of gradient, a function can be derived within the zero crossing itself using a threshold to only accept zero crossings of sufficient gradient. A flow chart illustrating a process that returns the number of zero crossings as a variable *numberofzeros* is given in Figure 5.19. This process incorporates filtering based on a constant *scalefactor*. This is used as a threshold to eliminate smaller zero crossings and was used as a variable to find an optimal value.

Evaluation of these functions will be addressed in the results chapter, where the use of these features for rice grain analysis will be implemented.

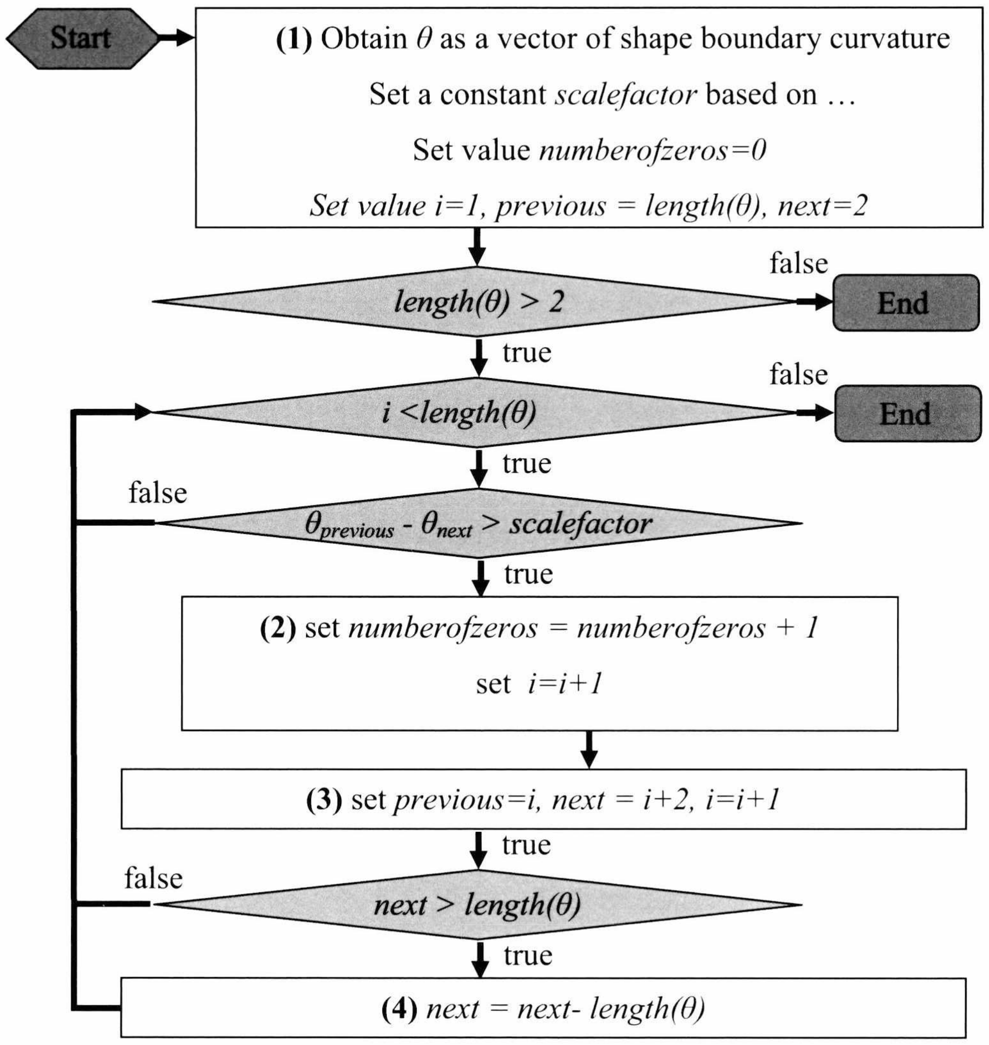

Start
(1) Obtain θ as a vector of shape boundary curvature
Set a constant scalefactor based on …
Set value numberofzeros=0
Set value i=1, previous = length(θ), next=2
length(θ) > 2
false
End
true
i <length(θ)
false
End
true
false
θprevious - θnext > scalefactor
true
(2) set numberofzeros = numberofzeros + 1
set i=i+1
(3) set previous=i, next = i+2, i=i+1
true
false
next > length(θ)
true
(4) next = next- length(θ)

Figure 5.19. Zero Crossing Detection.

5.8 Grain Segmentation Function

This section covers the rule-based segmentation function used to partition an image of physically touching rice grains. The section begins with a general overview, and expands the various complex subsections in turn. Several functions described earlier, such as curvature and the FKZ algorithm are employed in this section, although further analysis of those forms of data are expanded upon in this section.

5.8.1 Process Overview

The process of segmentation implemented for this task is achieved through the use of a simple decision function that requires several properties of the shape, and repeats several times to perform individual segmentations. The input requirement for this process is a Thresholded Image, produced through the earlier described threshold process. This process is directly influenced by several previous publications (Paliwal and Visen 2001, Wang 1998), although the decision function approach was tailored specifically for application on rice grain images.

The NexImage camera system was employed with the image processing setup for the capture of images of Basmati rice grains. The grains were positioned in contact with one another against a contrasting chroma keyed background. Images obtained were subject to the Image threshold process using a chroma key to subtract the contrasting background. This approach can vary dependent on camera employed, yet this ultimately leads to a binary image. An analysis of this image is performed by a segmentation decision function, which determines if each shape within the image requires segmentation. The segmentation function is nested within the decision function, and is called if several criteria are met for the segmentation of the shape. Finally, the analysis process is run extracting shape features from each segmented shape of appropriate size. Small fragments caused by any noise, debris or other anomalies in the captured image are ignored using this criterion.

The image capture and processing software was written using a combination of Directshow, C#, and Matlab Compile Runtime (MCR) functions. The key segmentation algorithm is described in the following sections.

5.8.2 Segmentation Decision Function

Taking the processed binary image as input, an analysis function determines if 2D shapes (shape in this case refers to a grain or several grains touching) within the image require segmentation. The segmentation function is called from within this decision function on the basis that several criteria are met. Following these steps an analysis process is run permitting measurement of each segmented shape. The result of the segmentation function is given in several examples as well as the description of the function itself. The decision function accepts the grain for segmentation based on its feature values. This method uses several constant values partly derived from previous work on the imaging of segregated grains of rice (Hobson, Carter and Yan 2007). The function works by first establishing a recursion criterion, given that no more than four segmentations are to be performed in a given case; and the process repeats a maximum of four times. Next, each individual shape is individually labelled through connected component analysis. Following this is an assessment of these shapes with features derived from them. Figure 5.20 illustrates the result of applying segmentation with the criteria for the decision function being met. The overall process is illustrated in Figure 5.21, with the segmentation function itself encapsulated in a single process described later.

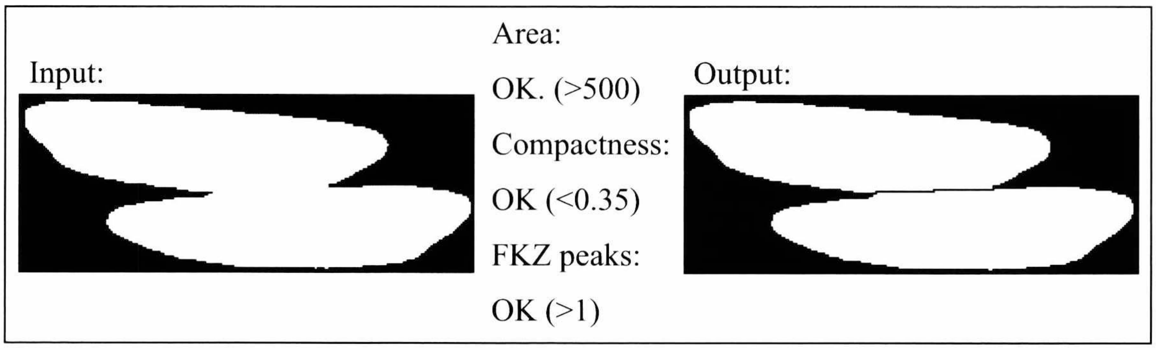


Figure 5.20. Example input and output images using decision criterion from Figure 5.21.

The process begins with elimination of artefacts in the image, omitting small objects below an arbitrary minimum size value (500 pixels in the Figure 5.21 case), selected relative to the image size and grain size expected. Next, the detection of any interior holes is performed by a combination of image arithmetic operations. Any holes found can be used to derive interior shape boundaries, as several grains in contact in a circular fashion may have a gap in the middle of them. The area of the hole is measured and a minimum size criterion of 25 pixels is given to prevent any small image noise related holes being vetted. For the next decision it has been previously established that all valid rice grains of all varieties (Sakai, Yonekawa and Matsuzaki 1996, Hobson, Carter and Yan 2007) have moderately high shape compactness or R_c as defined earlier. This uses parameters A as the (complete filled) shape area and accurate length P as the outer shape perimeter. A shape with low compactness is regarded as irregular. A value of 0.35 was used as a threshold, as this was considerably lower than the lowest values recorded for a valid single grain using previous analysis methods, yet often occurs in irregular thresholded grains or contacting grains. This feature alone is not sufficient so an additional concaveness feature based on Fernandez's FKZ algorithm (Wang 1998) is employed. The concaveness feature is derived using a simplified 1x5x5 mask M. M is consequently a mask for the 5x5 area for each point *j* in the shape boundary and B is the source binary image. The FKZ peak mentioned in figure 5.20 is an isolated point greater than 20 Shapes with more than one peak will be automatically accepted by the segmentation decision function. The number of peaks within this data for contacting grains will still be higher than 1, as peaks will be present for the concave curves of the grains.

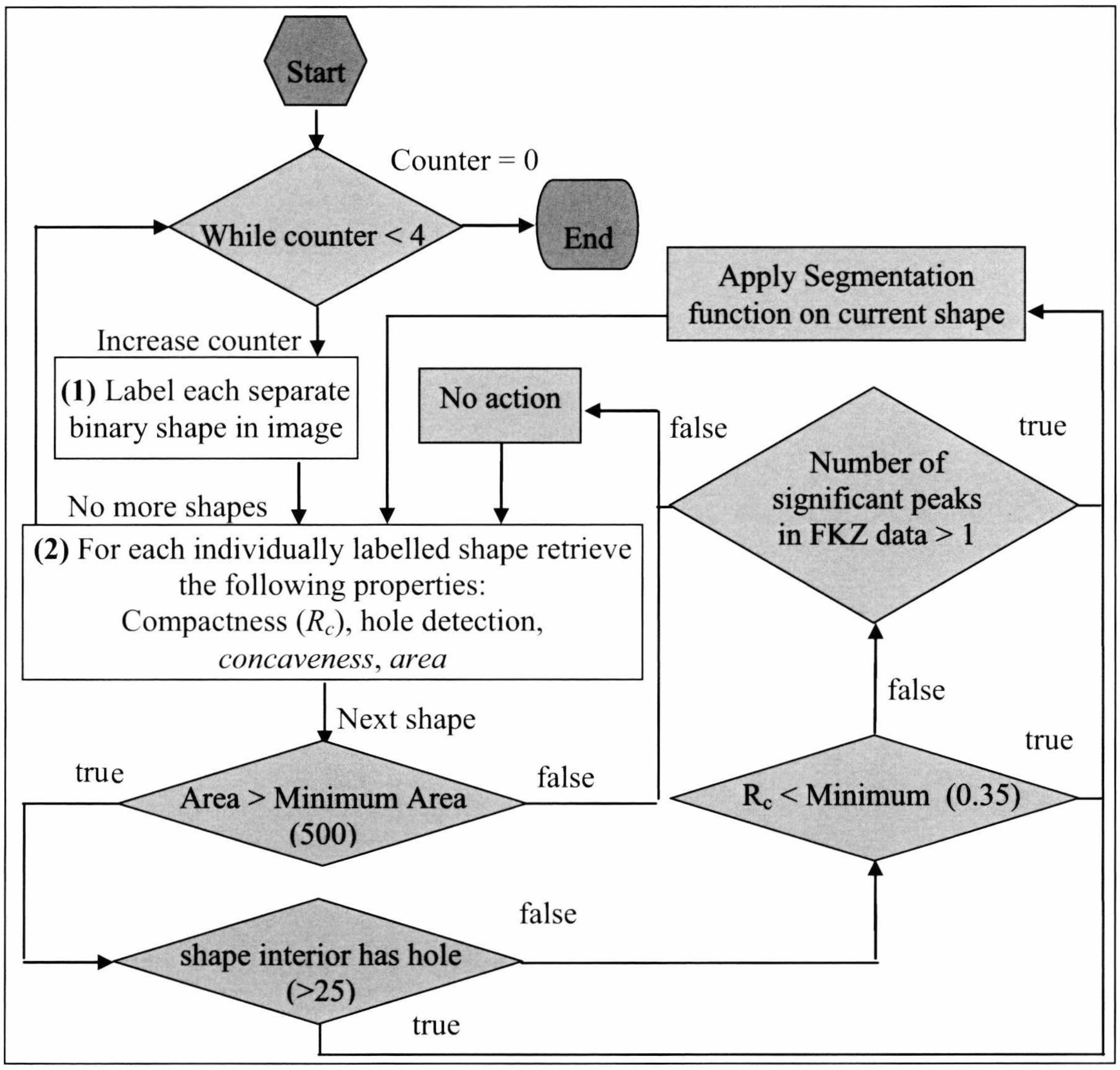

Start
Counter = 0
While counter < 4
End
Increase counter
(1) Label each separate binary shape in image
No more shapes
(2) For each individually labelled shape retrieve the following properties: Compactness (R_c), hole detection, concaveness, area
Next shape
Area > Minimum Area (500)
true
false
shape interior has hole (>25)
false
true
No action
Apply Segmentation function on current shape
Number of significant peaks in FKZ data > 1
false
true
R_c < Minimum (0.35)
false
true

Figure 5.21. Process for Segmentation Decision Function.

The final rules for the decision function to be applied on a shape are as follows:

The decision function uses these features to determine if the segmentation processes should be applied. The manner in which these rules are applied are as follows:

- The Grain area greater must be greater than a predetermined area threshold, i.e. $A > A_T$
- At least one of the following criteria must also be met:
  - At least two FKZ peaks must be detected from the shape boundary.
  - Compactness must lower than the threshold $R_c < C_T$.
  - A special case exists if a 'hole' is detected in the shape interior. This detection is performed here by connected component analysis of the inverse of the image. The area of these holes should also be above a predetermined threshold (25 pixels was used in this case), or $H > H_T$

If these sets of criteria are met, the shape can be subject to the segmentation function. This proceeds by a complete evaluation of the concave regions of the shape. This process requires these points as 'candidate split points' etween which to determine the 'split path' which are the the set of points between which to draw a 'split line' in order to segment the target shape. This process repeats four times through the decision function, each time drawing a split line if a path is found.

5.8.3 Allocation of Split Paths

Assuming the criteria are met to the initial segmentation decision, significant concavities can be investigated as potential locations for drawing a segmenting line or 'split path' as described in Figure 5.22. This approach examines the curvature of the shape boundary in order to find regions of irregular concavity as candidate split paths. Once the appropriate 'split points' marking the position of the path are found, a split path can be determined and a segmenting line drawn in the manner illustrated in the output example of Figure 5.19. Six steps follow with one significant choice dependent on the number of significant split points detected. The first step (step 1) is the identification of candidate points as significant points of curvature on boundary regions (both from the shape perimeter and the perimeter of any interior holes). Candidate points are added to a list of points designated '*angles*'. Steps 2 and 3 record the properties of those points, first individually, then as pairs of points joined to form candidate paths. Next, the outcome varies according to the number of points detected on the shape perimeter. Shapes with only two points are considered to be only two adjacent grains contacting at a single point. In this case less stringent segmentation rules are applied to ensure segmentation is only performed if the mean orientation of these points towards one another is within 90 degrees. A more restrictive 45 degree criterion is employed if there are more split points. Valid paths are added to the list '*anglelist*' only if these requirements are met. The first path in the list will be drawn on the image as the split line in step 6.

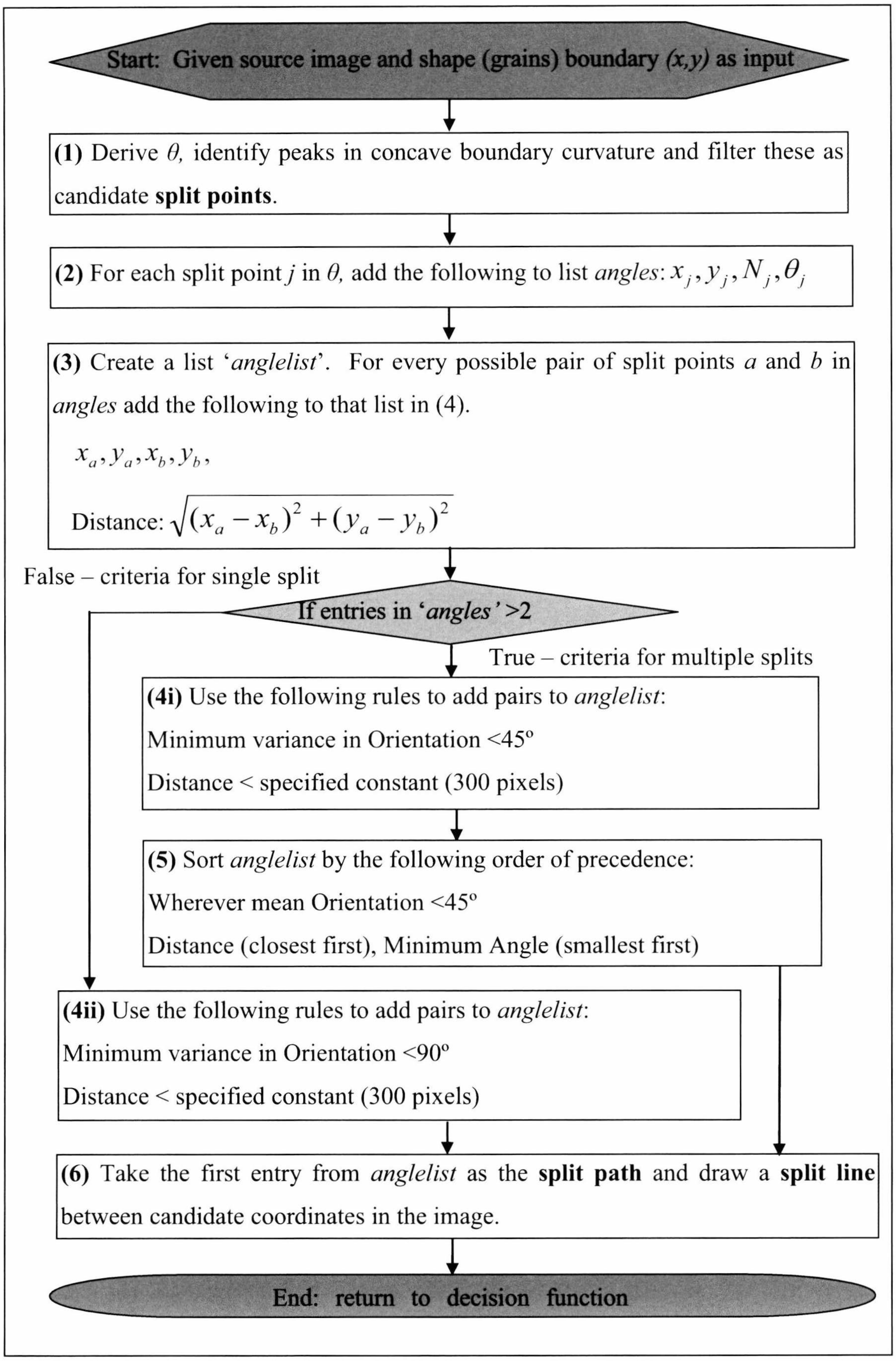

Start: Given source image and shape (grains) boundary *(x,y)* as input
(1) Derive *θ,* identify peaks in concave boundary curvature and filter these as candidate **split points**.
(2) For each split point *j* in *θ*, add the following to list *angles*: x_j, y_j, N_j, θ_j
(3) Create a list '*anglelist*'. For every possible pair of split points *a* and *b* in *angles* add the following to that list in (4).
$x_a, y_a, x_b, y_b,$
Distance: $\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$
False – criteria for single split
If entries in '*angles*' >2
True – criteria for multiple splits
(4i) Use the following rules to add pairs to *anglelist*:
Minimum variance in Orientation <45°
Distance < specified constant (300 pixels)
(5) Sort *anglelist* by the following order of precedence:
Wherever mean Orientation <45°
Distance (closest first), Minimum Angle (smallest first)
(4ii) Use the following rules to add pairs to *anglelist*:
Minimum variance in Orientation <90°
Distance < specified constant (300 pixels)
(6) Take the first entry from *anglelist* as the **split path** and draw a **split line** between candidate coordinates in the image.
End: return to decision function

Figure 5.22. Flow chart for segmentation function procedure.

5.8.4 Detection of Significant Split Points

The steps in the illustrated procedure in turn contain complex functions. Step 1 in Figure 5.23 involves obtaining the curvature of the shape from internal and external perimeters and extracting significant peaks from that data. First, this curvature must be defined. Figure 3 illustrates the use of boundary points from the discrete boundary of a shape from a binary image to derive curvature. Coordinates of each boundary pixel are stored in vectors *x* and *y*. Each point *j* from *x* and *y* is used in curvature functions to record the curvature for that point. The curvature of every boundary point is then stored in a vector, which is subject to statistical filtering to separate points of interest from the rest of the curve data. The detection of split points from this curvature is part based on polygonisation (Wang 1998). The goal of this stage is to break the shape down from the entirety of the curvature to only a few select points of significant curvature.

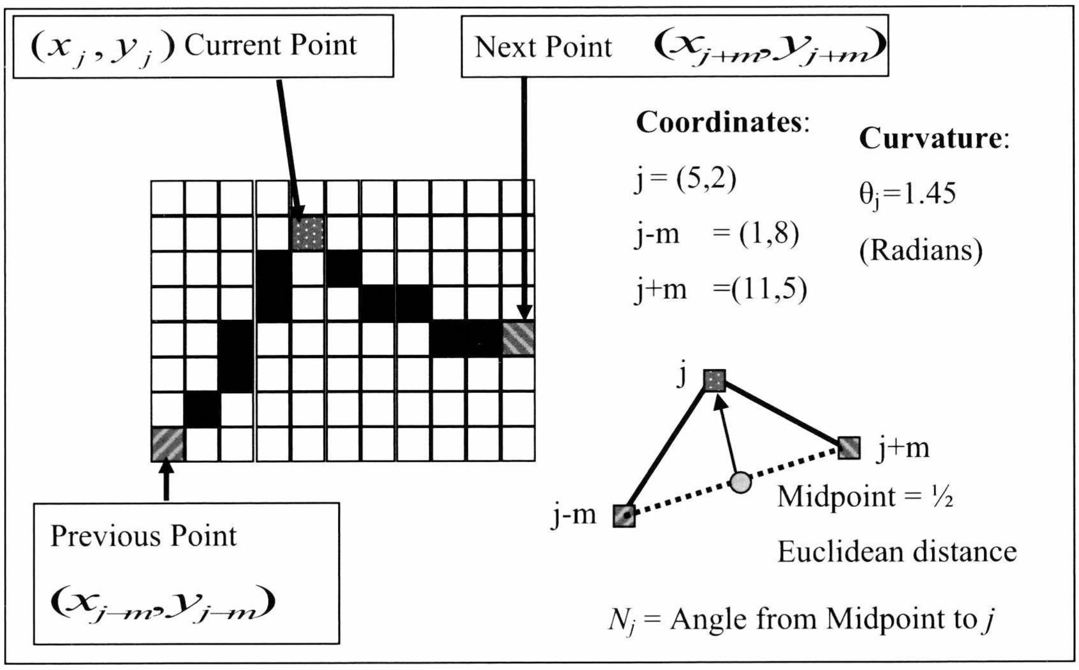


Figure 5.23. Coordinates of boundary point *j* with curvature θ in 2D

Curvature is represented by vector θ where $-\pi \geq \theta_j \leq \pi$. The values of θ_{j-m} and θ_{j+m} are derived using tangent angle functions of the *m*-distant boundary points. An arbitrary distance value of 6 pixels used for the curve function was employed as it was proven effective for the scale and resolution of the grains within the images used in testing. The two points are derived from the boundary of the current shape using the coordinates of each point j, j+m and j-m, where m=6 for the entire shape boundary.

Figure 5.24 gives an example pre-processed image and resultant segmented result. Figure 5.25 shows the plot of the thresholded image in Figure 4's exterior boundary curvature θ, and resultant peaks from that data. The pixels on the shape perimeter where the shape is most concave denotes the two points where the edge points of the contacting area between two separate grains. By finding these points and examining them further, the remainder of the decision making to segment the shape can be made. To isolate these points, significant curvature peaks are filtered from the background curvature of the shape according to a specific scale of curvature. This scale must be sufficiently tolerant to ignore lesser peaks caused by grain eye curves and other small artefacts. These lesser curve features are generally far smaller than the significant curves of the edge of the contacting concave perimeter. A small eye feature can be seen in Figure 5.25 as the third highest positive peak, which is still significantly smaller than the two concave peaks of interest. These positive peaks are marked in Figure 5.25.

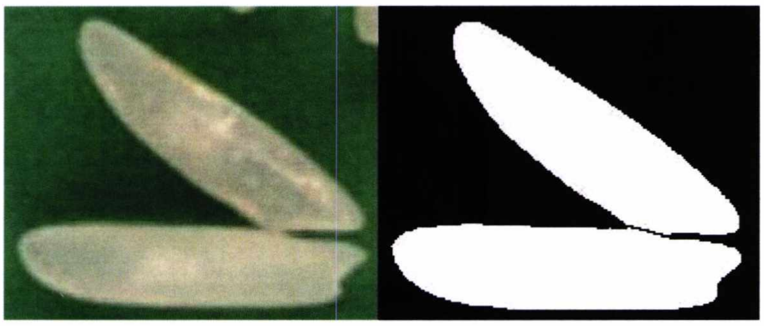


Figure 5.24. Source grain image (left) and segmented result (right)

In order to isolate peaks of interest the curve data is filtered using a simple statistical analysis. For this function only values where θ_j>0 were used in deriving the mean and standard deviation. These values are designated $(\theta_\mu+\theta_\sigma)$ and are combined with a modifier that is variable dependent on the scale of curvature found within the shape. If mean curvature is less than 0.5 (a general case found in grains without the right significant peaks), values are multiplied by 1.3. This arbitrary value increases the scale to avoid detecting some minor curves as significant peaks. In addition, for the boundary of interior holes in multiple contacting shapes, the same was applied with curvature θ inverted. The output of this process for the curve is the marked points (dotted lines) in Figure 5.25. The orientation of these angles is then determined as value N_j for the respective θ_j. This value is the angle from a midpoint directly (Euclidean) between θ_{j-m} *and* θ_{j+m} to the point θ_j, and is illustrated in Figure 3. These valid angles are added to list *angles*.

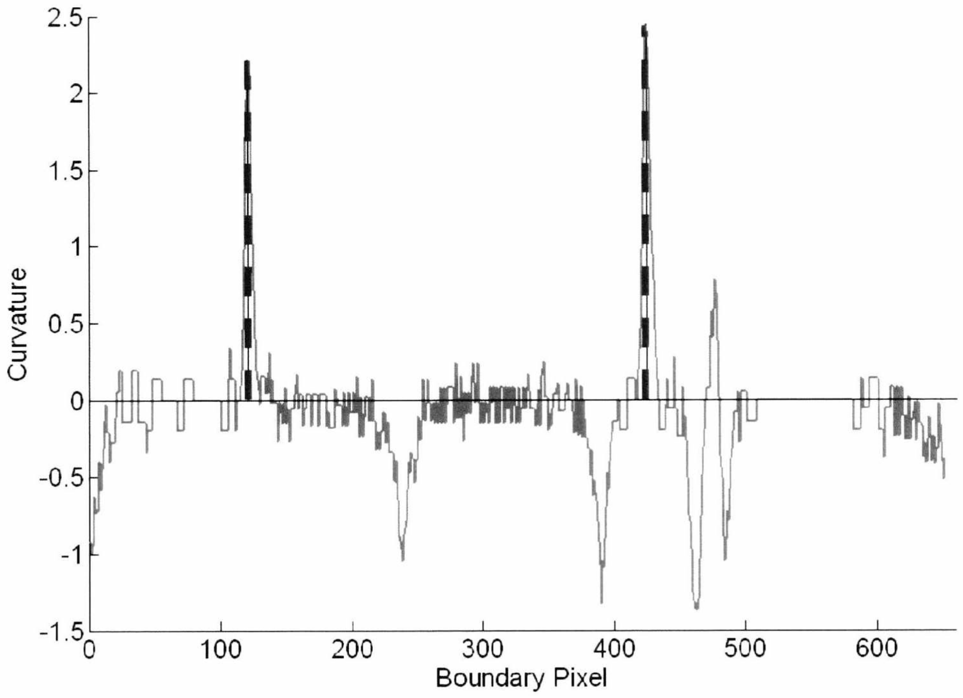


Figure 5.25. Curvature plot (3) of boundary from Figure 3.

Step 3 creates a new list *anglelist* pairing each possible combination of entries in angles as points *a* and *b*, in order to determine relationships between the two split points as potential split paths. Calculating the orientation between each pair of peaks in step 3 is achieved by comparing the individual value N_j to the direct angle to the point. This gives the orientation as values θ_{ab} and θ_{ba} for each point respectively. These values and the Euclidean distance between the points are used in step 4 to validate that pair of split points as a candidate split line. Finally, the most likely candidate is segmented in step 6, dependant on wherever the single or multiple split criterions are met.

5.8.5 Drawing Split Lines

Upon the final allocation of potential angle pairs from the vetted entries in *anglelist*, split lines for each shape can be drawn on the original image. Drawing a clearly divisive 4-way connected split line is a straightforward and robust approach to segmenting the shapes. A simple function was written for this task permitting 4-connection line drawing. It is worth noting that any segmentation approach involving removing a line of pixels is one that performs a permanent alteration to the shape and a deletion of the subset of pixels that compose the line. This produces the type of split line visible in the grain image in Figure 5.23. The complete algorithm for deriving the split line employed is given in Figure 5.26.

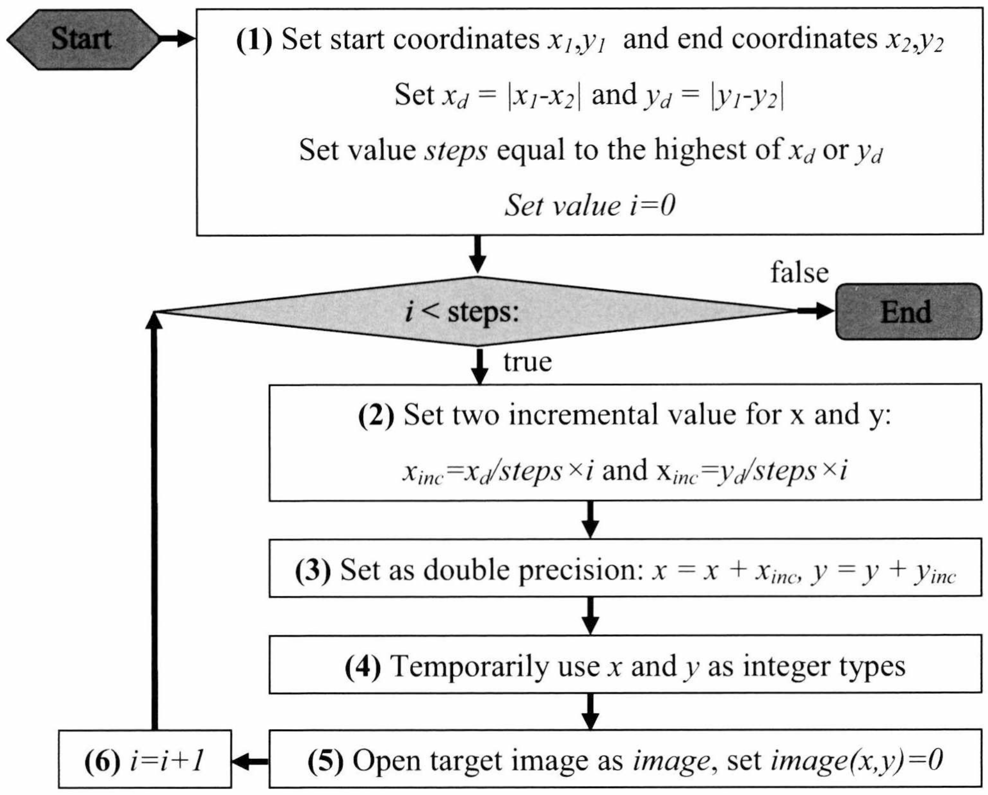

Start
(1) Set start coordinates x_1,y_1 and end coordinates x_2,y_2
Set $x_d = |x_1-x_2|$ and $y_d = |y_1-y_2|$
Set value *steps* equal to the highest of x_d or y_d
Set value i=0
$i <$ steps:
false
End
true
(2) Set two incremental value for x and y:
$x_{inc}=x_d/steps \times i$ and $x_{inc}=y_d/steps \times i$
(3) Set as double precision: $x = x + x_{inc}$, $y = y + y_{inc}$
(4) Temporarily use x and y as integer types
(5) Open target image as *image*, set *image(x,y)=0*
(6) $i=i+1$

Figure 5.26. Line drawing algorithm.

5.9 Summary

The descriptive properties of grains need to be determined as to wherever or not they are useful to the given task. This can be done part by contrast with requirement with shape and appearance requirements set down by rice organisations mentioned in previous chapters. The physical challenge of an intelligent rice sorting machine having contacting grains is also one that has been addressed in this section. Also the possibility of examining the texture of rice for identifying specific grain types may be some use, and applicable in practical engineering tasks relating to grading and vision based identification. The ultimate goal of this section was to describe the features retrieved from images and how they are retrieved, as well as summarise the comparative analysis of features for identification and classification. This leads now to the next chapter where results of image capture, processing, and analysis are discussed.

Chapter 6

Results and Analysis

6.1 Introduction

The previous chapters discussed means to obtain metrics for characterisation of the target materials. In this chapter results of application of the image processing and features are described. The first step in assessing these materials was sample preparation for imaging using the discussed static imaging set-ups. Next was capture of images of these materials. Several different imaging devices were used for obtaining data on target grains. At different stages in the work slight variations in image processing were applied dependent on the employment of background subtraction and type of material imaged. Shape and texture measurements are then retrieved, and evaluated in this chapter per experiment.

Many of the features considered here are not only applicable to rice grain characterisation. In recognition of this, the results in this chapter are divided into two parts. First shape features described in chapter five are evaluated as a more widely applicable subject of research. The rest of this chapter is focussed on the primary scope of this thesis, the characterisation of rice grains. This separation was result of consideration during the course of research that features and image processing techniques were more widely applicable beyond the analysis of rice.

These include the different rice grain analysis tests. These include the classification of rice grains by shape, the textural difference between white and brown grains, sizing differences between basmati and non-basmati rice, curvature features, and grain whitening. Where applicable, effectiveness of extracted features for characterising and identifying grain samples are described in the appropriate section. Results include the ability to differentiate specific basmati varieties, brown, white and special grain varieties, grain 'eye' features as a quality measure, general shape characterisation of multiple grain varieties, and segmentation of grains through image processing Techniques developed are also transferable to other grain types and entirely separate fields of research.

First it is important to describe what results are expected from this system. In this case, any metrics that are indicative of general properties of the grain, particularly if relevant to requirements by international organisations mentioned in previous chapters. Ways of identifying different types of rice, identifying unique features for certain grain types and comparing results for different sample sets are given here. Also assessed are the manner in which there are recorded. Several examples of the grain images are also shown in order to demonstrate image quality from the setup and cameras used. And finally the segmentation function is tested in contrast with images of grains that are separated, in order to assess the impact of segmentation on measurement of rice grains through digital imaging.

Testing is performed through analysis of the recorded shape and textural features. Several typical approaches are taken in expressing feature analysis results:

- Derivation of mean and standard deviations per sample. This was used as a way of showing the general properties of results for a specific feature for each sample, in an attempt to show difference per feature per sample.

- Cluster results with an unsupervised hierarchical approach. This can be implemented to show the differentiation between clusters in terms of distance between cluster centres based on the values of several features. Examples relevant to this task can be found in work by Carter (Carter and Yan 2005) and in literature by Costa and Caesar. (2000). Such an approach can indicate how effective certain combined features are at highlighting the difference between classes.

6.2 Testing of Shape Features on Theoretical Shapes

Work by Carter (2005) is reinvestigated and extended to consider further analysis of two dimensional shape, with applications to rice grains, particle imaging, and general two dimensional shape analysis. In that research, digital images of different shapes of different scale and rotation are evaluated. In this research, a wider range of rotations and more controllable scale is implemented to improve the experiment. More features are also investigated. The nature of this test is as follows. To compare the effectiveness of different shape descriptors a number of simple geometric shapes are created and exposed to the shape features in chapter 5 using their region and boundary properties. This test is to show the ranges of values for different types of shape descriptors and show trends in errors due to scaling. Past reasons for doing this include demonstrating a relationship between the complexity of shapes and shape factor. Similar discussions were made by Podczeck (Podczeck 1997, 1999). Shapes with more sides generally have a higher shape factor. While this analogy is somewhat simplistic with complex shapes, that is essentially the capacity of the shape factor descriptor. However, as other researchers have recognised the importance of multiple shape descriptors, this is expanded with futher investigation of a wider range of shape descriptors. These also include calculation of some of the theoretical shape values for a number of geometric shapes.

6.2.1 Generating images of test shapes

In order to create the digital images required for this test a C# graphics application was created to expressly produce a large number of accurate images of rotatable shape images. Image rotation was achieved using through an algorithm for generating shape images automatically. The application permitted the specification of shape types, sizes, and in the case of complex concave shapes such as stars, the ratio of concavity of the shape by specifying several parameters such as size, concavity ratio, and number of sides. The code of key functions of this application is described with full examples provided in the appendix, and working applications will be retained at the University of Kent. Shape description from chapter five is applied on the boundaries and regions of resultant shapes images.

6.2.2 Shape feature testing

In order to explore shape description using the functions described, we apply them to a set of shapes generated by this application. First, shapes images are generated by algorithm of specific size, concavity, and orientation ranges. In this experiment, four shape ranges and 45 rotation ranges are employed, with the same shape rotated at one-degree intervals in order to eliminate the impact of rotational variance and gauge impact of digitisation due to scale and rotation. Shape descriptor values for each shape are calculated and tabulated in this chapter. Each table and figure set represents the values for a single shape descriptor. Where possible, the theoretical value for the shape descriptors for that shape has been determined using trigonometry and calculation. This also permits the calculating of error between digital and theoretical versions of shapes. While it is impractical to attempt to perform such calculation for every shape and every shape descriptor many are calculated. Radial aspect ratio is the simplest to calculate and is consequently the most complete set of theoretical descriptors, as well as Compactness. Some of the diameter features are more complex due to the difference between radial and diameter measurements for many shapes so limited error calculations were made only in limited cases.

The tables and figures are split according to shape types. One set is composed of entirely concave shapes of increasing numbers of sides, from triangles to circles. Another set is composed of entirely convex shapes, of varying degrees of convexity. Most are stars and crosses of the same convexity index – the same ratio between the longest and shortest distances were used. A ratio 0.375 for the shortest length compared to the longest gave a ratio very close to that of a symmetrical 5-point star. This ratio was modified for two other 5-point stars, giving an additionally concave and convex star, with ratios of 0.5 and 0.25 respectively. Figure 6.1 shows the varying concave stars, labelled in later tables as a ccv (more concave), 5-point (midrange) and cvx (more convex) star. Figure 6.2 shows example images of all other shapes considered.

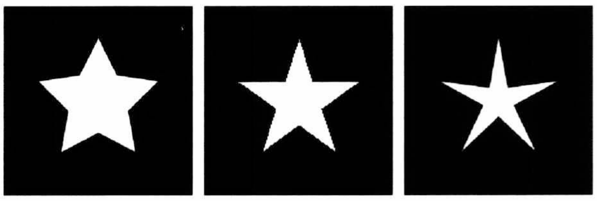


Figure 6.1. Images of the three stars of increasing concavity.

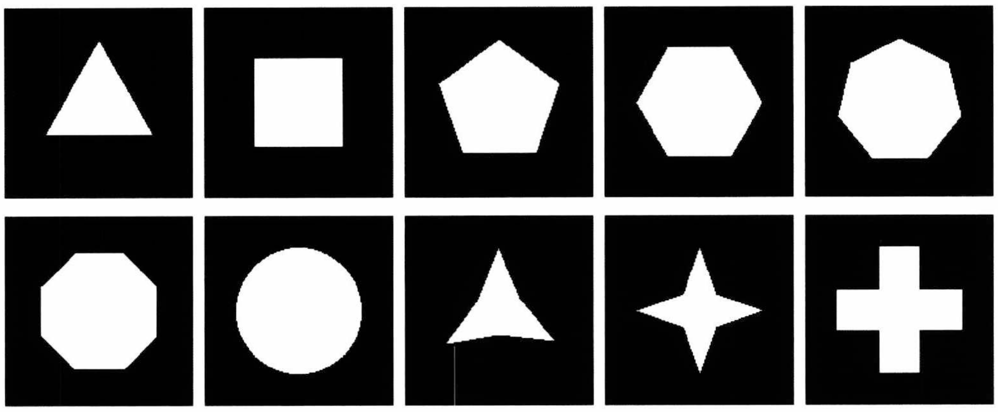


Figure 6.2. Images of other shapes produced for test.

This section records the mean values for all rotated shapes across four distinct shape size ranges, in order to illustrate the impact of aliasing on the measurement quality of that shape feature. Where possible, theoretical values for each shape feature for a continuous version of the shape have been determined and listed in conjunction with mean values from the sets of rotated shape images. This permits a derivation of margin of error between the discrete digital shape and the expected theoretical value. The tabulated descriptor values are provided over the following pages for each shape and each of the following shape features. First the shape images were automatically generated using the following values as their highest dimensions. Different image sets of different radii lengths were used as the longest dimension for each shape, then N-sided polygons were derived for the convex shapes connecting the N-points directly that shared this distance. The dimensions of each of these image sets is given in table 6.3. For the concave shapes, the distance was derived with midpoints that were equal in distance by the dimension multiplied by a scaling constant. This was defaulted at 0.375 with different values for some stars.

Table 6.3. Scales of each image set, with mean area per set.

	Set 1	Set 2	Set 3	Set 4
Longest radial length (pixels)	100	75	50	25
Mean area for set in pixels				
Triangle	5248	2619	1306	315
Square	8086	4029	2009	487
Pentagon	9592	4794	2398	577
Hexagon	10498	5233	2612	631
Septagon	11036	5519	2762	666
Octagon	11404	5704	2855	686
Circle	12674	6336	3170	762
3 pt Star	3295	1578	787	180
4 pt Star	3592	1713	854	196
5 pt Star	3722	1780	889	203
Cross	8782	4485	2164	531
Concave Star (ccv)	24884	14184	6203	1492
Convex Star (cvx)	49775	28008	12437	3101

Source images were generated using routines found in the appendix using the longest radial length and appropriate concavity ratio value . In the remainder of this section, tables list and graphs plot the four different size sets for the different shape descriptors in order to illustrate trends in the results of features.

These trends are two-fold. First are the trends in the feature results themselves, such as expected results for shapes. This would be for example, in the case of aspect ratio, values approaching 1 for shapes with all dimensions equal such as circles. The second trend set would be the amount of aliasing for each image set. Each set is plotted, and where possible compared to the theoretical value for that shape feature to denote error and accuracy of the resultant feature from the shape of that scale and quality of representation. Figure 6.4 shows the concave regular polygon shape trends, and Figure 6.5 shows the trends for the convex (stars, crosses) shapes. Concave features in a shape result in significant variation in shape measurement with application in grain and particulate imaging measurements.

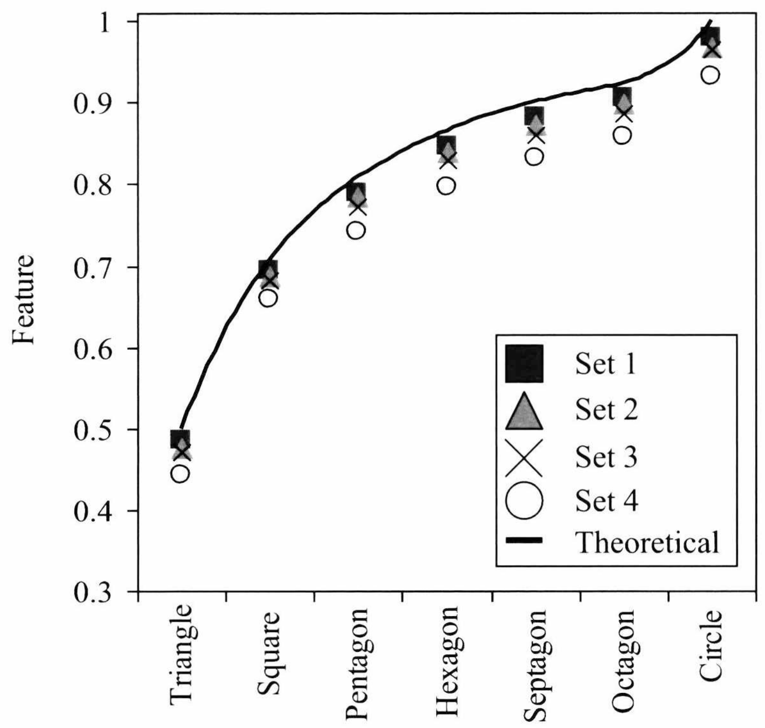


Figure 6.4. Radius Aspect Ratio Trends for Convex Shapes.

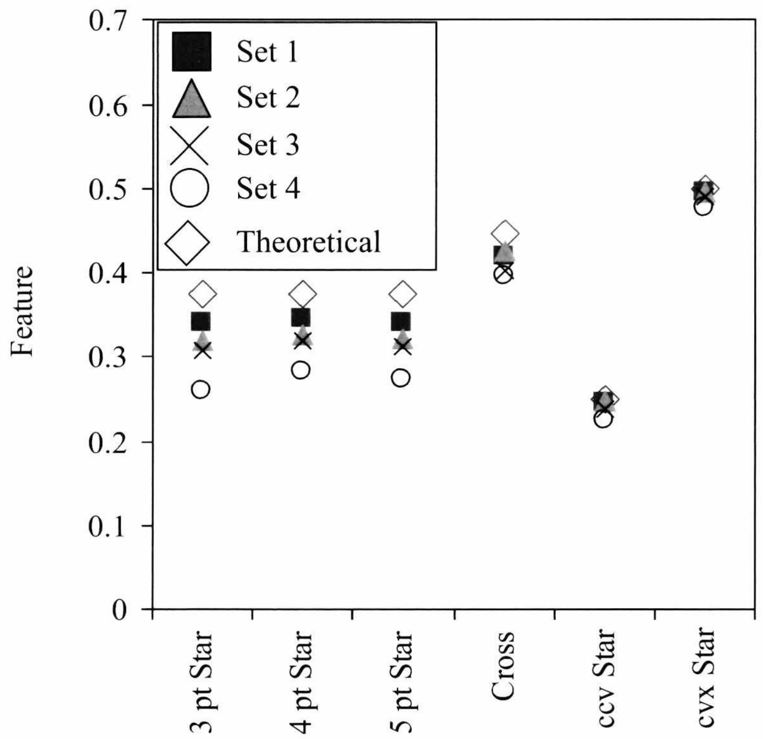


Figure 6.5. Radius Aspect Ratio Trends for Concave Shapes.

The concave shape trend is coherent with expected results with uniform shapes approaching aspect ratio=1. A good indication of aliasing is possible by contrasting the results of the four different scales with the theoretical value.

Table 6.6. Radius Aspect Ratio.

Shape	Non Perpendicular Radius Aspect Ratio								
	Theory	Set 1	Error (%)	Set 2	Error (%)	Set 3	Error (%)	Set 4	Error (%)
Convex Shapes									
Triangle	0.500	0.484	3.3	0.476	4.9	0.470	6.3	0.442	13.0
Square	0.707	0.693	2.0	0.687	2.9	0.682	3.7	0.658	7.5
Pentagon	0.809	0.790	2.4	0.784	3.2	0.773	4.6	0.742	9.1
Hexagon	0.866	0.846	2.3	0.838	3.3	0.830	4.3	0.796	8.8
Septagon	0.901	0.881	2.3	0.872	3.4	0.861	4.7	0.832	8.3
Octagon	0.924	0.905	2.0	0.898	2.9	0.887	4.1	0.857	7.8
Circle	1	0.980	2.1	0.968	3.3	0.966	3.5	0.932	7.3
Concave Shapes									
3 pt Star	0.375	0.339	10.6	0.320	17.2	0.308	21.8	0.260	44.2
4 pt Star	0.375	0.344	9.1	0.327	14.7	0.319	17.7	0.282	33.2
5 pt Star	0.375	0.341	10.0	0.323	16.3	0.313	20.0	0.273	37.5
Cross	0.447	0.419	6.8	0.425	5.2	0.403	10.9	0.396	12.8
cvx Star	0.25	0.245	1.9	0.247	1.0	0.239	4.5	0.223	12.0
ccv Star	0.5	0.495	1.0	0.495	1.0	0.490	2.0	0.477	4.9

Aspect Ratio for both Radius in Table 6.6 and Diameter in Table 6.7 is consistent for regular shapes, closing in to a value of 1 with shapes with more sides. Concave shapes are more varied and the degree of concavity is directly proportional to the result. Error is also consistent for each size with each set following in the same trend. There is also a clear difference between radial and diameter approaches dependent on certain shapes. Certain geometries produce highly varied results from the radial approach.

The diameter based approach is more complex and theoretical shape calculations become impractical. A limited number of shapes have been evaluated for error in this set in the results in Table 6.7.

Table 6.7. Diameter Aspect Ratio.

Shape	Non Perpendicular Diameter Aspect Ratio								
	Theory	Set 1	Error (%)	Set 2	Error (%)	Set 3	Error (%)	Set 4	Error (%)
Convex Shapes									
Triangle	0.771	0.811	+5.0	0.808	+4.6	0.807	+4.5	0.778	+0.9
Square	0.707	0.703	0.6	0.702	0.8	0.699	1.2	0.696	1.7
Pentagon	0.940	0.898	+4.7	0.891	5.6	0.885	6.2	0.848	10.8
Hexagon	0.866	0.856	1.2	0.853	1.5	0.850	1.9	0.838	3.4
Septagon	0.972	0.950	2.4	0.943	3.0	0.933	4.2	0.908	7.1
Octagon	0.924	0.914	1.1	0.911	1.4	0.905	2.1	0.894	3.4
Circle	1.000	0.983	1.7	0.976	2.5	0.976	2.5	0.951	5.1
Concave Shapes									
3 pt Star	0.687	0.815	16.0	0.807	-15.4	0.783	12.7	0.685	-1.8
4 pt Star	0.375	0.382	5.6	0.365	9.2	0.365	10.2	0.347	-15.2
5 pt Star	0.761	0.746	2.0	0.719	5.8	0.689	10.4	0.613	24.1
Cross	0.447	0.437	2.3	0.433	3.2	0.426	4.9	0.418	6.9
cvx Star	N/A	0.637	-	0.638	-	0.635	-	0.615	-
ccv Star	N/A	0.932	-	0.928	-	0.922	-	0.898	-

The diameter based approach actually has a lower error than the radial approach, however this can be misleading. Unlike the regular trend seen in the radius based approach, the resultant diameter varies significantly dependent on the geometry of the shape and the same types of trends do not appear. Only in some cases will the diameter approach match the radial one. However this likely includes the roundish shape of many food grains. Error trends are also unusual particularly with regard to the concave shapes, which appear to increase and decrease between shapes. However, at time of submission it should be noted there is a risk of calculation error in the three-point star. The author must admit this result could be inaccurate.

The next feature considered is compactness which in Table 6.8. This returns considerably different results than shown in the other descriptors. This feature can reveal different trends not spotted through considering aspect ratio alone.

Table 6.8. Compactness feature

Shape	Compactness								
	Theory	Set 1	Error (%)	Set 2	Error (%)	Set 3	Error (%)	Set 4	Error (%)
Convex Shapes									
Triangle	0.605	0.560	7.2	0.568	5.6	0.577	4.0	0.615	+2.5
Square	0.785	0.724	8.5	0.733	7.1	0.743	5.7	0.787	+0.2
Pentagon	0.864	0.795	8.8	0.803	7.5	0.813	6.3	0.854	1.2
Hexagon	0.907	0.833	8.9	0.843	7.6	0.852	6.4	0.894	1.4
Septagon	0.932	0.855	9.0	0.864	7.8	0.874	6.7	0.915	1.9
Octagon	0.948	0.869	9.1	0.879	7.9	0.886	6.9	0.925	2.5
Circle	1.000	0.904	10.7	0.907	10.2	0.923	8.4	0.980	2.0
Concave Shapes									
3 pt Star	0.444	0.389	14.1	0.382	16.2	0.386	14.9	0.406	9.4
4 pt Star	0.341	0.301	13.3	0.294	15.8	0.298	14.4	0.317	7.6
5 pt Star	0.267	0.229	16.6	0.224	19.5	0.227	17.5	0.242	10.4
cross	0.436	0.393	10.9	0.410	6.6	0.403	8.3	0.447	2.5
cvx Star	N/A	0.129	-	0.133	-	0.131	-	0.133	-
ccv Star	N/A	0.382	-	0.386	-	0.388	-	0.394	-

Compactness error is unusual in that it decreases with smaller scales. This is misleading given that the smaller the shape, the more consistent the value, as the simplicity of the shape results in greater compactness. If a circle was composed of 1 pixel all values would be 1. This circle would be mistakenly be classed as a perfect circle.

The Shape factor results are given in Table 6.9 and Table 6.10. These correlate with the aspect ratio, although following an inverted trend. In Table 6.10 the same can be said with the diameter based approach. Where possible, theoretical shape descriptor values have been determined to highlight error in measurement at that scale.

Table 6.9. Radius shape factor.

Shape	Radius Shape Factor								
	Theory	Set 1	Error (%)	Set 2	Error (%)	Set 3	Error (%)	Set 4	Error (%)
Convex Shapes									
Triangle	0.328	0.389	+15.6	0.396	17.2	0.403	18.7	0.431	24.0
Square	0.172	0.198	+13.4	0.203	15.5	0.207	17.1	0.227	24.3
Pentagon	0.191	0.126	41.9	0.130	43.7	0.136	46.3	0.158	53.8
Hexagon	0.134	0.088	51.6	0.094	43.0	0.098	36.4	0.119	12.3
Septagon	0.099	0.067	48.4	0.072	37.2	0.078	26.2	0.095	4.3
Octagon	0.076	0.052	45.3	0.057	34.5	0.063	21.6	0.080	4.5
Circle	0	0.010	0.9	0.016	1.6	0.017	1.7	0.035	3.4
Concave Shapes									
3 pt Star	0.419	0.540	22.4	0.561	25.3	0.577	27.3	0.636	34.1
4 pt Star	0.419	0.511	18.0	0.529	20.7	0.538	22.0	0.580	27.7
5 pt Star	0.419	0.509	17.6	0.530	20.9	0.543	22.8	0.600	30.2
cross	0.447	0.384	16.6	0.378	18.1	0.398	12.4	0.406	10.2
cvx Star	0.515	0.622	17.1	0.619	16.8	0.631	18.4	0.656	21.5
ccv Star	0.328	0.349	6.2	0.350	6.2	0.355	7.6	0.367	10.7

Table 6.10. Diameter shape factor.

	Set 1	Set 2	Set 3	Set 4		Set 1	Set 2	Set 3	Set 4
	Convex Shapes					Concave Shapes			
Triangle	0.106	0.109	0.111	0.134	3 pt Star	0.106	0.111	0.125	0.190
Square	0.190	0.191	0.192	0.194	4 pt Star	0.467	0.483	0.483	0.503
Pentagon	0.054	0.059	0.062	0.088	5 pt Star	0.146	0.164	0.185	0.243
Hexagon	0.082	0.084	0.086	0.092	Cross	0.378	0.368	0.385	0.375
Septagon	0.027	0.031	0.036	0.052	ccv Star	0.223	0.222	0.224	0.238
Octagon	0.047	0.049	0.053	0.060	cvx Star	0.106	0.111	0.125	0.190
Circle	0.009	0.013	0.013	0.028					

From examining the shape features a number of important conclusions can be made about general shape description:

- Individual features give limited descriptive power, yet even individually some differentiation between classes is possible. For instance the highly concave and highly convex shapes are distinguished by aspect ratio, and there is a gradual increasing trend with the number of sides.

- Several features used in conjunction can reveal more information about complex shapes. This can be expressed as several simple descriptive values. Part of this research intended to find if several different simple features can be used in conjunction to provide more descriptive power, in the hope this may assist in grain shape analysis. An example can be given through contrasting the aspect ratio feature with the compactness feature shows that shapes can be distinguished differently with each feature. The varying concavity shapes could be more clearly distinguished with compactness, yet the dynamic range of values for this feature gives a marginally less clear distinction than aspect ratio.

- At the scales considered compactness generally reduces in accuracy when larger objects are examined.

- Comparing these results with those recorded by Carter in his shape analysis, a different set of trends exist. In Carter's results, many shapes were observed with a decreasing overestimated relative error, eventually turning into a negative error value. This is likely due to the use of larger scales of images returning a trend of falling error. As the intention here was to track the impact of aliasing, small shape sizes were used to the point where considerable aliasing is noticeable. The maximum error carter recorded for his smallest shapes was round 3%, where the small scale here results in much higher margins of error.

As expected the range of values for the different aspect ratio approaches was extremely different, and the trend far less linear than the radius approach. Using full diameter lengths requires measurements through the entire shape from boundary to boundary. With more complex shapes, the measurements appear to become particularly erratic, however general convex shapes still follow a similar trend to the radial approach. The difference between the radial approaches is not necessarily a disadvantage. This is simply a different approach resulting in a different trend for the feature values for shapes.

Figure 6.11 shows a plot of shape area against Aspect Ratio Error. Since the aspect ratio has been determined in theory and practice for all shapes, the values for all sets and varieties of shapes has been determined. This is shown via the Trend line drawn from the data. Also a few sample sets of shapes are displayed in the graph – Triangles, Squares, Circles, Crosses and 5-point Stars. The 4 mean values for each shape are shown in the data, generally conforming to the trend. Overall, shape complexity increases the likelihood of error, yet this is compounded further with lower resolution representations of shapes. A high resolution circle was inevitably going to have the lowest margin of error due to its simplicity. Crosses and Stars, prone to aliasing at lower resolutions, require a minimum size to be accurately represented. The general trend suggests objects with areas of less than 1000 pixels will have shape measurement errors greater than 5%. Shapes with areas <300 pixels will likely experience errors greater than 10% of their shape descriptor value. However the observed trend suggests the error will not reduce substantially in the order of >1000s pixels. This suggests an appropriate scale for objects considered in the digital image, although more aliased shapes such as crosses are less predictable. The digitisation results in poor measurement in the case of aspect ratio. In other shape features almost the opposite occurs.

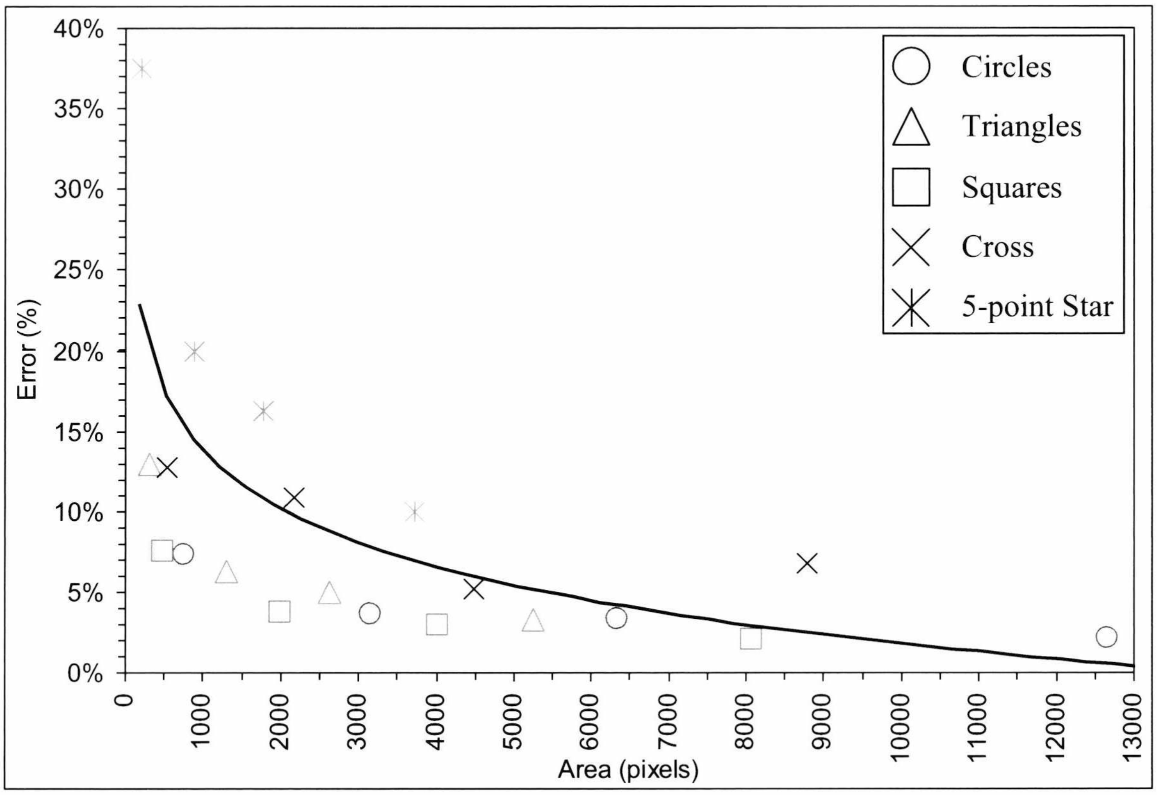


Figure 6.11. Relative error of mean radial aspect ratio vs. mean shape area.

Figure 6.12 shows that with compactness the trend reverses. The trend line derived for all shapes considered (square, star, cross, triangle, circle) highlights the increase of error consistently with area, indicating this feature is not entirely scale invariant. The danger of error varies dependent on the feature. Smaller objects do increase in accuracy of compactness up to a point. Although extremely small objects succumb to aliasing and begin to approach 1, appearing more as a circle than a distinct shape.

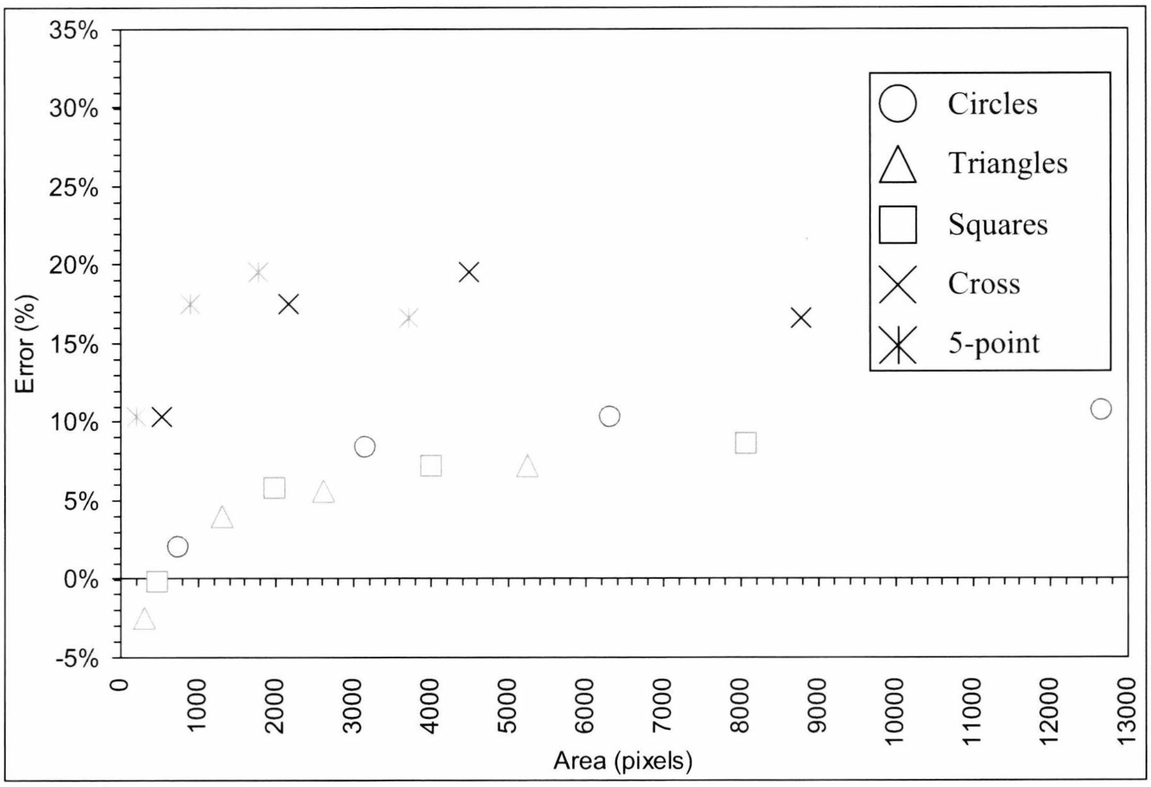


Figure 6.12. Relative error of mean compactness vs. mean shape area.

6.3 Image Sets Created In the Course of Research

Image acquisition of the various materials considered here have relatively simple requirements, which the controlled conditions of the image system are intended to accommodate. Expanding on the previous set-up created by Carter (2005), the imaging box permits precise acquisition of 2D grain and particulate images from above in a relatively uniform illumination environment against an interchangeable contrasting background to improve the reliability of image processing.

The images created for different tests are listed here in this section, listing the different experiments and their purpose. A number of different challenges have been identified for image acquisition:

- Ensuring only intact individual objects are measured. This includes the removal of edge contacting objects and the control of imaging conditions in order to minimise poor quality image thresholds resulting in poor representations of grains. Much of this has been considered in past research although segmentation is new and some alternate methods of applying a threshold are considered.

- Assessing the impact of resolution on threshold quality and measurement of resultant objects in terms of shape, size and texture.

- Assessing the impact of segmentation on the quality of grain representations, and attempting to produce a 'good segmentation' of grains. This is a highly subjective concept and is situation specific.

The following image sets were created in research.

- 2D Test images. Set of images of different sized, different oriented simple shapes for the shape description testing described in the previous chapter. One set of different sized binary images generated by a drawing algorithm. Total of 45 rotations of 4 different scales of 13 different shapes, giving 2340 images of shapes.

- 8 Rice Varieties. Samples of commercial rice images for general shape and texture description. Two sets of data were devised: An Initial set of 9 varieties (two control groups of white basmati rice) obtained with CCIR camera. Total number of grains imaged in the order of several hundred. One obtained with the SD20 camera once initial imaging box complete. Several hundred grains imaged (scale 0.033mm/pixel).

- Basmati & Non-Basmati. Comparative test of size and shape of milled basmati and non-basmati samples with an imaging setup created by Carter. One set of processed results from Carter's system. 2500 grains per set imaged. Second set contrasted with results from the static system using the 5-megapixel SD20 sensor. 228 grains imaged (114 grains/set, 0.033mm/pixel).

- Basmati eye curves. Creation of image training set of basmati rice grains with and without eye curve features, followed up with curvature analysis of the eye features. One set obtained with the SD20 Camera, containing images of grains with the eye feature and without. (0.034mm/pixel)

- Rice segmentation. Images of rice grains in contact with one another requiring segmentation. One set of images obtained with the modified celestron CCD consisting of sets of contacting and segregated grains. This data is compared to other rice sets. 146 grains imaged.

It has been established that a number of features can be extracted from images of rice grains. These include general shape features that can quantify and distinguish rice varieties using pure samples of whole grains. Simultaneous research (Cox and Budhu 2008) with general shape descriptors draws similar conclusions to those made here – In that case, several shape parameters taken together can reveal much about the characteristics of particles. In this case these characteristics are considered for the general identification of rice grain varieties from each other. We also identify some unique rice grain features in line with whitening detection and eye curves, and the use of front lighting permits consideration of the texture of milled and brown grains.

In the present work a digital imaging approach has been devised in order to investigate different types of characteristics to identify different rice varieties. The tests results of rice analysis are now provided.

Eight different common rice varieties were used in tests for defining features. These include existing standards for grain length and diameter based aspect ratio features, but also successfully show the effectiveness of compactness as a feature. Results from The texture feature from chapter 4 are also given through the same processing of the grain images. Several shape features, taken together are used to characterise different rice varieties and identify similar varieties. This work also shows these features generally remain constant for different samples of the same type of rice, even after the rice milling process. All of these techniques are employed in an inexpensive imaging system that is non-intrusive and non-destructive. A highly effective yet simple imaging setup and processing system is established, permitting image acquisition, image processing, and feature extraction. Features are assessed using unsupervised clustering techniques, showing the dissimilarity between different varieties to a degree that would allow successful identification. Rice grain image examples obtained with the respective cameras are given in figure 6.13 through to 6.15.

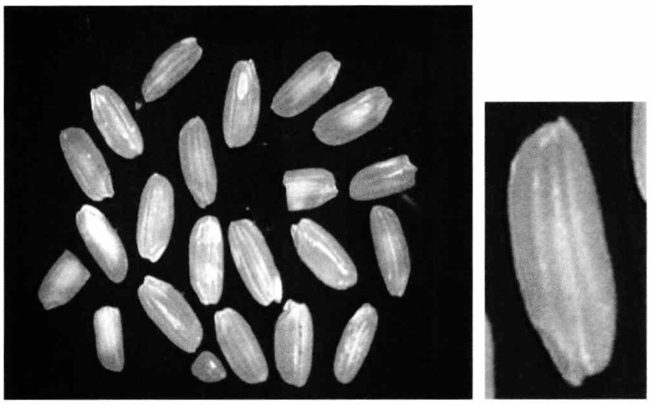


Figure 6.13. ROI and single object imaged with a Sony monochrome CCIR.

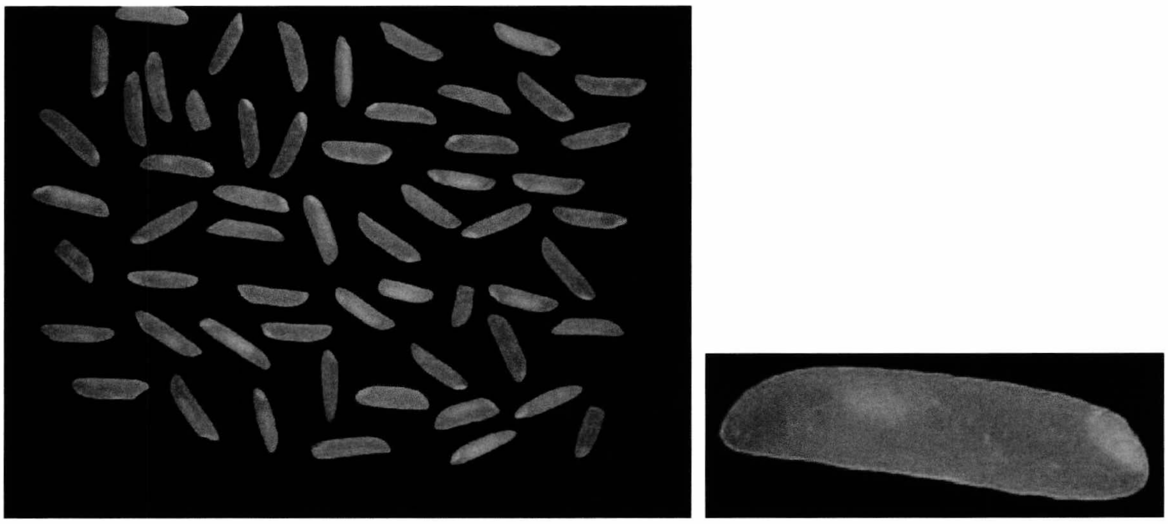


Figure 6.14. ROI and single object imaged with a Canon SD20 camera for set 2.

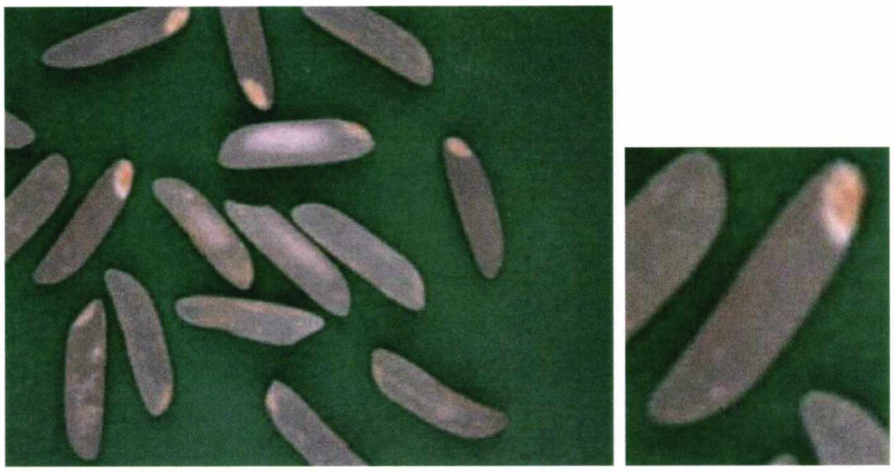


Figure 6.15. ROI and single object imaged with Celestron NexImage CCD.

6.4 Classification of different varieties of rice grains by shape and texture

This section relates to two series of experimental tests conducted using the imaging setup for the published IMTC Conference Paper 'Characterisation and Identification of Rice Grains through Digital Image Analysis'. The first series has produced a set of images of rice grains for eight rice varieties. These were obtained from supermarkets and as a control measure included two different commercial brands of white basmati rice labelled WB1 and WB2. The CCIR greyscale CCD camera was used for this test. From each image, 40-50 individual rice grains were extracted. A slightly higher number were actually positioned within the imaging area yet in a number of cases a small proportion of grains were not reliably thresholded by Carter's function. This was likely due to the imaging with the older camera system and occurred towards image edges, in less uniform areas of illumination, Efforts to modify the function were pursued, including incorporating morphological operators in the counting stage, yet this had a negligible effect on improving the threshold quality. This did only effect a small proportion of grains, usually one or two per image, and the majority of grains were reliably measured. The poorly thresholded grains were omitted through manual identification and image manipulation of the binary images prior to further analysis.

In this early test scale was also not properly considered, yet grains within the images had areas in a range of 2000-5000 pixels. The results from the first test series are first discussed in Figure 6.16. These results incorporate the mean values and standard deviations for all grains of each type recorded from all images in this test set. Error bars in the graph denote the standard deviation. The purpose of this is to show the dynamic range of values for the shape descriptors and the likely values for that rice variety. These features were diameter based Aspect Ratio and Shape Factor, and Compactness.

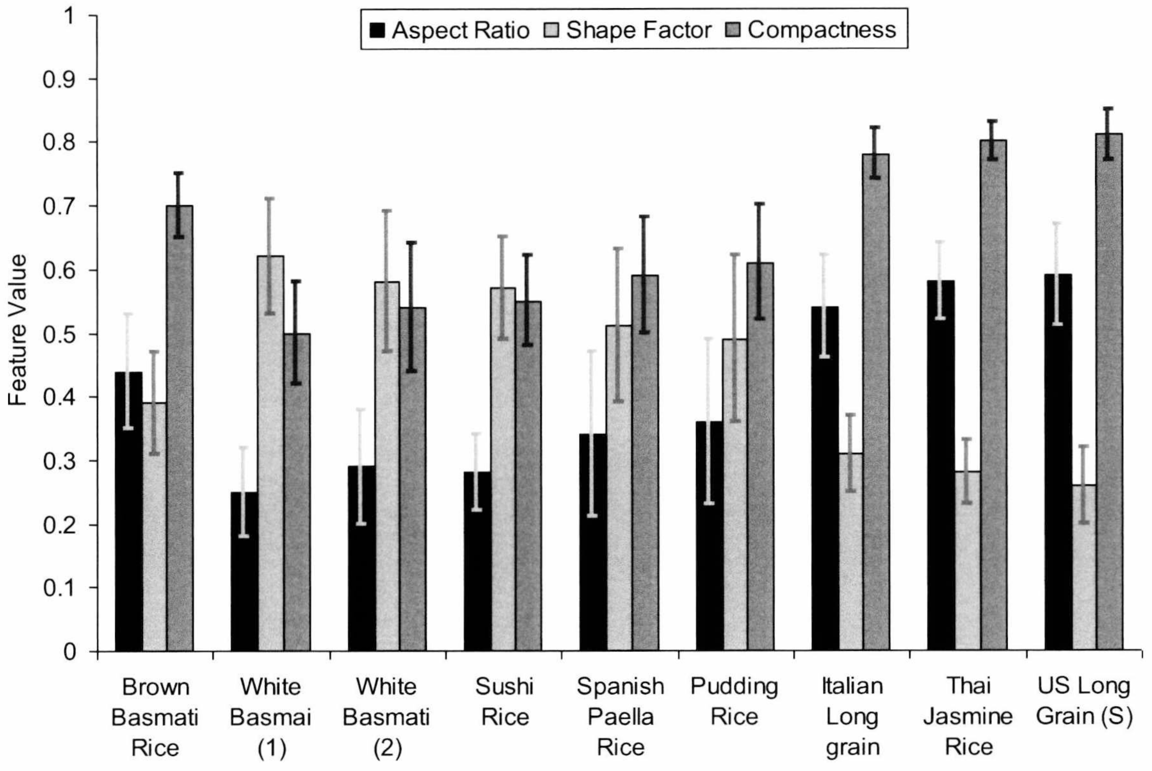


Figure 6.16. Initial shape features from the low-resolution images.

The second test series used images of each grain variety acquired within the desktop image capture box, using the Canon SD20 commercial camera configured to capture in greyscale in order for comparison to the Camera system used by Carter. Grains were positioned on a dark contrasting background in repeat of the earlier test with the CCIR sensor. One crucial difference was the use of a narrower area of imaging within the 5-megapixel image to interrogate for grains. This area was a central rectangle more uniform in illumination than the entirety of the imaging area. In this later test grains were captured with an area of 7000-12000 pixels, relative to the different in scale of the camera system employed. This test did not repeat the WB1 and WB2 due to the similarity of results between the samples. Both of these tests involved application of the initial set of shape descriptors investigated as from the earlier low-resolution test..

Figure 6.17 shows the mean values and standard deviations for shape features from the high-resolution images. The features shown in this table are as stated in the nomenclature. The shape features appear similar in value to those of the low-resolution images. Not surprisingly, standard deviation of the parameters from the high-resolution images is lower than that from the low-resolution images. Yet the low resolution results correlate round the same points as the high resolution sets. This supports the use of a relatively low-resolution camera for reliable, cost-effective shape feature extraction. Example high resolution images of each rice type is provided in figures 6.18 to 6.21.

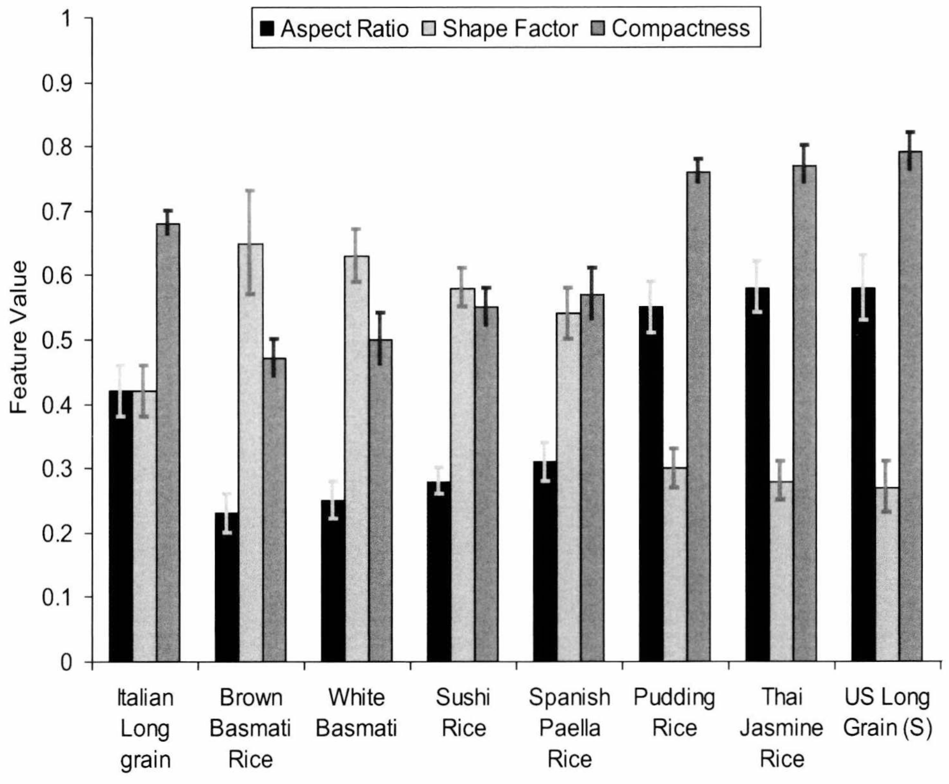


Figure 6.17. Shape features from the high-resolution images

The texture features are notably different as shown in Figures 6.22 and 6.23. The edge detection is shown to be more optimised with the greater resolution and resultant scale of grains captured. The values recorded for Re is more spread ranging from 10-2 to 10-3 for brown basmati and textured surface Italian long grain grains, and to 10-4 for white basmati. This suggests a marked improvement in performance for this measure at a higher resolution.

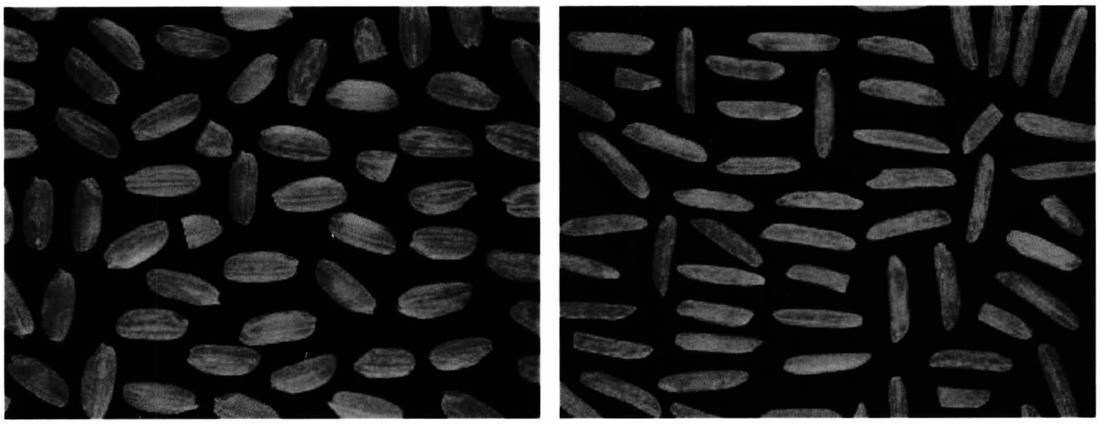


Figure 6.18. Italian Long Grain (left) and Brown Basmati (right) imaged with the SD20.

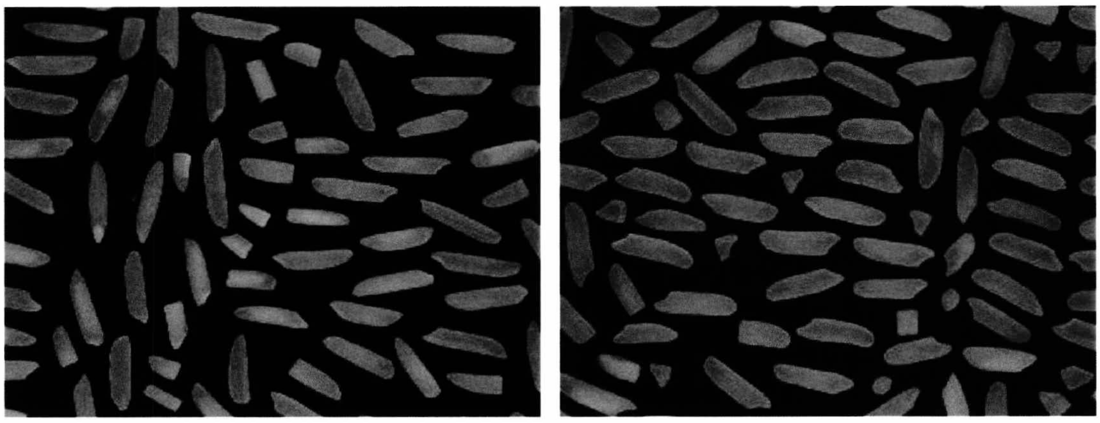


Figure 6.19. White Basmati (left) and Thai Jasmine (right) imaged with the SD20.

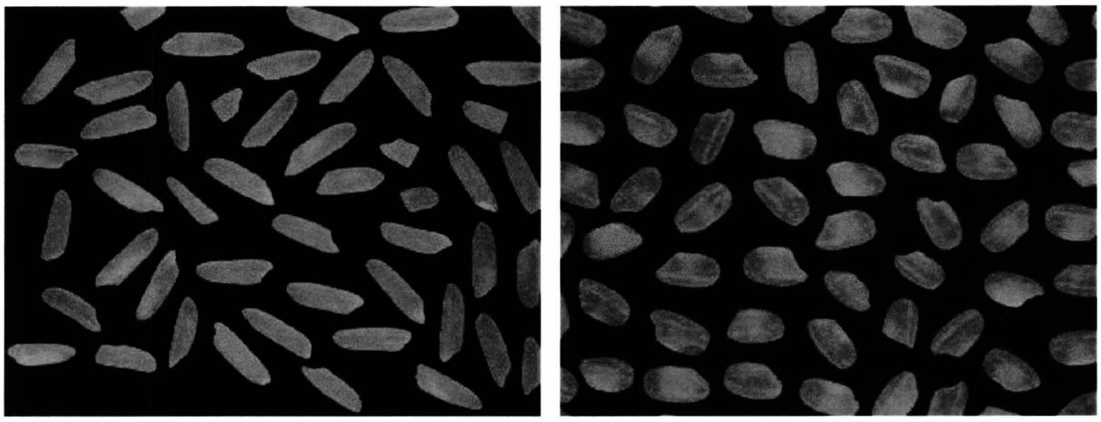


Figure 6.20. US Long Grain (left) and Spanish Paella (right) imaged with the SD20.

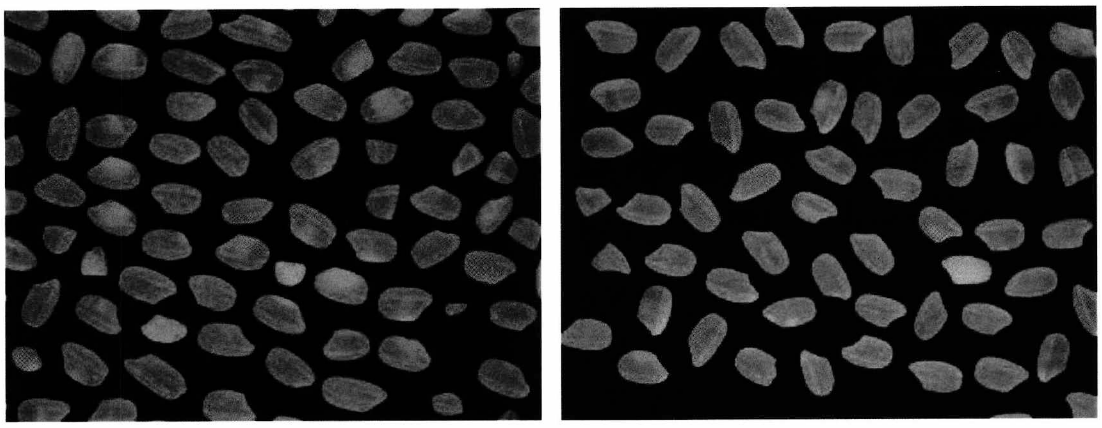


Figure 6.21. Pudding Rice (left) and Japanese Sushi Rice (right) imaged with the SD20.

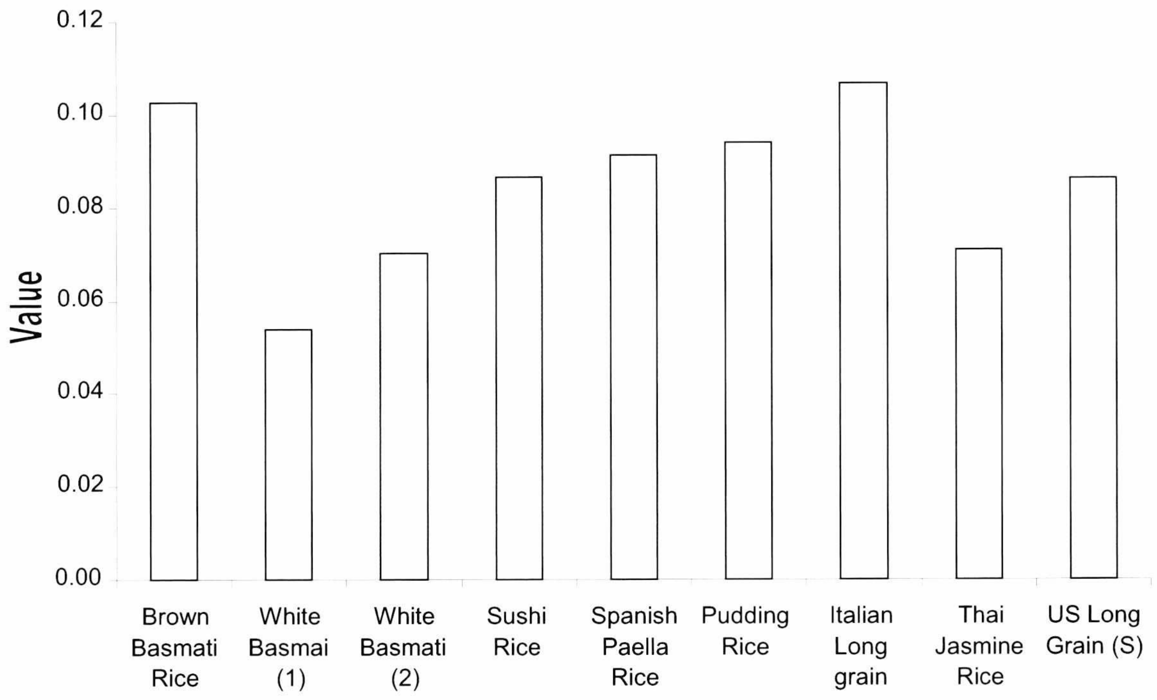


Figure 6.22. Mean texture features from the low-resolution images.

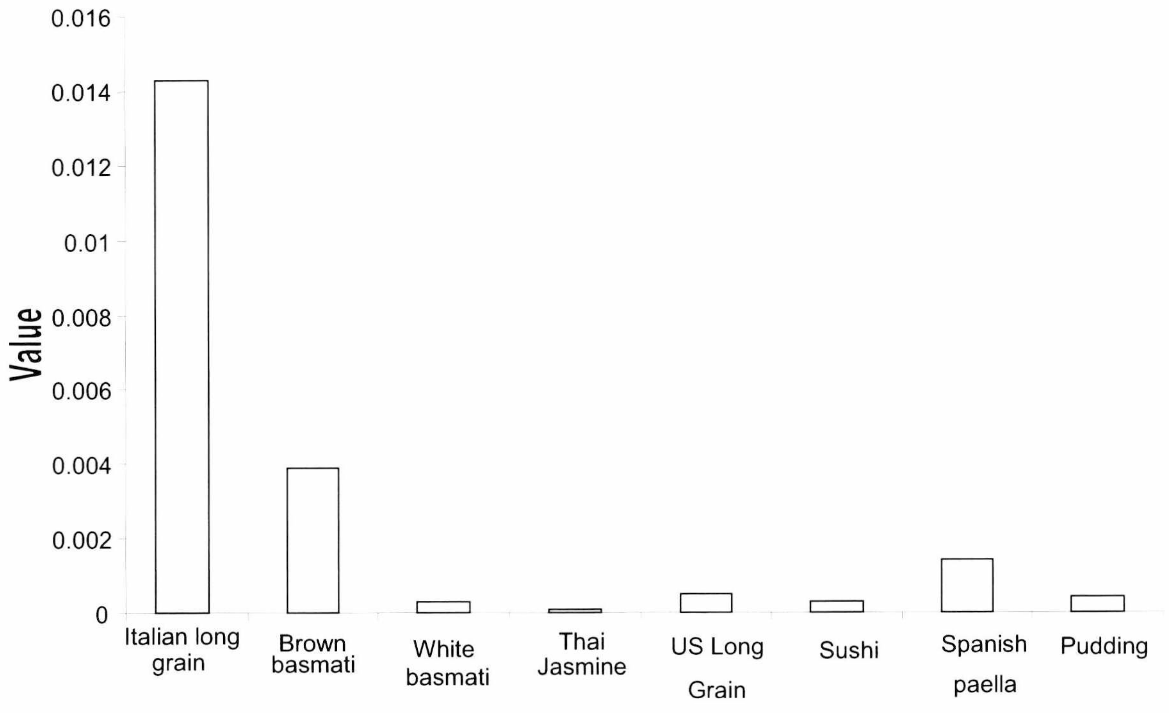


Figure 6.23. Mean texture features from the high-resolution images.

To show the similarity between the different rice samples hierarchical clustering is performed using mean feature values. This produces the dendrogram as shown in Figure 6.24, by applying Ward's linkage to the values. These include aspect ratio, compactness and the edge/area texture feature. Labelling on this diagram is sourced from the abbreviations of the rice grains names as follows: Sus: Sushi. PUD: Pudding Rice. SPN: Spanish Paella. ITL: Italian Long Grain. BB: Brown Basmati. WB: White Basmati. JAS: Thai Jasmine Rice. USL: US Long Grain. The dendrogram correlates well with the key shape and texture differences of the long/medium/short grains and brown/white rice.

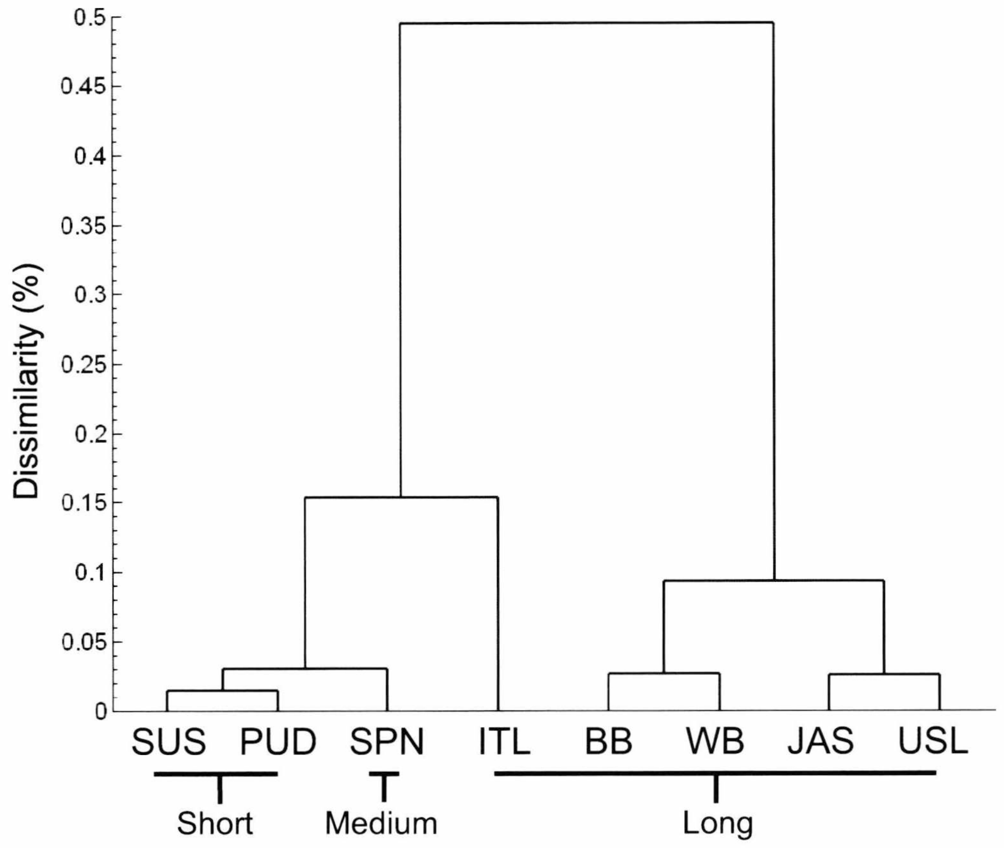


Figure 6.24 Dendrogram of results based on low-resolution images.

The results of the grain length, as shown in Table 6.25, correlate well with IRRI and BRC expectations for rice types. The distinctions returned are expected for two grains in particular - Paella is actually a medium grain despite its shape and Italian a long likewise. The unusual roundness of the Italian Long Grain makes it easy to distinguish with these shape description properties. The average length is critical for differentiation from short grains. This bears a strong resemblance to the way in which shape differentiates Italian Long grain from other long grain varieties.

The same trends exist between the two test series for types of grains. However, all features in the second series have a significantly smaller standard deviation due to the higher image resolution. Compactness appears to offer a complementary descriptive power to aspect ratio and shape factor.

The similarity between the two sets of results suggests that even an inexpensive low-resolution imaging sensor can provide valid shape description and classification of different rice grains. In both series of images, there is a clear separation between long, medium, and short grains. A full distinction of varieties only becomes apparent when applying a range of different features. Texture distinction only appears apparent for Brown Basmati and Italian Long grain varieties, appropriate for characterising the rough surfaced un-milled grains.

Table 6.25. Grain lengths recorded (mm).

Rice Variety	L_a	σ	Classed as	Actual class
Italian long grain	6.71	0.29	Long	Long
Brown basmati	7.85	0.46	Long	Long
White basmati	6.96	0.55	Long	Long
Thai Jasmine	7.09	0.36	Long	Long
US long grain	6.89	0.54	Long	Long
Spanish paella	5.65	0.28	Medium	Medium
Sushi	4.77	0.24	Short	Short
Pudding	4.88	0.34	Short	Short

Strong similarity exists between the brown and white basmati. This is significant in that despite a slight loss in mass through the milling of the outer layer of rice, the general shape characteristics are sufficiently intact for the same shape values to apply to milled and unmilled varieties of rice. Certainly this similarity is sufficient to differentiate from medium and short grain varieties. Taken together, several shape features reveal characterising trends for the different types of rice, with high and low values respectively and consistently for different features.

6.5 Comparative Testing between Basmati and Non-Basmati Samples

A set of images was taken in order to compare results with a series of experimental analysis runs conducted using a purpose-built rice imaging scanner [just insert necessary ref]. The main advantage of this system was that it permitted the rapid analysis of a much larger sample than the static tests due to a feed mechanism and conveying 'turnstile', however the system only records and saves mean values and distributions for each sample. While this permitted an evaluation of a large number of rice grains in a short span of time, modifications to the system were needed to better analyse the data. The only immediately permissible modification to this system was the size ranges of the distributions. Types of features could not be changed at the time, so this limited analysis to area, longest dimension length, mean diameter length, shape factor and aspect ratio. While this permitted some general assessments of trends, it did not explicitly match the required standards of food organisations described in chapter 2. To this end, the static imaging system was used to look at these features specifically.

Samples of two pure milled rice varieties were donated for research by the School of Biological Sciences at the University of Wales, Bangor. These two samples were designated by name and variety numbers as HBC19 (Basmati) and PAK386 (Not Basmati, although a long grain type rice). These consisted of milled white long grains, of extremely similar physical appearance. Measurements considered were to determine if the Basmati variety consistent matched the properties of a Basmati sample described in Chapter 2.

The conveyor system was employed first. This system took images at regular intervals in time, measuring any whole grains within the imaging area. The general properties of these grains are added to various tallies shown in the user interface – mean diameters, areas, shape features, and plots of size and shape distribution across specified ranges. These features are generally self explanatory. Solids concentration is not particularly relevant to this situation and can be ignored. The number of individual grains counted is significant for the scale of sample sizes recorded.

Although it could be possible to determine chalkiness, lack of specific knowledge on constructing valid training and testing sets denied this possibility at the time. The focus here was left on size and aspect ratio properties. The other was similar non-basmati long grain rice, of slightly smaller size. The shape and size of these grains was recorded using Carter's evaluation system. Five sample runs were conducted for each rice variety, each time recording 500 grains giving a total of 2500 readings for each rice variety. However the system in its present state permitted only limited evaluation of rice, in the form of an area and shape feature value distribution, as well as mean values. The distribution of the aspect ratio and mean diameter length recorded for grains was plotted for the combined mean value of all five runs in Figures 6.26 and 6.27. These show the results for a typical run of the basmati and non-basmati samples. There are two methods for expressing length used, the mean diameter used in the turnstile system, and the mean length in the static imaging approach. The difference between these methods is given in Chapter 5 (Figure 5.32).

The results are impacted by a number of physical problems intrinsic to imaging, yet several solutions were developed in collaboration with Dr Carter to improve imaging. First, the poor lighting of the system resulted in many grains towards the edge of the image area being poorly thresholded. This was mitigated by the use of a circular mask region to only obtain grains from the center of the image area. Second, the issue of contacting grains still occurred even with a reduced feed rate. This was managed by ignoring objects of significantly large size, and by significantly slowing the feed rate of the system's conveyor. However, the risk exists that pairs of broken grains could have been wrongly identified with the system as whole grains if in contact.

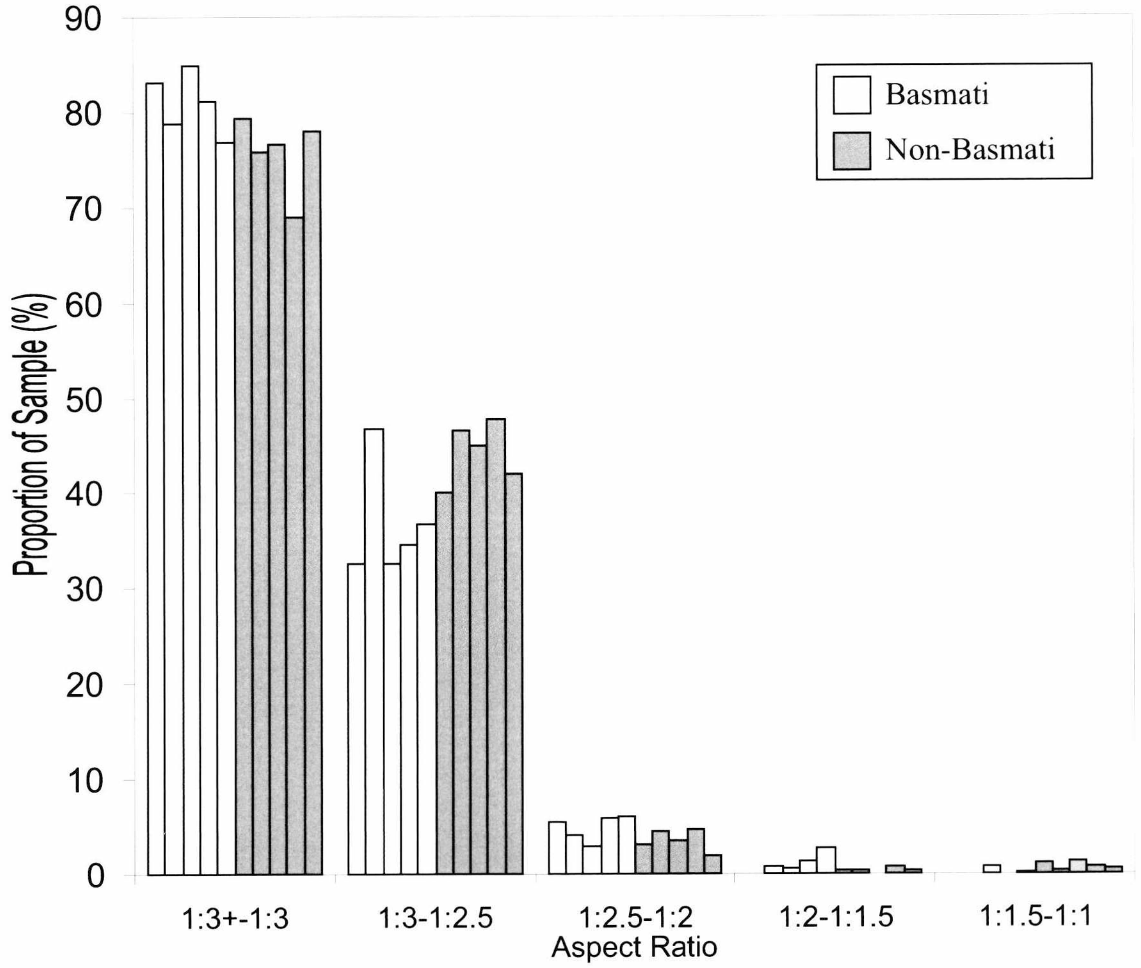


Figure 6.26. Results returned from Carter's System: Aspect Ratio distribution of the grain samples.

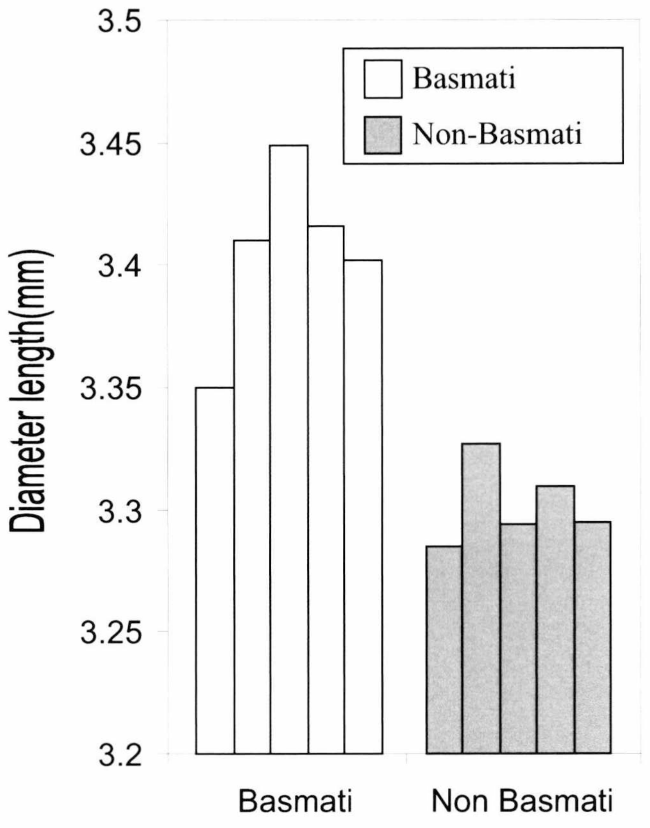


Figure 6.27. Mean diameter of the five runs of grain samples.

While there is some indication of difference between the samples in the data returned by carter's system, the trends are not particularly transferable to the requirements for measurement by the British Retail Consortium and other organisations. By adapting the static imaging system, it is possible to return measurements closer to those sought by the organisation, particularly the highly relevant grain length property.

A training set of 50 Basmati and 50 Non-basmati grains was created from thresholded images of grains obtained with the SD20 camera. Figure 6.28 denotes the mean length per grain for each sample of Rice obtained with the static imaging setup on the left, and the proportion of the sample above the required standard on the right. Although in both cases the mean is above the requirement, the desired proportion of the samples is not above the required standard for the non-basmati rice. This is indicative that the rice is generally shorter, and is coherent with the size distribution trends recorded in figure 6.26. Similarly, aspect ratio trends are given in Figure 6.29 using radius and diameter based aspect ratios.

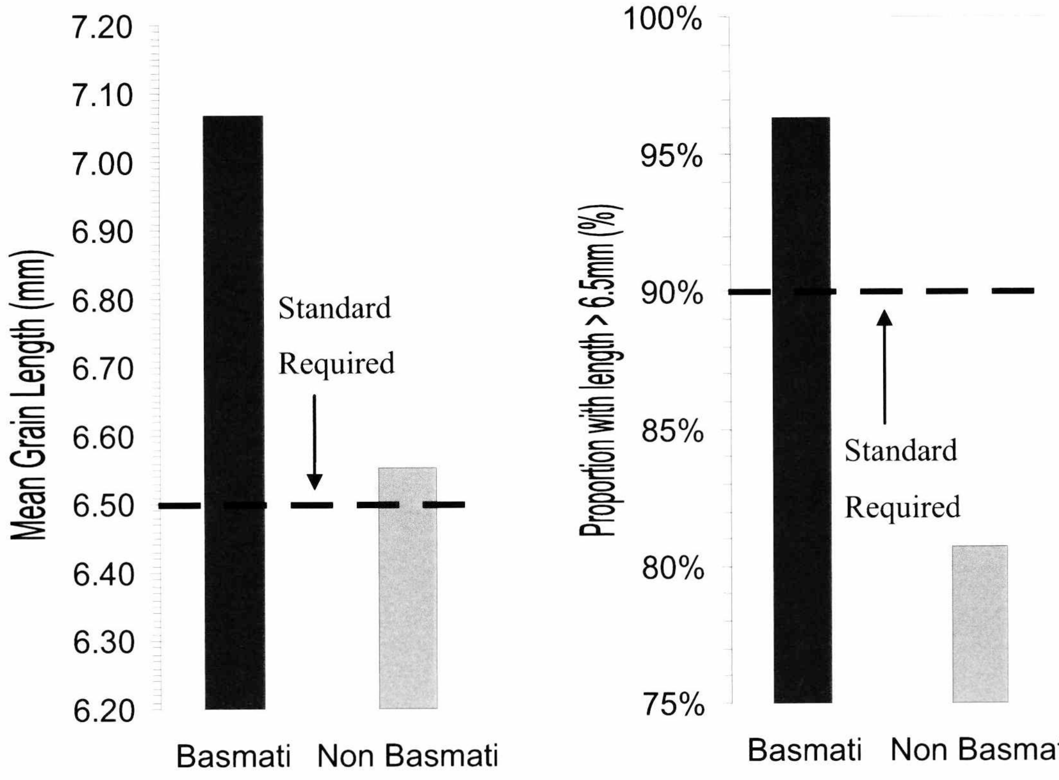


Figure 6.28. Static image results for Basmati and Non-Basmati grain lengths.

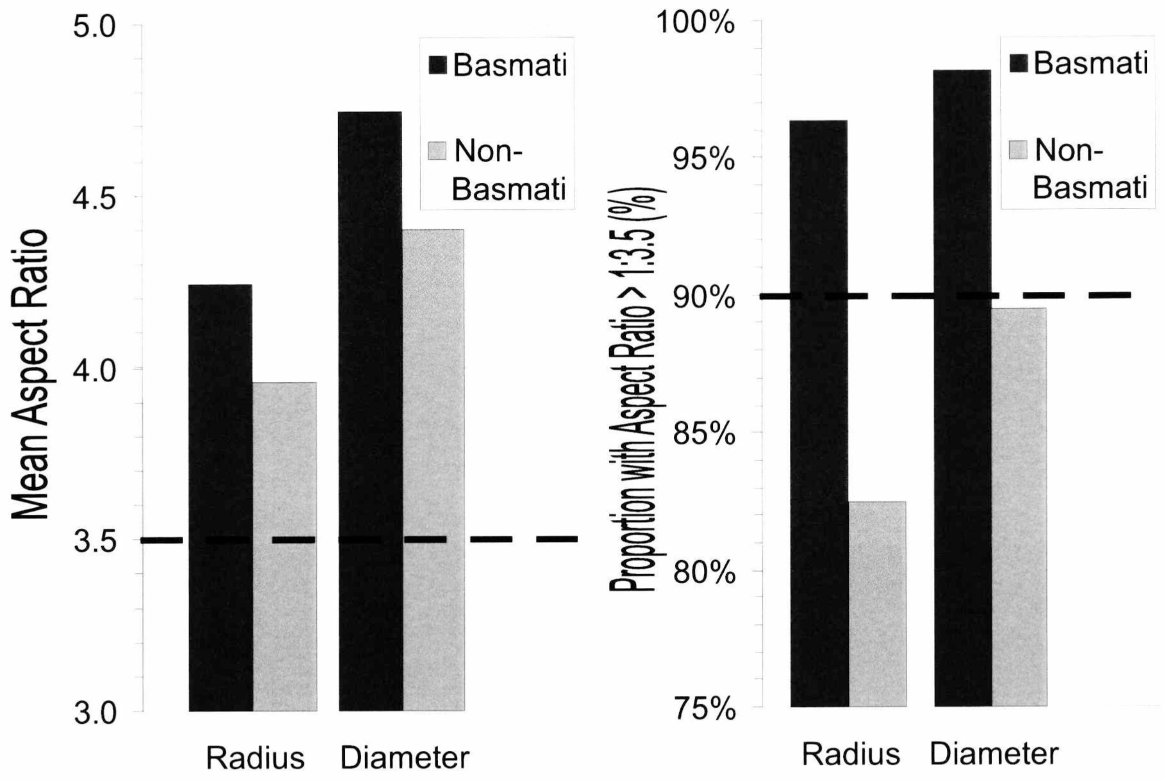


Figure 6.29. Static image results for Basmati and Non-Basmati aspect ratio.

The BRC standards for Basmati Rice require that no more than 7% of the sample not be classifiable as Basmati rice, and no more than 10% be broken grains. Assuming the case is that 10% of the grains being broken constitute grains of insufficient aspect ratio and size, the results here are coherent with the trends. However, there is a clear difference in range of values for the choice of aspect ratio method. There is some degree of subjectivity of the relevance of these results, however the different trends between the non-basmati and basmati samples is visible in both the shape and sizing results from the static system and the distribution results of the dynamic system.

6.6 Dedicated rice grain eye detection

The detection of the IRRI eye feature relates to the presence of the aforementioned concave 'eye socket'. Digital image processing methods have already permitted repeatable characterization and measurement of rice grains and other types of materials (Carter, Yan and Tomlins 2005). Such classification tasks are often aimed at finding criteria to distinguish different varieties by their general shape properties rather than a single feature within the same variety. There is very little reported work in the literature on this topic. In the present work we attempt to explore methods for the detection of the presence of specific curvature, illustrating these approaches with idealized grain examples. In practice this requires image processing of grains to produce a two-dimensional binary representation. These are optimal for shape description based techniques such as curvature functions (Xu, Luxmoore and Deravi, 1997, 1998). In these tests gradient functions of a curvature shape signature are investigated and contrasted with Fourier methods for detecting the presence or absence of the desired curvature, indicative of the presence and location of the eye feature. Specific Fourier coefficients are identified based on the curvature to find the eye feature in order to demonstrate that they can be used efficiently for practical applications.

The detection of these features are of relevance to rice grain quality measures and attempts to detect the presence or absence of rice grain embryos are not widely reported in existing literature. Efforts to describe it from information available from IRRI sources have led to a classification illustrated in Figure 6.30, with an annotated image of rice grains captured with the camera system developed as part of this research. Here can be seen a clear instance of the presence of the grain embryo, its absence after being physically removed, and an unclear case without an embryo that might not be so easily classified based solely on the shape of the grain. The socket like feature commonly occurs in milled rice. While cases exist where shape only may be sufficient to determine embryo absence, texture and intensity properties are also relevant in order to say confidently that the characteristics of the grain are with or without the presence of an embryo.

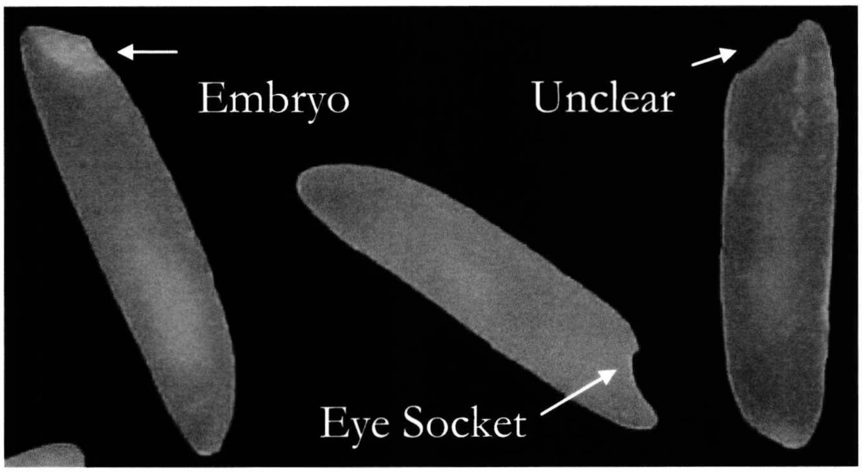


Figure 6.30. Grains with an embryo, eye socket, or a potentially unclear edge.

6.6.1 Imaging for rice grain eye detection

The 5-megapixel colour CCD camera was employed again with the static imaging box with chroma keying of grains against a coloured background. Following spatial illumination testing and scale calibration, the grain samples are subject to testing. The result of the image processing is a static binary image of rice grains from which shape descriptors can be derived. Figure 6.31 shows a processed image of Long grain rice with and without the presence of the distinctive rice grain eye feature. This was produced with the application of Carter's image threshold function. Grains with eye curvature present have a distinct end clearly visible in contrast to the rounded ones. Analysis of this image proceeds with labelling of each object in the image, then extraction of the boundary pixels of each shape, interpreting it as a 'signature' for further analysis.

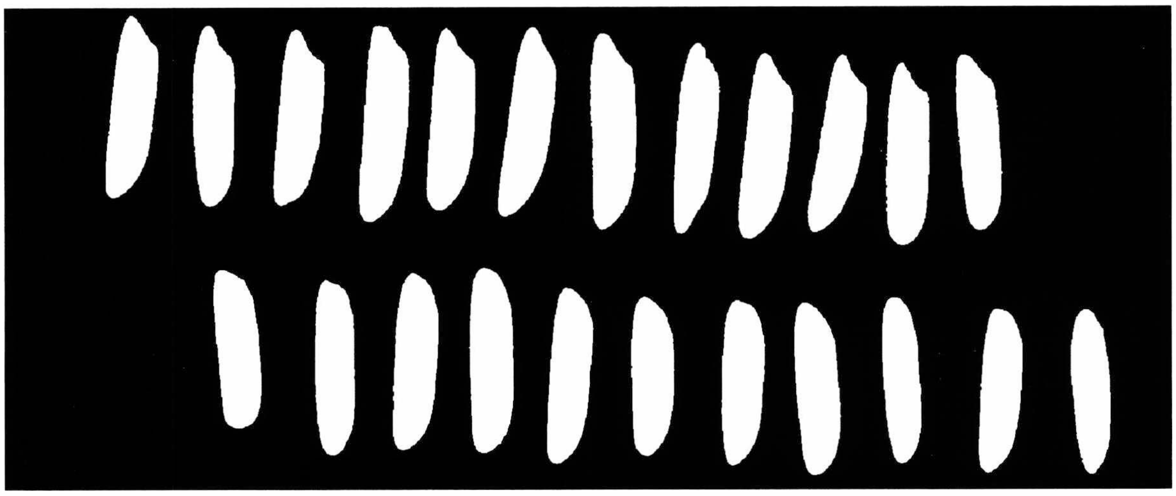
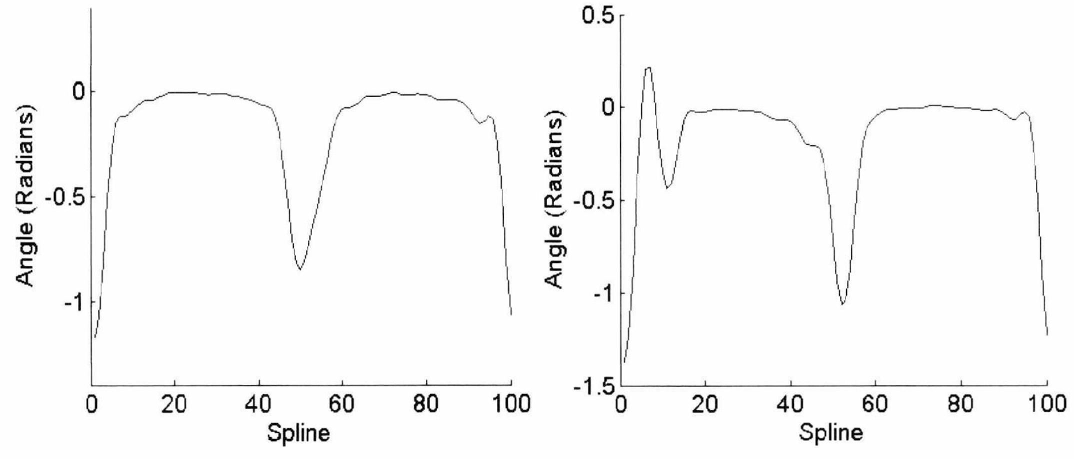


Figure 6.31. Example long grain rice with (top) and without (bottom) the presence of the curvature feature, denoting an eye socket.

A total of 224 milled grains of the HBC19 Long Grain Basmati rice were captured in images, processed and had their signatures recorded. These samples were provided by the University of Bangor, along with a highly similar variety of non-Basmati rice used in another test.

6.6.2 Curvature Analysis for Eye Detection

Half of the images obtained were grains specifically positioned so that curvature on one end was present. Half were grains where no curvature existed. The data was divided into two sets both of 114 grains rounded/with embryos or curved/with eye sockets. For initial testing and validation purposes subsets of 50 rounded and 50 eye grains were used. Figure 6.32 shows the averaged curvature angle, illustrating the trend sought between these two types of rice grain shape. This is normalised to the same scale as Figure 5.17, then sorted from the point of most negative (convex) curvature in order to provide a consistent representation [need to describe this process]. Plots are given for the two 50 grain training sets of the rounded (Figure 6.32a) and the eye (Figure 6.32b) grains. The 'hills' and 'valleys' in Figure 6.38b represent the desired feature. There is a strong correlation between what has been observed here with the theoretical results of similar shapes given earlier in chapter 5.

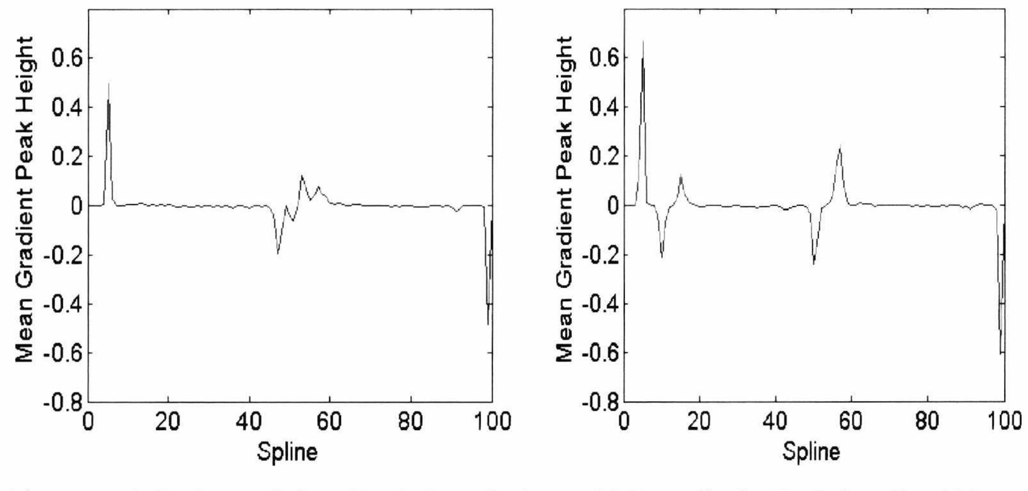


(a) Mean of Round Grain Training Set (b) Mean of Eye Grain Training Set

Figure 6.32. Signatures of Curvature derived from constructed training sets.

6.6.2.1 Gradient Peaks of Curvature

By counting six peaks in the gradient approach a visible set of positive and negative peaks is detectable. The mean values are plotted for the six largest gradient peaks from the two training sets in Figure 6.33, resulting in a clearly visible set of positive and negative gradient peaks. These peaks are consistent in the sorted data and have reasonable correlation with the simplified theoretical shapes.

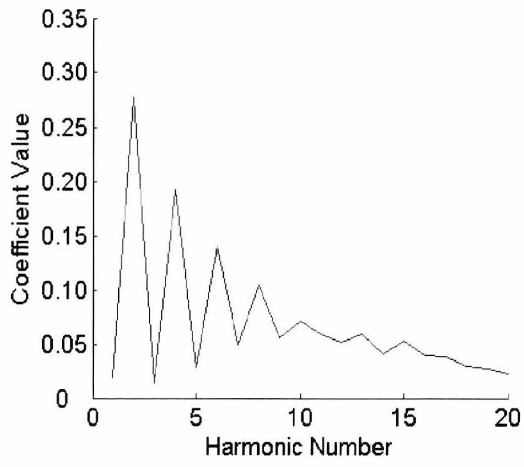


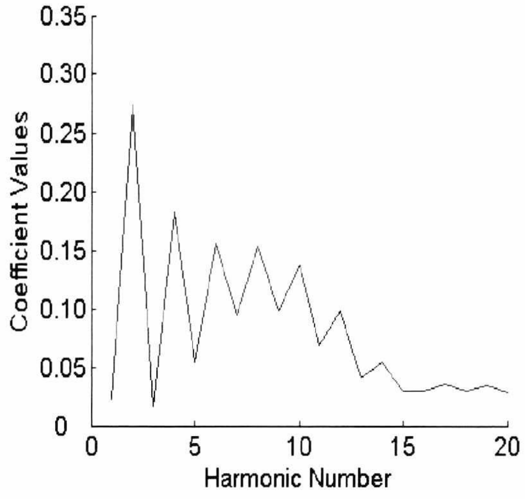
(a) Round Grain Training Set (50 grains) (b) Eye Grain Training Set (50 grains)

Figure 6.33. Significant gradient peaks with respect to functions *G* and *F* (note the correlating peaks for the different sets).

6.6.2.2 Fourier Coefficients of Curvature

The resultant Fourier coefficients and gradient peaks from the training sets are also plotted to highlight significant differences in the sets. The plot for the mean values of the first 20 Fourier coefficients for the entire training sets are shown in Figure 6.34. It can be seen that a significant difference exists between the two grain shapes from 7th to 12th coefficients. The even numbered harmonics reflect significant curvature in the shape perimeters. Much like in the gradient approach, the eye set results have six significant peaks again, and the round set results have four. This is applicable to differentiation between the two grain shapes.





(a) Round Grain Training Set (50 grains) (b) Eye Grain Training Set (50 grains)

Figure 6.34. Means of the first 20 Fourier coefficients of curvature.

Further analysis of the plots suggests that the Fourier descriptors detect a difference between (a) and (b) in Figure 6.33. The gradient peak approach is useful for highlighting the significant number of peaks in the curvature which directly correlates with the number of significant Fourier descriptors. If only the appropriate Fourier coefficients are used, then the Fourier approach will be simpler to implement than the peak approach. This would be beneficial for an autonomous classification system. However, there is still some overlap between even the best coefficients shown in Figure 6.34 suggesting alternative approaches were still worthy of consideration.

6.6.3 Analysis of Features

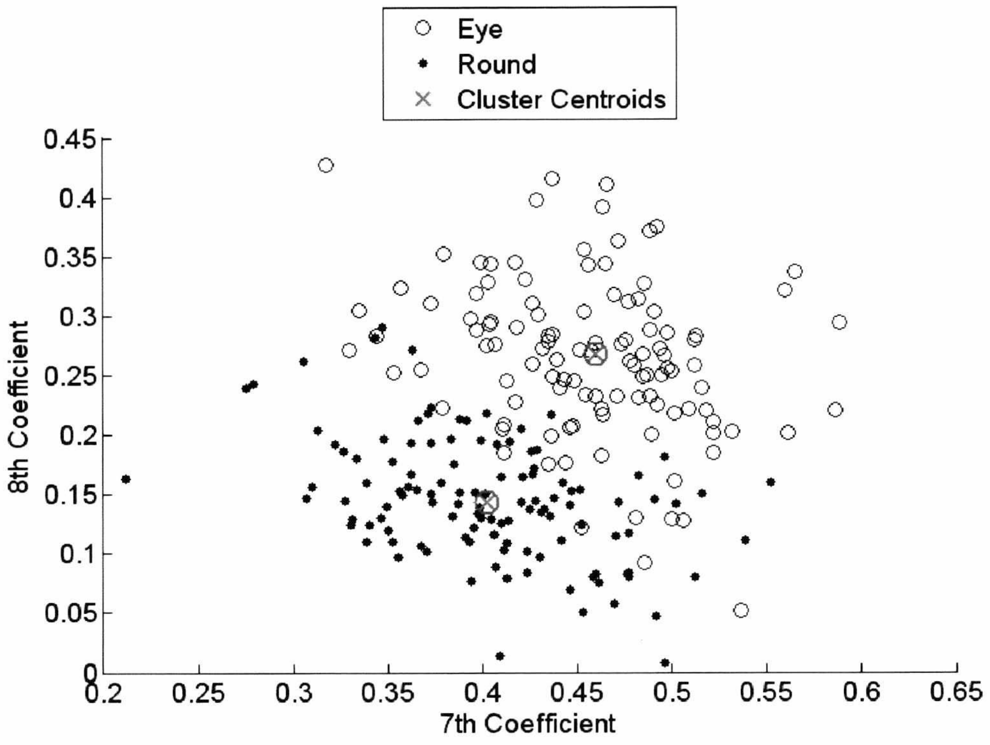
Clustering of this data makes it possible to assess wherever the data is linearly separable by identifying the proportion of grains which are positively and falsely identified as possessing the eye curve feature. This is achieved through combining a typically unsupervised k-means clustering approach with knowledge of the training sets to determine the accuracy of cluster assignment. The clustering approach is used to partition the data into two sets.

This approach was taken separately with the different analysis methods – separate clustering accuracy results are produced and given for the Function of gradient, Fourier Coefficients, and numbers of zero crossings. These results are discussed and compared in terms of accuracy in representing the training sets.

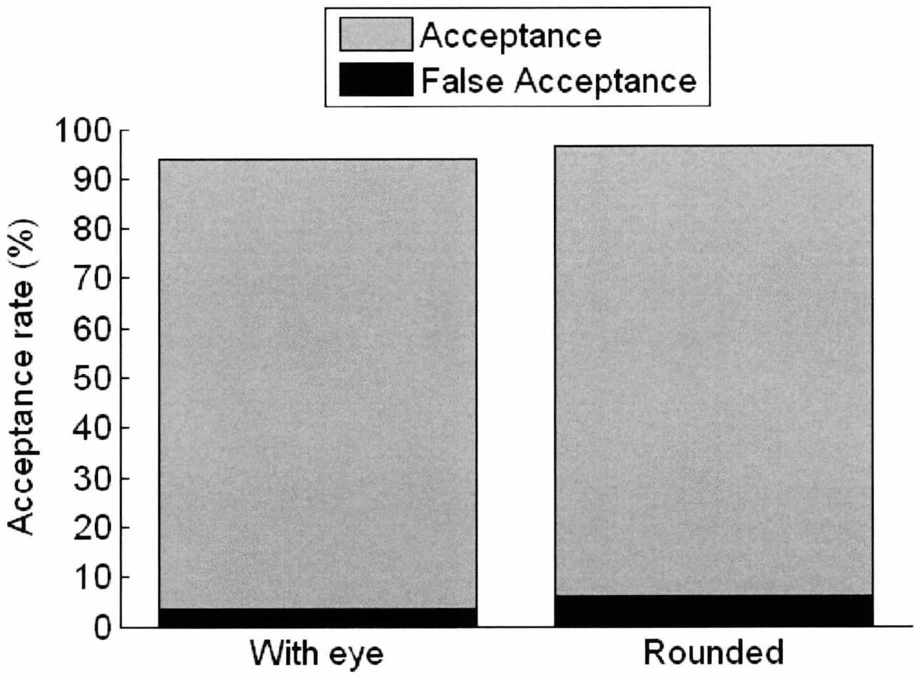
6.6.3.1 Analysis of Fourier Coefficients

The entire set of 228 grains was combined for a cluster analysis. K-means clustering was employed, with 2 cluster centres to reflect the two sets of grain types. For this clustering a MATLAB k-means function was employed. The following parameters were set used in its final implementation: A city distance function for the distance between data points and cluster centres. A total of five replications were used in all cases to ensure the random cluster allocations were sufficiently consistent.

In initial research the 7th to 12th Harmonics were used as the input data. Within the two sets, 93.9% of the 'eye' set were correctly assigned to the same cluster, and 96.5% of the 'round' set were correctly assigned. The other grains were misclassified in the other cluster. The inaccuracy in the eye set can be accounted to irregular curvature in a small proportion of the grains, which is less obviously apparent to a human observer as well as the system. 3.5% of round grain were inaccurately classed as eye grains, and 6.1% of eye grains were inaccurately classed as round. This conclusively indicates a good standard of accuracy using these features. Figure 6.35 illustrates the plotted coefficients 7 and 8 (a) and a final bar graph (d) for the accuracy of the k-means classifier based on coefficients 7 through to 12.



(a) Distribution of coefficients 7 and 8.



(b) Plotted accuracy of the k-means clusters.

Figure 6.35. Plots of coefficients of curvature, with results for clustering with coefficients.

The investigation of Fourier coefficients was expanded to considerer a broader range of combined coefficients. By employing a strategy of optimization, the clustering function was repeated applied with different sets of coefficients in order to find the most accurate set. This led to and expansion of the coefficients used. In the end, the training sets were improved in accuracy to 99.1% accurate for the round set classification, and 93.9% accuracy for the eye set classification. This suggests high reliability for the detection of the eye feature, yet a slight risk of false acceptance of rounded grains as curved ones, possibly due to grain elongation like the 4th grain in figure 9. This was achieved by utilizing the 2nd through to the 12th coefficients. This final set is considered adequate for reliable eye detection with the Fourier approach. Additional coefficients appear to only hamper the clustering process as there is no longer a significant detectable variation among the other coefficients, and utilising only the 7th to 12th coefficients also causes a slight loss in reliability. Tables 6.36 denotes the clustering accuracy for the different analysis strategies employed, and generally the best results obtained. The results of using an increased number of less reliable coefficients (1-30) are also less effective than the subset of ones identified as potentially effective in Figure 6.35.

Table 6.36. Acceptance/Rejection rates for different Coefficient analyses of grains.

Coefficients	Correct eye detection	Falsely classified with Eyes
2-12	99.1%	6.1%
7-12	96.5%	6.1%
1-30	93.0%	7.9%

6.6.3.2 Analysis of Gradient Peaks

Results for the gradient peak approach are less accurate. The results generally cluster according to the number of peaks, where less than or equal to 4 generally the condition for the round grains. Greater than or equal to 5 was the condition for the eye feature grains. However when applied to the complete set of 228 grains the present clarity of distinction is not as high as that of the Fourier approach. The distribution of peaks and accuracy are shown in figure 6.37 and 6.38. Plotting the peak distribution for the sets of rounded and eye grains make it immediately apparent that a proportion of grains are easily misclassified with the k-means approach. Round grains are successfully assigned to a k-means cluster with an accuracy of 89.5%, and eye grains with an accuracy of 78.9%. The false classifications leading to false acceptance in the wrong clusters for the grains is also higher than the Fourier approach, at 10.5% and 21.1% for round and eye grains respectively. It may be possible to refine the gradient approach however adapting the long established Fourier coefficients for this task has been proven effective.

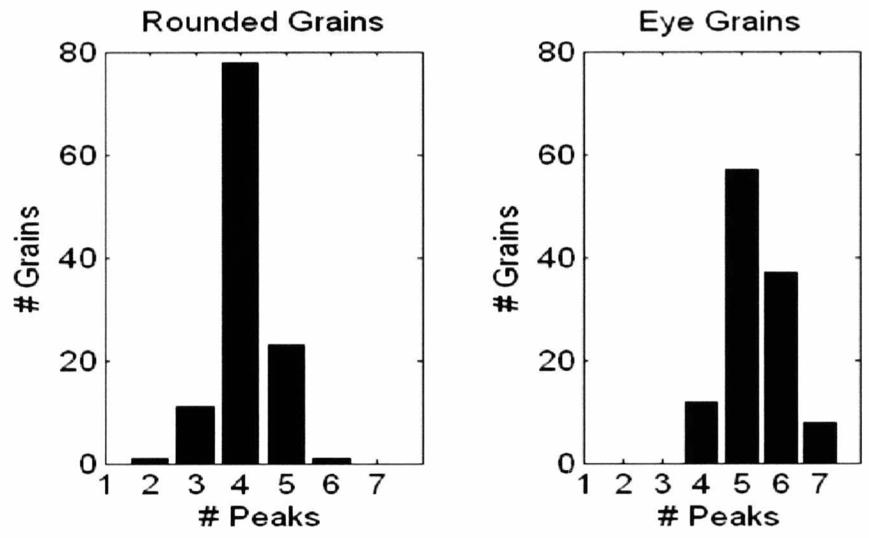


Figure 6.37. Plots of peak distributions for different sets.

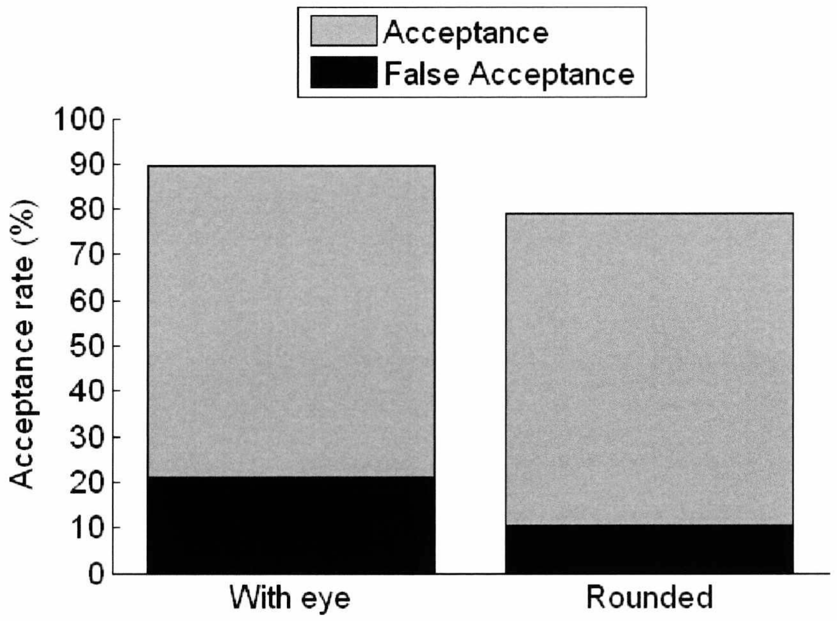


Figure 6.38. Accuracy of resultant k-means cluster assignments.

6.6.3.3 Analysis of Zero Crossings of Gradient

The number of zero crossings was retrieved from the gradient of the curvature of grains from a training set of 112 rounded and 112 eye grains. The zero crossings for the 224 grains are shown in Figure 6.39 for both complete sets of eye and rounded grains. This set was derived with a gradient threshold 0.12. This gradient threshold was determined by the processing of the training sets with a zero crossing identification function. The full function is extensive but essentially follows the following steps:

- For the recorded curvature for each grain in the round set and each grain in the eye set, retrieve the gradient.
- Apply the zero crossing detection function at a range of magnitudes (0.040 to 0.140 ranging by 0.001).
- Record, sorting by which set the grain is from, the number of zero crosses detected for each grain at each magnitude.

This produces a matrix of zero crossings of grains for each set by the different magnitude. Figure 6.39, showing the sets for one specific magnitude shows there is a clear set of grains with four zero crossings which can be positively identified using this criterion. There is a set of grains with 3 zero crossings of both grains, although the overwhelming majority are eye grains. The results suggest a threshold of three zero crossings support a high probability that the grain has the eye feature.

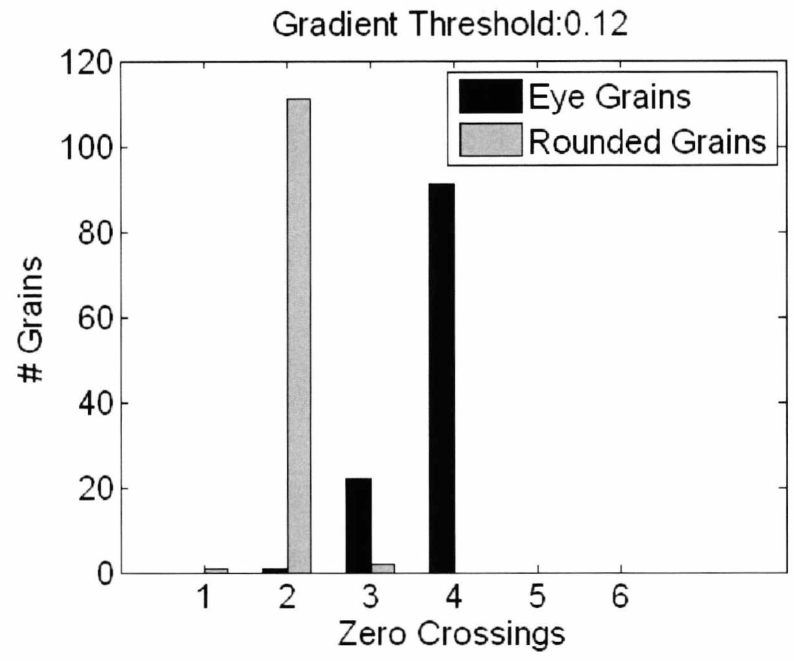


Figure 6.39. Zero crossings for the training set grains at the specified threshold.

The range of thresholds are explored to find an optimal setting for the gradient magnitude. This approach requires a supervised approach to analysis, and determining an optimal threshold point from a training set is pivotal for its implementation. An investigation was taken to find an optimal setting for the gradient magnitude threshold. This led to an effort to find a threshold that would produce a minimal amount of false eye acceptances and false round grain acceptances. To do this, the gradient threshold point was reapplied to produce an accuracy value for each level. Figure 6.40 shows the resultant plot of classification accuracy for the k-means clustering based on the number of zero crosses in the gradient, dependent on the minimum threshold for the crossings. The data is approximated due to its scale using a 6th order polynomial trend line and is intended to be indicative of an optimal threshold point for the zero crossing gradient.

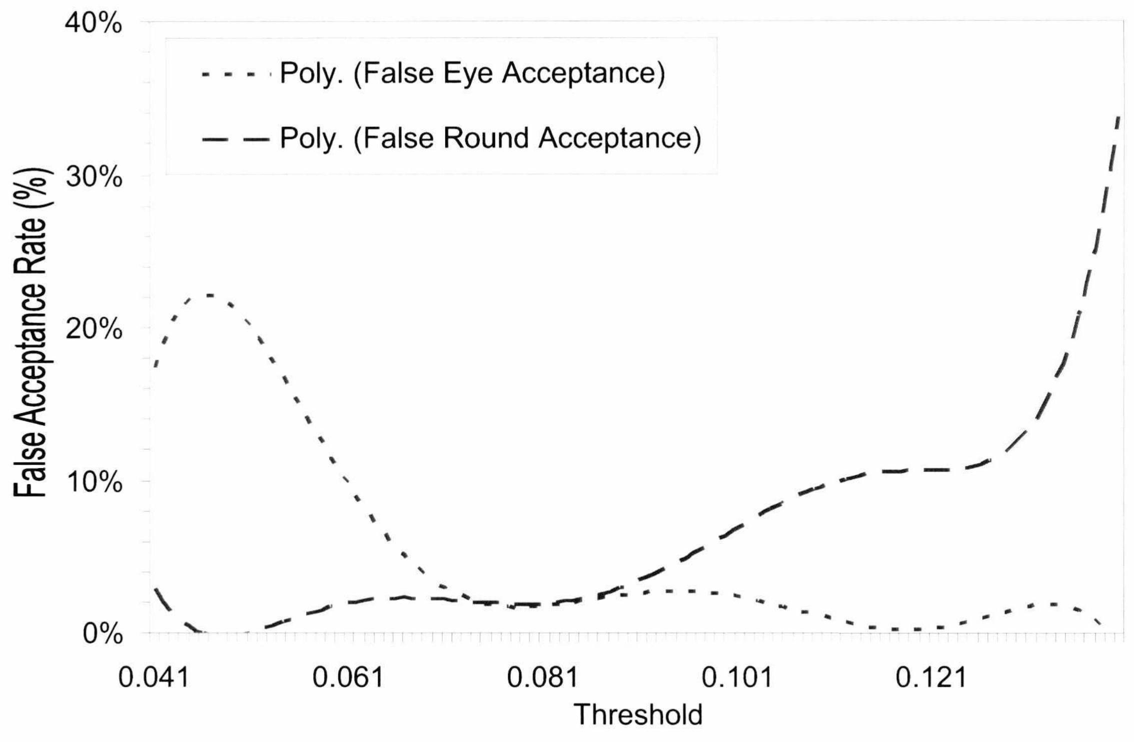


Figure 6.40. Trendlines for acceptance/rejection rates for rice grain eye detection at different zero crossing magnitudes.

The trend in this data is as follows. The higher the threshold, the decreasing of the likelihood of round grains being falsely classified as eyes. This continues up to a point where the eye grain acceptance rate itself begins to fall. The best balance, as was the case with fourier coefficients, is to minimise the risk of false acceptances and maintain high acceptances. Thresholds in the range of 0.8 to 0.1 appear to have lower risks. A closer evaluation of each set of results for this range is given in Figure 6.41. The trend suggests an accuracy of 99-100% is achievable yet this is offset by a slight risk of false acceptance (nominally less than 5%). In any case, a smaller set of eye grains with a higher number of detected zero crossings will always be correctly detected. This is proportional to the number of grains with four zero crossings shown earlier in Figure 6.39.

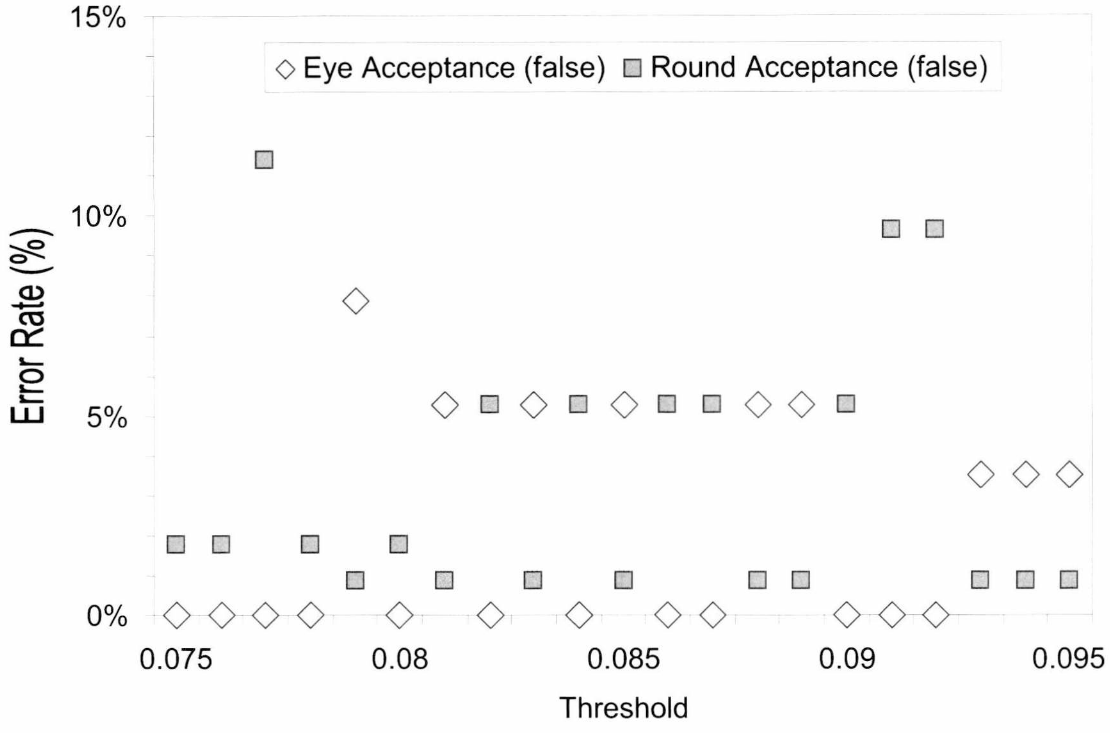
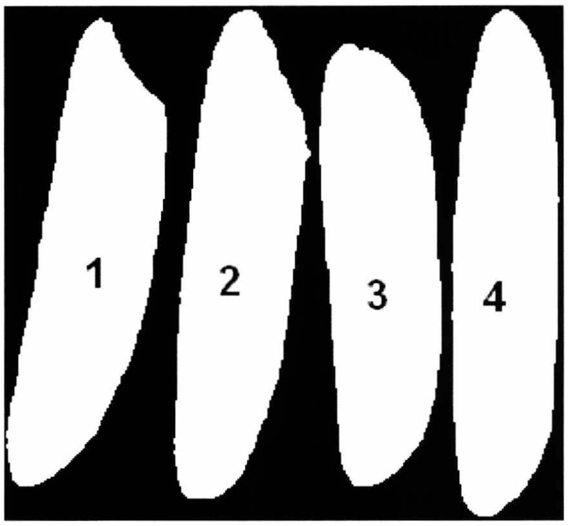


Figure 6.41. Threshold range 0.075 to 0.095 from Figure 6.45.

Results show a high degree of reliability for detection of rice grain eyes using a minimum value for the gradient magnitude threshold in the region of 0.075 to 0.095. Error rates between 0% and 6% are possible in this range with the highest false acceptance being with eye grains falsely classified as rounded. Performance is comparable to the Fourier coefficient approach.

6.6.3.4 Discussion of eye detection results

A small part of the error that exists can be attributed to the quality of the training and testing sets used. Although care was taken to ensure grains in each set had definitive curvature present or absent, other factors may have impacted this in measurement. The curvature recorded in a number of grains was notably weak in part of the eye curve set, which in turn became difficult to detect with either approach. The orientation of some of the grains within the image also occluded most of their curvature. This is entirely possible in practice as the curvature is often only visible at certain orientations of the grain. However, given that the majority of grains were effectively positioned to highlight their present curvature, the grain sets are considered to reflect reality. A number of grains in the rounded set were particularly elongated, which appears to have been sufficient to distort the end results. These factors need to be accounted as issues in accuracy. Figure 6.42 shows a number of grains in the eye and rounded set that fall into the accepted and false accepted categories, according to the earlier clustering of the coefficients.



(1) Recognised Eye. (2) Unrecognised Eye.
(3) Recognised Rounded. (4) Unrecognised Rounded.
Figure 6.42. Four grains from the final set.

Methods were employed to diminish error in final testing, such as the training to find an optimal feature set for the Fourier coefficients and for the gradient threshold range. However borderline cases such as Figure 6.42 (2) and (4) are still capable of failure. To conclude, Fourier methods are quite reliable when specific sets of optimal Fourier coefficients have been identified, compared to the high risk of false positives in use of the gradient function.

6.7 Discussion of Rice chalkiness and embryo detection efforts

The whitening of grains was detected using the textural information of the grain. To specifically locate a patch of whitening in the grain, the method illustrated in chapter 4 was employed. The mean subtraction and morphological operations employed resulted in many instances where whitening and embryo features, denoted by a bright patch, became visible. However, this process was not reliable in all cases.

Depending on the positioning of the presence of whitening within a grain, the whitening feature can be connected to the embryo feature. This can make detection of the embryo difficult if attempting to distinguish the two. It is important not to confuse the embryo with whitening, particularly if the whitening is in a grain which clearly does not possess an embryo, for instance, if the grain had prior been assessed as possessing the eye socket curve as described in the previous section.

6.8 Segmentation and analysis of grains

Segmentation is practical given the contact that often occurs between conveyed particles in the imaging system devised in the Embedded Lab (Carter and Yan 2005). The solution for that system was to omit objects that were above a certain scale. While this was functional, it does not assist in situations where the majority of grains may be in physical contact. Indeed, utilising Carter's system required very gradual feeding of grains into the system to minimise physical contact between grains. This resulted in not all grains passing through the system being assessed. Limited segmentation of small numbers of contacting objects is worthwhile to improve this process. The intention was to implement this functionality and maintain the robustness of grain measurements recorded.

A total of 146 rice grains were imaged using the imaging box as described in Section II. The images have a resolution of 640*480, approximately 0.0034mm scale per pixel and contain sets of 2, 3 and 4 contacting grains as illustrated in Figure 6.43. The images are divided into three sets according to the number of contacting grains. The images were subject to a threshold converting them to binary and processed with the segmentation function.

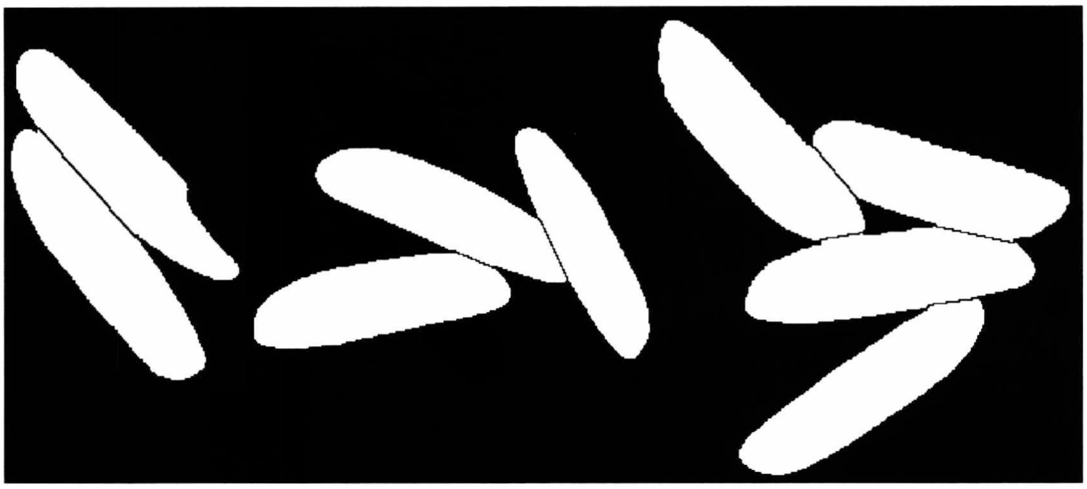
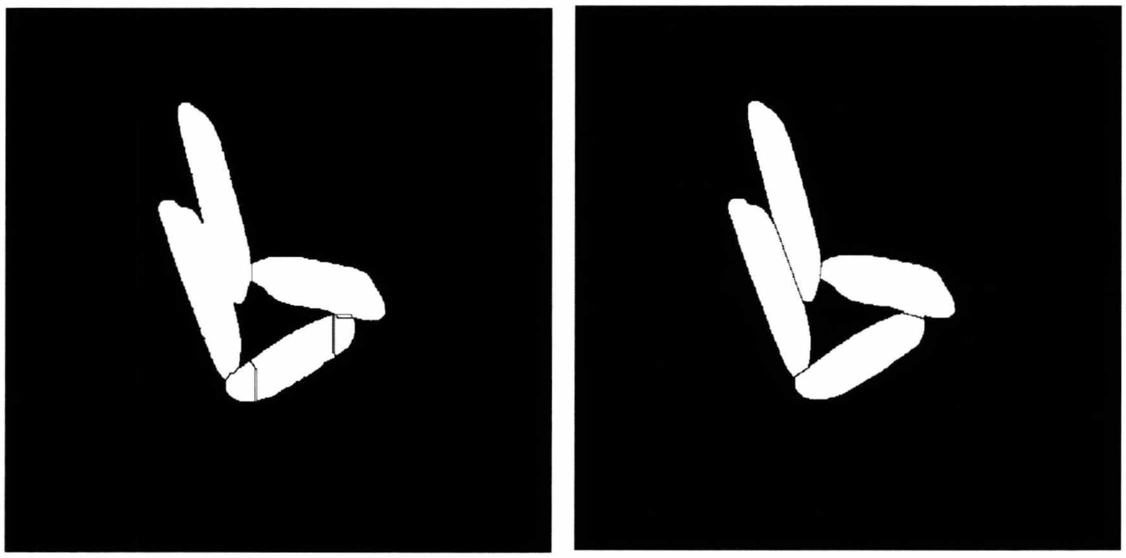


Figure 6.43. Contacting rice grains segmented with concave split line approach.

In order to contrast the effectiveness of segmentation in a specific scenario, a watershed method was employed for purpose of comparison. Figure 6.44 illustrates the segmentation results. The watershed method demonstrated in 6.44(a) is the method described in chapter 4. and the concave approach in 6.44(b) described in chapter 5.



(a) Watershed Approach. (b) Concave Split Line Approach.

Figure 6.44. Comparison of segmentation methods.

Precise split between the contacting grains is debatable, although in Figure 6.44(a) it is clear false splits have occurred as well as valid splits being missed. Instead, the test images for the split line approach shown in Figure 6.44(b) represent a significant improvement in aesthetic, although further analysis can be performed to truly verify how well the segmented grains can be measured.

6.8.1 Analysis of Segmented Grains

Following segmentation, shape analysis of the images in a manner employed in section 6.5 is applied as a validation of the measurement of the grains. This has permitted the comparison of segmented image results with previous results involving segregated grains. This is illustrated by the demonstration of recorded shape feature values for the resultant final segmented images of rice grains. Table 6.45 summarizes the individual results for each image set, the mean result for all sets, and the previous result for high resolution grain images from Figure 6.16. Sizes of sample sets, the number of grains were recorded for each instance. It is noted that every image in the test sample was segmented at the correct location on the shape perimeter.

Table 6.45. Compactness features for sets of Basmati rice grains.

Image set	R_c	σ	Sample Size
Two contacting grains	0.49	0.03	58
Three contacting grains	0.48	0.03	48
Four contacting grains	0.49	0.03	40
All segmented grains	0.49	0.03	146
Compactness, Figure 6.16.	0.50	0.04	168

The results suggest shape features returned concur with segregated grains almost exactly, with the same general range of values returned from both sets regardless of the contact of the grains.

Early testing of the function exposed a problem where less than five iterations of the algorithm are be used. In some cases where four grains are in contact, more than four split lines may emerge. While this is rare, five split lines can occur with four grains touching. An example is given in Figure 6.46, which shows an image obtained from an additional grain image set. This was corrected with an alteration in the algorithm decision function loop, increasing the iterations from four to five, or until no further grains meet the split criteria of the decision function. No significant oversegmentation was detected in the images following these changes.

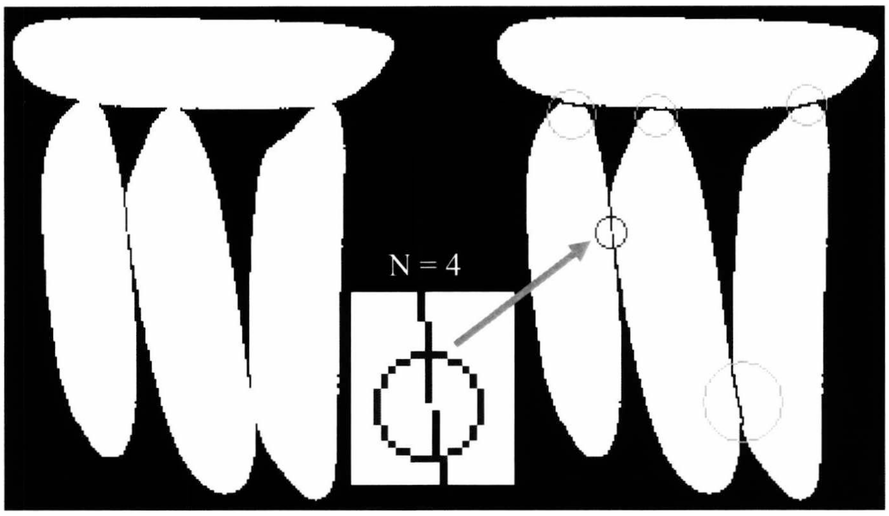


Figure 6.46. Grains with ten split points with only four split lines drawn.

6.9 Summary

The imaging of rice grains was achieved and analysis was successfully performed returning a range of descriptive properties of rice grains and indicating a number of useful functions for grain analysis. General shape descriptors could distinguish general rice varieties although specific grains required more descriptive analysis. The subtle eye curve shape feature can be derived from the shape boundary of rice grains. Some distinction is possible between pure samples of basmati and non-basmati varieties. Segmentation can be employed for specific varieties of rice in order to individually measure grains in contact. These different elements of research support one another within the context of an imaging system for the purpose of characterisation and classification of rice grains. Development of image processing systems can be employed following these principles for the benefit of the commercial sector.

Chapter 7

Conclusions and Recommendations for Future Work

7.1 Introduction

This thesis reports on the design, implementation and experimental evaluation of a flexible imaging system and series of image processing techniques for application upon static images of rice grains. Additionally a number of simple image processing features have been evaluated for their useful for this task, and for wider potential applications. The system is non-destructive and non-intrusive, with capture software integrated into an entirely portable system that requires only low voltage external power and a laptop computer capable of supporting the image processing application. This image processing application includes some automatic implementation of the many calibration tests conducted during research, such as spatial dependency. Some aspects are separately performed using environments such as Matlab, which can easily be installed on the target machine and carry out further image processing and statistical analysis routines based upon the source images from the capture system.

The initial stages of this work consisted of analysis of a range of features useful for two-dimensional shape analysis. While this was initially considered for a broader range of applications within image processing and shape analysis, the optimisation of many of these features for rice grain characterisation was pursued and became the primary objective of this thesis over time.

The next phase of work involved construction of the physical image setup and the obtaining of images of rice grains. This first began with the use of an existing image capture system to obtain the first set of general rice images. Further work led to the development and implementation of a superior system based in a highly portable box. While initially crude, this system was extended and improved to support a range of different camera devices more flexible the initial high resolution commercial camera.

Later cameras were capable of immediate capture via USB and could immediately be captured and processed with an attached laptop. Using this approach, more image sets could be obtained in a repeatable controllable manner. Further to this, other imaging systems were employed for comparative analysis, including a system capable of imaging large samples of flowing grains.

A simultaneous final phase took the processed images obtained from each set and performed relevant analysis according to each type of experiment. This returned a range of feature values in accord with the many features developed in the course of research. Analysis of the recorded data could be performed over a greater period of time and with greater opportunity for experimentation in order to test a range of features and feature analysis techniques.

This chapter presents conclusions of this research. These are that digital imaging technology can be used to derive useful metrics for the assessment of food grains, deriving a wide range of useful measured properties. This information can be of use for various reasons such as fraud detection, individual grain type characterisation, grading, and incorporating into a larger measurement system for sorting and other tasks. The focus on rice grains here has permitted the identification of several unique features and research into properties of relevance to the commercial sector, particularly with regard to fraud detection and food quality grading. The capture of images of these types of materials can be performed easily with low-cost technology, and a number of image processing techniques can be employed to enable analysis using widely available technology.

7.2 Conclusions

The conclusions that can be drawn from the results presented in the thesis are broken down into several sections. As well as measurement data significant end products of this research are the basic methodologies for appropriate image processing, the physical image acquistion system and the follow-up image processing and analysis routines. Part of these processing and analytical functions can be carried out immediately on the capture of images using a laptop based capture program, and recorded images and features can be saved for further examination. A large volume of images of rice grains were created in the course of this research which has been retained in the Instrumentation, Embedded Systems and Control Laboratory. As part of the research strategy, from such images the derivation of meaningful measurements and characteristics of the different rice grains could be attained.

As part of achieving the ultimate goal in this research, three questions were posed in chapter 1 section 3. The first of these related to how imaging could specifically show useful measurements. In the literature review several alternative methods to imaging were discussed, what made imaging distinct was the speed, ease and low cost of imaging approach over other methods.

For the second point, low cost hardware was tested and effectively employed for the task. The prevalence of digital cameras provided a wide selection of possible sensors which varied in quality according to cost. Even very inexpensive sensors and low cost setups can provide usable measurements.

The third point and forth points were addressed by using this setup to obtain and process images. A range of useful features could be applied in rice imaging. From the experimental work conducted, distinction of long, medium, and short grains was found to be possible in a variety of rice grains using simple shape, length and texture features. Aspect ratio and Shape factor, and Compactness all produced similar ranges of values. Radius and diameter derived approaches did differ in ranges of values, yet distinct. A combination of length and shape features can also be employed for the distinction of highly similar varieties where adulteration is concerned.

The curvature of grains can be detected by examining the curvature of the grain two dimensional top down image of its boundary. Minor concavities can be seen in the boundary itself, and are still detectable in transformations of the boundary, which can be subject to further analysis and classification tests to distinguish grains with and without this feature. Grains with the feature were detected with 99.1% accuracy using the 2nd to the 12th discrete fourier coefficients, and with a 6.1% false acceptance rate of grains without the feature. Furthermore the distinction of contacting grains contacting in a two-dimensional plane is possible based on the detection of significant concavity.

Despite the obvious power of features such as fourier transformation, the simpler features are still applicable in various practical domains. In the experimental research conducted, length has been shown to be immediately useful for general classification of long, medium and short grains. A separate set of tests showed length and aspect ratio are of use in distinguishing basmati and non-basmati in accordance with official guidelines. The compactness ratio feature was used in several tests and in part of the segmentation function. This also addressed the third point in section 1.3 regarding the improvement of image processing The employment of this type of segmentation approach was highly effective for assisting in the imaging of rice grains. Not only did this feature give differences in long, medium and short grains, this feature had advantages over the other shape features employed. Its derivation is simpler, requiring only calculation of boundary length and area as opposed to a choice of method to determine grain length. Other features such as aspect ratio and shape factor were of use, yet require careful consideration of the choice of method for determining grain length. Different approaches return different ranges of values for measurements and it is not clear which approach is distinctly superior in this case.

Further experimental work derived through the use of edge detection permitted a method texture based analysis believed to be novel at time of writing. This novel texture feature was also shown in experimental work to be capable to distinguish brown and white grains in situations where shape would not be of use.

7.2.1 Acquisition and Processing of Rice Grain Images

As mentioned in the previous section, the image sets successfully created have been a major success of this research. A number of experiments and tests were required and contributed to the success of this process in order to ensure the quality of these images was consistent. Through the employment of controlled conditions and careful testing a reasonable standard of imaging was attained, from which further analysis was possible. Several lessons were learned in the creation of this acquisition environment. This included how to carry out several methods to calibrate and test the environment in order to get the best possible image quality from the system. This helped minimised shortcomings in the system. The employment of spatial tests and detection of lens distortion permitted the elimination of potential problems. The ability to use a chroma key to maximise the effectiveness of colour cameras and improve the segmentation of background and objects was also beneficial to imaging conducted here. The ability to employ and test a variety of cameras kept a wide range of options open in the course of research for possible experiments to conduct. Ultimately this permitted the use of a variety of cameras for the imaging of rice grains, and a diverse variety of image sets were created for several different experiments.

Following the successful acquisition of images of sufficient quality, processing could be successfully employed to remove redundant irrelevant information, leaving only the key properties of the grain images fit for measurement. The development and implementation of segmentation functions permit the further evaluation of grains with imaging as well as offering the opportunity to explore this area of research. A range of segmentation approaches were employed, from background removal to the distinction of contacting grains. A successful segmentation function was created for separating grains in a complex scene, and it was shown that the impact of this function on the measurable characteristics of the grains was negligible.

7.2.2 Shape Features

A range of simple shape features were investigated to consider impact of aliasing across a range of size ranges relevant to this research and on a number of different shapes. Ultimately a number of shape features were assessed in order to find their expected range of values for different geometries, in the hope that features with a good range of values relevant to rice grains could be found. This study indicated simple shape features had individually limited descriptive power, yet as each feature responded differently, several could be used in conjunction to offer potentially more descriptive ability. Features such as aspect ratio, shape factor and compactness particularly express a distinct range of values for a variety of simple shapes. Also certain shape features such as compactness were found to reveal properties which in turn were highly useful in other applied image processing such as the segmentation of rice. Compactness is likely to be the most widely useful simple feature in this task. While the aspect ratio and shape factor methods considered could be used to distinguish the long, medium and shorter grain varities, compactness can also fill this role as well. There is also some complexity in choosing the radius/diameter approach with the aspect ratio and shape factor features. Comparatively compactness was shown to be a simple, effective and reliable indicator of distinction between grains, with simpler computational requirements and ease of use.

7.2.3 Measurements of rice grains

With distinct image sets, processing and measurement could be performed with the objective of gauging some characterising properties of rice grains, and attempting to use the properties of rice derived from images in a commercially relevant way. It is however difficult to draw precise parallels with all existing standards. Many such standards are often confidential. Some standards relate to different types of physical properties altogether such as the chemical composition of rice. Despite this some it was possible to obtain a number of useful properties. These include the distinction of basmati and non-basmati samples which could be performed automatically based on the measurement approach developed in this research

The results presented have demonstrated that the separation of short, medium and long, as well as brown and white rice varieties, is achievable using a combination of the descriptors including average length, aspect ratio, shape factor, and compactness. Texture description using an edge detection based feature is also possible for the differentiation between brown and white rice, even from a grayscale image. These techniques work well with commonly available low-cost imaging hardware. While this was based upon existing measurement research it was expanded with simultaneous employment of what was at time of writing believed to be a novel texture feature. This texture feature, incorporating edge detection was able to distinguish brown and white and otherwise distinctly textured materials in this research and elsewhere beyond the scope of this thesis (Hobson, Carter and Yan 2007).

The detection of grain 'eye' features was persued based on reference in IRRI literature, and the distinctiveness of the feature in grain shape. Using the curvature of grains permitted detection of this type of distinct shape feature, that research suggested was of impact to grain quality as it was a position where grain breakage could occur, and broken grains are often an indicator of a lower quality of rice. A number of methods were considered for the detection of this feature, and the several features investigated. These were the following features derived from the curvature of the shape boundary: Fourier coefficients. Gradient peaks derived with functions of the gradient of the curvature. And finally, the zero crossings of the gradient of the curvature. Through use of these methods detecting the eye feature from a defined training set was achieved with a varying degree of success. While all cases had some false positives and assigned rounded grains as members of the set of eye ones, these were relatively few. The results of the employment of a small set of Fourier harmonics displayed a particularly high level of accuracy and low number of false positives when said harmonics were clustered into training sets of grains with and without the eye curve feature.

This research permitted testing of segmentation functions in a practical way, of potential relevance to research and to commercial sectors. A novel segmentation decision function was created, with an effective decision making function, capable of repeatedly and successfully segmenting suitable rice grains in images. The segmentation was conducted with follow up evaluation of the shape of the resultant grains, indicating that the impact of this segmentation was negligible on the measurement of grain shape.

All grains could be subject to all standard forms of measurement described earlier in the results. A large number of different rice grain varieties were imaged, measured and general shape and textural properties were recorded, giving general characteristics regarding these qualities of rice grains. Brown and White grains can be distinguished as well as long, shorter and some special variety grains.

It is anticipated that the analysis strategy employed may be useful in the wider field of pharmaceutical imaging, medical, agricultural, as well as geology and particle imaging. Software and characterisation methods employed are immediately transferable, however variations in hardware are needed for the different materials.

7.3 Future work

A number of challenges relating to the digital imaging of rice grains were identified during the course of research which warrant further investigation. These include:

- Rice grain disease assessment such as measurement of 'shrunk grain panicles' (Zhang et al 2008). Such disease concerns may be extensively addressed in future agricultural and bioengineering research.
- Use of imaging in conjunction with other methods such as DNA analysis (Tan 2000) to reveal general characteristics. Again, this may have bearing on rice quality, grading, and disease concerns.
- In the course of research, Grain Chalkiness was briefly investigated. More work on the assessment of grain chalkiness is feasible as a significant factor in rice quality, and is sufficiently visible to be a candidate for imaging based measurement.

Textural analysis methods used in research can be expanded and applied in other areas of research. One example where this feature might be applicable is the comparison of whole crystals and agglomerates (Yu 2007). These methods are also transferable to a range of other geological and medical fields. The hardware setup developed can be used for further experimentation with rice, other granular material of similar scale, or adapted with different optical packages for different scale material imaging such as fine powders.

The setup constructed can be used for further experimentation, with the present optics or with a number of different camera systems. The box can, with the proper position of parts, support a range of C-mount or custom mount camera systems. The changeable sample tray can support a different background material. Furthermore, the system is small enough to be portable and with USB cameras can be made entirely so. The system can function with software that can be installed on a conventional laptop capable of running windows XP and the matlab compile runtime. This forms an entire system for image capture, complicated image processing and analysis. This approach is low-cost, flexible and extendable.

Further static imaging research is possible with the imaging box. The physical setup presently uses a simple illumination setup although it in future work a more dynamic lighting arrangement could be implemented. A controllable directional lighting system could be employed under the same controlled conditions to implement other imaging techniques. One example would be employing variation of illumination (Chantler 1995) to achieve an imaging method known as photometric stereo texture, which permits building an 'arbello' representation of the surface texture of target materials, offering a whole new avenue of texture analysis with which the edge based feature may be applicable. Further improvements to the box could also permit different types of illumination, such as varying colour or infrared.

The many research groups, institutes and process machinery manufacturers mentioned in this research are still very much interested in the imaging based analysis of food grains and major research publications are still being produced by them. Several manufacturers, such as the Buhler group discussed in chapter two, are also producing relevant instrumentation and awareness of these companies is relevant to any future work, particularly for studying rice.

References

1. http://www.asiarice.org/sections/whatsnew/BRbulletin.html
2. http://www.specs.com/products/Kamina/Kamina.htm
3. Soille, P. (2004) Morphological Image Analysis–Principles and Applications. Springer-Verlag New York, Inc.
4. http://waltonfeed.com/self/rice.html
5. http://www.knowledgebank.irri.org/
6. United States Department of Agriculture, Foreign Agricultural Service. World Agricultural Production figures, 2009. URL: http://www.fas.usda.gov/wap/circular/2009/09-08/toc.asp
7. Department for environment, food and rural affairs, UK. http://www.defra.gov.uk/
8. PRA6/PRA15610 Product Sales and Trade: Grain Mill Products. Office of National Statistics, UK. http://www.statistics.gov.uk/
9. Survey on Basmati Rice (2004). F. S. Agency, UK Government. http://www.food.gov.uk/
10. Atkinson, R. W. (1881). The Chemistry of Saké-brewing, Tokio Daigaku.
11. Bayer, B. E. (1976). Colour Imaging Array. http://www.google.co.uk/patents. U. Patent. United States, Eastman Kodak Company. 3,971,065.
12. Bowman, E. T., Soga, K. Drumond, W. (2000). Particle Shape Characterisation Using Fourier Analysis, University of Cambridge, Department of Engineering.
13. Braidotti, G. (2007). Chalky Rice. Partners Magazine, Australian Center for International Agricultural Research (www.aciar.gov.au).
14. Brodatz, P. (1966). Textures: a photographic album for artists and designers, Dover, New York 1966.
15. Brosnan, T. and Sun, D.-W. (2004). Improving quality inspection of food products by computer vision--a review. Journal of Food Engineering 61(1): 3-16.
16. Brosnan, T. and Sun, D.-W. (2002). Inspection and grading of agricultural
17. and food products by computer vision systems – a review. Computers and Electronics Agriculture 36: 193-213.
18. Chakravarty, S. N. (2007) Basmati Rice Fraud in U.S. Markets Exposed.
19. Chantler, M. J. (1995). Why illuminant direction is fundamental to texture analysis. Vision, Image and Signal Processing, IEE Proceedings - 142(4): 199-206.

20. Choudhary, R. Paliwal, J. Jayas, D. (2008). Classification of cereal grains using wavelet, morphological, colour, and textural features of non-touching kernel images. Biosystems Engineering 99(3): 330-337.
21. Cox, M. R. and Budhu, M. (2008). A practical approach to grain shape quantification. Engineering Geology 96(1-2): 1-16.
22. Forsyth, D. Ponce, J. (2003). Computer Vision: A modern Approach. Upper Saddle River, New Jersey, Pearson Education LTD.
23. Mumford, D., Shah, J. (1989). Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems. Communications on Pure and Applied Mathematics 42(5): 577-685
24. Davies, E. R. (1990). Machine vision : theory, algorithms, practicalities / E.R., London : Academic, 1990.
25. de Carvalho, M. Lotufo, R. Couprie, M. (2007). Morphological segmentation of yeast by image analysis. Image and Vision Computing 25(1): 34-39.
26. Zhang, D. Lu, G. (2004). Review of shape representation and description techniques. Pattern Recognition, Pergamon, Elsevier 37: 1-19.
27. Zhang, G. Jayas, D. Deng, J. Halhead, J. (2005). Separation touching grain kernel in machine vision application with Hough Transform and morphological transform Written for the presentation at the CSAE/SCGR Meeting, Winnipeg, Manitoba.
28. Gonzalez, R. C. Woods, R. E. Eddins, S. (2004). Digital image processing, Addison-Wesley Reading, Mass.
29. Graham, R. (2002). A Proposal for IRRI to Establish a Grain Quality and Nutrition Research Center, IRRI Discussion Paper Series.
30. Sortex Z+ Documentation. http://www.buhlergroup.com/Docs/35634EN.pdf. The Buhler Group. http://www.buhlergroup.com/
31. Hobson, D. M., Carter, R.M. Yan, Y. (2007). Characterisation and Identification of Rice Grains through Digital Image Analysis. Instrumentation and Measurement Technology Conference Proceedings, 2007 IEEE: 1-5.
32. Hobson, D. M., Carter, R.M. Yan, Y. (2007). Differentiation between Coal and Stone through Image Analysis of Texture Features. Imaging Systems and Techniques, 2007. IST'07. IEEE International Workshop on: 1-4.
33. The Segmentation Comparision Project. Department of Computer Science and Engineering, University of Florida. http://marathon.csee.usf.edu/range/seg-comp/SegComp.html

34. British Retail Consortium. http://www.brc.org.uk/

35. International Rice Research Institute. http://www.irri.org

36. Kett . http://www.kett.com/

37. Number of rice varieties worldwide.
http://www.riceassociation.org.uk/About/varieties.htm

38. Hideshi, S. Motonobu, N. Takamasa, M. Jun, K. (2005). X-ray Fluorescence Analysis of Rice and Rice Bran with a Dry Battery X-ray Generator. Bunseki Kagaku 54(4): 321-324.

39. Horton, R. E. Drainage Basin Characteristics, Transactions of the American Geophysical Union, 1932.

40. Lie, J. Lysaker, M. Tai, X.C. (2006). A Binary Level Set Model and some Applications to Mumford-Shah Image Segmentation. IEEE transactions on image processing 15(5): 1171-1181.

41. Juliano, B. (1993). Rice in Human Nutrition, Food and Agriculture Organization of the United Nations (FAO). Rome, IT.

42. Keating, S. (2006, October 10, 2006). Food detective: basmati rice. The Times Online Retrieved October, 2006, from www.timesonline.co.uk.

43. Kindratenko, V. (1997). Development and Application of Image Analysis Techniques for Identification and Classification of Microscopic Particles Antwerp, University of Antwerp. Ph.D thesis.

44. Kindratenko, V. (2003). On Using Functions to Describe the Shape. Journal of Mathematical Imaging and Vision 18(3): 225-245.

45. Kumar, S. Ong, S. Ranganath, S. Ong, T. Chew, F. (2006). A rule-based approach for robust clump splitting. Pattern Recognition 39(6): 1088-1098.

46. Litwiller, D. (2001). CCD vs. CMOS: facts and fiction. Photonics Spectra. 35: 154-158.

47. Loncaric, S. (1997). A Survey of Shape Analysis Techniques. Pattern Recognition 31(8): pp. 983-1001.

48. Loncaric, S. (1998). A survey of shape analysis techniques-from automata to hardware. Pattern Recognition 31(8): 983-1001.

49. Vincent, L. Soille, P. (1991). Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations. IEEE Transactions on pattern analysis and machine intelligence 13(6): pp. 583-598.

50. Costa, L Caesar, R. (2000). Shape analysis and classification theory and practice. Pennsylvania Institute of technology, CRC Press.
51. Magan, N. and Evans, P. (2000). Volatiles as an indicator of fungal activity and differentiation between species, and the potential use of electronic nose technology for early detection of grain spoilage. Journal of Stored Products Research 36(4): 319-340.
52. Malamas E, Petrakis, E. Zervakis, M. Petit, L. Legat, J (2003). A survey on industrial vision systems, applications and tools. Image and Vision Computing 21(2): 171-188.
53. Hentschel, M. Page, N. (2003). Selection of Descriptors for Particle Shape Characterization. Particle & Particle Systems Characterization 20(1): 25-38.
54. McBader, S. and P. Lee (2002). A programmable image signal processing architecture for embedded vision systems. Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on.
55. Mehmet, S. & S. Bulent (2004). Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic Imaging 13(1): 146-168.
56. Mukherjee, Ray, (2004). Level set analysis for leukocyte detection and tracking. Image Processing, IEEE Transactions on 13(4): 562-572.
57. Visen, N. Paliwal, J. Jayas, D. and White, N (2004). Comparision of two neural network architectures for classification of singulated cereal grains. Canadian Biosystems Engineering 46.
58. Visen, N. Paliwal, J. and Jayas, D. (2001). Identification and segmentation of occluding groups of grain kernels in a grain sample image. Journal of Agricultural Engineering 79(2): 159-166.
59. Otsu, N. (1979). A threshold selection method from gray level histograms. IEEE Trans. Systems, Man and Cybernetics 9: 62-66.
60. Theanjumpol, P. Ripon, S, Karaboon, S. Uwapanit, K. Thanapornpoonpong, S. Vearasilp, S. (2005). Aromatic Thai Rice Identification by Near-Infrared Reflectance Spectroscopy. Conference on International Agricultural Research for Development, Hohenheim, Stuttgart http://www.tropentag.de/.
61. Felzenszwalb & Huttenlocher (2004). Distance transforms of sampled functions, Cornell Computing and Information Science TR2004-1963 http://people.cs.uchicago.edu/~pff/dt/.

62. Paliwal, J, Visen, N, et al. (2003). Cereal Grain and Dockage Identification using Machine Vision. Biosystems Engineering 85(1): 51-57.
63. Whelan, P. Molloy, D. (2001). Machine Vision algorithms in java. London, Springer-Verlag..
64. Podczeck, F. (1997). A shape factor to assess the shape of particles using image analysis. Powder Technology 93 47-53.
65. Podczeck, F., Rahman, S. Newton, J. (1999). Evaluation of a standardised procedure to assess the shape of pellets using image analysis. International Journal of Pharmaceutics 192(2): pp. 123-138(16).
66. Pons, M. Plagnieux, N. (2005). Comparison of methods for the characterisation by image analysis of crystalline agglomerates: The case of gibbsite. Powder Technology 157(1-3): 57-66.
67. Pratt, W. & Adams J. (2007). Digital Image Processing. Journal of Electronic Imaging 16: 029901.
68. R M Carter, Y. Yan, K. Tomlins. (2005). Digital imaging based classification and authentication of granular food products. Institute of physics publishing, measurement science and technology 17: 235-240.
69. R M Carter, Y. Yan. (2005). Measurement of particle shape using digital imaging techniques. Institute of Physics Publishing, Sensors & Their Applications XIII 15: 177-182.
70. R M Carter, Y. Yan. (2005). On-Line measurement of size distribution and volumetric concentration of pneumatically conveyed solids using digital imaging techniques. Electronic Engineering. Canterbury, University of Kent at Canterbury (UKC). BEng: 289.
71. R M Carter, Y. Yan., S. D. Cameron (2005). On-line measurement of particle size distribution and mass flow rate of particles in a pneumatic suspension using combined imaging and electrostatic sensors. Flow measurement and Instrumentation, Elsevier B.V 16: 309-314.
72. R. Pydipati, T. F. B., W. S. Lee (2006). Identification of citrus disease using color texture features and discriminant analysis. Computers and Electronics in Agriculture(52): pp. 49-59.
73. Ravilious, K. (2006) Buyer beware: the rise of food fraud New Scientist Volume, DOI:

74. Ruttler, D. (2001, Updated 2008). Webcam comparison: Logitech QuickCam Pro 3000, Logitech QuickCam Express, Lifeview Robocam http://www.dansdata.com/webcams.htm.
75. N Sladoje, I Nyström, P K Saha (2004), Measurement of digitized objects with fuzzy borders in 2D and 3D, Image and Vision Computing, 23, pp. 123 – 132.
76. N Sladoje, I Nyström, P K Saha (2003), Perimeter and Area Estimations of Digitized Objects with Fuzzy Borders, Discrete Geometry for Computer Imagery: 11th International Conference, DGCI, Naples, Italy, 2886, pp. 368-377.
77. Delwiche, S. Webb, B. (1996). Quality characteristics in rice by near-infrared reflectance analysis of whole-grain milled samples. Cereal chemistry 73(2): 257-263.
78. Sakai, N. Yonekawa, S. Matsuzaki, A. (1996). Two-dimensional image analysis of the shape of rice and its application to separating varieties. Journal of Food Engineering 27(4): 397-407.
79. Schäefer, M. (2002). Digital Optics: Some Remarks on the Accuracy of Particle Image Analysis. Particle & Particle Systems Characterization 19: 158-168.
80. Sun, D.-W. (2000). Inspecting pizza topping percentage and distribution by a computer vision method. Journal of Food Engineering 44(4): 245-249.
81. Tomanovich, S. (2006). Vision System Measures Scallops, http://www.vision-systems.com/.
82. Tschumperlé, D. (2004). The CImg Library. D. Tschumperlé, GREYC UMR CNRS 6072, Image group. http://cimg.sourceforge.net/.
83. Tuceryan, M. and Jain, A. (1998). The Handbook of Pattern Recognition and Computer Vision, World Scientific Publishing Co.: 207-248.
84. Mezaris, V. Strintzis, M (2003). Still Image Objective Segmentation Evaluation using Ground Truth. Proceedings of the Fifth COST 276 Workshop on Information and Knowledge Management for Integrated Media.
85. Metzler, V, Lehmann, T. Mottaghy, K. Spitzer, K. (1999). A novel method for quantifiying shape deformation applied to biocompatibility testing. Journal of the International Society of Artificial Internal Organs (ASAIO) 45(4): pp. 264-271.
86. Liu, W. Siebenmorgen, T. Chen, H. (1998). Digital Image Analysis Method for Rapid Measurement of Rice Degree of Milling. Cereal Chemistry 75(3): 380-385.
87. Wang, W. (2006). Seperation and Identification of Touching Kernels and Dockage Components in Digital Images. Canadian Biosystems Engineering 48: 7.1-7.8.

88. Wang, W. (1998). Binary image segmentation of aggregrates based on polygonal approximation and classification of concavities. Pattern Recognition 31(10): 1503-1524.

89. Woolfe, M. & Primrose, S (2004). Food forensics: using DNA technology to combat misdescription and fraud. Trends in Biotechnology 22(5): 222-226.

90. www.codeproject.com. Codeproject Website.

91. www.mathworks.com. The Mathworks website and File Exchange.

92. Xu, K. Luxmoore, A. Deravi, F. (1997). Comparison of Shape Features for the Classification of Wear Particles. Engineering Applications of Artificial Intelligence 10(5): 485-493.

93. Xu, K. Luxmoore, A. Deravi, F. (1998). Integration of neural networks and expert systems for microscopic wear particle analysis. Knowledge-Based Systems 11(3-4): 213-227.

94. Tan, Y. F. Xing, Y. Z. Li, J. X. Yu, S. B. Xu, C. G. and Zhang, Q (2000). Genetic bases of appearance quality of rice grains in Shanyou 63, an elite rice hybrid TAG Theoretical and Applied Genetics 101(5-6): 823-829.

95. Yadav, B. & Jindal V. (2001). Monitoring milling quality of rice by image analysis. Computers and Electronics in Agriculture 33(1): 19-33.

96. Yu, Z. Chow, P. Tan, R (2007). Quantification of particle morphology by boundary Fourier transform and generic Fourier transform. Chemical Engineering Science 62(14): 3777-3786.

97. Zhang, D. and G. Lu (2004). Review of shape representation and description techniques. Pattern Recognition 37(1): 1-19.

98. Zhang, G., D. S. Jayas, et al. (2005). Separation of Touching Grain Kernels in an Image by Ellipse Fitting Algorithm. Biosystems Engineering 92(2): 135-142.

99. Levner, I. & Zhang, H. (2007). Classification-Driven Watershed Segmentation. IEEE Transactions on Image Processing 16(5): 1437-1445.

100. Zhang, J. and T. Tan (2002). Brief review of invariant texture analysis methods. Pattern Recognition 35: 735-747.

101. Zhang, Y. J. (1995). Influence of segmentation over feature measurement. Pattern Recognition Letters 16(2): 201-206.

102. Zhang, Y. J. (1996). A survey on evaluation methods for image segmentation. Pattern Recognition 29(8): 1335-1346.

103.Ziedan M, J. X., Williams R A (2006). Errors implicit in digital particle characterisation. Chemical Engineering Science, Institute of particle science and engineering, School of process, Environmental and Materials Engineering, University of Leeds.

104. Zhang, Z. Lang, Y. Pan, M. Yang, J. Zhu, Q. (2008). Symptom of the Shrunk-Grain Panicle and the Change Characteristics in Its Grain. Agricultural Sciences in China 7(1): 10-18.

Nomenclature

A	-area in pixels
A_o	-Fourier descriptor
A_n	-Fourier descriptor
A_T	- area threshold
a_x	- x coordinate of first manhattan distance point
a_y	- y coordinate of first manhattan distance point
B_n	- Fourier descriptor
b_x	- x coordinate of second manhattan distance point
b_y	- y coordinate of second manhattan distance point
C_e	-Straight line distance between two points
C_v	-FKZ plot (concaveness)
C_m	-Chess board distance between two points
Cq	-Functional distance between two points
CT.	-compactness threshold
d_{mean}	-Mean diameter length
d_{max}	-Maximum diameter length
d_{min}	-Minimum diameter length
E	- number of edge pixels for a specific connected component
C_n	-Fourier harmonics of boundary pixels
$d_{rms.d}$	-Root mean square deviation
Elongation	-Elongation Ratio
F_f	-Form Factor
F_i	-Individual point I of the function of gradient
f_k	-Fourier function of boundary pixels
G_i	-Individual point i on Gradient plot G from the gradient of a shape signature
G_x	-x mask for sobel edge detection
G_y	-y mask for sobel edge detection
$\|G\|$	-pixel value returned as result of sobel edge detection
G_α	-edge direction for canny edge detection
h'	-extension of h from start pixel to new location

h	-distance from image center to the pixel of interest during pincushion correction
H	-Number of holes in a 2D binary object
H_T	.Holes threshold
$I, I1, I2$	-Representation of different images in flow charts.
m_p	-millimetres per pixel
m_x	-x coordinate of pixel, distance in millimetres to x center
m_y	-y coordinate of pixel, distance in millimetres to y center
N *or* n	-length of boundary or array.
N_j	-Angle of curvature between a midpoint and a significant point of curvature
Numberofzeros	- number of zeros returned by scaling factor detection
O_{img}	-output image for distortion correction.
P	*-perimeter length in pixel lengths*
P_i	-current point on a shape boundary the list of boundary pixels
P_{List}	*- list of boundary pixels represented as coordinates x and y*
q_h	-euclidean distance of pixel to center
R_a	-Aspect ratio, radius or diameter based (specified where used)
R_e	-Ratio of edge pixels to area
r_{mean}	-Mean radius length
r_{max}	-Maximum radius length
r_{min}	-Minimum radius length
R_c	–Ratio of compactness
S_f	-Shape factor, diameter or radius based (specified).
Scalefactor	-zero crossing scaling factor
T	-threshold
T1	-threshold 1 for canny edge detection
T2	-threshold 2 for canny edge detection
$x_{a,} y_{a}, x_{b}, y_{b},$	-Coordinates of candidate split path
$x_{j,} y_j$	-Candidate Split point coordinate from a shape boundary
$x_{1,} y_{j2}, x_{2,} y_2$	-Start and end coordinates for a split line
x_d, y_d	-distance for line drawing for split line.
x_p and y_p	-individual pixel coordinates in shape boundary
X	- height of original image for distortion correction
X_n	-mapped new pixel location for distortion correction

$X_{n'}$	-mapped old pixel location for distortion correction
$X_{n'mm}$	-vertical new mapped pixel location for distortion correction in mm
Y	-height of original image for distortion correction
Y_n	-mapped new pixel location for distortion correction
$Y_{n'}$	-mapped old pixel location for distortion correction
$Y_{n'mm}$	-horizontal new mapped pixel location for distortion correction in mm
Z	distance from the focal point to the pixel of interest during pincushion correction
z	used to determine zp
zp	distance from point of interest to 'imaginary' concave surface in the plain of Z_p
θ	-angle used during pincushion correction. -AND tangent angle curvature
θ_j	-Boundary curvature coordinate
θ_{ab}, θ_{ba}	-Split path candidates, represented by coordinates and angle.

Appendix A

Publications and Dissemination

Following papers were published during the course of the research:

1. Hobson, D.M. and Carter, R.M. and Yan, Y. (2006) Quantifying the Shape of Coal and Biomass Particles through Digital Imaging. In: 6th European Conference on Coal Research and its Applications, Canterbury, UK, September 5-7, 2006.

   105.

2. Hobson, D. M., Carter, R. M., Yan Y. (2007). Characterisation and Identification of Rice Grains through Digital Image Analysis. Proceedings of the IEEE International Instrumentation and Measurement Technology Conference, Warsaw, Poland, date? May 2007.

   106.

3. Hobson, D. M., Carter, R. M., Yan Y. (2007). Differentiation between Coal and Stone through Image Analysis of Texture Features. IEEE International Workshop on Imaging Systems and Techniques, IST'07, Krakow, Poland, [Dates] May 2007.

   107.

4. Hobson, D. M., Carter, R. M., Yan, Y. (2009). Rule Based Concave Curvature Segmentation for Touching Rice Grains in Binary Digital Images'. Proceedings of IEEE International Instrumentation and Measurement Technology Conference, I^{2}MTC, Singapore, May 5-7, 2009.

Appendix B

Software and Algorithms for Image Capture and Processing

This section includes code examples of functions, and describes the software content of the folder/disc (available from the Instrumentation, Control and Embedded Systems Laboratory at the University of Kent) containing code and applications produced during this project. This includes Matlab, C++, and C# code.

Functions for Image processing

The image processing toolbox of Matlab contains many functions which can accelerate development of image processing systems. This code can be compiled and executed within a DLL file attached to another application, or run as independent applications. There is one condition for these types of deployment. The application must be run on a machine with either Matlab or the freely available Matlab Compile Runtime Installed on it. This does however permit the entire range of Matlab toolboxes to be deployed, including plot tools, image processing functions, and data acquisition.

The following functions were created in the course of this research.

Matlab Function for Spatial Dependency

The spatial dependency test is embedded in the C# application in the DLL. This function library requires a converted Matlab array and parameters as input. The functioning of this matlab routine is as follows. First, the user provides a source image of the calibration target. This should only be the target area, and can be produced by using the crop function in the image capture application. Next, the user must specify the number of calibration targets in rows and columns (width and height). There will always be a matrix of targets required for this function. Finally, pressing the 'Illumination Test' button will begin the spatial dependency evaluation. This will first apply the adaptive threshold function so please wait a moment. Next, several screens will display, the ultimate one being a binary image of the calibration targets. These should appear to be solid and complete representations of each calibration target. If this is not so, the image must be taken again under better conditions. Assuming this is the case, the user should click on each target from each column in the first row of the image. Then press the right mouse button. The user should repeat this action for every row in the image. At this stage, the final output will be processed and the spatial plots as demonstrated in Chapter 3 will be displayed for the calibration image. The user can then proceed to save these plots as images for further review or proceed with other tasks.

```
% draws the spatial dependency test graphs for a specified input:
% im1 = calibration image
% numcirclesx/y = number of calibration targets in the image
% x and y denote rows and columns:
% Eg:  a 4x3 grid of 12 circles is 4,3.
% select one column of circles at a time. eg:
%
% x x x -<1st time (left click, left click, left click, right click.)
% o o o
% o o o

function [meanintensity meanarea] =
spatial_dependency_test(im1,numcirclesx,numcirclesy)

    if size(im1,3)>1
    im2 = rgb2gray(im1);
    else
        if max(max(im1))<=1
        im2=uint8(im1*255);
        else
        im2=uint8(im1);
        end
    end

im3 = IST_Threshold(im2);
meanintensity = zeros(numcirclesy,numcirclesx);
meanarea = zeros(numcirclesy,numcirclesx);
currX=1;currY=1;

% select one each loop.
% select in the correct order left>right row by row.
for currY=1:numcirclesy

    currentshapeimage = bwselect(im3);
    im3 = logical(uint8(im3-logical(currentshapeimage)));

  for currX = 1:(numcirclesx)

    currentshapeimage = uint8(bwlabel(currentshapeimage)); %
    CurrentThold = logical(currentshapeimage-1);
    CurrentImg = ~CurrentThold;
    CurrentImg = bwlabel(~currentshapeimage-CurrentImg);
    currentshapeimage = logical(
                         uint8(bwlabel(currentshapeimage)) -
                         uint8(bwlabel((~currentshapeimage-CurrentImg))
                         )
                        );
    areas = sum(sum(CurrentImg))
    mean_intensity=0;

    for x=1:size(im2,1)
        for y=1:size(im2,2)
            if CurrentImg(x,y) == 1
                mean_intensity = mean_intensity + double(im2(x,y));
            end
        end
    end

    mean_intensity = mean_intensity/areas
    meanintensity(currY,currX)=mean_intensity;
    meanarea(currY,currX)=areas;
```

```
        mean_int = mean(mean(meanintensity));
        mean_area = mean(mean(meanarea));

    end

end

createmeanintensityfigure(meanintensity);
createmeanareafigure(meanarea);

function createmeanintensityfigure(zdata1)

figure1 = figure('PaperSize',[20.98 29.68],'Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,...%,'ZTick',[750 800 850 900 950 1000],...
    'YTick',[0:(size(zdata1,1))],...
    'XTick',[0:(size(zdata1,2))],...
    'FontSize',12);

view([-43.5 36]); grid('on'); hold('all');
% Create surf
surf(zdata1,'Parent',axes1);
colormap(gray);
% Create xlabel, ylabel, zlabel, colorbar
xlabel('X  (targets)');  ylabel('Y  (targets)');  zlabel('Intensity  per
target');
%title('Intensity per target','FontSize',12);
colorbar('peer',axes1);

function createmeanareafigure(zdata1)

figure1 = figure('PaperSize',[20.98 29.68],'Color',[1 1 1]);
% Create axes
axes1 = axes('Parent',figure1,...%,'ZTick',[750 800 850 900 950 1000],...
    'YTick',[0:(size(zdata1,1))],...
    'XTick',[0:(size(zdata1,2))],...
    'FontSize',12);

view([-43.5 36]); grid('on'); hold('all');
% Create surf
surf(zdata1,'Parent',axes1);
colormap(gray);
% Create xlabel, ylabel, zlabel, colorbar
xlabel('X (targets)'); ylabel('Y (targets)'); zlabel('Area per target');
%title('Area per target','FontSize',12);
```

colorbar('peer',axes1);

Matlab Function for Lens Distortion Correction

The spatial dependency test is embedded in the C# application in the DLL. This function library requires a converted Matlab array and parameters as input. The functioning of this matlab routine is as follows. First, the user provides a source image of the calibration target. This should only be the target area, and can be produced by using the crop function in the image capture application. Next, the user must specify the number of calibration targets in rows and columns (width and height). There will always be a matrix of targets required for this function. Finally, pressing the 'Illumination Test' button will begin the spatial dependency evaluation. This will first apply the adaptive threshold function so please wait

```
%   this   function   alters   image   layout   incorporating   a   limited
acknowledgement
% of lens distortion.
% currently just for pincussion, future work with trapezoidal possible.

% Example runs of this function:
% DistortionCorrection(imread('x:\imgs\distort.jpg'),93,48,-0.25,0.005);
% DistortionCorrection(imread('x:\imgs\distort.jpg'),100,84,-0.05,0.005);

function OutImg =  DistortionCorrection(
                   InputImg,distance,imwidth, coefficient,resizevariable
                   )

% Variable Declarations:
InputImg = rgb2gray(InputImg);
ytot = size(InputImg,1);xtot = size(InputImg,2);
OutImg = uint8(zeros(ytot,xtot));

d = distance; iw = imwidth;
theta = atan((d/iw)); halftheta = theta/2;
X_tot = imwidth; mmPerPixel = X_tot/xtot;
Y_tot = mmPerPixel*ytot; h = sqrt((xtot*xtot)+(ytot*ytot));
hmm = (X_tot/xtot)*h; dD = sqrt((d*d)+((hmm/2)*(hmm/2)));
x = 1;y = 1;

while(y<=uint32(ytot/2))
        while(x<=uint32(xtot/2))

            xq = double(uint32(xtot/2)-x);
            yq = double(uint32(ytot/2)-y);
            xqmm = double(xq*mmPerPixel);
            yqmm = double(yq*mmPerPixel);
            %Trigonomic functions
            qh = sqrt(((xqmm*xqmm)+(yqmm*yqmm)));
            z=sqrt(((qh*qh)+(d*d)));
            zp=(dD-z);
            Alpha=acos((d/z));
            ext=(tan(Alpha))*zp;
            % Beta
            if(xqmm~=0)
                Beta=atan(yqmm/xqmm);
            else
                Beta=atan(0);
            end
            % Beta
            if x==xtot/2
            xnqmm=0;
            else
            xnqmm=((ext+qh)*cos(Beta));
            end
            ynqmm=sqrt(((ext+qh)*(ext+qh))-(xnqmm*xnqmm));

            xnq=(xnqmm/mmPerPixel); ynq=(ynqmm/mmPerPixel);

            % assign correct coordinate
            xn=uint32( (double(x-((xtot/2)-xnq))*coefficient)+x );
            yn=uint32( (double(y-((ytot/2)-ynq))*coefficient)+y );
            %correction of quadrants
            xi = xn; yi = yn;

            % write to output image
               if( (xi>0) && (xi<=uint32(xtot/2)) ...
                       && (yi>0) && (yi<=uint32(ytot/2)) )
                    OutImg(y,x) = InputImg(yi,xi,1);
                    xi = (xtot-uint32(xn));
                    OutImg(y,(xtot-x)) = InputImg(yi,xi,1);
                    xi = xn; yi = (ytot-uint32(yn));
                    OutImg((ytot-y),x) = InputImg(yi,xi,1);
                    xi = (xtot-uint32(xn)); yi = (ytot-uint32(yn));
                    OutImg((ytot-y),(xtot-x)) = InputImg(yi,xi,1);
               end
            x = x+1;
        end % end x
y = y + 1;
x=1;
end % end y
toc

if resizevariable>0

OutImg                                                                    =
OutImg(uint32(size(OutImg,1)*resizevariable):uint32(size(OutImg,1)-
(size(OutImg,1)*resizevariable)),uint32(size(OutImg,2)*resizevariable):ui
nt32(size(OutImg,2)-(size(OutImg,2)*resizevariable)));

end
```

Threshold Function for Rice Imaging

This adaptive threshold is based on Carter's implementation and is supplemented with morphological operations permitting the option of removing small irregularities in the object perimeter, and an option for inversion of orientation to apply the threshold in reverse.

```
    % Input parameters:
    % inputI = input (and output) image
    % orientation - true/false, denotes inversed threshold.
    % spur - true/false, apply morphological spur operation.
    % majority - true/false, apply morphological majority.
    % minimumobjectsize - int, minimum object size permissible.
    % previewimages - debug, show output on screen (depreciated).

function inputI = RobsAdaptiveThreshold(…
                        inputI,orientation,spur,…
                        majority,minimumobjectsize,previewimages)

if (size(size(inputI))==3) % Matlab greyscale formatting, rgb to grey.
    inputI = rgb2gray(inputI);
end

    Outarray = zeros(256,1); thresholdlevel = 1;
    ThresholdScale = 256; firstpeaklocation=1;

    for Tlevel = 1:ThresholdScale
        if orientation==true % store number of connected components.
           Outarray(Tlevel) = uint32(
                  max(max(
                        bwlabel(
                              im2bw(inputI,Tlevel/ThresholdScale)
                        )
                  ) )
            );
        else
           Outarray(Tlevel) = uint32(
                  max(max(
                        bwlabel(
                              ~im2bw(inputI,Tlevel/ThresholdScale)
                        )
                  ) )
            );
        end
           if(Outarray(Tlevel)>Outarray(firstpeaklocation))
               firstpeaklocation=Tlevel;
           end
    end

% smooth the histogram
SmoothedArray = Outarray; tpoint = 1; filterwidth=10;


    for i = 1:ThresholdScale
        for ic = 1:(filterwidth)


            members=0; before = uint32(i-ic); after = uint32(i+ic);
            if(before>0)
                SmoothedArray(i) = Outarray(i) + Outarray(before);
                members = members + 1;
            end
            if(after<=ThresholdScale)
                SmoothedArray(i) = Outarray(i) + Outarray(after);
                members = members + 1;
            end

            SmoothedArray(i) = SmoothedArray(i)/members;

        end
    end

endpoint=ThresholdScale; isdone=false; currentending=Outarray(endpoint);

    while isdone==false && endpoint>0
        if (Outarray(endpoint)>currentending)
            isdone=true;
        end
        endpoint = endpoint-1;
    end

lastpeaklocation=endpoint;   rangeOffset=4;   thresholdrange=rangeOffset;
Tlevel=lastpeaklocation; finishcondition=0; minimumheight=9;
% find the last peak work then backwards to first peak in SmoothedArray

    while(Tlevel>0)
        if(finishcondition==1)
        if(SmoothedArray(Tlevel) > SmoothedArray(firstpeaklocation))
                firstpeaklocation=Tlevel;
        end
        end
        if(finishcondition==0)
            if (SmoothedArray(Tlevel) >
               SmoothedArray(lastpeaklocation))
                lastpeaklocation = Tlevel; thresholdrange=rangeOffset;
            else
                if(SmoothedArray(Tlevel) > minimumheight)
                thresholdrange = thresholdrange-1;
                end
            end
        if (thresholdrange==0)
            finishcondition=1;
        end
        end

        Tlevel = Tlevel - 1;
    end
gradient = zeros(ThresholdScale,1);

    % this routine produces the smoothed array of gradient data.
     for i = firstpeaklocation:lastpeaklocation
         before = i - 3;
         after =  i + 3;
             if(before<1)
                 before=1;
             end
             if(after>ThresholdScale)
                 after=ThresholdScale;
                end

                gradient(i) =
                     SmoothedArray(after) - SmoothedArray(before);

                if(gradient(i)<0)
                    gradient(i) = 0-gradient(i);
                end
        end

FL = firstpeaklocation; LL = lastpeaklocation;
% find the point of lowest gradient between the two peaks
rangeOffset=uint8(((lastpeaklocation-firstpeaklocation)/4)/4);
thresholdrange=rangeOffset; Tlevel=lastpeaklocation-1; finishcondition=0;

    while(Tlevel>firstpeaklocation && finishcondition==0 )
        if (gradient(Tlevel) > gradient(lastpeaklocation))
            lastpeaklocation = Tlevel;
            thresholdrange=rangeOffset;
        else
            thresholdrange = thresholdrange-1;
        end
        if (thresholdrange==0)
            finishcondition=1;
         end
        Tlevel = Tlevel - 1;
    end

TheThresholdPoint=(lastpeaklocation);

    % finally, step backwards between the two peaks and find the point
    % of lowest gradient - use this point for the threshold.

    i=(lastpeaklocation-2);
    while(i>firstpeaklocation)
        if(gradient(i) <= gradient(TheThresholdPoint))
            TheThresholdPoint=i;
        end
    i = i-1;
    end



tp = double(TheThresholdPoint)/255; % the threshold point (in matlab)
      % Conditionals for inverse threshold.

      if orientation==true
            inputI = im2bw(inputI,tp);
      else
            inputI = ~im2bw(inputI,tp);
      end
      % Conditionals for applying morphological operations.
      if spur==true
            inputI = bwmorph(inputI,'spur',Inf);
      end
      if majority==true
            inputI = bwmorph(inputI,'majority',Inf);
      end

inputI = bwlabel(inputI); imHstLength = max(max(inputI));
imHst = zeros(imHstLength,1); % create list of objects by size.
```

```
for x=1:size(inputI,1)
       for y=1:size(inputI,2)
           Currentindex = inputI(x,y);
           if Currentindex > 0
                 imHst(Currentindex) = imHst(Currentindex)+1;
           end
       end
end


for z = 1:imHstLength % Remove from list objects < minimumobjectsize
    if(imHst(z) < minimumobjectsize )
       imHst(z) = 0;
    end
end

for x=1:size(inputI,1) % Remove from image objects in list = 0
    for y=1:size(inputI,2)
        if(inputI(x,y)>0 && inputI(x,y))
            if  ( imHst(inputI(x,y)) > 0 )
                inputI(x,y)=1;
            else
                inputI(x,y)=0;
            end
     end
    end
end
```

Matlab Feature Recording

This function records the descriptive features using the binary and source image

```
% return all shape and texture properties for an image
% featurelist = all feature parameters for a list of shapes
% pic - the binary image of the shapes
% pic2 - the original grey image
% selectorgetall - if 0, bwselect the desired shapes, else selects all.
% rather than all. eg:  1 = every 1 degree angle, 5... etc.
% numberofharmonics - the number of fourier descriptors of curvature to
use for analysis

function                                ReturnedShapes                               =
Grain_Analysis_Allfeatures(pic2,pic1,selectorgetall)

    if selectorgetall == 0
    pic = bwselect(logical(pic1));
    else
        pic=logical(pic1);
    end

  % debris filter, may need to be changed dependent on imaging scale...
    minimumsize=1000;

    % Get edge properties
    currentimg = edge(pic2,'canny');
    subtrackedcomponent = bwmorph(pic,'erode',1);
    currentimg = im2bw((currentimg + subtrackedcomponent),1);
    meangrey=meani(currentimg,pic,minimumsize);

% Declare parameters in structure.
ReturnedShapes = []; shapeproperties = struct();

% label individual particles and proceed
if(size(pic,3)>1)
if(max(max(pic))==1)
    size(pic)
Labelpic = bwlabel(pic(:,:,1));
else
Labelpic = bwlabel(rgb2gray(pic));
end
else
Labelpic = bwlabel(pic);
end

% start of master loop through list of objects
for i = 1:max(max(Labelpic))
```

```
area=[]; B=[]; L=[]; PerimeterLength=0;
% simple shape features
Compactness=0; Daspectratio=0; Dshapefactor=0; Ddrainage=0;
% radii diameters, including lists
Raspectratio=0; Rshapefactor=0; Rdrainage=0; minradius=0; maxradius=0;
meanradius=0; radii=[];
% diameter features
mindiameter=0; maxdiameter=0; meandiameter=0; diameters=0;

% curvature plot of shape
curvature=0;
% shape perimeter pixel list
boundary=[]; bending = [];

% get objects and fill holes in the shape area
CurrentThold = logical(Labelpic-1);
CurrentImg = imfill(~CurrentThold,'holes');
CurrentImg_unfilled = CurrentImg-~CurrentThold;
areaofshape = sum(sum(CurrentImg));
Labelpic = bwlabel(~(~Labelpic-CurrentImg));

[x,y,z] = size(CurrentImg);
cancontinue=0;

        for ix = 1:x % If edge contacting, can't continue.
        If((CurrentImg(ix,1)==1) || (CurrentImg(ix,y)==1))
            cancontinue=1;
        end
        end
        for iy = 1:y
        if((CurrentImg(1,iy)==1) || (CurrentImg(x,iy)==1))
            cancontinue=1;
        end
        end

        area = sum(sum(CurrentImg)); minimumarea=10;

        if area<minimumarea % If object small, can't continue.
            cancontinue=1;
        end
```

```
% will only proceed if not edge-contacting and area > specified value
        if cancontinue == 0

              meanintensity =
                  meani(pic2,logical(CurrentImg),minimumsize,area);
            % return features for shape
            shapeproperties = regionprops(bwlabel(CurrentImg),'all');
            shapeproperties.image_internal = [];
            shapeproperties.image_internal =
       CurrentImg_unfilled( shapeproperties.BoundingBox(2):
    (shapeproperties.BoundingBox(2) + shapeproperties.BoundingBox(4)-1)
       , shapeproperties.BoundingBox(1):(shapeproperties.BoundingBox(1)
       + shapeproperties.BoundingBox(3)-1));

            [rows,cols] = size(CurrentImg); % work out the sums for
            x = ones(rows,1)*[1:cols];      % x
            y = [1:rows]'*ones(1,cols);     % y centroids

            area = sum(sum(CurrentImg));
            meanx = sum(sum(double(CurrentImg).*x))/area; %
            meany = sum(sum(double(CurrentImg).*y))/area;
            % calc centroids

            tic;
            [boundary,L] = bwboundaries(CurrentImg,8,'noholes');
            innerboundary = {[]};
            % Record list of interior holes
            [innerboundary, l2] = bwboundaries(bwmorph(
                          CurrentImg_unfilled,'thicken',1),8,'noholes');
            % record time of chain coding
            curvecalculationtime = toc;
            shapeproperties.boundarytime = curvecalculationtime;

            B = boundary{:};
            diameters=[]; minDiameter=area*area;
            maxDiameter=0; meanDiameter=0;
            radii=[]; minradius=area*area; maxradius=0; meanradius=0;
            longestdiampoint1 = [];  longestdiampoint2 = [];
            shortestdiampoint1 = []; shortestdiampoint2 = [];
```

```
halflist = uint16(size(B,1)/2);
for Bi = 1:uint16(size(B,1))

    YDistCentroid =
      max(meanx,B(Bi,2)) - min(meanx,B(Bi,2));
    XDistCentroid =
      max(meany,B(Bi,1)) - min(meany,B(Bi,1));
    DistCentroid = sqrt( (XDistCentroid*XDistCentroid) +
                   (YDistCentroid*YDistCentroid));
    radii = [radii;DistCentroid];

    if DistCentroid > maxradius
        maxradius = DistCentroid;
    end
    if DistCentroid < minradius
        minradius = DistCentroid;
    end

    if Bi < halflist
    YDistDiameter = max(B(Bi+halflist,2),B(Bi,2))
                    - min(B(Bi+halflist,2),B(Bi,2));
    XDistDiameter = max(B(Bi+halflist,1),B(Bi,1))
                    - min(B(Bi+halflist,1),B(Bi,1));
    DistDiameter = sqrt((XDistDiameter*XDistDiameter) +
                   (YDistDiameter*YDistDiameter));
    diameters = [diameters;DistDiameter];

        if DistDiameter > maxDiameter
            maxDiameter = DistDiameter;
            longestdiampoint1 = [1 1; B(Bi,1) B(Bi,2)];
            longestdiampoint2 =
            [1 1;B(Bi+halflist,1) B(Bi+halflist,2)];
        end
        if  DistDiameter < minDiameter
            minDiameter = DistDiameter;
            shortestdiampoint1 = [1 1; B(Bi,1) B(Bi,2)];
            shortestdiampoint2 =
            [1 1; B(Bi+halflist,1) B(Bi+halflist,2)];
        end
    end
end
```

```
% determine other shape features
meanradius = mean(radii);
meandiameter = mean(diameters);
EquivelentDiameter = 2/(sqrt(area/pi));
% Compactness
Compactness = (4*pi) * (area/
    (shapeproperties.Perimeter*shapeproperties.Perimeter));
% Aspect ratio and shape factors
if (maxradius>0)
Raspectratio = minradius/maxradius;
end

RMSD = sqrt((
((maxradius-meanradius)*(maxradius-meanradius))
+
((meanradius-minradius)*(meanradius-minradius))
)/2);
Rshapefactor = RMSD/meanradius;

if (maxDiameter>0)
Daspectratio = minDiameter/maxDiameter;
end

RMSD = sqrt((
((maxDiameter-meandiameter)*(maxDiameter-meandiameter))
+
((meandiameter-minDiameter)*(meandiameter-minDiameter))
)/2);
Dshapefactor = RMSD/meandiameter;

% Form Factor
MaxFfactor=0; MeanFfactor=0; MinFfactor=0;
MaxFfactor = area/(maxDiameter*maxDiameter);
MeanFfactor = area/(meandiameter*meandiameter);
MinFfactor = area/(minDiameter*minDiameter);
% Basin Elongation
MaxElongation=0; MeanElongation=0; MinElongation=0;
MaxElongation  =
      (2*sqrt(area)) / (maxDiameter*sqrt(pi));
MeanElongation =
      (2*sqrt(area)) / (meandiameter*sqrt(pi));
MinElongation  =
      (2*sqrt(area)) / (minDiameter*sqrt(pi));
% Lemniscate ratio
MaxLemniscate = 0; MeanLemniscate = 0; MinLemniscate = 0;
MaxLemniscate  =
      ((maxDiameter*maxDiameter)*pi)  / (4*area);
MeanLemniscate =
      ((meandiameter*meandiameter)*pi)/ (4*area);
MinLemniscate  =
      ((minDiameter*minDiameter)*pi)  / (4*area);
% retrieve inverse tangent curvature plot of perimeter
angle=10;
m = 360/((angle)*2);
curvature=boundarycurvature(boundary{:},m);
curvecalculationtime=toc;
% harmonics of fourier curvature analysis
harmonics=fouriercurve(curvature,1:30);
harmonicstop=toc;
% signature plots of shape
shapeproperties.Radii =            radii;
```

```
        shapeproperties.Diameters =        diameters;
        shapeproperties.Curvature =        curvature;
        tic;
        shapeproperties.MiniCurve =
                          boundarycurvature_fixed(boundary{:},4);
        minicurvecalculationtime = toc;
        tic;
        % REMEMBER - internal curve needs to be inversed - 0-curve
        shapeproperties.innerBoundary = [];
        shapeproperties.MiniCurve_Internal = [];
        shapeproperties.MiniCurve_Counter = [];

counter=0;

        while length(innerboundary)>0 && counter<2
            temp = innerboundary(1); % inbound(1);

            if length(temp{:})>25
                shapeproperties.innerBoundary =
                    [shapeproperties.innerBoundary ; temp{:}];
                shapeproperties.MiniCurve_Counter =
                    [shapeproperties.MiniCurve_Counter;
                    (length(shapeproperties.MiniCurve_Internal) +
                    length(temp{:}))];
                shapeproperties.MiniCurve_Internal =
                    [shapeproperties.MiniCurve_Internal ; 0-
                     boundarycurvature_fixed(temp{:},4)];
            end

            if(length(innerboundary)>1)
            innerboundary = innerboundary(2:length(innerboundary));
            else
            innerboundary=[];
            end
        counter = counter+1;
        end
```

```
shapeproperties.fkz =
      fkz_algorithm(pic, boundary{:});
fkzcalculationtime = toc;
% stats of signature plots of shape kurtosis
 shapeproperties.Radii_kurtosis =
      kurtosis(radii);
 shapeproperties.Diameters_kurtosis =
      kurtosis(diameters);
 shapeproperties.Curvature_kurtosis =
      kurtosis(curvature);
% skewness
 shapeproperties.Radii_skewness =
      skewness(radii);
 shapeproperties.Diameters_skewness =
      skewness(diameters);
 shapeproperties.Curvature_skewness =
      skewness(curvature);
% shape perimeter pixel list
shapeproperties.Boundary = boundary{:};
%area perimeter features
shapeproperties.PerimeterLength=
      shapeproperties.Perimeter;
shapeproperties.ConvexRatio =
      area/shapeproperties.ConvexArea;
shapeproperties.EquivDiameter=     EquivelentDiameter;

% simple shape features
shapeproperties.Compactness=       Compactness;
% diameter based features
shapeproperties.Daspectratio=      Daspectratio;
shapeproperties.Dshapefactor=      Dshapefactor;
% radius based features
shapeproperties.Raspectratio=      Raspectratio;
shapeproperties.Rshapefactor=      Rshapefactor;
% radii
shapeproperties.minradius=         minradius;
shapeproperties.maxradius=         maxradius;
shapeproperties.meanradius=        meanradius;
% diameters
shapeproperties.mindiameter=       minDiameter;
shapeproperties.maxdiameter=       maxDiameter;
shapeproperties.meandiameter=      meandiameter;
```

```
    %Graveluis's compactness (1914)
    shapeproperties.GCompactness=
          shapeproperties.Perimeter / (2*sqrt(pi)*area);
    %Form Factor (Horton 1932)
    shapeproperties.MaxFfactor=        MaxFfactor;
    shapeproperties.MeanFfactor=       MeanFfactor;
    shapeproperties.MinFfactor=        MinFfactor;
    %Basin elongation (Schumm 1956)
    shapeproperties.MaxElongation=     MaxElongation;
    shapeproperties.MeanElongation=    MeanElongation;
    shapeproperties.MinElongation=     MinElongation;
    %Lemniscate ratio (Chorley et al 1957)
    shapeproperties.MaxLemniscate =    MaxLemniscate;
    shapeproperties.MeanLemniscate =   MeanLemniscate;
    shapeproperties.MinLemniscate =    MinLemniscate;
    % fourier harmonics of shape
     shapeproperties.harmonics =        harmonics;
    % digitisation error function
    shapeproperties.digitalerror =
          shapeproperties.Perimeter/(0.79* area);

[p1,pl1,np,sc1,sc2] =
    curvegradientanalysis(shapeproperties.Curvature,3,100);
curvegradientanalysistime=toc;

% Record times of functions
shapeproperties.curvecalculationtime =
          curvecalculationtime;
shapeproperties.minicurvecalculationtime =
          minicurvecalculationtime;
shapeproperties.fkzcalculationtime =
          fkzcalculationtime;
shapeproperties.curvegradientanalysistime =
          curvegradientanalysistime;
shapeproperties.harmonicstime =
          harmonicstop;
shapeproperties.splinedcurve = sc2;
shapeproperties.shrunkcurve = sc1;
shapeproperties.sortedcurve = pl1;
shapeproperties.sortedcurvepeaks = np;

% **** texture analysis
    meanintensity=0; stdofintensity=[];

    if max(max(pic2))>1
          shapeproperties.RiceTexture =
                uint8(pic2) - uint8(gray2ind(~CurrentImg,256));
          meanintensity = uint8(
 sum(sum(shapeproperties.RiceTexture)) / shapeproperties.Area);
    else
          shapeproperties.RiceTexture = uint8(CurrentImg*255);
          meanintensity = uint8(
 sum(sum(shapeproperties.RiceTexture)) / shapeproperties.Area);
    end

    meanimage = shapeproperties.RiceTexture;
```

```
            for x = 1:size(shapeproperties.RiceTexture,1)
             for y = 1:size(shapeproperties.RiceTexture,2)
                 if shapeproperties.RiceTexture(x,y)==0
                     meanimage(x,y)=meanintensity;
                 else
                     stdofintensity =
                     [stdofintensity; shapeproperties.RiceTexture(x,y)];
                 end
             end
             end

             if ~isempty(stdofintensity)
             stdofintensity = std(double(stdofintensity));
             else
                 stdofintensity=0;
             end

             shapeproperties.meanintensity = meanintensity;
             shapeproperties.stdofintensity = stdofintensity;
             shapeproperties.edgeartearatio = 0;

             % Chalkiness detection - not used.
             chalkiness = uint8(
                          shapeproperties.RiceTexture-meanintensity);
             chalkiness = im2bw(chalkiness,1/256);
             chalkiness = bwdist(~chalkiness,'quasi-euclidean') ;
             chalkiness = im2bw( uint8(chalkiness), 3/256 );
             chalkiness = spotsingrain( chalkiness,
             {shapeproperties.Centroid}, {shapeproperties.BoundingBox});

             shapeproperties.proportionof chalkiness =
                   sum(sum(double(chalkiness)))/shapeproperties.Area;

         % Add to list of all shapes.
         if shapeproperties.Area > minimumsize
             ReturnedShapes = [ReturnedShapes;shapeproperties];
         end

         end

end  % end of master loop for each object.

    for i=1:length(ReturnedShapes)
         ReturnedShapes(i).edgeartearatio = meangrey(i);
    end
```

Spurring

```
function im=spur(im) % assume image boundary is padded

width = size(im,1);height = size(im,2);
% Expand Image to avoid masking over the image edge
im = [[zeros(size(im,2)+2,1)];…
      [zeros(size(im,1),1) im(:,:) zeros(size(im,1),1)];…
       zeros(size(im,2)+2,1)]];
spurringperformed=true;

   while(spurringperformed) % Repeat while flag reset
      spurringperformed=false;
      for (x=2:width)
      for (y=2:height) % assume defined from image
         if(im(x,y)~=0)
              if sum(sum(im((x-1:x+1),(y-1:y+1)))) <= 2 % spur pixel
                  im(x,y)=0; spurringperformed=true;
              end
         end
      end
      end
   end
```

Inversion

```
function image=invert(image)

      for (x=1:width)
      for (y=1:height) % assume defined from image
            if (image (x,y)>0)
                  image (x,y)=0;
            else
                  image (x,y)=1;
            end
      end
```

Filling Holes

```
function image=fillholes(im)

      width = size(im,1);height = size(im,2);
      % Expand Image to avoid masking over the image edge
      im = [[zeros(size(im,2)+2,1)];…
            [zeros(size(im,1),1) im(:,:) zeros(size(im,1),1)];…
            [im2.';im2b;im2.']; zeros(size(im,2)+2,1)]];

      image =invert(im);
      LabelledImage = Label(image) % apply CCA
      background = LabelledImage (1,1);
      for (x=1:width)
      for (y=1:height) % assume defined from image
            if(image(x,y) == background
            image(x,y)=0;
            else
            image(x,y)=1;
            end
      end
      end
```

Extracting Holes from an image

```
function image=extract_holes(image)

      image = invert(image + invert(fillholes(image)));
```

Boundary Curvature Function

Returns arrays of several representations of the curvature data. Parameter m required for specifying distance between curvature points (see chapter 5).

```
% Function Boundarycurvature
% returns the following:
% curvature - the curvature of the shape as a 1 dimensional plot equal in
length to the 2D boundary vector.

function curvature = boundarycurvature(boundary,m)

b_length = length(boundary(:,1)); % boundary length
m = double(uint16(boundarylength/m));
boundary = boundary(1:boundarylength,:);

% create two lists for -
%boundary minus m
JminusM = [
    boundary((b_length-m): b_length,:); boundary(1:(( b_length-m)-1),:)
    ];
% boundary plus m
JplusM = [boundary((m): b_length,:); boundary(1:((m)-1),:)];
% in order for easy iteration down the list at
% the start and end of the list

allXminus =  boundary(:,1) - JminusM(:,1);
allXplus  =  JplusM(:,1)   - boundary(:,1);
allYminus =  boundary(:,2) - JminusM(:,2);
allYplus  =  JplusM(:,2)   - boundary(:,2);
curvature = atan2(allYplus,allXplus) - atan2(allYminus,allXminus);
mincurvature = 2; mincurvaturePoint = 1;

for j=1:size(boundary(:,1))
    if curvature(j)>pi
        curvature(j) = (curvature(j) - (2*pi));
    end
    if curvature(j)<-pi
        curvature(j) = (curvature(j) + (2*pi));
    end

    if curvature(j) < curvature(mincurvaturePoint)
        mincurvature = curvature(j);
        mincurvaturePoint = j;
    end
end

%use the minimum point of curvature as the frame of reference

if j>1
curvature = [ curvature(j:size(curvature,1)) ; curvature(1:(j-1)) ];
end
```

Fourier Curvature

This function returns the harmonics of curvature for grains of a specific range of fourier harmonics.

```
function [harmonics]=fouriercurve(curve,harm_range)

if size(curve,2)>size(curve,1)
    curve = curve';
end

harm_number = length(harm_range);

N = length(curve);

k=1:N;
n = 1:length(harm_range);
    if (min(harm_range)>1)
        n = n + min(harm_range)-1;
    end
kn = (k'*n)';

cosvalues = cos((2*pi*kn) / N);
sinvalues = sin((2*pi*kn) / N);

curveb = curve(:,ones(1,harm_number));

cosshrunk = curveb'.*cosvalues;
An = (2/N)*sum(cosshrunk');

sinshrunk = curveb'.*sinvalues;
Bn = (2/N)*sum(sinshrunk');

C = sqrt( (An.^2) + (Bn.^2) );

% PLOTTING DATA

% OUTPUT
harmonics = C;
```

Mean Object Intensity

Returns the mean intensity of each object specified in region I according to each separate region B.

```
% Return mean intensity of objects in imageB using intensity data of
% imageA edge/area ratio can be calculated by first applying edge
% detection.  Returns 2D array of intensity and area for objects.

function meangrey=meani(inputI,inputB)

binaryregions = bwlabel(inputB); %connected components
numberofobjects = max(max(binaryregions));
objectsize = uint32(zeros(numberofobjects,1));
objectintensity = uint32(zeros(numberofobjects,1));
meangrey = double(zeros(numberofobjects,1));

for x = 1:size(inputI,1) % record area/object and intensity/region
    for y = 1:size(inputI,2)
        if(binaryregions(x,y)>0)
            currentpoint = binaryregions(x,y);
            objectsize(currentpoint) =
                  uint32(objectsize(currentpoint))+1;
            objectintensity(currentpoint) =
                  objectintensity(currentpoint) + uint32(inputI(x,y));
        end
    end
end

for i = 1:size(meangrey,1) % store edge/area ratio.
    meangrey(i) = double(objectintensity(i))/double(objectsize(i));
end
```

FKZ Algorithm

Uses the recorded shape boundary as input (2D array) and returns plot of FKZ values

```
% input parameters:
% 2D binary image and a 2-vector list of x,y boundary coordinates.

function concaveness=fkz_algorithm(image,Boundary)

concaveness = zeros(length(Boundary));

for i=1:length(Boundary)

 if i==1
  x1 = Boundary(length(Boundary),1); y1 = Boundary(length(Boundary),2);
 else
  x1 = Boundary(i-1,1); y1 = Boundary(i-1,2);
 end

 x2 = Boundary(i,1); y2 = Boundary(i,2);

 if i==length(Boundary)
  x3 = Boundary(1,1); y3 = Boundary(1,2);
 else
  x3 = Boundary(i+1,1); y3 = Boundary(i+1,2);
 end

  concaveness(i) =
      concaveness(i) + fkz_mask(x2,y2,image,concaveness(i) );
end

function concaveness = fkz_mask(xvalue,yvalue,image,concaveness)

  counter1=-2;
  while (counter1<3)
   counter2=-2;    currentX = xvalue-counter1;
    if(currentX>0 && currentX<=size(image,1))
     while (counter2<3)
      currentY = yvalue-counter2;
       if(currentY>0 && currentY<=size(image,2))
        if(image(currentX,currentY)>0)
         concaveness = concaveness+1;
        end
       end
      counter2=counter2+1;
     end
    end
   counter1=counter1+1;
  end
```

C++ code for recording objects in images

Early in the course of research a C++ program was created for basic image processing and feature extraction from Binary images. This application includes a structure for storing images, and several functions. These include an 8-way labelling function, similar to MATLAB's bwlabel function, to identify individual objects. These functions also perform boundary detection and calculation of many of the basic shape descriptors.

This code used a separate main file incorporating the CImg library to load the image from a file and then convert it to the specified ImgFile structure for use with the ImgManip class. The CImg Library is available from the URL http://cimg.souceforge.net.

Header File

```
#pragma once

//represents image pixels, multiple formats considered in development
struct ImgFile
{
            unsigned int * pixel; unsigned char * data;
            unsigned int x; unsigned int y;
            unsigned int * red;
            unsigned int * green;
            unsigned int * blue;
};


//an instance of an object to be stored in a list of objects
typedef struct particleList
{
         unsigned int value;
         double centroidx; double centroidy;

          unsigned int * x;unsigned int * y;
            unsigned int * red;unsigned int * green;unsigned int * blue;
            unsigned int * edgeorder;unsigned int * originator;
            bool * isedge;
            int edgepixels;int blobsize;int perimeter;
            double shortestdistance,double longestdistance,
            double shortestperp, double longestperp,
            double perp_1,double perp_2,
            double aspectratio, double shapefactor,
            double paspectratio1, double pshapefactor1,
            double paspectratio2, double pshapefactor2,
            double averageratio,
            double compactratio, double Spreading, double Elongation,
            double max, double min, double mean, double p_mean,
            double inertia1, double inertia2,
            double circulararea,
        unsigned int particlecount; struct particleList *Lptr;
} particleList;
//pixel  coordinates,  distance  from  centroid,  wherever  is  edge  data,
pointer to next pixel.
//particle image data, centroid of particle, pointer to next particle
```

```
class imgmanip
{
public:
	imgmanip(void);
	imgmanip(int w, int h);
	~imgmanip(void);

	void ReadImage();
	void minimumpixels(unsigned int newmin);
	particleList * listBlobs(struct ImgFile * thisImg);

private:

	bool idx(unsigned int m,unsigned int n);
	unsigned int pixelnum(unsigned int m,unsigned int n);
	void chaincode(unsigned int x, unsigned int y);
	unsigned int chainloop(
				unsigned int x, unsigned int y,
				unsigned int * visitedX,
				unsigned int * visitedY,
				unsigned int * visitedCode,
				unsigned int ordercounter,
				unsigned int code);
	double ecDistance(int x1,int y1,int x2, int y2);
	void findcentroids();

	bool isinitialised;
	int width;
	int height;
	ImgFile * thisImg;
	particleList * part_list;
	particleList * temp_part_list;
	unsigned int minpixels;
};
```

cpp File

```
#include "StdAfx.h"
#include ".\imgmanip.h"
#include <cmath>
#include <fstream>
#include <iostream>
#include <string>

using namespace std;

imgmanip::imgmanip(void)
{
      part_list=NULL;
}

imgmanip::imgmanip(int w, int h)
{
      width = w;
      height = h;
      part_list=NULL;
}


imgmanip::~imgmanip(void)
{
}

void  imgmanip::minimumpixels(unsigned  int  newmin)  //Set  minimum  size
object to record
{
      minpixels = newmin;
}
```

```
void imgmanip::ReadImage(char *filename)
{

      ofstream fout(filename);

      printf("\nWriting file.");
      fout << "PARTICLE DATA\n";
      fout << "\n";

     int numberofParticles=0;


             fout << "\n Particle:\t"
                    << "x:\t"
                    << "y:\t"
                    << "Area:\t"
                    << "Perimeter:\t"
                    << "Edge Length :\t"
                    << "Error :\t"
                    << "\t"
                    << "Max Diameter:\t"
                    << "Min Diameter:\t"
                    << "Mean Diameter:\t"
                    << "MostDistance:\t"
                    << "LongestRadii:\t"
                    << "ShortestRadii:\t"
                    << "Mean Radii :\t"
                    << "\t"
                    << "Inertia 1:\t"
                    << "Inertia 2:\t"
                    << "\t";

       temp_part_list = part_list;

       //printf("\n blobsize-%i ",temp_part_list->blobsize);
       printf("\n minpixels-%i ",minpixels);
```

```
        while(temp_part_list){


              if(temp_part_list->blobsize >= minpixels){
              numberofParticles++;

            fout << "\n" << numberofParticles <<"\t"
                   << temp_part_list->centroidx  <<"\t"
                   << temp_part_list->centroidy  <<"\t"
                   << temp_part_list->blobsize <<"\t"
                     << temp_part_list->perimeter+1 <<"\t"
                     << temp_part_list->edgelength <<"\t"
                     << temp_part_list->error <<"\t"
                     << "\t"
                   << temp_part_list->max <<"\t"
                   << temp_part_list->min <<"\t"
                     << temp_part_list->mean_diameter << "\t"

                     << temp_part_list->mostdistance <<"\t"
                     << temp_part_list->longestdistance <<"\t"
                     << temp_part_list->shortestdistance <<"\t"
                     << temp_part_list->mean_radii <<"\t"
                     << "\t"
                     << temp_part_list->inertia1 <<"\t"
                     << temp_part_list->inertia2 <<"\t";

              temp_part_list = (*temp_part_list).Lptr;
        }

       fout << "\n";

       fout.close();
     printf("\nFile Written.\n");

}
```

```
// ********    *************    ****************    *******   ****
// This function labels the binary image objects so they can be
// independently assessed. - CONNECTED COMPONENT ANALYSIS FUNCTION -

particleList * imgmanip::listBlobs(struct ImgFile * thisImg)
{

  int x = (int)thisImg->x;
  int y = (int)thisImg->y;

  bool printimage=true;

  //     Source Image

     /* 2 Iterations:
          First iteration:  Scan image top->bottom
          scans left and top right pixels
          Second iteration:  Scan image bottom->top
          scans right and bottom left pixels
     */

      printf("\nCounting Seperate Objects...");

       unsigned int passedCounter=1;

     //FIRST ITERATION xxyy
      for(int xxyy=0; xxyy<x*y; xxyy++)
        {
                if(*(thisImg->pixel+xxyy)>0){

                             *(thisImg->pixel+xxyy) = passedCounter++;
```

```
/* >**
   ***
   *** */

if(xxyy - ((x)+1) > 0){
if(*(thisImg->pixel+xxyy-((x)+1)) > 0){
if(*(thisImg->pixel+xxyy-((x)+1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-((x)+1));}
else {*(thisImg->pixel+xxyy-((x)+1)) = *(thisImg->pixel+xxyy);}
}
}

/* *>*
   ***
   *** */

if(xxyy - (x) > 0){
(*(thisImg->pixel+xxyy-(x)) > 0){
if(*(thisImg->pixel+xxyy-(x)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-(x));}
else {*(thisImg->pixel+xxyy-(x)) = *(thisImg->pixel+xxyy);}
}
}

/* **>
   ***
   *** */
if(xxyy - ((x)-1) > 0){
if(*(thisImg->pixel+xxyy-((x)-1)) > 0){
if(*(thisImg->pixel+xxyy-((x)-1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-((x)-1));}
else {*(thisImg->pixel+xxyy-((x)-1)) = *(thisImg->pixel+xxyy);}
}
}

/* ***
   >**
   *** */
if(xxyy - 1 > 0){
if(*(thisImg->pixel+xxyy-(1)) > 0){
if(*(thisImg->pixel+xxyy-(1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-(1));}
else {*(thisImg->pixel+xxyy-(1)) = *(thisImg->pixel+xxyy);}
}
}

                }
           }//end xxyy

          passedCounter=1;

            //SECOND ITERATION - xxyy
         for(int xxyy=(x*y)-1; xxyy>=0; xxyy--)
            {
                     if(*(thisImg->pixel+xxyy)>0){
```

```
/* ***
   ***
   >** */
if(xxyy + ((x)-1) < x*y){
if(*(thisImg->pixel+xxyy+((x)-1)) > 0){
if(*(thisImg->pixel+xxyy+((x)-1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy+((x)-1));}
else {*(thisImg->pixel+xxyy+((x)-1)) = *(thisImg->pixel+xxyy);}
}
}

/* ***
   ***
   *>* */
if(xxyy + (x) < x*y){
if(*(thisImg->pixel+xxyy+(x)) > 0){
if(*(thisImg->pixel+xxyy+(x)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy+(x));}
{*(thisImg->pixel+xxyy+(x)) = *(thisImg->pixel+xxyy);}
}
}


/* ***
   ***
   **> */
if(xxyy + ((x)+1) < x*y){
if(*(thisImg->pixel+xxyy+((x)+1)) > 0){
if(*(thisImg->pixel+xxyy+((x)+1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy+((x)+1));}
else {*(thisImg->pixel+xxyy+((x)+1)) = *(thisImg->pixel+xxyy);}
}
}

/* ***
   **>
   *** */
if(xxyy + 1 < x*y){
if(*(thisImg->pixel+xxyy+(1)) > 0){
if(*(thisImg->pixel+xxyy+(1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy+(1));}
{*(thisImg->pixel+xxyy+(1)) = *(thisImg->pixel+xxyy);}
}
}

/* >**
   ***
   *** */
if(xxyy - ((x)+1) > 0){
if(*(thisImg->pixel+xxyy-((x)+1)) > 0){
if(*(thisImg->pixel+xxyy-((x)+1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-((x)+1));}
else {*(thisImg->pixel+xxyy-((x)+1)) = *(thisImg->pixel+xxyy);}
}
}
```

```
/*  *>*
    ***
    ***  */
if(xxyy - (x) > 0){
if(*(thisImg->pixel+xxyy-(x)) > 0){
if(*(thisImg->pixel+xxyy-(x)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-(x));}
else {*(thisImg->pixel+xxyy-(x)) = *(thisImg->pixel+xxyy);}
}
}

/*  **>
    ***
    ***  */


if(xxyy - ((x)-1) > 0){
if(*(thisImg->pixel+xxyy-((x)-1)) > 0){
if(*(thisImg->pixel+xxyy-((x)-1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-((x)-1));}
{*(thisImg->pixel+xxyy-((x)-1)) = *(thisImg->pixel+xxyy);}
}
}


/*  ***
    >**
    ***  */
if(xxyy - 1 > 0){
if(*(thisImg->pixel+xxyy-(1)) > 0){
if(*(thisImg->pixel+xxyy-(1)) < *(thisImg->pixel+xxyy) )
{*(thisImg->pixel+xxyy) = *(thisImg->pixel+xxyy-(1));}
else {*(thisImg->pixel+xxyy-(1)) = *(thisImg->pixel+xxyy);}
}
}

                    passedCounter = *(thisImg->pixel+xxyy);

            } //end xxyy
        }
```

```
unsigned int currentblob=0;
unsigned int blobcount=0;
bool areedgeparticles=false;

 // *** Loop to detect & remove edge contacting blobs.

  for(int xxyy=0; xxyy<x*y; xxyy++)
{

        if(currentblob=0 | *(thisImg->pixel+xxyy) < currentblob)
        {
              currentblob = *(thisImg->pixel+xxyy);
        }

        if( (xxyy<x || xxyy>(x*y)-x
              || (xxyy%x)==0 || (xxyy+1%x)==0)
              && (*(thisImg->pixel+xxyy) > 0) )
        {

              areedgeparticles=true;
              currentblob = *(thisImg->pixel+xxyy);

              for(int tempxxyy=0; tempxxyy<x*y; tempxxyy++)
              {
                  if(*(thisImg->pixel+tempxxyy)==currentblob)
                    {*(thisImg->pixel+tempxxyy)=0;}
              } // end for

        } //end if
  } // end for
```

```
// *** Loop to create list of different blobs.
        currentblob=0;
        int imagesize = x*y;

        for(unsigned int xxyy=0; xxyy<imagesize; xxyy++)
      {
               bool blob_alreadyvisited=true;
                    // *** scan through image until find next blob
              while((*(thisImg->pixel+xxyy)==currentblob
 | *(thisImg->pixel+xxyy)==0) && xxyy<(x*y)){
                                  xxyy++;
                    }

                    // *** if found next blob
                    if(*(thisImg->pixel+xxyy)>currentblob)
                {
                              blobcount++;
                       currentblob = *(thisImg->pixel+xxyy);
              unsigned int currentblobsize=0,lastpoint=0;
              // *** if member of current blob, increment counter.
                    for(int tempxxyy=xxyy; tempxxyy<x*y; tempxxyy++){
                          if(*(thisImg->pixel+tempxxyy)==currentblob)
                                {
                                      currentblobsize++;
                                }
                          }

                          // *** create a new instance of a particle
                          particleList * curr = new particleList;
                          // *** allocate memory
                          (*curr).x =
(unsigned int*)malloc((currentblobsize)*sizeof(unsigned int));
                          (*curr).y =
(unsigned int*)malloc((currentblobsize)*sizeof(unsigned int));
                          (*curr).isedge =
(bool*)malloc((currentblobsize)*sizeof(unsigned int));
                          (*curr).edgepixels =
(int)malloc(sizeof(int));
                          (*curr).value =
(unsigned int)malloc(sizeof(unsigned int));
                          (*curr).blobsize =
(unsigned int)malloc(sizeof(unsigned int));

                          // *** set value = current blob
                          (*curr).value = currentblob;
                          (*curr).blobsize = currentblobsize;

                          int edgepix=0;
                          int allpix=0;
                          unsigned int currentpointer=0;
                          int thispointer=0;

                          int tempxxyy =
(unsigned int)malloc(sizeof(unsigned int));



       // *** loop through all pixels in the image
       for(tempxxyy=x;
             tempxxyy<=(imagesize)-x; tempxxyy++)
       {
```

```
                    // *** particle edge detection routine
                    //get the x and y coordinates of each pixel.

                         thispointer = tempxxyy;

                if(*(thisImg->pixel+tempxxyy)==currentblob){
        *((*curr).x+currentpointer) = (tempxxyy)%x;
        *((*curr).y+currentpointer) = (tempxxyy-((tempxxyy)%x))/x;

               // 'Edge Pixel' Criteria:
               // count number of empty neighbours
               // must have 1 adjacent 4-way neighbours.
               // this algorithm will have trouble with particles
               // with holes.  convex hull or filling algorithm
               // might need to be applied to shape beforehand.

/* *X*
   X*X
   *X* -x||-1||+1||+x */

        thispointer = tempxxyy;

        if( *(thisImg->pixel+tempxxyy-(x))==0 ||
              *(thisImg->pixel+tempxxyy-(1))==0 ||
              *(thisImg->pixel+tempxxyy+(1))==0 ||
              *(thisImg->pixel+tempxxyy+(x))==0 )

        {*((*curr).isedge+currentpointer) = true;edgepix++;

        }else
        {*((*curr).isedge+currentpointer) = false;}
        // *** true/false denotes edge.
        currentpointer++;
        allpix++;
        }//end if


    }//end for tempxxyy
```

```
      // *** End loop through all pixels in the image
      // *** set number of edge pixels detected.
       (*curr).blobsize = (unsigned int)allpix;
       (*curr).edgepixels = edgepix;
      // *** allocate memory for edge data order.

                              (*curr).edgeorder            =           (unsigned
int*)malloc((edgepix+2)*sizeof(unsigned int));
                              (*curr).originator           =           (unsigned
int*)malloc((edgepix+2)*sizeof(unsigned int));

                              // *** initialise edgeorder list.
                              for(int tempcounter=0; tempcounter<edgepix;
                                   tempcounter++){
                              *(curr->edgeorder+tempcounter) = 0;
                              }
                              // *** repeat for next object.
                        (*curr).Lptr = part_list;
                        part_list = curr;
                  }//end  if
                        // *** End if found next blob
       }// end xxyy
      // *** End Loop to create list of different blobs.

      // *** Output edge data
       int numberofParticles=0;

       temp_part_list = part_list;
       while(temp_part_list){
       numberofParticles++;
       (*temp_part_list).value = numberofParticles;
       temp_part_list = (*temp_part_list).Lptr;
       }

       temp_part_list = part_list;

       int tplcounter=1;
       while(temp_part_list){
             if((*temp_part_list).edgepixels>0){

             //Find centers of gravity per shape
             findcentroids();

             chaincode(*((*temp_part_list).x),*((*temp_part_list).y));

             int sortededges = 0;
             int tmpx=0,tmpy=0;
             int orig=0;
             int edgeC=0;

             }
             temp_part_list = (*temp_part_list).Lptr;
             tplcounter++;
       }

       return part_list;
}
// **** chainloop() - chain coding function
unsigned int imgmanip::chainloop(
      unsigned int x, unsigned int y,
	unsigned int * visitedX, unsigned int * visitedY, unsigned int *
visitedCode,
	unsigned int ordercounter, unsigned int code)
{

	if(*(visitedCode+ordercounter)>0 && *(visitedCode+ordercounter)<9)
	{}else{
	*(visitedX+ordercounter)=x;
	*(visitedY+ordercounter)=y;
	*(visitedCode+ordercounter)=code;

//Loop through specified particle's image data.
//Look for the first (top left) edge pixel, then produce a list of
// neighbours in clockwise order around the edge of the shape.

	unsigned int previouscode;

		if(code>2){ code=code-2; }else{
			if(code==2){code=8;}
			if(code==1){code=7;}
		}
		previouscode=code;

	while(ordercounter < temp_part_list->edgepixels) {

		unsigned int x2,y2;

		while(ordercounter < temp_part_list->edgepixels
		 && ordercounter>=0){
			switch(code){

				/* 781
				   6 2
				   543      - order of coding */
				case 1:
					x2=(x)+1;y2=(y)-1;break;  // top right
				case 2:
					x2=(x)+1;y2=(y);break;    // right
				case 3:
					x2=(x)+1;y2=(y)+1;break;  // bottom right
				case 4:
					x2=(x);y2=(y)+1;break;    // bottom
				case 5:
					x2=(x)-1;y2=(y)+1;break;  // bottom left
				case 6:
					x2=(x)-1;y2=(y);break;    // left
				case 7:
					x2=(x)-1;y2=(y)-1;break;  // top left
				case 8:
					x2=(x);y2=(y)-1;break;    // top
				default:
					break;
			}

			if(idx(x2,y2)==true)
			{

				int tempvisitcount=0;
				bool isvisited=false;
				while(tempvisitcount<=ordercounter)
				{
```

```
                                    if(x2==*(visitedX+tempvisitcount)
                                     && y2==*(visitedY+tempvisitcount))
                                    {isvisited=true;}
                                    tempvisitcount++;
                              }

                              if(!isvisited){
                              //Recur for next point
                              ordercounter =
  chainloop(x2,y2,visitedX,visitedY,visitedCode,ordercounter+1,code);
                              }
                         }
                              if(code<8){code++;}else{code=1;}
                              if(code==previouscode)
                              {return ordercounter;}
                    }
               }

               }return ordercounter;
}
```

```
// chain coding function for allocating chain code memory
// and for recording shape properties
void imgmanip::chaincode(unsigned int x, unsigned int y)
{

	if(temp_part_list->edgepixels>0)
	{

	unsigned int * visitedX = (unsigned int*)
malloc(1+(temp_part_list->edgepixels)*(sizeof(unsigned int)));
	unsigned int * visitedY = (unsigned int*)
malloc(1+(temp_part_list->edgepixels)*(sizeof(unsigned int)));
	unsigned int * visitedCode = (unsigned int*)
malloc(1+(temp_part_list->edgepixels)*(sizeof(unsigned int)));
	unsigned int * firstX = (unsigned int*)
malloc(1+(temp_part_list->edgepixels/2)*(sizeof(unsigned int)));
	unsigned int * firstY = (unsigned int*)
malloc(1+(temp_part_list->edgepixels/2)*(sizeof(unsigned int)));
	unsigned int * secondX = (unsigned int*)
malloc(1+(temp_part_list->edgepixels/2)*(sizeof(unsigned int)));
	unsigned int * secondY = (unsigned int*)
malloc(1+(temp_part_list->edgepixels/2)*(sizeof(unsigned int)));

	temp_part_list->edgedistance =
(double*)malloc(1+(temp_part_list->edgepixels)*(sizeof(double)));
	temp_part_list->edgeangle =
(double*)malloc(1+(temp_part_list->edgepixels)*(sizeof(double)));


	double * ecDist =
(double*)malloc((temp_part_list->edgepixels/2)*(sizeof(double)));

	bool notdone=true;
	unsigned int ordercounter=0;

	int code=2;
	ordercounter =
chainloop(x,y,visitedX,visitedY,visitedCode,ordercounter,code);

	int numberedgepixs=0;
	temp_part_list->edgelength=0;

	for(int edgelist=0;edgelist<=ordercounter;edgelist++)
	{
		if(*(visitedCode+edgelist)%2!=0)
		{
			temp_part_list->edgelength += 1.414213562;
		numberedgepixs++;
		}else{
			temp_part_list->edgelength += 1;
		}
	}

	// DIGITISATION ERROR FUNCTION
	temp_part_list->error =
		(0.0000179211111*temp_part_list->edgelength)/
		(0.79*(0.0000179211111*temp_part_list->blobsize));

	// **** edgepixels is invalid for shapes that contain holes -
	// this is a solution.
	temp_part_list->perimeter = ordercounter;
```

```
        /* produce a new list of ecludean distances from the two lists
           find the maximum and minimum distances. */

        double min=0, max=0, mean=0;
        double      longestradii=0,     shortestradii=0,     longest12radii=0,
shortest12radii=0;
        double shortestperp, longestperp, radii_1, radii_2;
        double p_min1=0, p_max1=0, p_mean1=0, perp_1=0;
        double p_min2=0, p_max2=0, p_mean2=0, perp_2=0;
        double current_perpdiameter=0, radii_p1,radii_p2;
        double current_diameter,current_radius,current_perpendicular;
        double       mean_diameter=0,       mean_radii=0,       mean_perp=0,
mean_perp_radii=0, mean_robs=0;
        int currentlongest1=0,currentlongest2=0;

        //retrieve diameters/radii

        for(int xc=0; xc<(temp_part_list->perimeter); xc++)
        {
                radii_1 =
ecDistance( *(visitedX+xc)  , *(visitedY+xc), temp_part_list->centroidx,
temp_part_list->centroidy);
                // retrieve all distances to centroid
                *(temp_part_list->edgedistance+xc) = radii_1;
                // and angles
                *(temp_part_list->edgeangle+xc) =
                        returnAngle(temp_part_list->centroidx,
temp_part_list->centroidy,*(visitedX+xc), *(visitedY+xc));
        }

        //find two most distant points.
        for(int xc1=0; xc1<(temp_part_list->perimeter); xc1++)
        {
                for(int xc2=0; xc2<(temp_part_list->perimeter); xc2++)
                {
                        if(temp_part_list->mostdistance                  <
ecDistance(*(visitedX+xc1),*(visitedY+xc1),*(visitedX+xc2),*(visitedY+xc2
)))
                        {temp_part_list->mostdistance                    =
ecDistance(*(visitedX+xc1),*(visitedY+xc1),*(visitedX+xc2),*(visitedY+xc2
));}
                }

        }
```

```
      for(int xc=0; xc<(temp_part_list->perimeter+1)/2; xc++)
      {
            int opposite = xc + (int)
((temp_part_list->perimeter+1)/2);
            int perpvalue = xc + (int)
((temp_part_list->perimeter+1)/4);
            int perpopposite=0;
            if(xc < (int)(temp_part_list->perimeter/4)){
                perpopposite    =    (temp_part_list->perimeter+1)    -
((int)((temp_part_list->perimeter+1)/4)-xc);
            }else{
                  perpopposite    =    xc    -    (int)((temp_part_list-
>perimeter+1)/4);
            }

// *** set the distances for lengths through shape - through centroid.

            radii_1   =   ecDistance(  *(visitedX+xc)   ,   *(visitedY+xc),
temp_part_list->centroidx, temp_part_list->centroidy);
            radii_2      =      ecDistance(      *(visitedX+opposite),
*(visitedY+opposite),     temp_part_list->centroidx,     temp_part_list-
>centroidy );
            current_diameter    =    ecDistance(    *(visitedX+xc)    ,
*(visitedY+xc), *(visitedX+opposite),*(visitedY+opposite) );

            radii_p1    =    ecDistance(    *(visitedX+perpvalue)    ,
*(visitedY+perpvalue),    temp_part_list->centroidx,    temp_part_list-
>centroidy);
            radii_p2    =    ecDistance(   *(visitedX+perpopposite)    ,
*(visitedY+perpopposite),   temp_part_list->centroidx,   temp_part_list-
>centroidy);
            current_perpendicular = ecDistance( *(visitedX+perpvalue) ,
*(visitedY+perpvalue),           *(visitedX+perpopposite)            ,
*(visitedY+perpopposite) );

            if((radii_1 < shortestradii || shortestradii==0)
                  && (radii_2 != longestradii))
                  {
                   shortestradii=radii_1;
                   shortest12radii=radii_1;
                   shortestperp=radii_p1;
                  }
            if((radii_1 > longestradii || longestradii==0)
                  && (radii_2 != shortestradii))
                  {
                   longestradii=radii_1;
                   longest12radii=radii_1;
                   longestperp=radii_p1;
                  }

            if((radii_2 < shortestradii || shortestradii==0)
                  && (radii_1 != longestradii))
                  {shortestradii=radii_2;shortestperp=radii_p2;}
            if((radii_2 > longestradii || longestradii==0)
                  && (radii_1 != shortestradii))
                  {longestradii=radii_2;longestperp=radii_p2;}

//non-perpendicular  method  (find  literally  the  longest  and  shortest
//diameters)
            if( current_diameter < min   || min==0 )
                  {min=current_diameter;}
            if(    current_diameter    >    max           ||    max==0    )
{max=current_diameter;currentlongest1=xc;currentlongest2=opposite;}

//perpendicular  method  (get  perpendicular  as  1/2  way  round  the  chain
//code)
            if(current_diameter<p_min1 || p_min1==0)
                  {p_min1=current_diameter;p_max1=current_perpendicular;}
            if(current_perpendicular<p_min1 || p_min1==0)
                  {p_min1=current_perpendicular;p_max1=current_diameter;}

            //Rob's method (double radii length using half the shape)
            if((2*radii_1)<p_min2 || p_min2==0)
                  {p_min2=(2*radii_1);}
            if((2*radii_1)<p_min1 || p_min1==0)
                  {p_max2=(2*radii_1);}

            mean_diameter = mean_diameter + current_diameter;
            mean_radii = mean_radii + radii_1 + radii_2;
            mean_perp = mean_perp + current_perpendicular;
            mean_perp_radii = mean_perp_radii + radii_p1 + radii_p2;
            mean_robs = mean_robs + (radii_1*2);
      }

            // ***
            // perpendicular:  find opposing points
            double DiameterAngle = 0;
            double PerpendicularAngle = 0;
            double DiameterDistance = 0;

            // determine angle between longest points
            DiameterAngle                                                   =
returnAngle(*(visitedX+currentlongest1),*(visitedY+currentlongest1),*(vis
itedX+currentlongest2),*(visitedY+currentlongest2));

            // determine perpendicular angle
            if (DiameterAngle > 90)
            {PerpendicularAngle = DiameterAngle - 90;}
            else
            {PerpendicularAngle = DiameterAngle + 90;}

      //loop between currentlongest1 and currentlongest2
            int xc=0;
                  if(currentlongest1>currentlongest2)
                        {xc=currentlongest1;}
                  else
                        {xc=currentlongest2;}

            //rob's method
            p_max2=(longest12radii*2);
            p_min2=(shortest12radii*2);

            //calculate mean
            mean = (max + min)/2;
            mean_robs = mean_robs  / ((temp_part_list->perimeter+1)/2);
            mean_diameter    =    mean_diameter     /     ((temp_part_list-
>perimeter+1)/2);
            mean_perp = mean_perp / ((temp_part_list->perimeter+1)/2);
            mean_radii = mean_radii /
                  ((temp_part_list->perimeter+1)/2);
            mean_perp_radii = mean_perp_radii /
                  ((temp_part_list->perimeter+1)/2);
            p_mean1 = p_mean1 /
                   ((temp_part_list->perimeter+1)/2);
            p_mean2 = p_mean2  /
                   ((temp_part_list->perimeter+1)/2);

    unsigned int pi = 3.14159265; double Longest,Shortest,rmsd;

      // inertia based
      temp_part_list->shortestdistance = shortestradii;
      temp_part_list->longestdistance = longestradii;
      temp_part_list->inertia1 =
longestradii*temp_part_list->blobsize;
      temp_part_list->inertia2 =
shortestradii*temp_part_list->blobsize;
      temp_part_list->max = max;
      temp_part_list->min = min;
      temp_part_list->mean = mean;
      temp_part_list->mean_diameter = mean_diameter;
      temp_part_list->mean_radii = mean_radii;

      }
}
```

```
// identify if pixel x at coordinates m,n is an edge pixel.
bool imgmanip::idx(unsigned int m,unsigned int n)
{
	for(int xxyy=0;xxyy<temp_part_list->blobsize;xxyy++)
	{
		if(*(temp_part_list->x+xxyy) == m
		    && *(temp_part_list->y+xxyy) == n )
		{
			return *(temp_part_list->isedge+xxyy);
		}
	}
	return false;
}

// return identifer for pixel based on coordinates.
unsigned int imgmanip::pixelnum(unsigned int x,unsigned int y)
{
	return x+(y*height);
}
// return euclidean distance.
double imgmanip::ecDistance(double x1,double y1,double x2, double y2)
{
	int xX,yY;

	if(x1>x2)
	{
		xX = x1-x2;
	}else{
		xX = x2-x1;
	}

	if(y1>y2)
	{
		yY = y1-y2;
	}else{
		yY = y2-y1;
	}
	return sqrt((double)((xX*xX)+(yY*yY)));

}
```

```
// return angle between two points in degrees
double imgmanip::returnAngle(double x1,double y1,double x2, double y2)
{
            double x12=0; double y12=0;
            unsigned int pi = 3.14159265;
            x12 = x1-x2;
            y12 = y1-y2;
            double anglemodifier = 0;
            double angle = atan(y12/x12)*(180/pi);

            // where x2 y2 are centroid coordinates, x1 y1 perimeter.
            // accelerate this code by using radians instead.

      if (x1<x2) // adjust according to direction of coordinates.
      {
                  anglemodifier = 270;
      }else{
            if (y1<y2)
            {
                  if (x1>x2)
                  {
                        anglemodifier = 90;
                  }else{
                        anglemodifier = 0;
                  }
            }else{
                  if (x1>x2)
                  {
                        anglemodifier = 90;
                  }else{
                        if (y1>y2)
                        {
                              anglemodifier = 180;
                        }else{
                              anglemodifier = 0;
                        }
                  }
            }
      }

      angle = angle + anglemodifier;

      if(angle<0){angle = 0 - angle;}

      if(angle>360){angle = angle - 360;}

            return angle;
}
```

```
//Find Centroid Routine **********
void imgmanip::findcentroids(){

     if(temp_part_list) {

            int blobsize = (*temp_part_list).blobsize;
            unsigned int cenX=0,cenY=0;
        double CurrX,CurrY;
            int currentpixel=0;

        while(currentpixel<blobsize) {
                  cenX += *(temp_part_list->x+currentpixel);
                  cenY += *(temp_part_list->y+currentpixel);
                  currentpixel++;
        }

            CurrX = (double)cenX/(double)temp_part_list->blobsize;
            CurrY = (double)cenY/(double)temp_part_list->blobsize;

            temp_part_list->centroidx = CurrX;
            temp_part_list->centroidy = CurrY;
     }
}
   //End of Find Centroid Routine **********
```

C # Application for creation of theoretical shapes

It would be impractical to describe the entire application; however the functions for generating shapes will be described here. The entire application can be found on the attached disk under *ImageRotater*.

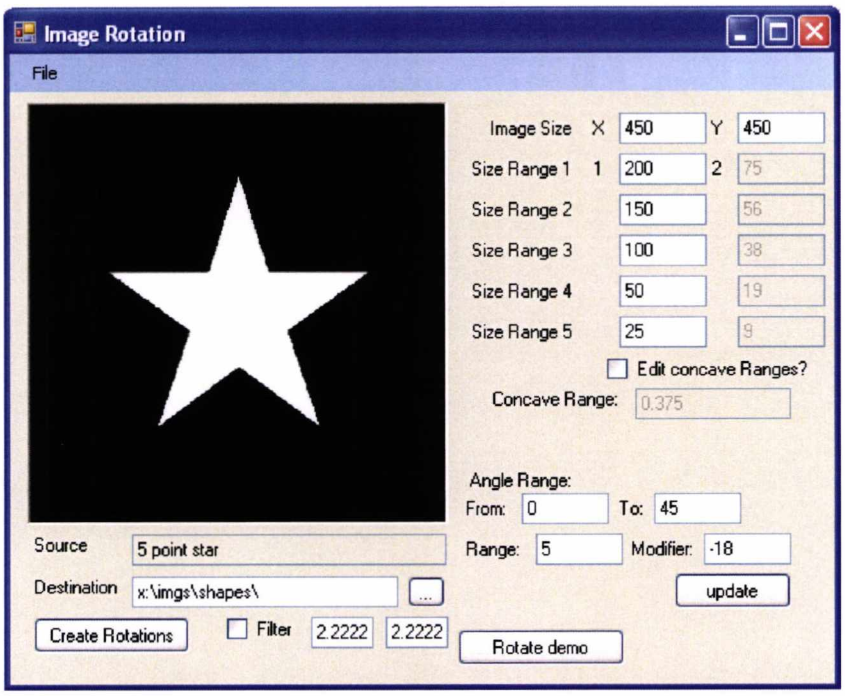


Image Rotation

File

Image Size	X	450	Y	450
Size Range 1	1	200	2	75
Size Range 2		150		56
Size Range 3		100		38
Size Range 4		50		19
Size Range 5		25		9

☐ Edit concave Ranges?

Concave Range: 0.375

Angle Range:

From: 0 To: 45

Range: 5 Modifier: -18

Source: 5 point star

Destination: x:\imgs\shapes\

☐ Filter 2.2222 2.2222

Several key points in the functionality of this software will now be outlined. The application takes a number of input parameters for generating shape images. Five different output shapes are produced using different size scales. The image size of the largest image is set in user interface then the dimensions of the longest and shortest axes. The shortest is generated automatically for applicable shapes, otherwise longest is used for regular geometric shapes. This shortest value is proportional to the longest axis by the value of the 'concave range'. By default, this value was approximated to 0.375. This was determined to be a valid ratio for the three types of star discussed earlier. The file... menu permits loading of an image, or the generation of an image of a specific binary shape.

The following functions are used to generate the various images of shapes. Squares, Triangle, Concave and Convex Polygons, and four-pointed crosses.

```
        private Image SquareImage(int width, int height)
        {
            Bitmap newImg = new Bitmap(initimage());

            for (int x = 0; x < newImg.Width; ++x)
                for (int y = 0; y < newImg.Height; ++y)
                {
                    newImg.SetPixel(x, y, Color.Black);
                }

            Image newImage = newImg;

            Graphics g = Graphics.FromImage(newImage);
            GraphicsPath jeff = new GraphicsPath();
            Point[] jeffpoints = new Point[4];
jeffpoints[0] = new Point((int)(newImg.Width * 0.25), (int)(newImg.Width
* 0.25));
jeffpoints[1] = new Point((int)(newImg.Width * 0.75), (int)(newImg.Width
* 0.25));
jeffpoints[2] = new Point((int)(newImg.Width * 0.75), (int)(newImg.Width
* 0.75));
jeffpoints[3] = new Point((int)(newImg.Width * 0.25), (int)(newImg.Width
* 0.75));

            jeff.AddPolygon(jeffpoints);
            //RegionData jeff = new RegionData();
            //jeff.
            Region bob = new Region(jeff);
            g.FillRegion(Brushes.White, bob);

            return newImage;

        }
        private Image TriangleImage()
        {
            Bitmap newImg = new Bitmap(initimage());

            for (int x = 0; x < newImg.Width; ++x)
                for (int y = 0; y < newImg.Height; ++y)
                {
                    newImg.SetPixel(x, y, Color.Black);
                }

            Image newImage = newImg;

            int trianglesidelength = 200;
            int                         triangleheight                =
Convert.ToInt16((double)(trianglesidelength                           *
Math.Sin((60/(180/Math.PI)))));
            MessageBox.Show(triangleheight.ToString());

        int topx = (int)(newImg.Width  / 2);
        int topy = (int)(newImg.Height / 2) - (int)(triangleheight / 2);

            Graphics g = Graphics.FromImage(newImage);
            GraphicsPath jeff = new GraphicsPath();
            Point[] jeffpoints = new Point[3];
jeffpoints[0] = new Point(topx, topy);
jeffpoints[1] = new Point(topx - (int)(trianglesidelength / 2), topy +
triangleheight);
jeffpoints[2] = new Point(topx + (int)(trianglesidelength / 2), topy +
triangleheight);
            Region bob = new Region(jeff);
            g.FillRegion(Brushes.White, bob);

            return newImage;

        }
        private Image ConvexPolygon(int sides, int radius)
        {
            Bitmap newImg = new Bitmap(initimage());
double angleadjust = rotationangle + double.Parse(AngleModifier.Text);

            for (int x = 0; x < newImg.Width; ++x)
                for (int y = 0; y < newImg.Height; ++y)
                {
                    newImg.SetPixel(x, y, Color.Black);
                }

            Image newImage = newImg;

            Graphics g = Graphics.FromImage(newImage);
            GraphicsPath jeff = new GraphicsPath();
            Point[] jeffpoints = new Point[sides];

            for (int i = 0; i < sides; i++)
            {
                double sideangle = angleadjust / (180 / Math.PI);
                if (i > 0)
                {
sideangle = sideangle + ((360 / (double)sides * i) / (180 / Math.PI));
                }
int xpos = Convert.ToInt16(radius * Math.Cos(sideangle));
int ypos = Convert.ToInt16(radius * Math.Sin(sideangle));
jeffpoints[i] = new Point((newImg.Width / 2) + xpos, (newImg.Height / 2)
+ ypos);
            }

            jeff.AddPolygon(jeffpoints);
            Region bob = new Region(jeff);
            g.FillRegion(Brushes.White, bob);

            return newImage;

        }
```

```
private Image ConcavePolygon(int sides, int radius1, int radius2)
        {
            // minimum sides=6

            Bitmap newImg = new Bitmap(initimage());
            double        angleadjust        =        rotationangle        +
double.Parse(AngleModifier.Text);

            for (int x = 0; x < newImg.Width; ++x)
                for (int y = 0; y < newImg.Height; ++y)
                {
                    newImg.SetPixel(x, y, Color.Black);
                }

            Image newImage = newImg;

            Graphics g = Graphics.FromImage(newImage);
            GraphicsPath jeff = new GraphicsPath();
            Point[] jeffpoints = new Point[sides];

            for (int i = 0; i < sides; i++)
            {
                double sideangle = angleadjust / (180 / Math.PI);
                if (i > 0)
                {
sideangle = sideangle + ((360 / (double)sides * i) / (180 / Math.PI));
                }
                int xpos = 0; int ypos = 0;
                if (i % 2 == 0)
                {
xpos = Convert.ToInt16(radius1 * Math.Cos(sideangle));
ypos = Convert.ToInt16(radius1 * Math.Sin(sideangle));
                }
                else
                {
xpos = Convert.ToInt16(radius2 * Math.Cos(sideangle));
ypos = Convert.ToInt16(radius2 * Math.Sin(sideangle));
                }

jeffpoints[i] = new Point((newImg.Width / 2) + xpos, (newImg.Height / 2)
+ ypos);
            }

            jeff.AddPolygon(jeffpoints);
            Region bob = new Region(jeff);
            g.FillRegion(Brushes.White, bob);

            return newImage;

        }
```

```
private Image CrossImage(int sides, int radius1, int crossspread)
        {
            // minimum sides=8=4*2

Bitmap newImg = new Bitmap(initimage());
double angleadjust = rotationangle + double.Parse(AngleModifier.Text);

            for (int x = 0; x < newImg.Width; ++x)
                for (int y = 0; y < newImg.Height; ++y)
                {
                    newImg.SetPixel(x, y, Color.Black);
                }

            Image newImage = newImg;

            Graphics g = Graphics.FromImage(newImage);
            GraphicsPath jeff = new GraphicsPath();
            Point[] jeffpoints = new Point[sides*3];

for (double currentside = 1; currentside <= sides; currentside++)
            {
                double sideangle = angleadjust / (180 / Math.PI);
                double previousangle = 0;
                double nextangle = 0;

sideangle = sideangle + ((double)(currentside - 1) / (double)sides) * (2
* Math.PI);

                int xpos1 = 0; int ypos1 = 0;
                int xpos2 = 0; int ypos2 = 0;
                int xpos3 = 0; int ypos3 = 0;

double subdistance = Math.Sqrt((crossspread * crossspread) + (radius1 *
radius1));
double subangle = Math.Atan((double)crossspread / (double)radius1);
                double currentangle1 = sideangle - subangle;

xpos1 = Convert.ToInt16(subdistance * Math.Cos(currentangle1));
ypos1 = Convert.ToInt16(subdistance * Math.Sin(currentangle1));

jeffpoints[Convert.ToInt16(currentside    *    3    -    3)]    =    new
Point((newImg.Width / 2) + xpos1, (newImg.Height / 2) + ypos1);

subangle = Math.Atan((double)crossspread / (double)radius1);
                double currentangle2 = sideangle + subangle;

xpos2 = Convert.ToInt16(subdistance * Math.Cos(currentangle2));
ypos2 = Convert.ToInt16(subdistance * Math.Sin(currentangle2));
jeffpoints[Convert.ToInt16(currentside    *    3    -    2)]    =    new
Point((newImg.Width / 2) + xpos2, (newImg.Height / 2) + ypos2);

double currentangle3 = sideangle + (45 / (180 / Math.PI));

xpos3    =    Convert.ToInt16(Math.Sqrt((crossspread    *    crossspread)    +
(crossspread * crossspread)) * Math.Cos(currentangle3));
ypos3    =    Convert.ToInt16(Math.Sqrt((crossspread    *    crossspread)    +
(crossspread * crossspread)) * Math.Sin(currentangle3));

jeffpoints[Convert.ToInt16(currentside    *    3    -    1)]    =    new
Point((newImg.Width / 2) + xpos3, (newImg.Height / 2) + ypos3);
```

```
        }
        jeff.AddPolygon(jeffpoints);
        Region bob = new Region(jeff);
        g.FillRegion(Brushes.White, bob);

        return newImage;


}
```

C # Application for Frame Grabbing, with incorporated MATLAB image processing

It would be impractical to describe the entire operation of this open source application, however it utilises Directshow under Windows to stream video and grab frames from USB digital cameras. Camera settings can be controlled, and obtained images can be processed and saved. The later processing is carried out using Matlab functions compiled into a DLL and included into the application. This permitted easier development of the image processing routines into the complete program. The code for this application is included in the attached disk under *SampleGrabberNet.*

The original application was a C# video frame grabbing utility found on the open source community codeproject.com. This was modified to support buffering of the image for processing, and running Matlab image processing functions with a range of control parameters adjustable in a settings dialog box. Ultimately this was intended to be the basis for a rapid prototyping of image processing, feature extraction and feature analysis methods for digital images. This achieves this capacity, and future expansions on the program can be made to substitute the Matlab code with other languages such as C++, or implement the matlab functions in simulink for embedded devices.
The code for the c# application is structured as follows.

The application consists of two c# Visual Studio projects:

- DShowNet:
  Contains connectivity for DirectX DirectShow video driver acquisition.

- SampleGrabberNet:
  Contains implantation of video driver and functions through a GUI.

All work in this project focussed on SampleGrabberNet.

A static class was created to embody all image processing and perform it on Bitmap files passed between the video driver via the GUI to the class. This static class, *ImageProcessing.cs* contains several implementations of referenced and included matlab .DLL files compiled using the Matlab C Compiler (mcc).

ImageProcessing.cs creates instances of the following classes:

```
public static ImageThreshold.dotnetclass threshold_process
       = new ImageThreshold.dotnetclass();

public static return_image.dotnetclass segmentation_process
       = new return_image.dotnetclass();

public static Grain_Features_List.dotnetclass analysis_process
       = new Grain_Features_List.dotnetclass();
```

These are the three distinct Matlab image processing DLLs created with the MCC. These references are included in the declaration of the Visual Studio Project, after including imports the DLLs.

The use of these DLLs also requires the inclusion of the Matlab Array structures found in the DLL MWArray.DLL included in source.

The target machine must also have the Matlab compile runtime installed on the target machine in order to execute Matlab scripts.

The implementation of the Matlab functions is given below.

Return_Threshold_Image_Of returns the processed image based on the source image.
Analyse_Images performs analysis and can save results to a specified file. This requires colour image data and processed binary image to function.

Return_Threshold_Image_Of

```
public static Bitmap Return_Threshold_Image_Of(Bitmap captured_image)
        {
            Bitmap  threshold_image  =  new  Bitmap(captured_image.Width,
captured_image.Height);

            try
            {
                System.Array imagedata = new int[3, captured_image.Width,
captured_image.Height];

                for (int y = 0; y < captured_image.Height; y++)
                {
                    for (int x = 0; x < captured_image.Width; x++)
                    {
                      // Turn the bitmap into an array object.
                      Color currentcolour =
                      captured_image.GetPixel(x, y);
                      //color is ARGB - alpha, red, green, blue
                      imagedata.SetValue((int)currentcolour.R, 0, x, y);
                      imagedata.SetValue((int)currentcolour.G, 1, x, y);
                      imagedata.SetValue((int)currentcolour.B, 2, x, y);
                    }
                }
          System.Array morphologicals = new bool[4]
{enable_spurring, enable_majority, remove_edgeobjects, preview_images};
          System.Array minimumobjectsize = new int[1] { 500 };
          System.Array Tvalue = new int[1] { threshold_value };
          System.Array ThreshType = new int[1]
{ Convert.ToInt16(threshold_type) };

          // Convert all objects into MWArray types and run function.
          MWArray ThresholdedImage = threshold_process.ImageThreshold(
          new MWNumericArray(imagedata),
          new MWNumericArray(ThreshType),
          new MWLogicalArray(morphologicals),
          new MWNumericArray(minimumobjectsize),
          new MWNumericArray(Tvalue));

          // Begin to convert returned image back to bitmap.
          MWNumericArray ThldImg = (MWNumericArray)ThresholdedImage;
          System.Array arraydata =
          new int[captured_image.Width, captured_image.Height];
          arraydata = ThldImg.ToVector(MWArrayComponent.Real);
```

```
                int currentmember = 0;
                int c_x = 0;
                int c_y = 0;
                while (currentmember < arraydata.Length)
                {
                    int firstvalue =
                        Convert.ToInt16(arraydata.GetValue(0));
                    if
(Convert.ToInt16(arraydata.GetValue(currentmember)) > 0)
                    {
                       threshold_image.SetPixel(c_x, c_y, Color.White);
                       imagedata.SetValue(255, 0, c_x, c_y);
                       imagedata.SetValue(255, 1, c_x, c_y);
                       imagedata.SetValue(255, 2, c_x, c_y);
                    }
                    else
                    {
                       threshold_image.SetPixel(c_x, c_y, Color.Black);
                       imagedata.SetValue(0, 0, c_x, c_y);
                       imagedata.SetValue(0, 1, c_x, c_y);
                       imagedata.SetValue(0, 2, c_x, c_y);
                    }

                    c_x++;

                    if (c_x == captured_image.Width)
                    {
                        c_x = 0;
                        c_y++;
                    }
                    currentmember++;
                }



                // Segment the shape if specified
                if (enable_segmentation)
                {
                  System.Array mincompactness = new double[1] { 0.35 };
                  System.Array minarea = new int[1] { 50 };

                  //create data arrays for all values
                    MWNumericArray mw_imagedata =
                    new MWNumericArray(imagedata);
                    MWNumericArray mw_mincompactness =
                    new MWNumericArray(mincompactness);
                    MWNumericArray mw_minarea =
                    new MWNumericArray(minarea);

                    // Run segmentation Function - first set parameters
                    MWCellArray Passed_Parameters = new MWCellArray(6);
                    Passed_Parameters[1] = mw_imagedata;
                    Passed_Parameters[2] =
                    (MWNumericArray)minimumcompactness;
                    Passed_Parameters[3] = (MWNumericArray)minimumarea;
                    Passed_Parameters[4] = (MWNumericArray)repeatlimit;
                    Passed_Parameters[5] =
                    (MWNumericArray)angledistance;
                    Passed_Parameters[6] =
                    (MWNumericArray)highrange_fkz;
                // Run segmentation Function - with Cell
                MWArray ReturnedImage =
              segmentation_process.return_image(Passed_Parameters);

                // Begin to convert returned array back to image.
                MWNumericArray sourceimage =
                (MWNumericArray)ReturnedImage;

                arraydata =
                sourceimage.ToVector(MWArrayComponent.Real);

                currentmember = 0;
                c_x = 0;
                c_y = 0;

                while (currentmember < arraydata.Length)
                {
                 if ((byte)arraydata.GetValue(currentmember) > 0)
                 {  threshold_image.SetPixel(c_x,  c_y,  Color.White);
                 }else
                 { threshold_image.SetPixel(c_x, c_y, Color.Black);
                 }

                    c_x++;

                    if (c_x == captured_image.Width)
                    {
                        c_x = 0;
                        c_y++;
                    }
                    currentmember++;
                 }
            }
        }
        catch (StackOverflowException err)
        { // Error handling here.
        }
        return threshold_image;
    }
```

Analyse_Images

```
public static void Analyse_Images(
                        Bitmap captureimage, Bitmap processed_image)
        {
            try
            {

                System.Array image1data =
                  new int[3, captureimage.Width, captureimage.Height];
                System.Array image2data =
                  new int[1, captureimage.Width, captureimage.Height];

                for (int y = 0; y < captureimage.Height; y++)
                {
                   for (int x = 0; x < captureimage.Width; x++)
                   {
                    Color currentcolour = captureimage.GetPixel(x, y);
                    //color is ARGB - alpha, red, green, blue
                    image1data.SetValue((int)currentcolour.R, 0, x, y);
                    image1data.SetValue((int)currentcolour.G, 0, x, y);
                    image1data.SetValue((int)currentcolour.B, 0, x, y);

                    currentcolour = processed_image.GetPixel(x, y);
                    //color is ARGB - alpha, red, green, blue
                    image2data.SetValue(currentcolour.R, 0, x, y);
                   }
                }

                System.Array selectorgetall =
                  new int[1] { selectorgetall_flag };

                  // get the shape features and save them to a temp
                  // file in the specified default path.
                  // warning - this may need exception handling for
                  // handling file access.

                analysis_process.Grain_Features_List(
new MWNumericArray(image1data), new MWNumericArray(image2data),
new MWNumericArray(selectorgetall), new MWCharArray(data_file_name));
            }
            catch (Exception err)
            {
            }
        }
```

Segmentation algorithm

The Matlab Segmentation Function also calls several subfunctions. These are also included on the research materials disc/folder and include the following methods found in the /RiceSegment folder. The compiled algorithm can be found in RiceSegment/dotnet for use in Visual Studio.

Segmentation decision function

```
complete_segment_poly_parameters(
                      sourceimage,minimumcompactness,minimumarea,
                      repeatlimit,angledistance,highrange_fkz)
```

complete_segment_poly_parameters functions as the decision function described in chapter N. From the input image *sourceimage* individual grains are analysed to determine if their initial basic features (compactness, area, fkz) obey the initial rules (specified by the input parameters *minimumcompactness*, *minimumarea*, *repeatlimit*, *angledistance*, *highrange_fkz*). If this is the case, the segmentation function *polyapprox_segment* is called, using the properties of the grain and image to apply segmentation.

Primary segmentation function

```
[OriginalImage angles anglelist] = polyapprox_segment(
                           OriginalImage,MiniCurve,Boundary,
                           minimumdistance,MajorAxisLength,Image,
                           FillRegion,BoundingBox, previouslevel,
                           MiniCurve_Internal,innerBoundary,
                           MiniCurve_Counter,angledistance)
```

polyapprox_segment is used to segment the curve. Using the boundary coordinates of the shape, the boundary curvature is derived with the method *boundarycurvature_fixed*. Upon obtaining the plot of boundary curvature the process *polygonise_curve* filters this curve leaving only significantly concave points remaining. This can be correlated with coordinates on the shape boundary (a list of equal length that starts at the same point).
Different criteria are employed for the external boundary of the shape and the interior boundary of any large holes above a certain size threshold. The next step in this process is to create the initial list of split points using *calculate_curveangles*. Then the list of split paths is created using *return_angle_pairs_for*. The split paths are sorted in order, and invalid paths are removed using then the segmentation line is determined using *retrievelinesforimage*. This is then drawn on the original image, a morphological spurring operation is employed to clean away any loose spur pixels, and the algorithm reverts to the decision function again.

Deriving Boundary Curvature

```
[curvature]=boundarycurvature_fixed(boundary,m)
```

This returns a 1 dimensional curvature plot of the shape boundary based on the boundary coordinates and scaling factor m, where m is the distance between points in the list of points.

Locating Candidate Split Points

```
filteredcurves = polygonise_curve(
            MiniCurve,minimumdistance,Boundary,BoundingBox,md)
```

polygonise_curve is used to return a filtered curve plot and requires the following parameters:

MiniCurve is the curvature plot of the shape boundary.
Minimumdistance specifies minimum distance between peaks. If a peak is found, any other peaks within that range are removed. Value of 4 used consistently. Found to be reliable in practice.
Boundary and *BoundingBox* are depreciated and were used during development.
Md is a modifier that increases the scale of the peaks. This was initially set to a value of 1.5, although this function increases the value in cases where the shape boundary is significantly low. It was found that the significant peaks of low length boundaries are more difficult to detect and a greater tolerance is needed. This function accounts for this with a modifier for boundaries with a length <100, and boundaries with a low mean and standard deviation of curvature.

This function returns the plot *filteredcurves* which is equal in length to *MiniCurve*, which is the filtered plot of significant peaks.

Returning Candidate Split Points

calculate_curveangles returns the complete list of individual split points. The input variables are the original boundary coordinates, the filtered list and several control variables.

```
angles = calculate_curveangles( newboundary,newcurve,newfilter,
                                Boundary, outerend,angledistance)
```

newboundary represents the list of coordinates of candidate split points identified in the original boundary's curvature peaks.
newcurve is depreciated and was used in debugging. Remove from future versions.
newfilter represents the filtered curvature plots. All entries except valid concave points are zero. Points that are not equal to zero can be correlated with entries in the list *boundary* is

used to determine the location and angle of the midpoint which is used to determine the orientation of the curvature. The *angledistance* variable is used here to ensure the same angle used to derive the plot of curvature is used to derive the orientation of that curve. These properties are then used to create a list of all possible angles in the shape called *angles* and is composed of the following:

```
[Boundary(i,1) Boundary(i,2) angleN distN newfilter(1) newX newY dist0]
```

This gives angles as the x,y boundary coordinates.
angleN is the angle of the mid point to the boundary point.
distN the distance between the mid point and the boundary point.
Newfilter(1) represents the curvature of the point.
newX and *newY* represent coordinates of the mid point to the boundary point. (debugging)

Returning candidate split paths

```
anglelist = return_angle_pairs_for(angles,maximum_distance)
```

return_angle_pairs_for returns the list of properties for each possible pair of split points (to become the candidate split paths) on the shape boundary. These properties are:

```
[x1 y1 x2 y2 minangle distN minimumsail curvature_1 curvature_2 1 0]
```

X and *Y*s denote the coordinates of the two split points.
minangle denotes the smallest angle between the two points.
distN denotes the distance between the two points.
minimumsail previous denoted the minimum saliency (this is depreciated)
curvature_1 and *curvature_2* denote the curvature of the two points.
A true/false value is also returned dependent on whenever the difference in orientation between both angles is <45 (the first 1 variable). The last variable is depreciated.

However before the candidate split paths are added to the list, they must match the following criteria.

If there are only 2 candidate split points the following criterion are used:
Minimum difference in orientation <90

If there are more then the following are used:
Minimum difference in orientation <45

This is performed through the sub method *angle_pairs*.

```
function anglelist = return_angle_pairs_for(angles,maximum_distance)
```

```
% If there are more than 2 points, use criteria for multiple split
points.
% else use criteria for 2 points (more flexible)

if isempty(angles)
    anglelist=[];
    else
        if size(angles,1 )> 2
            anglelist = angle_pairs(angles,maximum_distance,45,45);
        else % USE CRITERIA FOR PAIR POINTS
            anglelist = angle_pairs(angles,maximum_distance,45,90);
    end
end
```

In both cases in *angle_pairs* orientation is determined as the variable *minangle*:

```
minangle = min([abs(angleN1-currAngle) abs(angleN2-currAngle_iii)]);
```

where

angleN1 and *angleN2* are the orientation of the points.
currAngle1 and *currAngle2* are the angle from the second point to the first. These are derived using sub method *determineangles*.

```
function                             anglelist                            =
angle_pairs(angles,maximum_distance,lowest_angle,maximum_angle)

 anglelist=[]; ii=1;

while (ii <=size(angles,1))

   currX = angles(ii,2); currY = angles(ii,1);
   currAngle = angles(ii,3); curvature_1 = angles(ii,5);
   iii=1;

   while curvature_1>0 && (iii <= size(angles,1))

       curr_iii_X = angles(iii,2); curr_iii_Y = angles(iii,1);
       currAngle2 = angles(iii,3); curvature_2 = angles(iii,5);

       if (currX == curr_iii_X && currY == curr_iii_Y)||curvature_2<=0
           iii=iii+1;
       else

        [angleN1 angleN2] = ...
          determineangles(currY,curr_iii_Y,currX,curr_iii_X);
        minimumangle = false;
        minangle = min([
            abs(angleN1-currAngle) abs(angleN2-currAngle2)]);

            if minangle<maximum_angle
               minimumangle=true;
            end

        minimumangle_both = true;
        minangle_both =
            (abs(angleN1-currAngle)+abs(angleN2-currAngle2))/2;
            if minangle_both<lowest_angle
                minimumangle_both=false;
            end

            distN = sqrt ...
                (((currX-curr_iii_X)*(currX-curr_iii_X)) + ...
                ((currY-curr_iii_Y)*(currY-curr_iii_Y)));

            maximumdistance = false;
            if distN < uint16(maximum_distance) %MajorAxisLength
             maximumdistance=true;
            end

            if minimumangle && maximumdistance &&
                    curvature_1>0 && curvature_2>0
                if minimumangle_both==true
                 anglelist = ...
                     [anglelist; currX currY curr_iii_X curr_iii_Y ...
                     minangle distN minimumsail curvature_1 ...
                     curvature_2 1 0];
                else
                 anglelist = ...
                     [anglelist;currX currY curr_iii_X curr_iii_Y ...
                     minangle distN minimumsail curvature_1 ...
                     curvature_2 0 0];
                end
            end
        iii = iii+1;
        end
    end
ii=ii+1;
end

if ~isempty(anglelist)
anglelist(:,11) = [anglelist(:,6) - mean(anglelist(:,6))];
end
```

determineangles is a function that the angles between two coordinates as *angleN1* and *angleN2*.

```
function                    [angleN1                 angleN2]               =
determineangles(currY,curr_iii_Y,currX,curr_iii_X);

            angleN1 = atan2(curr_iii_Y-currY,curr_iii_X-currX);
            angleN1 = angleN1 * 180 / pi;

            angleN2 = atan2(currY-curr_iii_Y,currX-curr_iii_X);
            angleN2 = angleN2 * 180 / pi;

            greaterthan360=false; lessthan360=false;

            if angleN<0
                angleN = 360+angleN;
                lessthan360=true;
            end
            if angleN>360
                angleN = angleN-360;
                 greaterthan360=true;
            end
            if angleN1<0
                angleN1 = 360+angleN1;
            end
            if angleN1>360
                angleN1 = angleN1-360;
            end
            if angleN2<0
                angleN2 = 360+angleN2;
            end
            if angleN2>360
                angleN2 = angleN2-360;
            end
```

Retrievelinesforimage uses the method *linecoords* to create a list of the pixel coordinates for split lines.

```
% uses linecords to return a list of line coordinates based on the
% anglelist.

function [retrievedlines anglelist] = retrievelinesforimage(anglelist)

    anglelist=update_anglelist(anglelist);
    retrievedlines = [];

    for i=1:size(anglelist,1)
        retrievedlines = [retrievedlines; linecords(
                                anglelist(i,1), anglelist(i,2),
                                anglelist(i,3), anglelist(i,4)
                           ) ];
    end
```

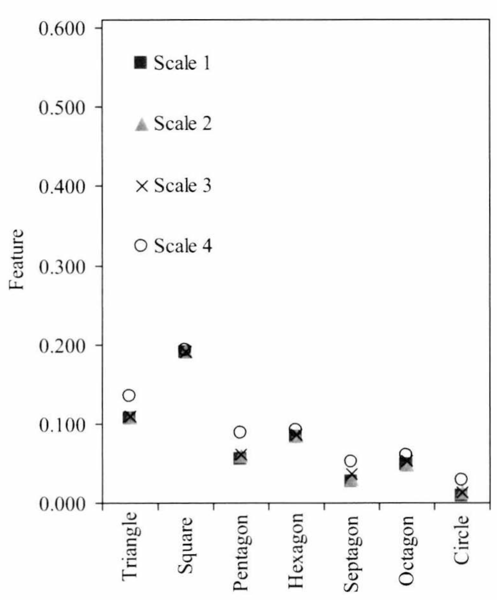
Linecoords is a simple line drawing algorithm that returns a list of pixels that are members of the line.

```
% returns the coordinates of the lines between points set x1/y1 and
x2/y2.
% returns a queen's distance line as 2D array (x,y) linecoordinates.

function linecoordinates=linecords(x1,y1,x2,y2)

linecoordinates = []; dx = x2-x1; dy = y2-y1; steps = 0;

% max x or y?
if (abs(dx)>abs(dy))
    steps=abs(dx);
else
    steps=abs(dy);
end

if steps~=0
xIncrement = (dx)/steps; yIncrement = (dy)/steps;
else
xIncrement=0; yIncrement=0;
end

previousx = 0;previousy = 0; x = x1; y = y1;
linecoordinates = [linecoordinates; uint16(x) uint16(y)];

for i=1:steps

    xvalue=0; yvalue=0;

    if xIncrement~=0
        x = (x+xIncrement);
        linecoordinates = [linecoordinates; uint16(x) uint16(y)];
    end

    if yIncrement~=0
        y = (y+yIncrement);
        linecoordinates = [linecoordinates; uint16(x) uint16(y)];
    end

    if (sqrt((abs(x-x2))^2)+((abs(y-y2))^2)==1)
        i=steps;
    end

end
```

Appendix C

Shape Features

In this section further tables and figures for shape features evaluated in the course of research are given. These features were beyond the primary scope of this thesis, although may be applicable in further research.



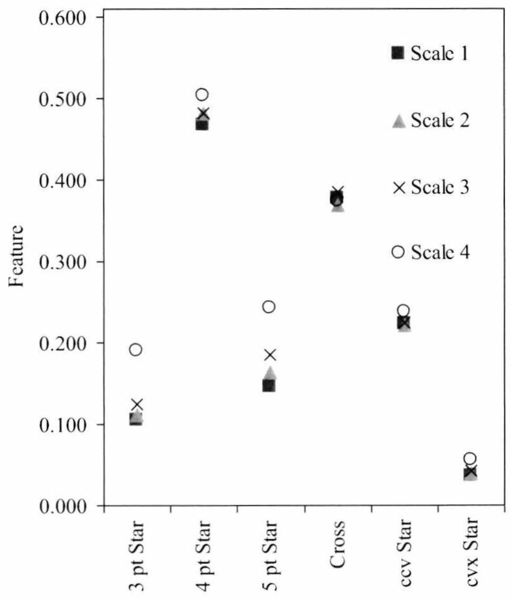


Figure C.1. Plot of Diameter based Shape feature measurements.

Table C.2. Convexity Ratio values.

	Set 1	Set 2	Set 3	Set 4
	Convex Shapes			
Triangle	0.973	0.963	0.954	0.924
Square	0.980	0.973	0.967	0.948
Pentagon	0.983	0.977	0.972	0.956
Hexagon	0.984	0.979	0.975	0.962
Septagon	0.985	0.980	0.976	0.967
Octagon	0.986	0.981	0.977	0.967
Circle	0.990	0.986	0.985	0.987
	Concave Shapes			
3 pt Star	0.761	0.731	0.731	0.704
4 pt Star	0.550	0.533	0.538	0.537
5 pt Star	0.485	0.474	0.480	0.493
Cross	0.701	0.715	0.701	0.727
ccv Star	0.313	0.319	0.317	0.314
cvx Star	0.621	0.622	0.624	0.630

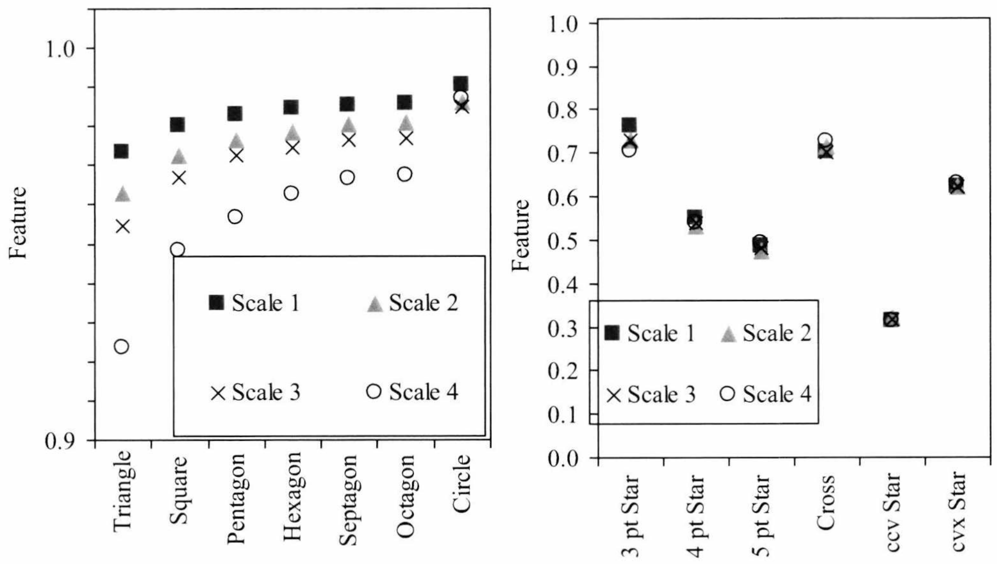
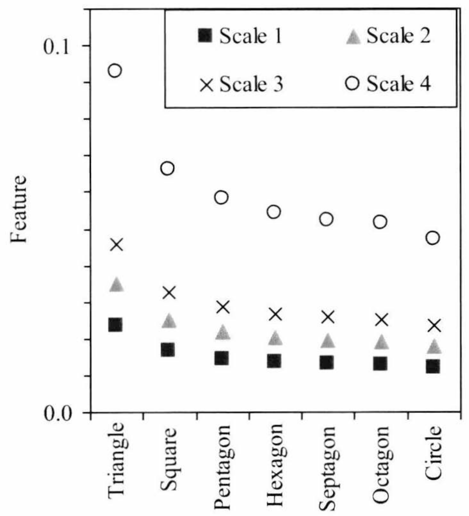


Figure C.3. Plot of Convexity Ratio values.

Convexity ratio supports its namesake, and offers a wider range for convex shapes and a consistent increase to a value of 1 for more perfectly concave shapes. Scaling impact is notably not significant for convex shapes.

Table C.4. Gravelius' Compactness values.

	Set 1	Set 2	Set 3	Set 4
Convex Shapes				
Triangle	0.023	0.035	0.046	0.093
Square	0.017	0.025	0.033	0.066
Pentagon	0.015	0.022	0.029	0.058
Hexagon	0.014	0.020	0.027	0.054
Septagon	0.013	0.020	0.026	0.052
Octagon	0.013	0.019	0.025	0.051
Circle	0.012	0.018	0.023	0.047
Concave Shapes				
3 pt Star	0.031	0.049	0.064	0.135
4 pt Star	0.034	0.053	0.069	0.148
5 pt Star	0.038	0.059	0.077	0.163
Cross	0.022	0.032	0.043	0.083
ccv Star	0.016	0.021	0.032	0.066
cvx Star	0.007	0.009	0.013	0.026



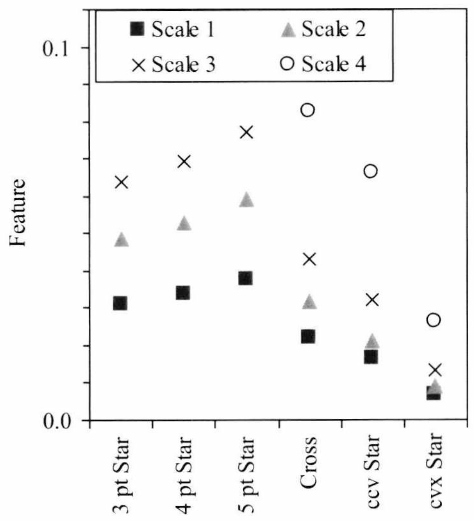
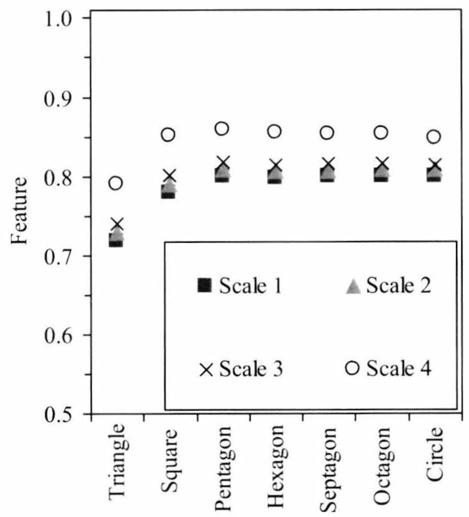


Figure C.5. Plot Of Gravelius' Compactness values

Gravelius Compactness trends tend to vary significantly with scale as result of digitisation. This error appears significant compared to the proportional variation between the values of shape descriptors, and feature values vary little between concave and convex shapes proportionally.

Table C.6. Mean Form Factor values.

	Set 1	Set 2	Set 3	Set 4
Convex Shapes				
Triangle	0.718	0.729	0.739	0.790
Square	0.780	0.791	0.802	0.851
Pentagon	0.799	0.808	0.817	0.858
Hexagon	0.798	0.806	0.815	0.855
Septagon	0.799	0.808	0.815	0.852
Octagon	0.800	0.808	0.816	0.852
Circle	0.800	0.808	0.815	0.847
Concave Shapes				
3 pt Star	0.648	0.648	0.659	0.708
4 pt Star	0.645	0.650	0.665	0.725
5 pt Star	0.661	0.668	0.685	0.759
Cross	0.805	0.820	0.828	0.879
ccv Star	0.515	0.522	0.524	0.531
cvx Star	0.704	0.707	0.711	0.724



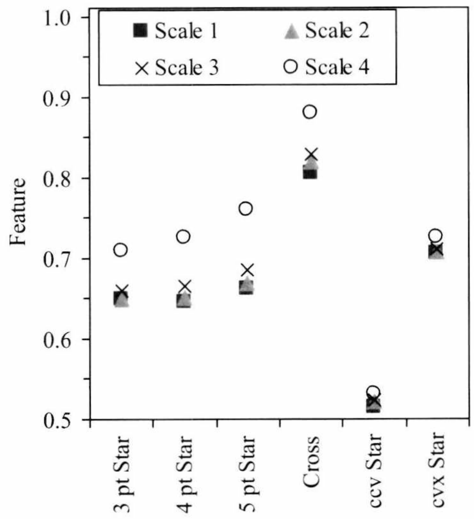
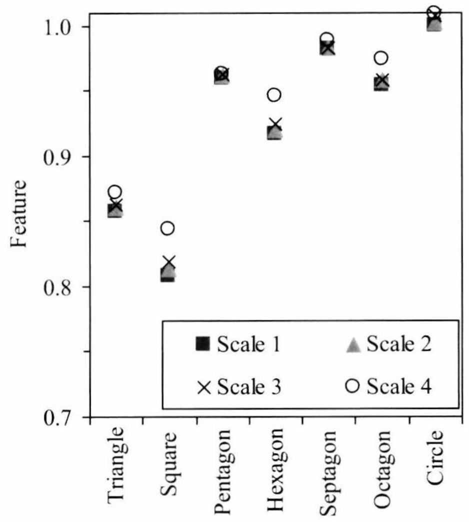


Figure C.7. Plot Of Form Factor.

Form Factor functions return highly consistent concave shape values and higher and lower feature values for shapes of significantly different concavity. However scale variance is proportionally high.

Table C.8. Mean Elongation values.

	Set 1	Set 2	Set 3	Set 4
Convex Shapes				
Triangle	0.857	0.861	0.863	0.872
Square	0.808	0.814	0.819	0.844
Pentagon	0.959	0.961	0.963	0.963
Hexagon	0.917	0.920	0.924	0.945
Septagon	0.982	0.983	0.983	0.988
Octagon	0.954	0.958	0.959	0.974
Circle	1.000	1.002	1.007	1.009
Concave Shapes				
3 pt Star	0.810	0.806	0.804	0.799
4 pt Star	0.603	0.599	0.607	0.627
5 pt Star	0.799	0.794	0.788	0.793
Cross	0.791	0.803	0.795	0.818
ccv Star	0.660	0.665	0.666	0.666
cvx Star	0.907	0.906	0.906	0.904



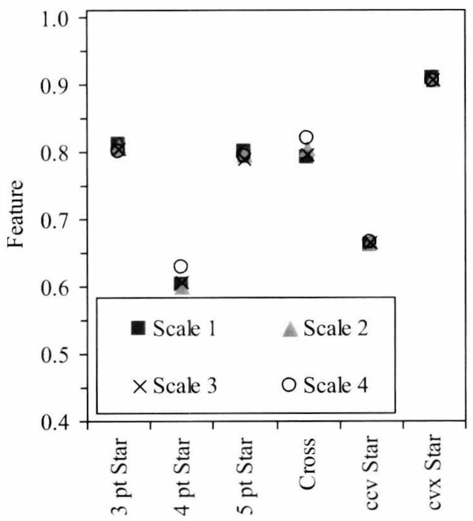


Figure C.9. Plot Of Mean Elongation

Trends for Mean elongation vary most notably with the number of sides of the shape, and with the level of concavity of the specific shape. Scale variance is relatively small.