An Automated Pollen Recognition System

A Thesis submitted to
Massey University, Turitea, Palmerston North, New Zealand
in fulfilment of the requirements for the degree of
Master of Engineering.

By
Gary Allen

December 2006

Institute of Information Sciences and Technology

Massey University

Abstract

A system was developed with the aim of demonstrating that the tedious tasks of classifying and counting pollen on slides could be performed automatically to a standard comparable with that of human experts. Automation of pollen classification and counting will advance the science and range of applications of palynology.

The system developed is a completely functioning prototype. After initial set up and training it is automatic in operation.

System tests have demonstrated that the concept is viable and that the prototype developed is at a stage that it is of practical use to palynologists. There are opportunities for improvements and added functionality. Now that the system is developed and characterised, it provides a benchmark for gauging the efficacy of future improvements and adaptations.

The system is presently adaptable to many different classification problems within palynology and would be adaptable for other automated microscopic classification or imaging tasks.

Acknowledgements

Thesis supervisors were Prof. Bob Hodgson and Rory Flemmer and their efforts are well appreciated.

Special thanks go to Prof. Bob Hodgson, who offered the project to me, and spent many hours in helpful discussions and reading reports and drafts.

The Pollen Research Team at Massey University includes: Prof. Bob Hodgson, Prof. John Flenley, Prof. David Fountain, Dr. Stephen Marsland, Greg Arnold. They are an integral part of this project and provided sound and experienced advice and help along the way.

Thank you to Dr. Roger Brown who checked my work on depth of field.

Many thanks are owed to Steve Denby and the crew at the special works and mechanical workshop of the Institute of Fundamental Sciences, Massey University, who built the AutoStage.

Manual pollen counters were: Xun Li (PhD), Kevin Butler (laboratory technician), Alistair Clements (student), Prof. John Flenley and Prof. David Fountain. Many thanks for the time taken to count these slides and again to David, for supplying the pollen for the slides.

Thanks to Xiuying Zou for conventional image capture of reference pollen.

And to my wife, Linda, who encouraged and supported my decision to study again, my love.

Contents

List of Tables and Figures

1 Introduction

This project is the aggregation, integration and augmentation of previous research outcomes in the design and development of a system now called “AutoStage”. With the development here of a complete working system, and an evaluation of it that shows it to be viable, a longstanding requirement for the study of pollen (palynology) is ready to be integrated into working laboratories.

AutoStage is an aid to palynology and all the important uses to which pollen identification can be put. The broad requirements of AutoStage are to locate pollen grains on a prepared microscope slide, extract feature data from images of the pollen grains, classify the pollen by genus and produce a count of each type. The system developed aims to be an intelligent assistant for use in palynology.

The core system operations are (described in subsequent paragraphs):

- the capture of images from a slide
- the segmentation of pollen in the images
- the extraction of features from pollen images
- the classification and count of pollen types

The capture of an image from a slide involves two focusable microscopes, lighting, and a mechanism to hold and move the slide with precision. A low magnification microscope allows quick coverage of the entire slide to determine pollen locations while a high magnification microscope captures images suitable for feature data extraction and identification of the pollen grains. The light source is a filtered and cooled halogen lamp. The slide holder is situated between the light source and the microscope objectives. The slide holder is movable in two dimensions with a resolution less than the smallest pollen of interest. Pollen grains are 10-100 microns diameter. Image capture system parts are detailed in §3, §4 & §5.

The segmentation of pollen in the images requires an algorithm to recognise shapes in a low magnification image that are likely to be pollen and ignore those that are not. A series of images is taken to cover the slide. The location on the slide of each likely pollen grain is determined from their position in the image. The high magnification microscope is driven to each location and segmentation is again performed to find objects, determine which was originally located, and test further for it being a valid pollen grain. If valid, then an image slightly larger than the bounding rectangle of the object found is saved for feature extraction. Details of segmentation are in §6.

The extraction of features from pollen images occurs after all the pollen grains have been found and their closely cropped images stored. The values of the grey levels of the image pixels are used to estimate geometric, textural and statistical qualities. The features used were determined in a previous study [74] and a description appears in §7.1.

The classification of pollen types is performed by an artificial neural network. The network is trained using features from images of known pollen types. The features from pollen images taken from slides are then able to be classified by the trained network. The numbers of pollen grains of each type found on a slide are reported. This is the main system output. Classification details are found in §7.2.

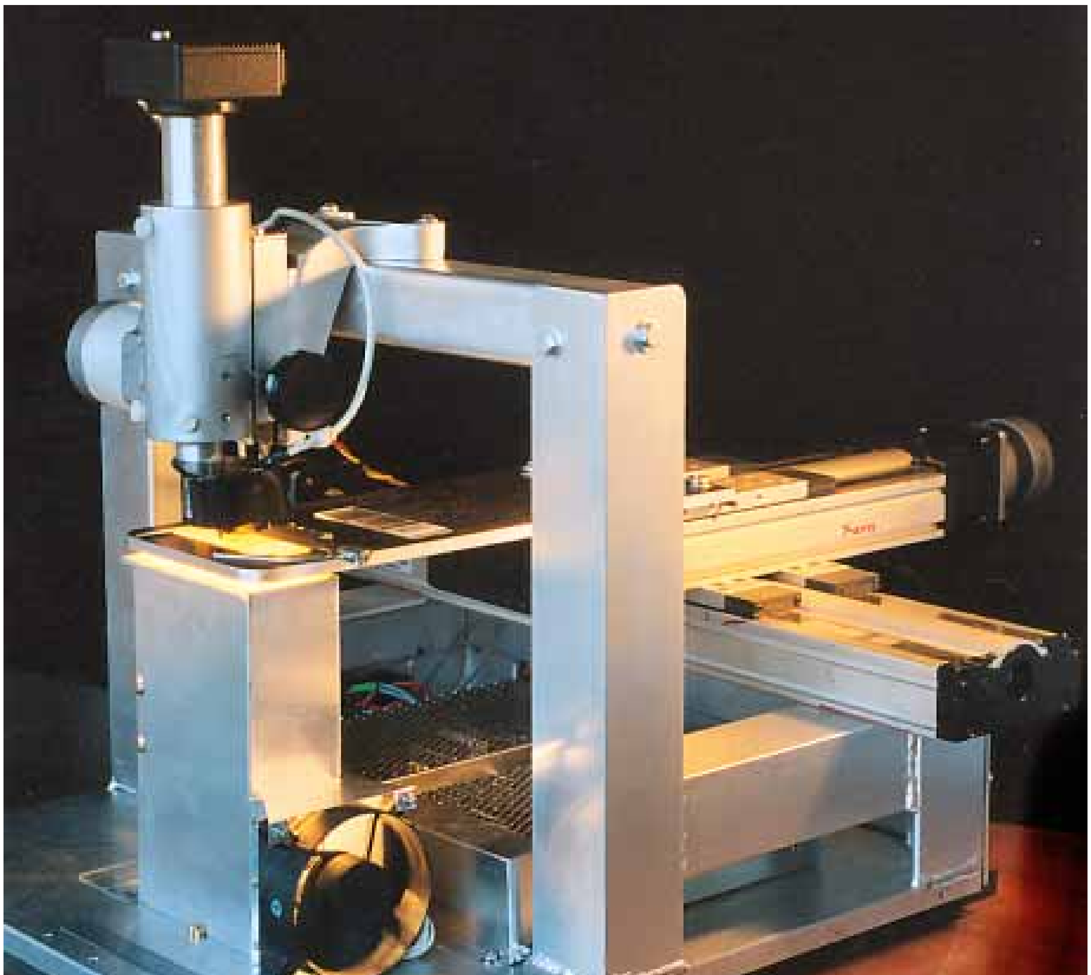


Figure 1-1: The AutoStage Prototype

1.1 Aims and Objectives

The Aim of this project is to build a workable system to:

1. demonstrate that automation of pollen identification and counting from slides is possible, with an accuracy comparable to that achieved by experienced palynologists
2. be suitable as an experimental platform to improve the accuracy and repeatability of pollen classification and counting, and enable a second generation system to be specified

The Objectives of this project are to build and evaluate a system as described above. Previous projects reported by Zhang [74] and Holdaway [34], have resulted in the determination of a set of features to allow pollen classification and the initial design of a suitable digital microscope respectively. Objectives for this project are to:

1. build and test the dual microscope system
2. develop a suitable lighting system
3. develop a two dimensional XY movement to securely hold and accurately position a microscope slide
4. develop focus capability and develop algorithms for auto-focussing
5. develop segmentation algorithms to locate pollen in microscope images and determine their precise location on the slide, and extract an image of the pollen suitable for features extraction.
6. define a classification system using previously selected features and to consider the multi-layer perceptron
7. integrate all the above into a coherent and operational system that will fulfil the aim of the project
8. test and evaluate the system with regard to the aim of this project

1.2 Design Specification: basic requirements

Features were developed to be used for a classification system. A high magnification visible light microscope was designed for capturing digital images suitable for feature extraction. These were considered when specifying other system components.

The specifications for the components of the system are listed below with implementations below each:

1. A slide holder was required to keep a slide located with precision and repeatability. It was required to maintain the slide orthogonal to the optical axis and allow light to pass through the slide.

    a. A standard microscope slide holder suited the requirements and was built into the structure. Orthogonality to the optical axis is achieved to a degree that the slide can be imaged by the low magnification microscope with a single focus setting. Pollen grain positions between the slide and cover slip vary enough that focussing on each grain using the high magnification microscope was required.

2. The precision movement of the slide holder, under computer control, was required to place a pollen grain of 10 microns diameter within the 0.5mm square field-of-view of a microscope. It would also need to recall that pollen grain to a position within the microscope field of view; preferably to the same location within a few microns. As the XY stage holds and moves the slide holder, the XY directions required orthogonality to the optical axis at any position.
    a. A precision commercial XY stage was built into the structure of the AutoStage. The X and Y linear movement sections are specified for repeatability of movement of 3 microns. The movement is driven by stepper motors that add to the repeatability error and 20 microns resulted in the maximum error.
3. A computer controlled focus mechanism capable of stepping at intervals of focus such that at least one and preferably more than one steps contain in-focus portions of a 10micron diameter pollen grain.
    a. A standard microscope focus mechanism was used to convert the rotary motion of a stepper motor to linear motion of the camera along the optical axis. The result just meets the requirement for the smallest pollen grain however the stepper motor driver does not drive the motors evenly and at every tenth step there is a larger step which can cause slight non-optimal focus. Depth of field has a corrective effect; however consistency suffers as a result.
4. A second microscope with a larger field of view for locating pollen grains on the slide with a minimal number of images was added. The constraints were to maximise field of view while remaining capable of detecting the smallest pollen of interest defined as 10 microns diameter. Auto-focus was considered to be a requirement.
    a. An available digital camera was used for the second microscope. An adjustment to the distance along the optical axis, from its lens to its sensor, was made to alter the magnification. A magnification of 1x met the detection and field of view constraints. The camera was attached to the high magnification camera so the focus mechanism could be shared.

5. Lighting suitable for both microscopes needed to be adequate for the integration times of the sensors and associated electronics. All images should be of consistent exposure.
    a. Stable power supplies to power the incandescent lamp affords constancy of illumination. A *Meanwell* S-150 series power supply is used. It is a mains in, 12 volt out, 150 Watt switched mode power supply. With the specified combined line and load regulation of 0.6%, the luminous intensity would change by less than 2%. Locations for four quartz halogen lamps is provided but one 35 Watt, 8000 candela lamp is used.
6. Frame construction required a physical stability and vibration damping to keep the microscope stationary and minimise blurring of the image.
    a. The frame construction and mechanical assembly was undertaken by the mechanical laboratory in the department of fundamental sciences at Massey University. A dense rubber mat was placed under the solid base with four shock absorbing feet added at the corners so the rubber is partially compressed by taking some of the weight with the feet taking the rest. This helps to isolate the unit from vibrations from the bench upon which it sits.
7. Software, suitable for quick development and rapid changes, while capable of performing all or most tasks required for the project, was desirable.
    a. Matlab is essentially a prototyping software package that fits the requirements well. It was used to develop the feature algorithms so no translation was required, and has built in functions for image processing. A Matlab toolbox for communication to the cameras was purchased which allowed the entire design to be written using the one programming language.

1.3 Contribution of the Author to this Project

The 43 features used for classification were selected in a previous project. The high magnification microscope was designed in a previous project. The author's contribution has been the design, development, integration and testing of the system including a performance comparison with expert palynologists, all tasks as outlined in the eight objectives of this report including controlling software as presented in appendix H. The AutoStage prototype was built in a mechanical workshop to a specification determined in this project.

1.4 Published Paper

A paper entitled “Automatic Recognition of Light-Microscope Pollen Images.” was published and presented at the Image and Vision Computing New Zealand 2006 conference [20]. The paper is reproduced in Appendix B.

2 Background

Introduction

This section introduces topics related to the project to add background knowledge, put the subsequent discussions into perspective, and cover literature in the various fields.

2.1 Palynology – the study of pollen

Pollen is technically termed “the multinucleate gametophyte generation of flowering plants” [8]. It carries the male gametes, or sex cells, for fertilization of plants. Wind and insects are the main carriers of pollen with the grains themselves having characteristics that promote, usually one or the other of the two methods of transport. Birds and bats help out the insects in that task. Spores, included in studies under palynology, are “asexual reproductive bodies of lower vascular plants” and “algae, fungi, and mosses”[8]. They are asexual in that they are themselves able to grow into an organism. Wind is an inefficient method of transport so, for wind dispersal, vast numbers of pollen or spores are produced: roughly one million per cone for pine and up to 500 million for a single marijuana plant shoot [8].

Palynology has numerous applications. Fossil pollen analysis (palaeopalynology) is used to identify the plant taxa, from which, can be deduced [8, 27]:

- vegetation variations with time
- climate and its temporal variation
- evidence of human activities including
  - land clearing
  - burning
  - atmospheric pollution (also natural, e.g. volcanic)
  - salinity
  - soil degradation and changes
- archaeological information
  - dating of sediment levels and ages of artefacts found
  - what people may have eaten
  - what was buried with them – flowers etc.
- oil deposit locations

Honey type, and location of origin, can be indicated by the pollen found in the honey (melissopalynology). Inhalant allergy sufferers can be advised of high pollen counts in the air (Aeropalynology) [18]. Forensic investigations can be aided by determining if an object has been in a certain general location by identifying the pollen grains attached [9].

The layers making up a pollen grain wall are labelled in Figure 2-1.

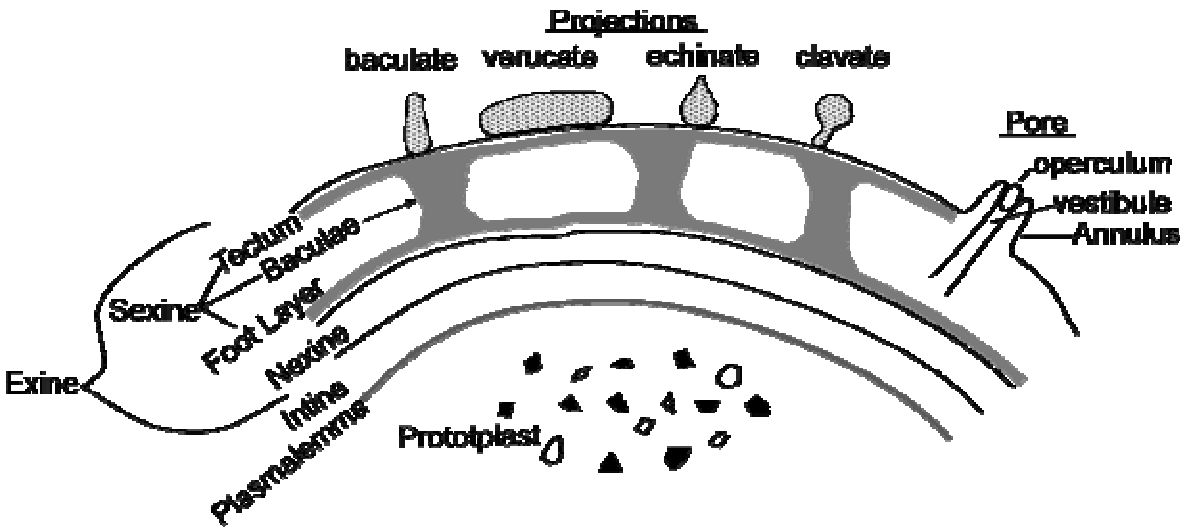


Figure 2-1: A section of pollen wall showing structure and some features.

The taxa, or type, of pollen can be determined by pollen morphology, or structure. Large data bases of images and diagrams are being built so pollen can be identified by referring to them. Pores, (holes), culpi, (furrows), and the numbers of them are clues to pollen taxa.

The exine (Figure 2-1) is made of sporopollenin. Sporopollenin is a substance composed of oxidative copolymers of carotoid and carotenoid esters. It is an extremely durable substance and can be found in anaerobic sediments dating back hundreds of millions of years. A pore is shown in the diagram but these may be elongated to a furrow and are called colpi [70]. The number of pores or culpi, and their arrangement on the sphere surface, is a strong indication of type. Surface features are used for identification using mathematical feature extraction in AutoStage. Some of the surface features can be seen in the scanning electron microscope images of pollen in Figure 2-2.

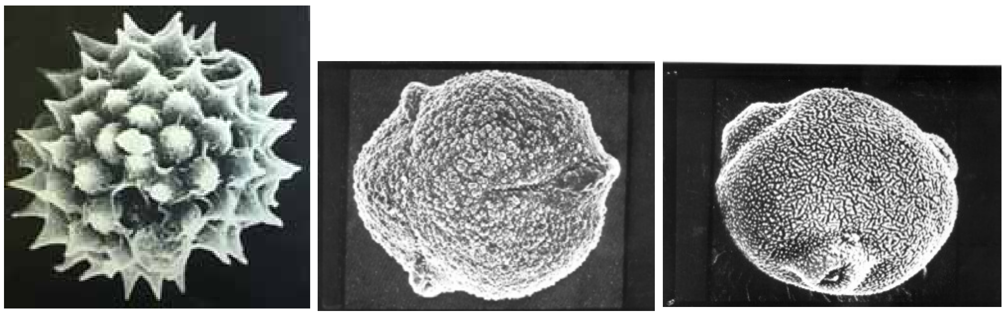


Figure 2-2: Golden Rod (echinate), Oak pollen(colpi) and Birch pollen (pores). Scanning Electron Microscope (SEM) images of pollen from "National Pollen and Aerobiology Research Unit" web site[58].

Modern "pollen rain", or the spatial distribution of pollen, is modelled to determine the vegetation taxa and climatic conditions that would have caused the pollen distribution found in sediment core samples [8]. Sample cores are extracted from lake beds, swamps etc., sliced horizontally to divide into samples of sequential depths, and analysed to make stratigraphic maps of the region. Carbon dating and other techniques are used to compare depth relationships to dates of deposits. The thickness of each core slice taken, determines the temporal resolution. Fine resolution [26], down to one year representing perhaps a millimetre of core depth, is often required but requires the preparation of hundreds of samples and many slides of pollen to prepare and count. A greater number of cores, taken in close proximity, will increase the reliability of the data and add information regarding localised variation. Regionalised layer variation is obtained by multiple core sampling at larger spacings. The counting of pollen from core samples requires experts to identify the pollen taxa as they count pollen grains on a slide. This is very time consuming and laborious work for trained people whose time could well be spent on less mundane tasks. This is one reason that AutoStage is being developed.

The AutoStage will also aid pollen counting by aeropalynologists who advise of pollen counts in the air for benefit of inhalant allergy and asthma sufferers. At present there are a number of pollen counting stations, mainly in Europe [58] and North America [3]. Many have volunteer based counting [2], where volunteers spend 2-3 hours per day, 3 days a week counting pollen captured in pollen traps.

A PhD student who uses Palynology as a prime research tool may spend up to 30 months of microscope work counting pollen. Any study using palynology would be advanced considerably if the counting were performed automatically and the time taken to collect research data reduced consequentially.

Preparation of samples is also time consuming and important to the outcome of the counting process. Fossil samples are treated to remove as much foreign materials as possible by sieving and chemical treatments. Density gradient centrifugation may have a part to play in the classification of pollen but at least it appears that it will be helpful in separating pollen from organic matter not removed by chemicals and sieving [16, 17, 56]. The problem of organic matter remaining is applicable to automatic pollen counting as the organic matter, as seen in the images in Figure 6-2, reduces segmentation effectiveness. Ultrasonic filtration [66], is an advance that may prove useful. Preparation is a separate part of the process but important to the aim of this project in two ways:

- preparation involves laborious, time consuming work
- preparation quality can influence the AutoStage processes and the reliability of the results

A requirements list has been developed to begin integrating preparation into the whole automatic pollen counting process (§3.2.1).

2.2 Microscopy

Microscopy is significant to palynology considering the 10 to 100 micron diameters of the objects under study. Of the many types of microscopy existing, the scanning electron microscope (SEM) would be a better tool than the light microscope, if it were not for the speed and cost of such systems [67]. Resolution is the advantage of SEMs as they are able to capture detail of pollen surfaces that are unable to be resolved or captured using visible wavelengths (see comparison images Figure 4-2 and Figure 4-3). The extra detail in SEMs holds information that can better distinguish between pollen types. A key to the utility of a system such as AutoStage is simplicity of use and affordability. Light microscopy is more suitable in those regards. The effort, therefore, is to optimise the images from a light microscope. The microscope, as designed and built, is adequate for a prototype and for the aim (§1.1) of this project. Any improvements to the microscopes for this project is a subject for future work (§9).

Pollen grains are usually counted and classified by viewing slide mounted samples under a microscope. Collection of pollen for aeropalynology often uses ‘sticky’ rods or tapes which are then placed on a slide. One project under development for automation of airborne pollen counts is “Microbus”, by Omnibus [10]. This project has a tape only system for continuous monitoring and includes a preparation module within a unit that may be located, in its entirety, at some remote site. AutoStage is developed for slide microscopy in a laboratory, however the use of other mediums is not precluded from future adaptations.

Digital microscopes are not new and are commercially available. For example, Keyence produce some functionally sophisticated microscopes, some of which use short wavelength laser to produce up to 1500x images [1] (see Figure 2-3). A digital microscope can readily be made at home [60]. The digital microscope was considered for use in this project by Holdaway [34], who made a careful analysis of the consequences in changing from conventional microscopy to direct digital imaging.

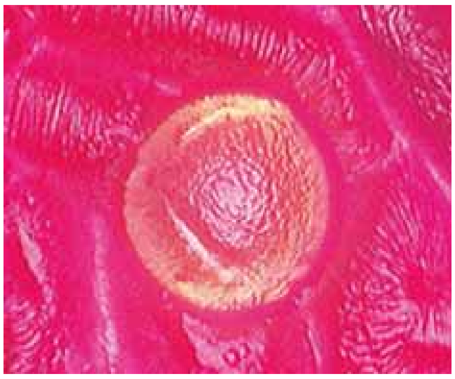


Figure 2-3: pollen grain on a camellia petal at 1500x from a commercial (Keyence) digital microscope.

The conventional microscope used for the capture of images for comparisons with images captured from AutoStage was an Olympus BX51 used at 40x objective magnification and a 10x eye-piece magnification giving 400x optical magnification. Images were captured via an Optronics magnaFIRE SS99802 digital camera with MagnaFIRE frame-grabbing software on a 2GHz Pentium computer.

Figure 2-4: Olympus BX51 as used to capture conventional microscope images for comparison studies

Lighting is an integral part of microscopy and affects the resolution and contrast of the image. The character of the light source produced by optics and the manner of transferring it to the object, affects the characteristics of the object that are represented in the image [23].

Differences of intended light paths, to actual paths through lenses, cause: chromatic aberration, spherical aberration, coma, astigmatism, distortion, chromatic differences in magnification, and curvature of field [23]. The use of a manufactured objective lens for AutoStage, designed to minimise most of these, is expedient. Two aberrations, "curvature of field" and "distortion", could be of issue and are considered in §4.2.1.

2.3 Automated Pollen Recognition

Flenley, was first to identify the need to automate in 1968 [14]. He stated that two problems existed: infinite possible orientations of pollen grains and partial focus of grain under the microscope. Since then, Flenley [33, 40-44, 64, 67, 68, 75] has inspired others to help overcome those and other problems of automation, culminating in this project and the development of AutoStage.

White was also early when he wrote about automation, concentrating on an image analysis system [72]. White's concentration on the imaging problem only, as is true for most authors, was noted by Rodriguez-Damian et al. [59]. This is understandable considering that a workable classification criterion should be developed before continuing with the overall design which may seem straightforward or obvious. However, it has been our experience that all is not obvious until the system as a whole is explored, by theoretical design and prototyping. France et al. [19], who have written a good account of what is required and what problems are inherent in an entire system, were credited with an holistic effort by Rodriguez-Damian et al. [59]. That publication was very recent which indicates continued interest in automated pollen counting.

"Nobody has yet developed a satisfactory automatic method for counting pollen" [58]. This statement is on the web-site of the United Kingdom's National Pollen and Aerobiology Research Unit. There are studies and projects [10, 19, 59] with an aim to automation but the statement seems to still hold true. AutoStage is very close to being of practical use.

Green, a proponent of numerical palynology, stated in 1997 that sporadic attempts over 30 years to produce automated systems, have produced no system of widespread use [27]. Numerical palynology attempts to mathematically model pollen dispersal and its relationship to the vegetation present. Given the complexities involved and interpretations of pollen diagrams often being made by “eyeballing” them, aids to interpretation are possible through the modelling. The collection of sufficient data is a limitation and automation of the counting of pollen would allow improved progress in this area.

Earlier interest in automation concentrated on the image recognition, but was hindered by slow computers with insufficient memory. Even now, the processing required takes some time but is not prohibitive.

A workable set of features has been selected by Zhang [33, 74, 75], and the study of image capture, resulting in a digital microscope proposal, by Holdaway [34], have paved the way to a complete system that will usefully automate palynology.

2.4 Neural Networks

Artificial Neural networks were originally formulated in order to model how the human brain was thought to operate. The first was introduced by McCulloch and Pitts in 1943. “Artificial” is often not used now, as neural networks have become a useful tool in their own right for machine learning and classification tasks. Neural networks are massively parallel distributed processes made up of simple processing units and emulate the brain in two ways [30]: 1) knowledge is acquired by the network from its environment, 2) interneuron connection strengths, known as synaptic weights, are used to acquire knowledge.

Hebb [31], a neuropsychologist, proposed a cellular level change to be the basis of learning, “when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A’s efficiency as one of the cells firing B, is increased.”

Haykin [30], expands and rephrases this as a two part rule, the second part was not contained in Hebb’s original rule:

1. If two neurons on either side of a synapse (connection) are activated simultaneously (synchronously), then the strength of that synapse is selectively increased
2. If two neurons on either side of a synapse (connection) are activated asynchronously, then that synapse is selectively weakened or eliminated.

A synapse as described above is called a “Hebbian synapse”. Weights in the model described next are equivalents to the synapses.

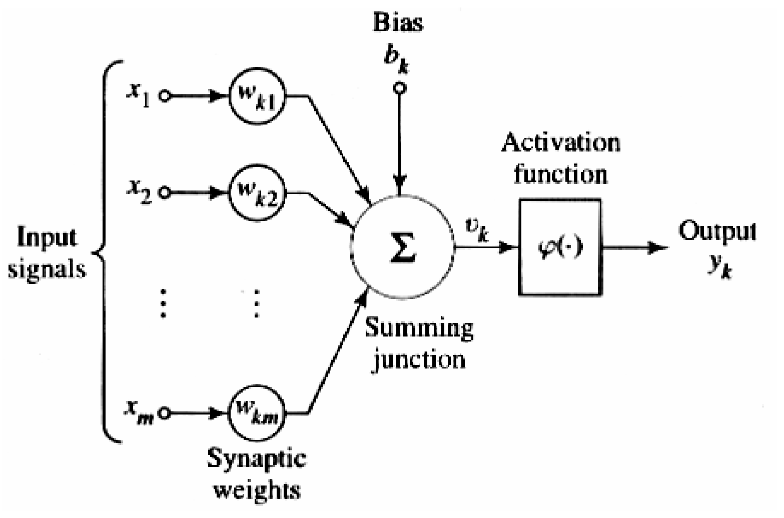


Figure 2-5: A model of a neuron [30]

A single neuron, shown in Figure 2-5, is a summing node with a number of inputs that are weighted, biased by what is essentially an extra input, and presented to an activation function to form the output. The bias eliminates any chance of the inputs summing to zero. This model is named a *McCulloch-Pitts model* [30].

The activation function shown as $\varphi(\cdot)$ in Figure 2-5, may be a simple decision to output 1 of the input, v_k, is positive and -1 if negative. Three simple activation functions are shown in Figure 2-6.

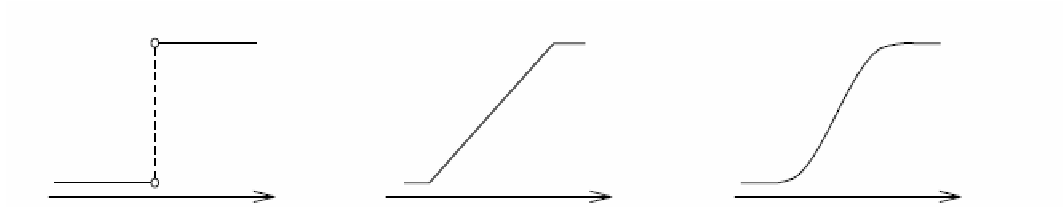


Figure 2-6: Three simple activation functions, $\tilde{\varphi}$:
From left to right: a) threshold, b) piecewise-linear, c) sigmoid.

Taking one neuron and forming a network creates an artificial neural network. Figure 2-7 below, shows a feed-forward fully connected network with one hidden layer. If the input nodes are weighted and connected directly to the output neurons then it is a single layer network. Using the Perceptron model of neurons, the network may have an additional “hidden layer”, as shown in Figure 2-7, to become what is called a Multi-Layer Perceptron, or MLP, which is the type used for AutoStage. It is possible to add more hidden layers.

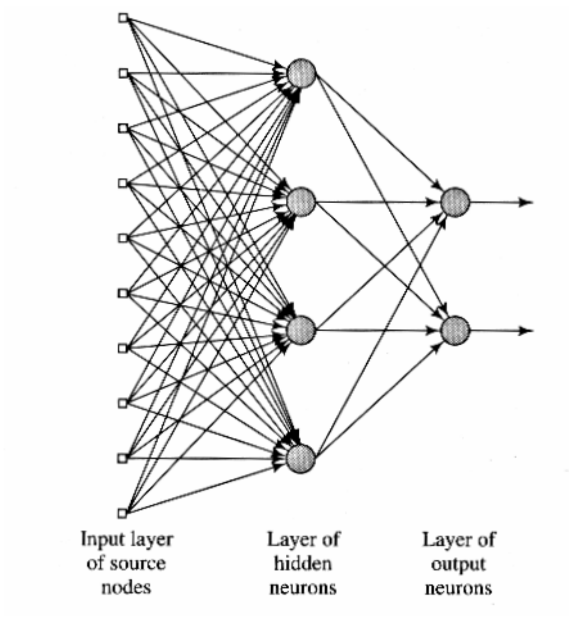


Figure 2-7: Fully connected feed-forward network with one hidden layer [30]

While ‘unsupervised’ neural networks will sort unknown data into groups of similar type, ‘supervised’ networks, as described here, are trained on known data and can then sort new data into the groups it has been trained on. Training, for a supervised network, occurs in the network described above by setting known data at the inputs, setting goals or targets for the neural network outputs as required to indicate the known inputs, and adjusting the weights until the output matches the targets set. Training a neural network is some method of successively altering the weights to arrive at the given target. An output error may be quantified by subtracting the network output from the target, squaring the result, and adjusting the weights with the aim of making this ‘sum of squares error’ as near zero as possible. The path this error value takes over successive weight changes is differentiated to find its slope and direction to determine by what value to adjust the weights to continue this gradient descent. Unfortunately, when a minimum in the error is found, this may be a local minimum and not the global minimum so various techniques are implemented in an attempt to overcome this problem.

The inputs are multiplied by the weights to the hidden summing nodes and then multiplied by more weights and summed for input to the activation function making this a “feed-forward” network. The error is calculated at the output and is fed back to the hidden layer; weights there are adjusted and fed back further to the input for the input weights to be adjusted and this process is called back-propagation (of error).

The amount of data in an image of pollen is generally too much for a contemporary computer running a neural network and data beyond that which is necessary may confound the neural network and actually cause degradation in performance. Consequently, features are extracted to represent the image using minimal data. Feature selection is important to the success of the classification as the data must, in some way, be separable by the network. A simple example is the given by using the size and shape of two pollen types.

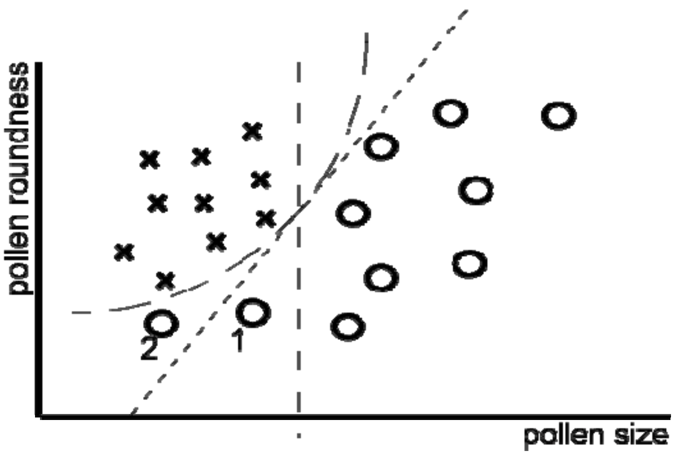


Figure 2-8: Data for size versus roundness of two pollen types graphed

In Figure 2-8, the data for two pollen types has ‘size” plotted on the x-axis and “roundness” on the y-axis. If the two numbered circular points did not exist, the pollen type could be determined simply by size being greater or less than the x-axis value of the vertical dashed line. Now adding in the measured pollen indicated by ‘1’, the angled dashed line may be used and the data can be transformed, perhaps by rotating the axes making the angled dashed line perpendicular to a new axis. If the numbered ‘2’ circular point is added then a non-linear division is required. Then, if the Os were actually a random mixture of two pollen types, another feature would need to be added that distinguished those two types and a three dimensional graph would be displayed. Adding more features adds dimensionality and we can no longer visualise them and must use mathematics for representation. Forty three features are used for representing images of pollen in this project. It is, however, possible to make a series of graphs of two features per graph, so that each feature is graphed against each of the others. If displayed in a matrix, patterns may be observed. Pattern recognition is something that humans do extremely well. Artificial neural networks also employ, "*the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories)*", which is the formal definition of pattern recognition given by Haykin [30].

2.5 System Building

Scientific and engineering methods have been used to develop a prototype of a system called AutoStage. Certain design criteria and constraints (§1.2) were considered and the prototype was built accordingly. Then the system was checked for conformation to the criteria. The criteria were sufficient for success but not over specified so that given a successful outcome, the system was inexpensive, adaptable, and easy to produce with room for enhancements.

Taguchi et al. [65] determined means for design of quality products to include system design, parameter design and tolerance design. System design uses scientific and engineering knowledge to produce a prototype. Parameter design and tolerance design use the prototype, and experience in its production, to help determine parameters to improve the prototype in a manner that is manufacturable. This project corresponds to the system design phase. The parameters defined here can be used to determine to what level the AutoStage can be developed to improve its performance in areas such as repeatability and speed of operation. Tolerance will then determine what level of component 'quality' and level of manufacturing processes will suffice to meet the parameter design. At that stage, the compromise of price, accounting also for ability to manufacture, versus capability, is considered. At this design stage, the prototype was developed to meet critical functional levels to meet the aim of the project. Steps were to:

1. identify basic requirements (listed in §1.2)
2. develop initial prototype (described in this document)
3. review functions with knowledge workers

The third item, the reviewing of functions, has occurred with Massey University internal reports and Pollen Group review meetings. Experienced palynologists are within that group and constitute the "knowledge workers".

Systems Building, like any complex operation, has many theories as to how to go about it in a methodical and time saving fashion. There are many traps to fall into while developing a product. For example, "creeping elegance" where, as something is developed to plan, new and exciting things are thought of to add to the design. The risk is that the product is not completed on time and many difficulties with the new idea are not foreseen because of the lack of early investigations and analyses. The timings of sub-module creation and intermediate testing are important, so that the testing results remain meaningful and not altered by subsequent additions or the interaction of the module with subsequent modules.

For the AutoStage, preliminary investigations of features and microscopy are done. System analysis and design are required to 'fit' the required parts together in this system integration project. Development is the main body of work with testing and implementation proving that the system is capable of meeting the aim of this project: to demonstrate capability of classifying and counting a slide as well as a trained palynologist.

3 The Capture of an Image from a Slide

Introduction

This section describes the preparation of the pollen bearing microscope slides and the system for focussing the microscopes and traversing the slides to enable the capture of images of pollen. An overview of the system is first presented so that the next sections, which detail the microscopes and lighting, are put into context. Figure 3-1 represents the AutoStage image capture mechanism.

3.1 Overview

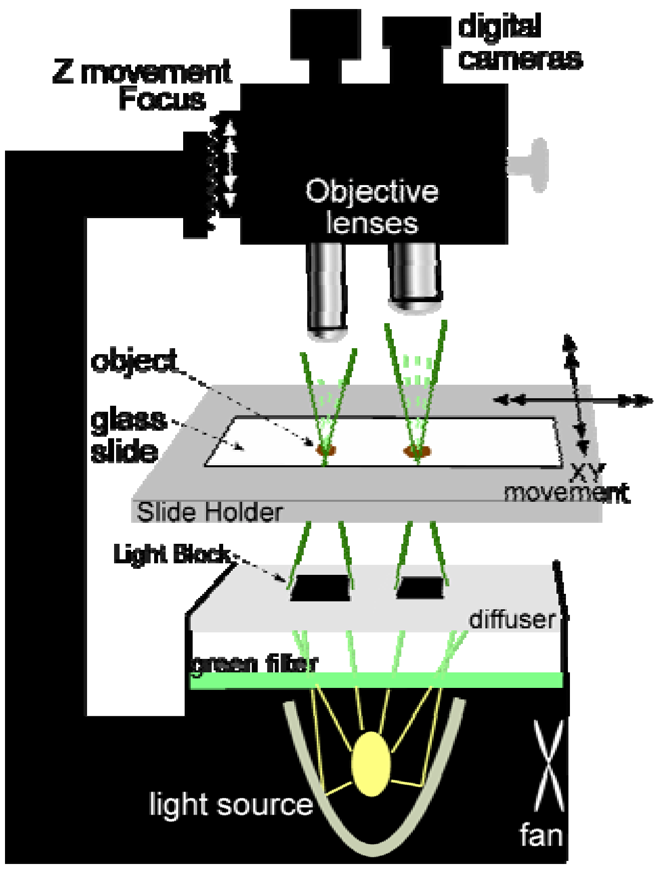


Figure 3-1: Elements of AutoStage Image Capture

A prepared slide is placed on the XY movement or "stage" where, under computer control, it is positioned between the light source and microscopes for transmission lighting microscopic image capture. An object of interest on the slide can, by the same XY movement, be positioned under either of the two microscopes. The Z direction of movement, under computer control, determines focus for both microscopes. The microscopes, once focussed, can then image the portion of slide in view; see Figure 3-2. Images are then uploaded to the PC for processing.

3.2 The Slide

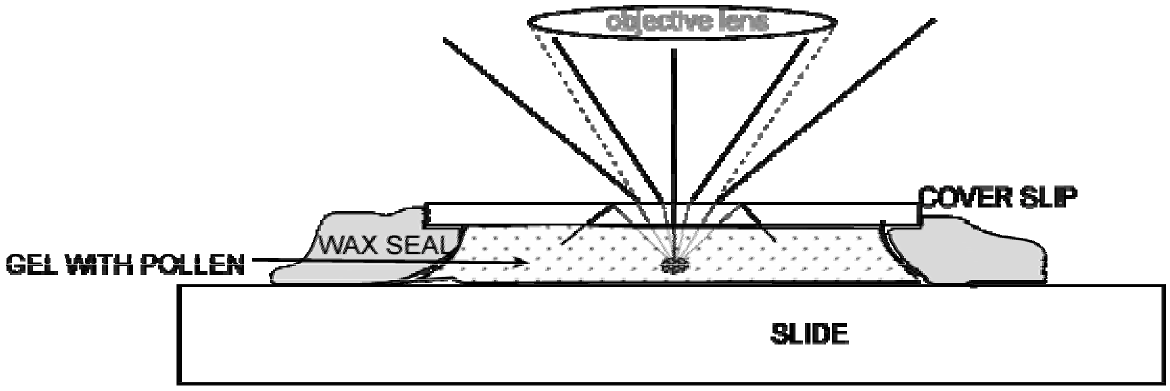


Figure 3-2: glass slide showing refraction effects.

As transmission lighting is used, light travels through the slide so the glass needs to be of a quality that does not distort the light. Light is modified within the slide by refraction, reflection and diffraction depending on the object features. The light emanating from the object toward the objective lens is refracted while transitioning from cover slip to air. It is refracted outward so that less arrives at the objective lens, consequently reducing resolution (§4.2.2). The refractive index of the cover slip glass should be as small as possible to limit refraction into the air. The cover slip glass itself must be flat, with smooth parallel surfaces. Oil immersion lenses are sometimes used in high magnification microscopy to reduce the diffraction effect by increasing the refractive index of the medium between slide and lens, making its refractive index nearer to that of glass so the dispersion of light is minimised.

Auto-focus can be adversely affected by objects on the bottom of the slide but especially on the top of the cover-slip, which is thin compared to the slide and its upper surface is very close to the focal plane of interest. In addition, the segmentation algorithms could be compromised and images captured would be degraded if dust or oil were present, even with out-of-focus contamination.

To improve the efficacy of the system the slides should be prepared in a prescribed and suitable manner. It is important that this be similar to current practice so that the system can be used without requiring changes to systems already in place.

3.2.1 Slide Preparation

The prescription proposed is for the pollen samples to be suspended in some setting gel with a refractive index similar to that of glass. Silicon oil may be desirable if the slides are to be checked on a conventional microscope, as are agar or glycerine if an aqueous medium is required. The suspension should have a concentration that results in no more than 500 pollen grains per slide to reduce clumping. Adding a drop of detergent to a last rinse before drying in any treatment of pollen will also help reduce clumping. The slide is placed on a warmer to help air bubbles escape the gel. The sample suspension volume is such that when dropped onto the slide and the cover slip is placed on top, the oil does not travel past the outer edges of the cover slip. Molten wax is dropped onto the slide at the edge of the cover slip to 'wick' under the cover slip and seal the pollen suspension in. the slide is removed from the warmer plate so the wax solidifies, contains the suspension and holds the cover slip firmly in place on the slide.

The slide surfaces can now be cleaned without moving the cover slip or pollen grains within the slide.

Fossil pollen grains are usually in mixtures of silicates, clays, vegetation etc and must undergo rigorous chemical and sieving treatments. These treatments are well documented and outside the scope of this study, however for automatic counting the slides should be cleaner and more sparse than are often used for viewing under a conventional microscope. Flenley et al. have suggested methods for improved removal of debris [17, 56].

3.3 The XY stage: specification and operation

Specification:

- 2.6 microns linear movement per step afforded by stepper motors
- 3 microns repeatability for each of two Parker 404XR linear positioners used for X and Y sections of the stage movements
- Measured: 20 microns repeatability when 1/10th micro-stepped stepper motors are used to drive the linear positioners
- 6.3 pixels per step on the high magnification microscope

- 0.7 pixels per step on the low magnification microscope
- variable steps per second: programmable with ramp-up and ramp-down

A prepared slide is placed manually on the slide holder and is fixed in place by a spring holding the slide against two lateral surfaces. This ensures a fixed placement for the slide. If the slide is removed and replaced, its relocation is such that a pollen in view under the high magnification microscope remains in the same view to within a few microns.

The slide is held between the light source and microscopes by the slide holder. The slide is held around its edges so light can pass through it. The slide holder is fixed to an XY stage designed for precision movement. Stepper motors drive the stage X and Y movements. The stepper motors have a step angle of 1.8º. The motors step by turning off the electromagnetic stator coil that holds them in place and turning on the neighbouring coil to attract the rotor to it. This may be done by gradually decreasing power to one coil while ramping up power to the next. By holding both coils at half power the rotor is held approximately half way in between. The drivers used for AutoStage are able to hold the rotor between coils, in this manner, at intervals of 1/10th of the 1.8º afforded by a single step. A 1/10th step translates into a linear motion of the X or Y stage movement of 2.6 microns. Although the XY stage is described as 'precision', there will be some 'slop' and 'stickiness' in the movement and the micro-stepping motors will not have the power to drive against the friction at small intervals. It was empirically found that using the 1/10th "micro-step", the stage can be relocated to a specific point with an offset maximum of 20 microns. The stepper motors are controlled from a PC via an RS232 serial connection to a commercial motor driver.

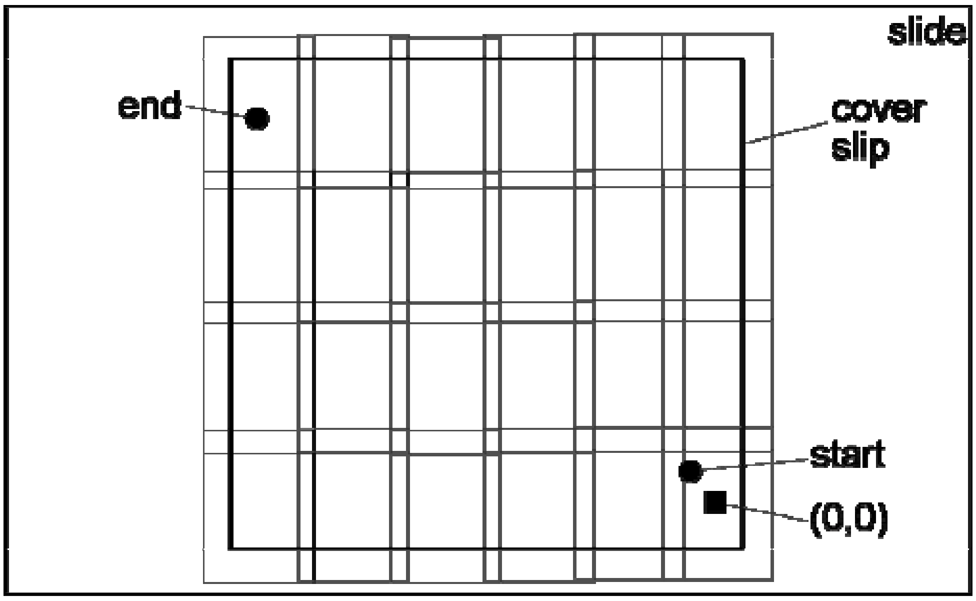


Figure 3-3: Diagram of low magnification images on a slide.

Figure 3-3 shows a slide and cover slip in plan view. The user drives the low resolution camera to the start position and marks the position by keyboard entry. The microscope is then driven to the far diagonal of some area of interest and a second position marked. The software calculates a series of evenly spaced overlapping images that will cover the area of interest completely. The centre of the final image is marked as the zero reference point for all subsequent locations. To store a location of a pollen grain in any image that relates to the reference point, considering Figure 3-4, the pollen location within the image is known by the number of pixels in rows, r, and columns, c, from the top right corner of the image. The row and column pixels to the centre of the image are subtracted so a pixel distance from the centre of the image is obtained. This is converted to 'steps' by a predetermined conversion factor (see pixels per step in the specification listing at the beginning of this section). The pollen grain distance from the image centre is added to the x,y distance from the reference point to the image centre, which is known in steps. The x,y distance in steps between the two microscope centres is known, so the high magnification microscope can now be driven to the reference point and hence to any pollen located in the low magnification imaging and segmentation operations.

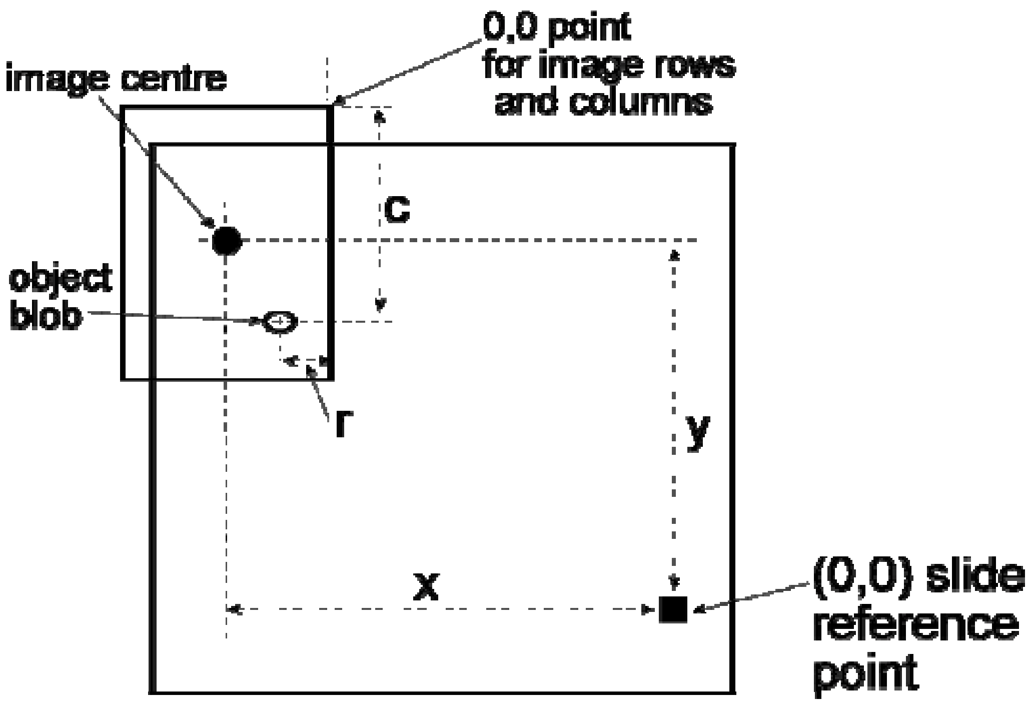


Figure 3-4: Location of an object in an image from the slide reference point. r and c are in pixels. Pixel distance from the centre is calculated and converted to steps to be added to x and y, known in steps.

3.4 Auto-Focus

Specification:

1. 12 microns of linear movement per step.
2. Approximately 10 seconds per focus

For consistency and ease of integration, the movement control mechanism for the 'z' movement, or focus, is the same as for the 'x' and 'y' movements: a stepper motor controlled by a commercial controller/driver unit that is under computer control via the serial port. The stepper motor rotational movement is converted to a linear movement by a standard microscope focus worm drive mechanism.

Auto-focus is performed once for the low magnification imaging of the slide. For each object imaged with the high magnification microscope an auto-focus is performed. The focus was developed to be completely automatic but a once-per-slide user pre-focus was added to increase reliability.

There are three levels at which a focus might be found: the bottom of the slide; the top of the slide under the cover slip where the pollen is; and the top of the cover slip. There is often dust, oil, finger marks or glass imperfections that show up at these levels. If the slide is very clean the focus algorithm could find the correct one by the magnitude of the parameter measured for finding focus. Built into the algorithm is a wide sweep for finding the three levels and deciding through a series of tests which local peak is the correct one.

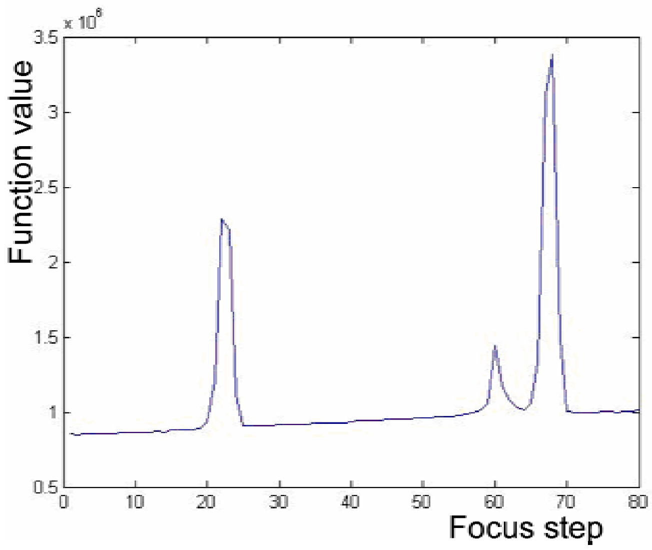


Figure 3-5: A gradient squared measure of focus showing the three focus levels of a slide. The peaks from left to right indicate: the slide bottom, the slide/cover-slip with the pollen suspension, and the top of the cover slip. The slide is quite dirty which is indicated by the outer surfaces having more detail and therefore higher focus measurement values.

It was decided that the risk of finding the wrong focal plane was reduced if the user set the focus manually while setting up the system and the algorithm then refocused around that point, assuming that the result would be more consistent using an auto-focus. The same user-initiated focus position, adjusted for the difference in height between the two microscopes, is used as a starting point in focussing for high magnification image capture.

An auto-focus capture sequence for each object found is performed in an upward direction. As with the focus sequence capture, driving the microscope to the focus position found is always in the upward direction. The upward direction of critical focus movements reduces mechanical backlash and stickiness effects thus reducing variation in the resulting position.

Stopping focus at or near the first focus peak found would speed up the focus process but risk focussing on dust or glass surface artefacts as was noted by Geusebroek et al. [22].

For the focus algorithm, a quantitative measure of focus is extracted from each image and stored as an array of numbers in the order of their capture. A local maximum algorithm finds any peaks in the array. A noise floor in the array is estimated to eliminate the smaller peaks. The peak values and the absolute maximum value are stored along with their position in the array. If the absolute maximum is at the beginning or end of the array then it is discarded. Remaining peaks are compared for height, and position within the array, and an algorithm determines which is most likely to be the peak representing the pollen focal plane.

A number of focus measures were gleaned from the literature [28, 39, 54, 61] and starting with the simplest and least computationally intensive, were trialled in images from the AutoStage to gauge their effectiveness. Eleven in all were trialled including: standard deviation of all grey levels; variance; a normalised variance measure; maximum of x,y direction gradient of neighbouring pixels; vollath4; vollath5; the derivative in x and y directions which is essentially a measure of the slope of values between pixels; a measure of power in all pixel values; a histogram measure; a measure of entropy in the image grey level values; and the Fourier transform where the mean of the largest 1000 values of the real portion of a fast Fourier transform of grey level values is calculated. Functions that required a threshold required extra considerations to determine the correct threshold and were found to be less effective with large variations in image type. Some of the better performing functions are shown in the graphs of function-value versus focus-step in Figure 3-10.

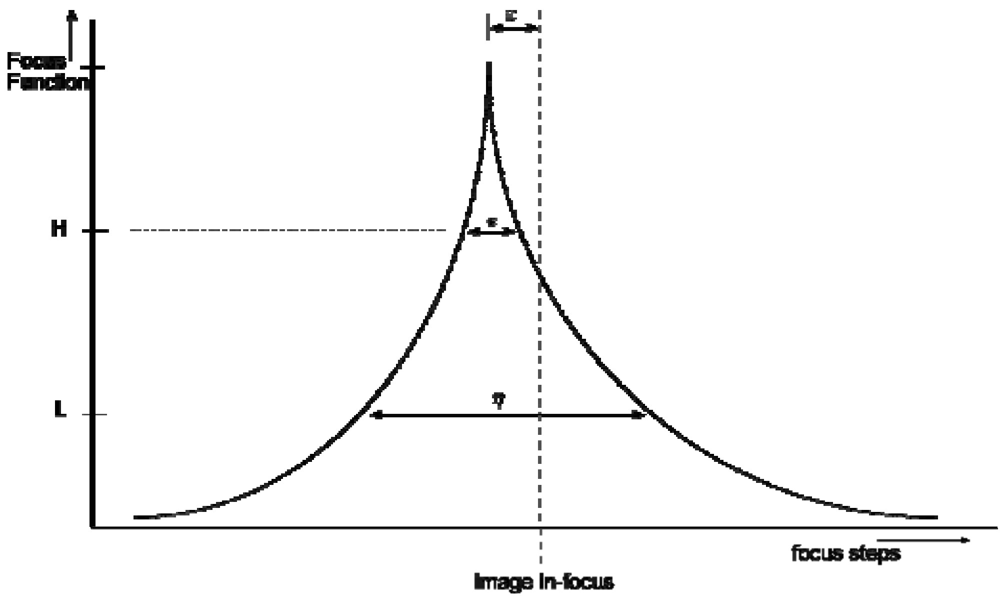


Figure 3-6: Focus Function. This figure from Groen et al. [28], defines the properties required of a good focus function. *v* should be small and reproducible while η is large, ideally ε = 0. This figure can be used to evaluate the graphs in Figure 3-10.

Groen [28], evaluated eleven functions and concluded functions based on squared derivatives and normalised standard deviation would have the required properties. Santos [61], concludes Vollath4 and Vollath5 to be top of their performance list, however, Kehtarnavaz [39], shows results showing squared gradient to be better when compared to Vollath5: Figure 3-7.

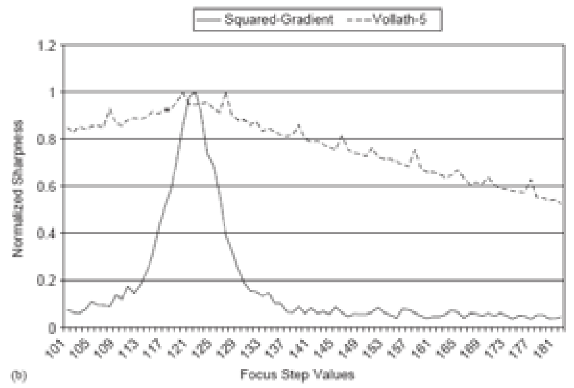


Figure 3-7: Focus measures; squared gradient compared to Vollath5 [39]

For images from AutoStage, the squared gradient produced very good results. Over a range of image types, all from AutoStage, the squared gradient was empirically more consistent in performance than others of similar effectiveness such as the derivative. In trialling the gradient measured in both x and y directions, the x direction was found to produce a better result more often. By choosing the maximum gradient value between x and y directions at each pixel, it was found to be even more consistent in performance so this was the chosen function and named “maxGrad” in the software code.
A Matlab function for gradient is used to obtain x and y gradients and the maximum for each value is determined, squared and then all values are summed. The gradient is described in the Matlab help file as:
{FX,FY} = GRADIENT(F) returns the numerical gradient of the matrix F. FX corresponds to dF/dx, the differences in the x (column) direction. FY corresponds to dF/dy, the differences in the y (row) direction. The spacing between points in each direction is assumed to be one.

For low magnification images, the maxGrad was found to be too sensitive to noise and introduced many peaks which would hinder the local peak finding algorithm. The large “width at a low percentage of the maximum”, (η in Figure 3-6), property of the normalised standard deviation, considered to be valuable by Groen [28], was found to reduce the noise peaks and proved to be more reliable for low magnification images. The width at high percentage of maximum, (ν in Figure 3-6), is also large but did not affect the result in the trials performed. The variance proved to be a better function given Groen's criteria (Figure 3-6) but as standard deviation showed equal performance for the images tested and was computationally less demanding the standard deviation of the image grey values was chosen for the low magnification auto-focus function.

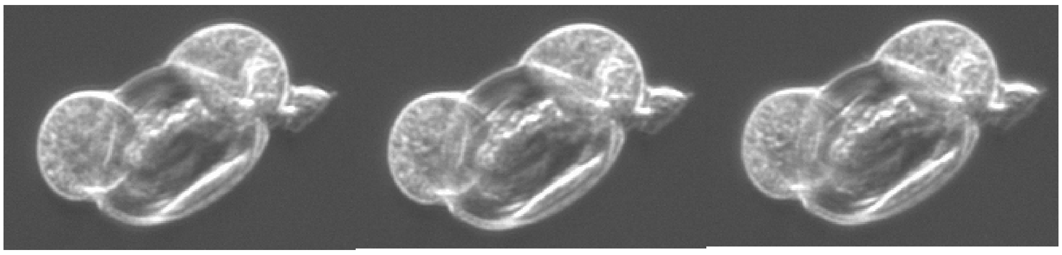


Figure 3-8: Cropped image series; #18, #19, #20. #19 was selected as 'in-focus' by the focus algorithm.

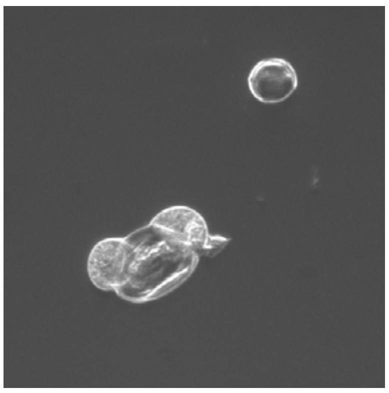
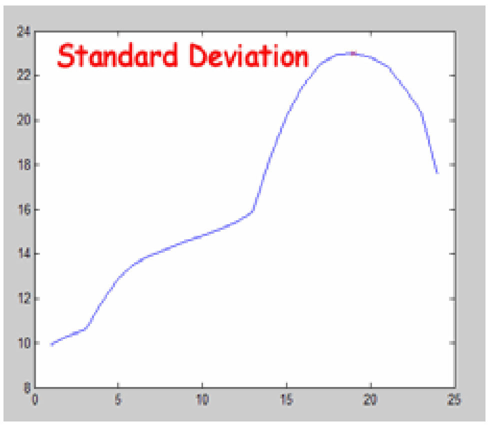



Figure 3-9: The entire In-focus image, #19 of series used to produce associated graphs, the first of which is shown above, to the right . More graphs of function values versus focus step are shown in Figure 3-10, and a second image series example is shown in Figure 3-11.

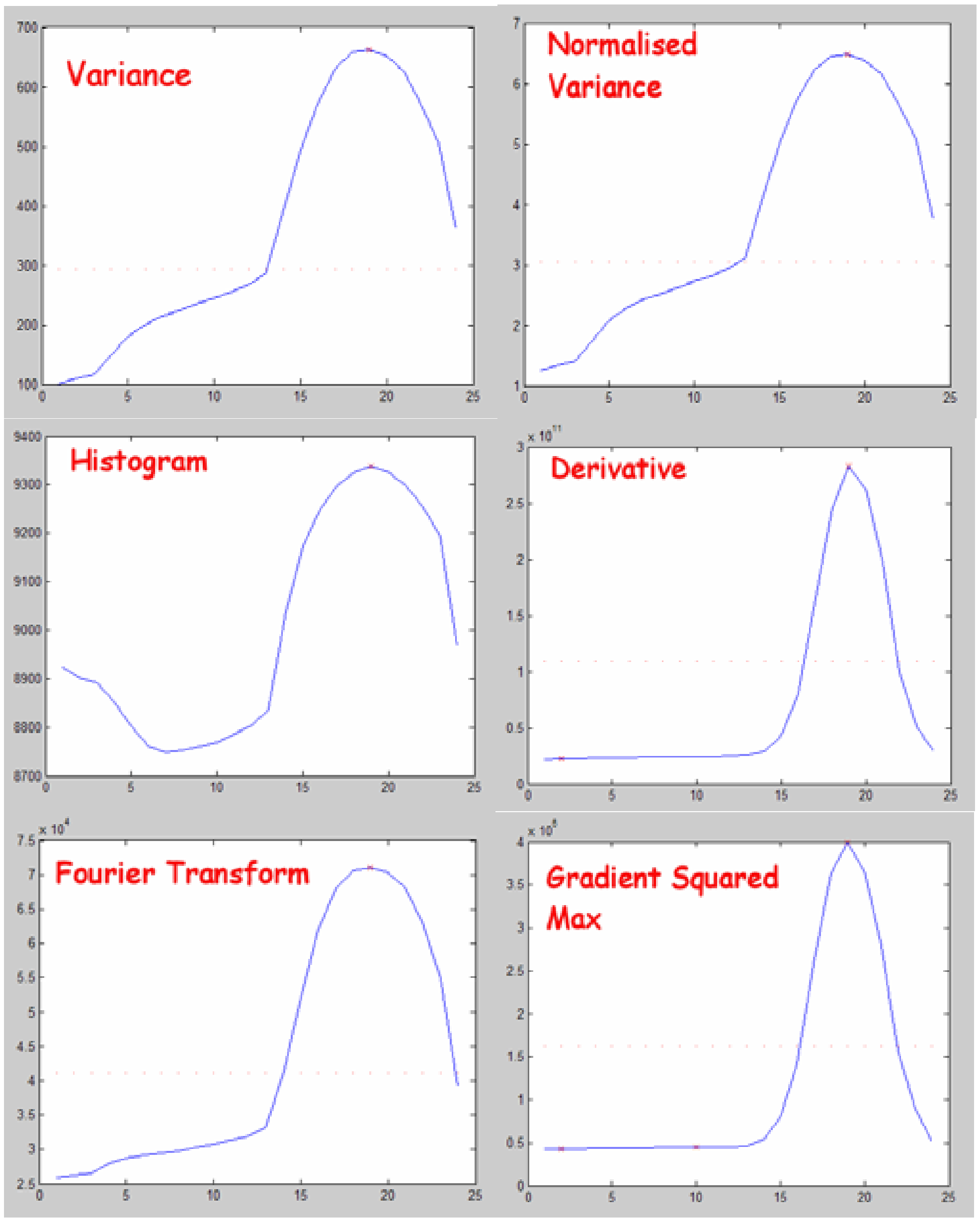


Figure 3-10: Focus graphs; focus algorithm values versus image series number. The focussed image is shown above. A cross marks any local peaks and a dotted line across marks a noise floor calculated level, below which peaks are ignored. All algorithms detect image #19 but the shapes vary as shown.

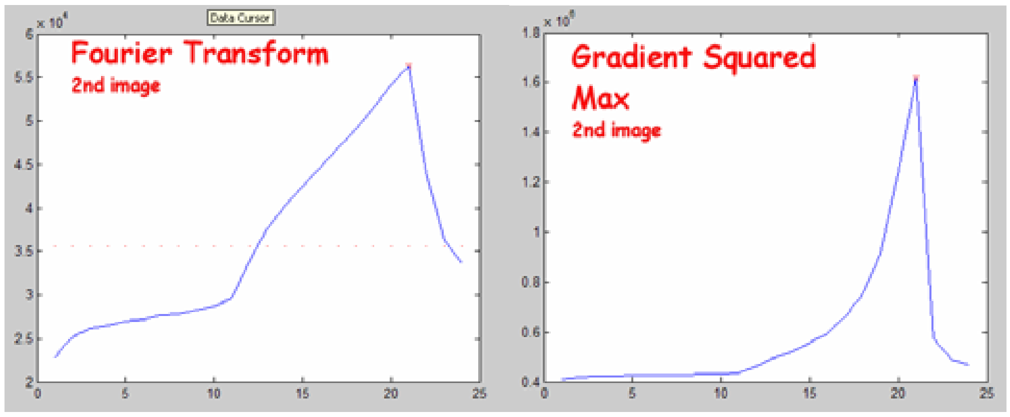


Figure 3-11: Focus graphs for a second image (shown below)

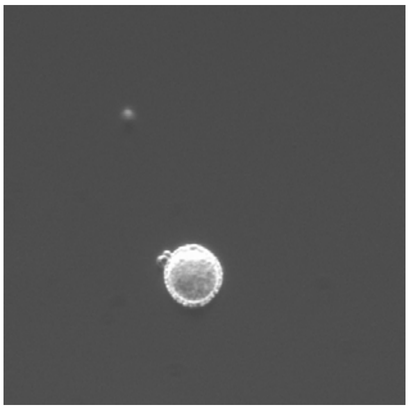


Figure 3-12: A second image series used for focus data: in-focus image #21.

4 Microscopes

Introduction

A suitable microscope was proposed for the project by Holdaway [34]. That design was reasonably faithfully followed for developing and building into the system and is described below. For this project a second, wide angle, or low magnification, microscope was also developed. In order to quantify any changes in the prototype it is important to have a base measure of parameters so measures such as resolution and depth of field are defined. These parameters may be used for parameter and tolerance design as described in §2.4.

4.1 Low Magnification Digital Microscope

specification:

- 640 x 480 pixels
- Magnification: 1x, a 10 micron pollen grain is about 7 (2.6^2)pixels.
- 1 pixel = 3.8 microns
- Field of view is 2.43mm x 1.82mm
- 100 low magnification images will cover the area under a standard rectangular microscope slide cover slip

The low magnification microscope was included to capture images of a slide with enough resolution to detect the presence of pollen grains of all sizes of interest (10-100 microns) and a field of view to allow a series of as few images as possible to cover an entire slide. The aim for the inclusion of the second microscope was to speed up the pollen location process. Slides are to be prepared with a relatively low concentration of pollen to avoid clumping (see §3.2.1) and the high magnification microscope would have therefore captured many images with no pollen in them. Any requirement of auto-focus for the low magnification images would have slowed the process, reducing or negating its usefulness.

An option had been to adapt one microscope for both tasks by making the high magnification microscope camera adjustable in distance from the objective lens to alter the magnification. That option required added mechanical complexity and perhaps another drive motor for the adjustment to be made automatically. Suitable cameras are now inexpensive and the mechanism to drive the slide under each would exist, so the decision was made to add a second microscope for the prototype.

The limiting factor for field of view was the need to detect the smallest pollen of interest (10 micron diameter). An optical magnification of 1x results in the area of an object of 10 microns diameter, being represented by about 7 pixels. Unity magnification, about 1/10th of magnification of the high magnification camera, was chosen. The resulting field of view is 2.43mm x 1.82mm.

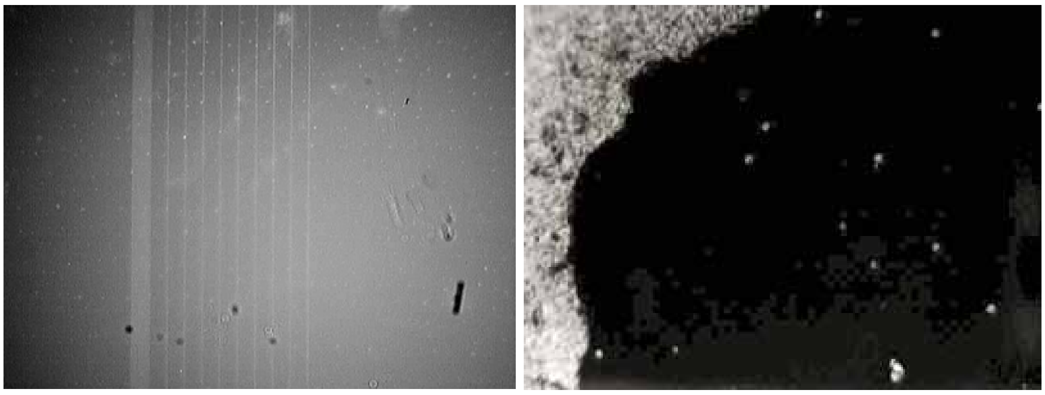


Figure 4-1: Low magnification microscope images showing, from left to right, 100micron spaced lines and pollen on a slide. The white mass near the edge is a wax seal.

The depth of field is large enough, and sensitivity to focus low enough for the system to focus once and capture the entire slide with a single focus setting. The images from this camera form a series that overlap slightly and segmentation removes any objects found touching the borders. Pollen found twice as a result of the overlap are removed by an algorithm described in §6.3. An existing problem with this approach is that any pollen found twice adds to the time taken by the system as a whole.

4.1.1 The Camera Sensor

The microscope was made by removing the camera printed circuit board and lens from an inexpensive “web-cam” and adding a plastic tube between its sensor and lens to effectively increase lens to sensor distance and increase the magnification. The tube length required was 5mm. The lens screws into its holder, which, in the camera, is a focus mechanism. It was useful for fine tuning the magnification.

The magnification of the low magnification microscope is 1x. A small, inexpensive sensor and lens module on a printed circuit board was modified by increasing the lens-to-sensor distance sufficient to achieve the required magnification. The sensor and lens combination was removed from an inexpensive "web-cam". The magnification was measured using a slide micrometer: a microscope slide with 11 etched lines 100 microns apart (1mm) and lines 10 microns apart between two of them. By taking an image and counting the pixels from line to line, the microscope could be calibrated (see Figure 4-1).

The camera was built into a housing, constructed at Massey University, and secured to the high magnification microscope. It has an 8mm sensor with 640x480 pixels that are about 9 microns square.

4.2 High Magnification Digital Microscope

Specification:

- 1024 x 768 pixels
- pixel size is 4.65 microns square
- sensor size has a 6mm diagonal
- Magnification 11.2x, a 10 micron diameter pollen is represented by about 580 pixels
- 1 pixel represents 0.415 microns of the object
- Field of view is 0.425mm x 0.318mm

Table 4-1: Microscope optical data with symbols used in formulae below.

unit	**symbol**	**value**
magnification	m	11.2
Image distance	v	207.4mm
Object distance	u	18.6mm
Object distance from front element		7mm
Lens Focal length	f	17mm
Numerical aperture	N	0.25
Objective lens aperture	a	2.5mm
Circle of confusion	c	0.0018mm
Wavelength of light	λ	550×10^{-6}mm
Refractive index of air	n	1.0

The high magnification microscope captures detail almost down to the resolution allowed by the wavelength of light being used (see §4.2.2, equation (4.3)). This microscope consists of a CCD sensor and image forming lens. The lens used is a standard microscope objective. The advantages of the microscope as designed are that it is easily fabricated, inexpensive and has a low *optical* magnification. The low optical magnification results in a larger depth of field than a conventional microscope with a similar ‘viewing’ magnification. The digital microscope viewing magnification is similar to the 400x conventional microscope used for capturing the images used in testing of the classification system (see §8.1). The additional magnification for viewing eventuates from the increase in pixel size from the 4.65 microns in the imaging sensor to the medium used to view the image. For example, 1024x768 pixels in the 6mm diagonal sensor is translated to 1024x768 pixels on say a 432mm (17”) diagonal computer screen, resulting in about 72x magnification and 790x magnification overall, taking the 11.2x optical magnification into account. This accounts for images viewed on a computer screen. For feature extraction, the resolution (§4.2.2), and correct sampling of it by the sensor (§4.2.4) is important.

The effects of aperture were discussed by Holdaway [34], and diffraction effects were found to be limiting the resolution. Aperture control is not implemented in AutoStage and the widest available aperture, limited by the objective lens, is used. In general, lighting intensity can be altered by:

- adjusting the size of an aperture, “stopping”
- reducing the intensity of the light source
- and in the case of the digital microscope, altering the sensor capture time or ‘shutter speed’.

Stopping down increases depth of field but also degrades resolution (§4.2.2). A pollen grain might be as small as 10 microns across. Some features, as shown in images from a scanning electron microscope (SEM, see Figure 4-2), are of sizes below the limits of resolution of this system, so resolution is the limiting factor here. No stopping function was added because resolution should not be reduced and image intensity is adjustable by the other means mentioned above.

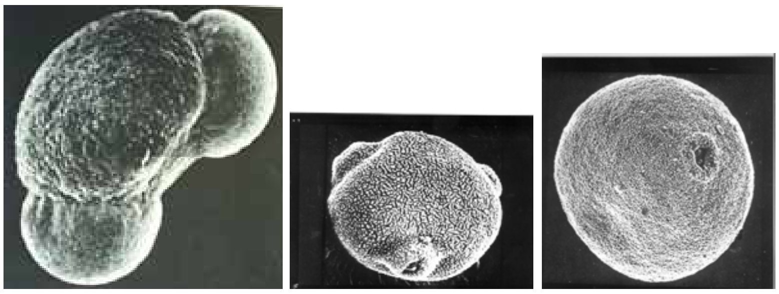


Figure 4-2: Scanning Electron Microscope (SEM) images of Scots Pine grain (left) Birch Pollen grain (centre) grass pollen grain (right). Scale is unknown. [58]

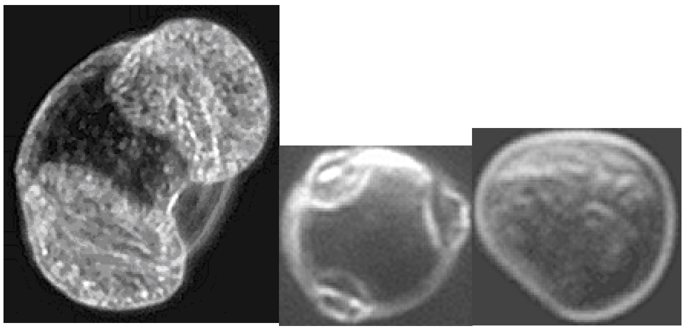


Figure 4-3: An interesting comparison of pollen captured on AutoStage, compared to the SEM images in Figure 4-2. From left to right they are Pine (Radiata), Silver Birch, and grass (Brown Top) A Silver Birch pollen may vary from 15 to 28 microns diameter. The pine pollen are about 50-70 microns across. Relative scale of images is approximate only.

4.2.1 Imaging Aberrations

A number of possible aberrations are introduced in §2.2. Most are dealt with conveniently by the use of a manufactured objective lens. Two that may not, "Curvature of Field" and "Distortions", are discussed below.

Curvature of field is caused by the light radiating outward from the lens making equidistant points a curved field. This might be corrected by a curved image sensor, however the sensor is small enough on the curved surface that the effect is not significant as the following shows. The diagonal of the image sensor from centre to one corner measures 3mm. Considering Figure 4-4, the maximum offset of the sensor from the true image, x, is calculated as follows:

$$\tan a = 3/207.4 \Rightarrow a = 0.829° \tag{4.1}$$

$$\sin a = 3\times10^{-3}/\left(207.4\times10^{-3}+x\right)$$
$$\Rightarrow x = \frac{\sin a\times207.4\times10^{-3} - 3\times10^{-3}}{-\sin a} \approx 50\times10^{-6}m \tag{4.2}$$

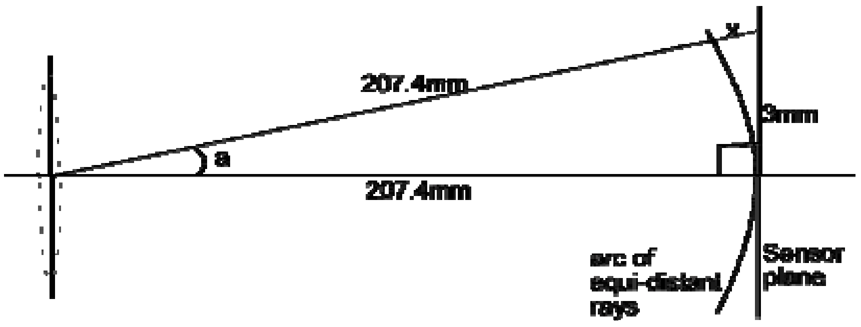


Figure 4-4: Field Curvature Distortion

To determine if this is significant, by similar triangles: $2.25/207.4 = c/x \Rightarrow c = 2.25\times50\times10^{-6}/207.4 \approx 524\times10^{-9}m$; where x, shown in Figure 4-4 and calculated in equation (4.2), is an error in focal position; and where c is the circle of confusion radius as in Figure 4-5.

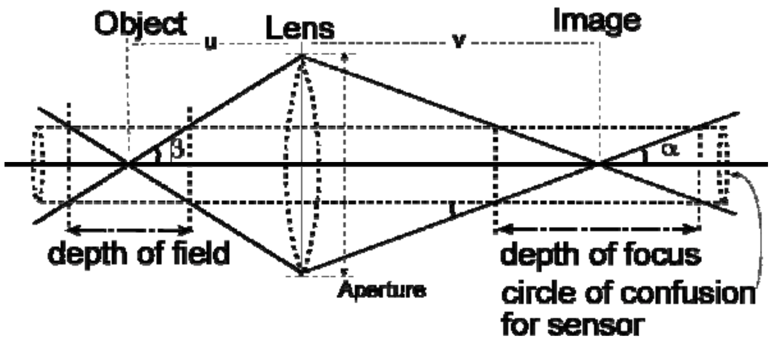


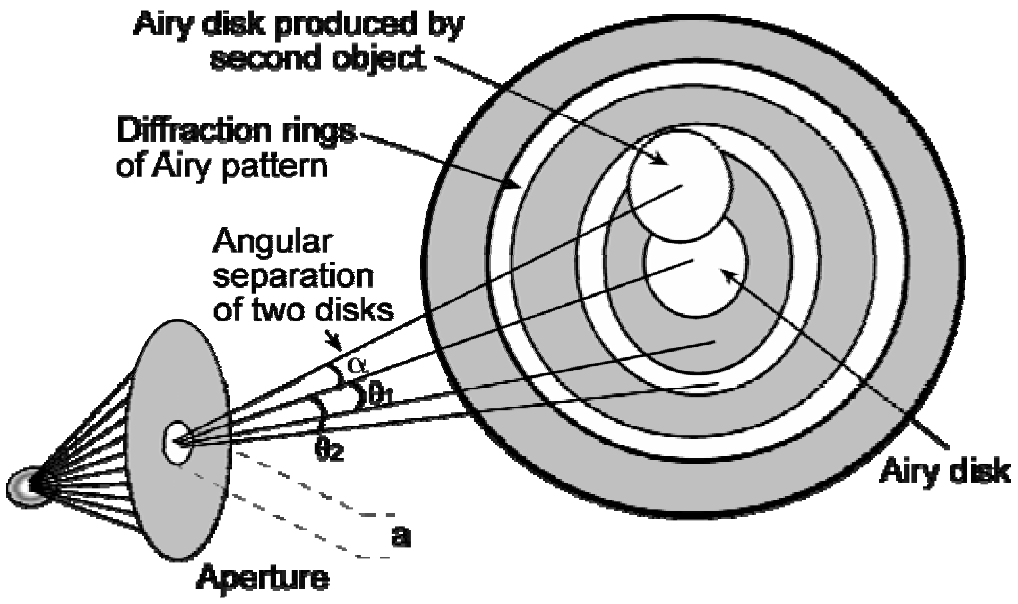
Figure 4-5: Circle of confusion for Depth of Focus

The circle of confusion of 0.5 microns, resulting from the shift of 50 microns along the focus field is not significant compared to the sensor element size of 4.65 microns.

Distortions are often caused by misplaced apertures, or “stops”. AutoStage requires as large an aperture as possible to maximise resolution (§4.2.2). This is achieved by having no “stops” other than the restriction afforded by the size of the objective lens. As the objective lens group is manufactured, aperture size is fixed, however if stops were ever to be considered, this effect would need to be revisited.

4.2.2 Resolution

The smallest resolution we can theoretically obtain is defined first by λ, the wavelength of the light, however diffraction effects must be considered. Diffraction adds to the resolution limit imposed by wavelength. When a point source passes through a limiting aperture, it produces a central bright “Airy disk” (Sir George Airy, 1801-1892) around the image point. There are also bright diffraction rings extending outward, but of decreasing intensity, the first of which has only 1.7% the radiant power of the centre of the Airy disk [62]. The Airy disk angular size is calculated by, $\sin\beta = 0.61 \times \lambda / a$. Rayleigh proposed that two points are resolved if the centre of one is no nearer than the first dark ring just outside the Airy disk [62].



Airy disk produced by second object
Diffraction rings of Airy pattern
Angular separation of two disks
α
θ1
θ2
a
Aperture
Airy disk

Figure 4-6: Light through an aperture showing Airy disk, diffraction patterns.

For a lens system such as in microscopes, the resolution depends on the diameter of the smallest light restriction in the objective and the lens-to-object distance. This characterises the light collecting ability of the lens and forms twice the angle β shown in Figure 4-9. which is called “angular diameter” of the lens.

The numerical aperture of a lens is measured as the sine of ‘β’, multiplied by the refractive index, ‘*n* ’ of the medium that light passes into, from the lens. The refractive index is often taken as 1.0 for air. When oil immersion lenses are used, for increasing resolution, the refractive index of the oil must be used. Numerical aperture is often quoted on lenses as an indication of their resolving capability.

Resolution depends on many things including conditions of illumination. Some formulae for resolution include the numerical aperture of the condenser lens in the lighting system. The Rayleigh criterion is often used, however that is a subjective value based on empirical observations and involving the optics of the eye, so it is not suitable here. The Sparrow criterion estimates a 26% increase in resolution over the Rayleigh criterion and is less arbitrary in its definition by stating that two objects are resolved if the combined intensity half way between their Airy disk centres is equal or less than the centre of the Airy disk of lower intensity [23]. It can be visualised, by considering Figure 4-7, that as the two points move closer together and the combined intensity forms a single peak then the two peaks are no longer distinguishable. This criterion does not depend on the human eye so is more appropriate for digital imaging as is used in AutoStage.

The Sparrow criterion is stated more formally:

when both, $\frac{\mathrm{d}}{\mathrm{dx}}[f(x)+f(x+\sigma_L)]=0$ and $\frac{\mathrm{d}^2}{\mathrm{dx}^2}[f(x)+f(x+\sigma_L)]=0$ [38],

f(x) is the instrumental response, and σ_L is the Sparrow limit.

The Sparrow limit: $R=\frac{0.47\times\lambda}{n\times\sin\theta}$, and Rayleigh limit: $R=\frac{1.22\times\lambda}{n\times\sin\theta}$.

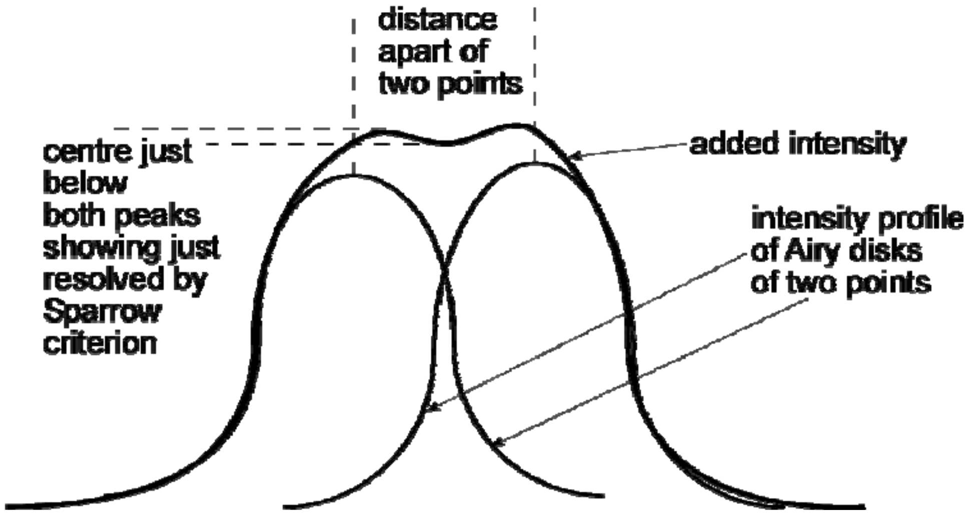

distance apart of two points
added intensity
centre just below both peaks showing just resolved by Sparrow criterion
intensity profile of Airy disks of two points

Figure 4-7: Airy disk intensity profiles (intensity versus distance across disk) showing Sparrow criterion

Resolution is defined by equation (4.3) in [71] as a reasonable approximation of resolution for common usage, so for AutoStage:

$$R=\frac{0.5\times\lambda}{n\times\sin a}=\frac{0.5\times550\times10^{-9}}{1\times\sin\left(\arctan\frac{2.25}{7}\right)}\approx0.9\times10^{-6}m \qquad (4.3)$$

What is essentially required for AutoStage is a practical maximisation of resolution. It does, however, need to be tempered with consideration for depth of field and particularly contrast. Adjusting parameters to minimise one will, in some cases, affect the others.

The slide, cover slip and pollen suspension gel, also have an effect on resolution. Light from the object to the objective lens is refracted outward by the transition from the cover slip into air so resolution is decreased as it effectively reduces the angular diameter of the objective lens; see Figure 4-8.

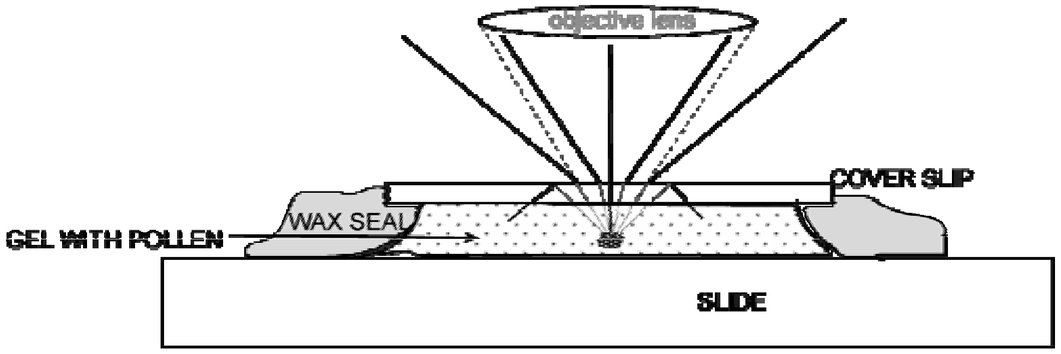


Figure 4-8: glass slide showing refraction effects.

4.2.3 Depth of Field

Depth of field and depth of focus are similar measures, with depth of field being the range of "in-focus" distance of the object from the lens and depth of focus being the range of distance the image is "in-focus" from the other side of the lens. Only one point in each case is perfectly in-focus, but the blurring increases as distance away from that point increases and *depth* of field/focus is therefore some decision as to what level of blur is tolerable. The size of a "circle of confusion", shown in Figure 4-9, is a measure of blur. In the case of the digital microscope with auto-focus, the sensor (image) is fixed in place, and the distance to the object is moved, or focussed, until determined "in-focus" by some algorithm. The image sensor is fixed and therefore defines the focal plane exactly. This leaves depth of field to be determined. As defined in Figure 4-9, depth of field is the range of object positions along the optical axis deemed to be in focus.

To quantify depth of field, first a circle of confusion is defined. It could be defined as the size of the image sensor elements as any movement of the light ray across the width of the sensor element makes no difference to the illumination detected, but this assumes the sensor detects the same level of light falling on any part of it and also assumes there is no Airy disk (see §4.2.2). An Airy disk size could be calculated and factored into the amount it could move before affecting a neighbouring sensor element.

With the image sensor position fixed and defined as the image plane, the object depth of field can be calculated. The image sensor has elements that are 4.6 microns square. Any blurring within this size is assumed not to be detected by the sensor, so the circle of confusion size we define as no *detectable* blurring to be tolerated. The Airy disk angular diameter is calculated as $\sin\beta = \frac{0.61\times\lambda}{a}$ [62]. Therefore, considering Figure 4-9, its diameter is:

$$\frac{0.61\times\lambda}{1\times\sin\left(\tan^{-1}\left(\frac{a/2}{u}\right)\right)} = 1.1*10^{-6}m \tag{4.4}$$

So, a point forming an Airy disk on a 4.65 microns square image sensor element, with the centres concurrent, has almost 1.8 microns to move before affecting a neighbouring image element. That defines a circle of confusion radius, usable for AutoStage.

Young et al. [73] say, "The depth-of-focus ($\Delta\mathbf{z}$) of a microscope system has been described by a number of authors. Unfortunately, these descriptions do not agree". That has been the result of searches for a suitable formula for AutoStage. A derivation of DoF for this project was performed with a result within the same range as the three equations, (4.5), (4.6) & (4.7) and is placed as an appendix in §A. The depth of field is estimated by these equations to be around 10 microns.

By geometric manipulations of diagrams such as Figure 4-9, and manipulation or differentiation of the lens equation, $1/u + 1/v = 1/f$, some change in u, away from f along the optical axis, will cause a corresponding change in v which will result in the circle of confusion increasing in size. The equations (4.5) and (4.6) were derived much in this way, however equation (4.7) was derived as explained in [73].

Table 4-1, explains the common parameters used in these equations.

Depth of Field may be calculated by [55]:

$$DoF = \frac{2\frac{f}{a}du(u-f)f^2}{f^4 - \left(\frac{f}{a}\right)^2 d^2u^2} \tag{4.5}$$

as cited by Holdaway, however there is no clear derivation offered.

Conrad [11], derives equations in detail and:

$$DoF_n = \frac{Ncu(u-f)}{f^2 + Nc(u-f)}, \qquad DoF_f = \frac{Ncu(u-f)}{f^2 - Nc(u-f)} \tag{4.6}$$

The subscripts in this equation for DoF indicate the portions of DoF from the object toward the lens (n), and from the object away from the lens (f). These are δu2 and δu1 respectively, in Figure 4-9.

Young et al. [73], use diffraction more directly than using the circle of confusion to derive an equation that is empirically tested:

$$\Delta z = \frac{\lambda}{4 \times n \times \left(1 - \sqrt{1 - N/n}\right)^2} \tag{4.7}$$

where Δz is depth of field.

Figure 4-9 shows the rays that would form an image and the boundary rays that would define a circle of confusion around the image sensor.

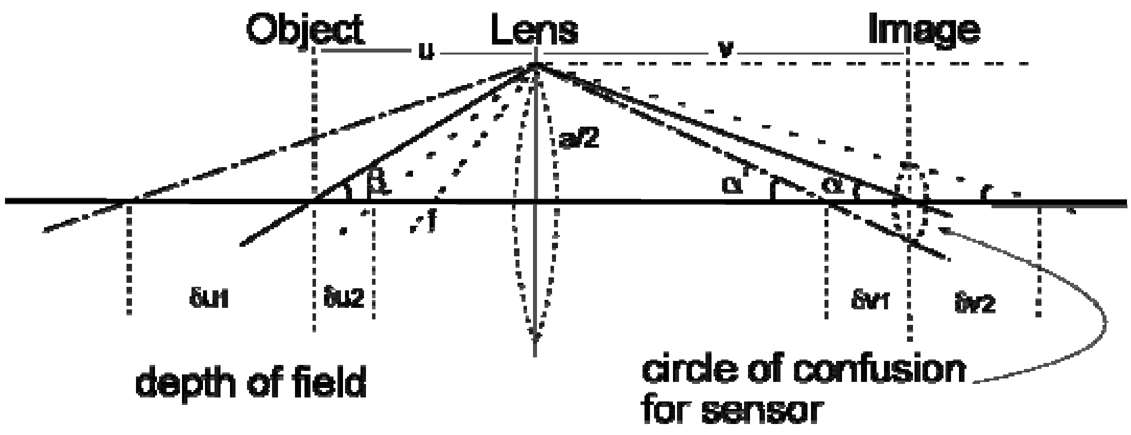


Figure 4-9: Ray Diagram for Depth of Field. An image forming lens and its half angular diameter, β. Image distance, v, divided by object distance, u, is magnification, m. For a given circle of confusion about the image point, there is an associated depth of field around the object point. A thick lens simply creates two principal planes that affect the values of parameters but do not affect the relative measures used in the associated equations.

Numerical aperture (NA) is a commonly quoted figure of merit, indicating resolution of an objective lens and is described in §4.2.1. One important area for further testing is to determine whether increasing NA to increase resolution, with the side effect of decreasing depth of field, would improve images for feature extraction. Depth of field may then be improved by the integration of the most-in-focus parts of each of a series of images taken in steps with the focal plane moving through the object. An example is shown in §9.3.2.

4.2.4 Magnification

The objective lens is a set of lenses designed to act as a single magnifying lens, but has corrections in the design to compensate for aberrations that occur (see §2.2 and §4.2.1). The objective lens has a nominal magnification of 10x and a numerical aperture of 0.25. A physical tube separates the objective lens from the eye piece in a conventional microscope but the standard measurement for “optical tube” is the distance from the image side focal point, *f*, to the real-image plane. Magnification is the optical tube length divided by the focal distance, *f* [23]. The objective lens has a 10x magnification when a JIS, Japanese standard 170mm optical tube is used. (160mm is the DIN or European standard). Therefore f = 170/10 = 17mm. The objective lens is being used with a magnification of 11.2 by moving the image sensor to 207.4mm from the lens. The new optical tube length is 207.4 - 17 = 190.4mm and 190·4/17 = 11.2. This was verified by using the micrometer slide as used to determine magnification for the low magnification microscope.

Given that the sensor is sampling the image, then it is expected the sampling theorem will hold. Sampling should therefore be at twice the maximum spatial frequency; that is half the minimum spatial resolution. Resolution is found to be about 0.9 micron in §4.2.2, so sample spacing required is at 0.45 microns or less. With a sensor pixel dimensions of 4.65^2 microns and a magnification of 11·2, the object area represented in a pixel is $(4.65/11.2)^2 \approx 0.42^2$ microns which satisfies the sampling criterion attributed to Nyquist [52], and stated more directly by Shannon [63]. Thus it is concluded that magnification is sufficient to capture all information that resolution affords the system.

4.2.5 The Image Sensor and Camera

The camera is a MicroPix™, model M1024. Connection to the computer is by IEEE 1394 serial data standard or “FireWire”. Using progressive scan it can transmit up to 30 frames per second. The MicroPix incorporates a Sony™ ICX204AL image sensor (see appendix in §11.1.2). The active sensor area is a rectangle of 5.952mm diagonal incorporating 1024x768 pixels that are 4.65 microns square.

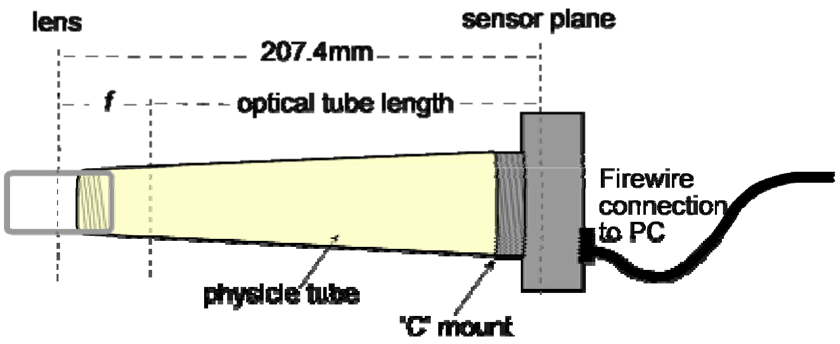


Figure 4-10: High magnification microscope construction

The lens mounting provided on the MicroPix is a standard 'C' mount. A physical tube to optically seal and fix the distance from the sensor to the objective lens, was manufactured at Massey University. A tube length of 207.4mm fixes the magnification at 11.2x as in Figure 4-10.

The Matlab image acquisition toolbox allows a data connection from the PC to the MicroPix camera via an IEEE 1394 (FireWire) serial connection.

Some resulting images are shown in Figure 4-11.

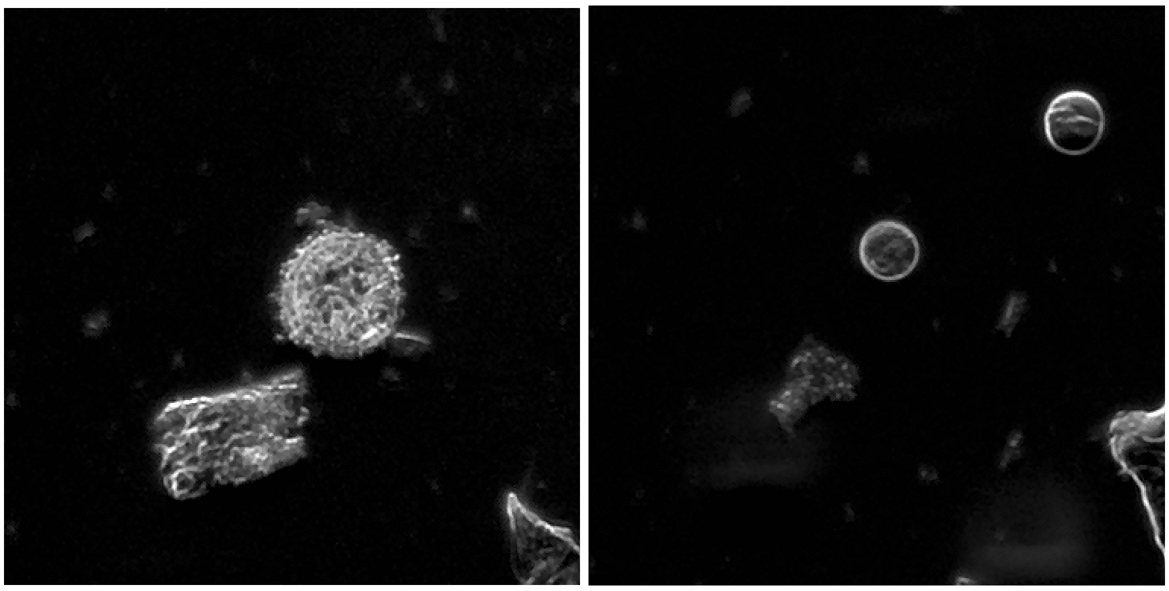


Figure 4-11: High magnification images cropped to the central portion (500x500) of the image. The pollen on the left is about 40 microns across and the pair in the right hand image are about 20 microns diameter.

5 Lighting

Introduction

Lighting sources were considered by Holdaway [34], and an incandescent lamp suggested as suitable on the grounds of "design simplicity". Lighting is considered here further.

A mains powered incandescent lamp outputs light of varying intensity due to the alternating supply voltage. With small image sensor integration times, the images would be of varying brightness unless appropriate capture timings were implemented. A simple and suitable alternative of a stable, regulated, direct current supply was chosen. A quartz-halogen lamp was found to be suitable for intensity output, size, price and availability. Heat was a problem so a cooling fan was added to mitigate the effects.

Light emitting diodes (LEDs) were considered as a more efficient visible light emitter. The output of a high-intensity LED is about 12 candella (Cd) compared to 8000Cd of the quartz-halogen lamp. Multiple LEDs using a timed and pulsed supply was considered but the incandescent lamp was selected, again for its simplicity.

Holdaway [34], noted that band-pass filtering the light will result in less chromatic distortion. This is theoretically true however the objective lens is corrected for chromatic aberrations. Most objective lenses are spherically corrected at the sodium yellow wavelength of 589.2nm [55]. The MicroPix camera is filtered internally so that its highest sensitivity is to the green part of the spectrum at about 550nm. Green to yellow/green light wavelength is in the 500-600nm range and human vision is most sensitive around 546nm. While no improvements could be discerned viewing images on a screen with such filtering, it was considered prudent to use a yellow/green filter to limit the band-width of the AutoStage lighting source.

The light forming the image, is reflected (specular, diffuse, or both), refracted and diffracted by the object [23]. The *plane of polarisation* of light can be affected by reflection and scattering [24]. Image formation is therefore complex, and beyond the intended scope here to fully define. Pollen are transparent in nature with fresh pollen containing gamete material inside the shell, while fossil pollen have decomposed, all except for the sporopollenin shell. Fossil and fresh pollen grains appear quite differently under the microscope. This prototype, AutoStage, allows for empirical measures of lighting quality by comparing classification results using different lighting schemes for various object types. The scheme adopted by initial design affords good resolution and excellent contrast. It was chosen as it performed best when examined on a computer screen. The results show an improvement in classification accuracy when compared to conventional light microscope images, indicating that the lighting choice was a good one. Further improvements may be possible, now that the system is operational.

Speckle is a phenomenon of coherent light, where reflection from surfaces with features in the order of the wavelength of the light, cause the reflected light to add constructively or destructively depending on the surface distances from the light source. This causes bright and dark spots in the image. Coherence of light is a continuum from coherent, as with laser light, to incoherent, as with diffusely reflected light. Directed light from a point source through a collimator lens, as used in many optical microscopes, forms light that is somewhat coherent, but not as coherent as laser. The lighting for AutoStage, being sourced from a wide scattering filter, is very much at the non-coherent end of the continuum.

5.1 Dark Field Illumination

Further research and experiments on lighting found that simply blocking the path from the light source directly into the objective lens, and adding a diffuser to allow light to "divert" to the object, created a simple form of dark-field lighting that increased contrast. Contrast is next to resolution in importance to imaging. If resolution is adequate but contrast is not, the distinction between two points may not be made as their grey colour level will be close in value.

An unexpected effect observed was the merging into the background of some detritus, as shown in Figure 5-1. An explanation offered is that opaque objects appear black and merge into the background while transparent objects (pollen) appear much lighter with features that are darkened but without blocking light entirely. Edges of opaque objects will show as the light reflects off them, toward the objective lens.

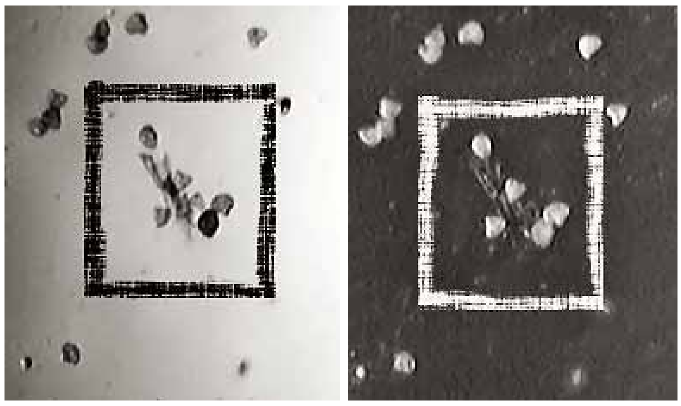


Figure 5-1: light-field and dark-field illumination (respectively) showing pollen grains standing out more amongst detritus in the dark-field image.

Dark field imaging is perhaps better obtained, and still reasonably simple, with more directed side lighting as shown in Figure 5-2. This would improve efficiency of the lighting and reduced heating. It was decided not to spend effort on modifying the lighting further until the system could be tested properly, and a benchmark obtained with which to show a cost effective advantage to the purpose of the system.

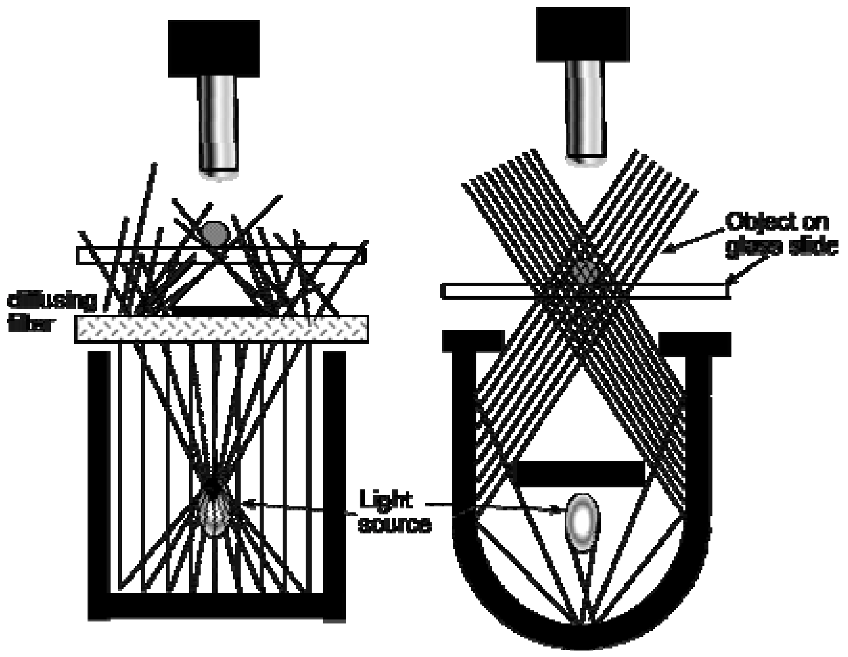


Figure 5-2: Dark-field lighting. Left: implemented. Right: simple alternative - in three dimensions this lighting forms a hollow cone of light with apex at the object and the objective lens inside the hollow.

Dark-field lighting is generally known to, and shown to have better contrast as in [49] where an increase of contrast from 10% to 80% was achieved in changing from light-field to dark-field microscopy and in [48] where, "The microscope uses a halogen bulb light source and a silicon vidicon camera detector. High-contrast images of defects are shown as bright detail on a dark background. For the same view, DFM gives image detail contrast as high as 100%, compared with 25% in bright-field illumination.”

There are many statements of objects smaller than the resolution of the system being “visualised” rather than resolved, as in [4]. In this case an object smaller than the wavelength of light used, appears in the image, although not to scale. Whether this exists in some degree for AutoStage, and whether it is an advantage or disadvantage has not been determined. It would be difficult to attempt to image particles smaller than 1 micron and be sure the resultant image showed those particles and not others – that is to say, ensure an absolutely clean slide and then have those particles only, placed on it. The article cited, [4], was attempting to show that the particles observed in the image were not simply clumps of the smaller particles.

6 The Segmentation of Pollen

Introduction

A segmentation scheme was proposed by Holdaway [34], but the algorithm required user input, was developed for a limited number of pollen types, tests differed from the proposed design due to a lack of background images, and lighting was different from that used here. A new segmentation scheme was therefore developed for AutoStage. Two segmentation algorithms were required to find pollen in images from each of the microscopes, but the basis of the algorithm is the same for each. Holdaway was successful for his image set but had problems with two pollen grains too close together. This is still the case here and "clumping" is an as yet partially unsolved problem. It is suggested, in discussion within the Pollen Group, that the counting of clumped pollen grains is a difficult image processing problem which will require a separate study. Palynologists within the group say that clumping is also difficult to do "by eye" and clumps of pollen are often ignored as too difficult to discriminate.

6.1 Segmentation and Segmentation of Touching Objects

It is stated by Pal and Pal [53] that, "it is known that no method [of segmentation] is equally good for all images…". They review some segmentation techniques including grey level thresholding, iterative pixel classification, Markov random field based approaches, neural network based approaches, surface based segmentation, edge detection and fuzzy set theory based methods. Edge detection was chosen as the basic technique here, as pollen against the dark background form sharp edges that are relatively easily detected. The problem of clumping was avoided by specifying sparsely populated slides, however from results it appears that clumping, and pollen touching other objects, cause a slight reduction in counting compared to manual counting. The Hough transform is likely to be a useful tool in later developments in sorting pollen in clumps as its ability to locate circles of certain radii could be used to pick out the pollen grains within one large blob. Some clumping examples are shown in Figure 6-1. As texture is already being successfully used for differentiating pollen then it would be a good candidate for use in clump splitting, although the implemented differentiation is between pollen types and for this problem, differentiation between pollen and non-pollen is required. It seems logical to combine edge and region detection as was done by Narandra [51], however a simpler and more specific method should be sufficient given the somewhat predictable nature of clumps of pollen. If a clump is larger in diameter than the largest pollen grain and/or irregular in shape (that is it would be rejected by the present segmentation algorithm) then it requires the detection of features or regional patterning and once defined as a clump, the edges of individual grains within that clump need to be detected.

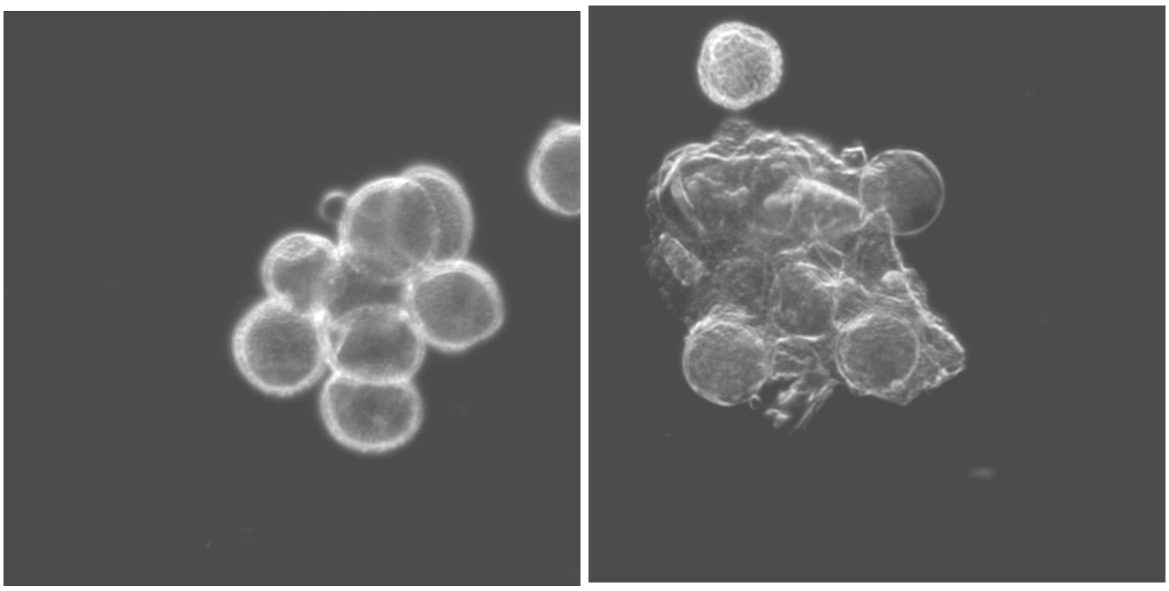


Figure 6-1: Examples of clumping. The image on the left shows clumping with overlapping and on the right, pollen grains overlap with detritus. The translucent nature of the pollen is apparent in these images.

6.2 The Segmentation of Low Magnification Images

The low magnification microscope images are varied in nature between images and across individual images. They may contain a black background with white pollen, added detritus, large white areas of the wax used to seal cover slips, or cover slip edges etc. Some examples are shown in Figure 6-2.

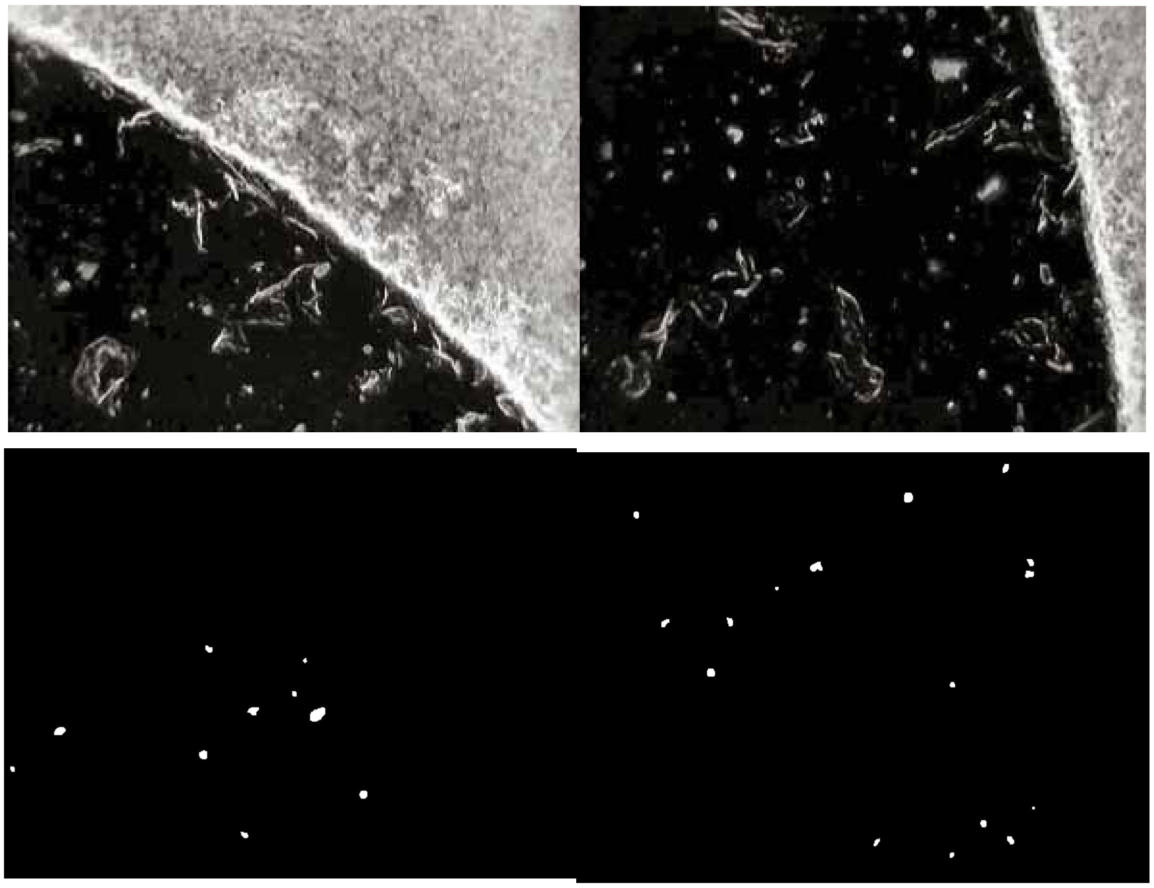


Figure 6-2: Two low magnification images and their segmented image below, from a series of images of a slide of fossil pollen from a core sample taken from Easter Island by Prof. John Flenley.

The wax seal required an added test for black object on white background as the speckled nature of it found many edges with many of those being of similar shape and size to pollen.

Details of the algorithms are found in the software description in §11.1.12.2.4. Essentially, the algorithm subtracts a background image, finds edges, joins any breaks in what may have been a connected edge and fills any resulting shapes enclosed by edges to create solid "blobs" formed in a binary image. A series of tests for size, shape and relative background intensity eliminate many of the blobs. Examples of blobs are seen in the segmented, low magnification images of Figure 6-2.

A background image for the system was taken without a slide in place so it could be subtracted from subsequent images to remove any constant effects from lighting, lens and sensor.

The image processing tasks to achieve forming blobs are a canny edge detector, dilate, fill then erode. The edge detector requires a threshold of the image. The threshold value varies from image to image but changes drastically when a large portion of white such as the wax seal appears in an image. To counter this unwanted effect the threshold was taken from the background image so it is constant across all images and is effective in finding edges for objects above the background intensity. The effect of this change is shown in Figure 6-3, between image numbers 2 & 3. Image 2 is missing edges for a large number of the less intense pollen grains in the image.

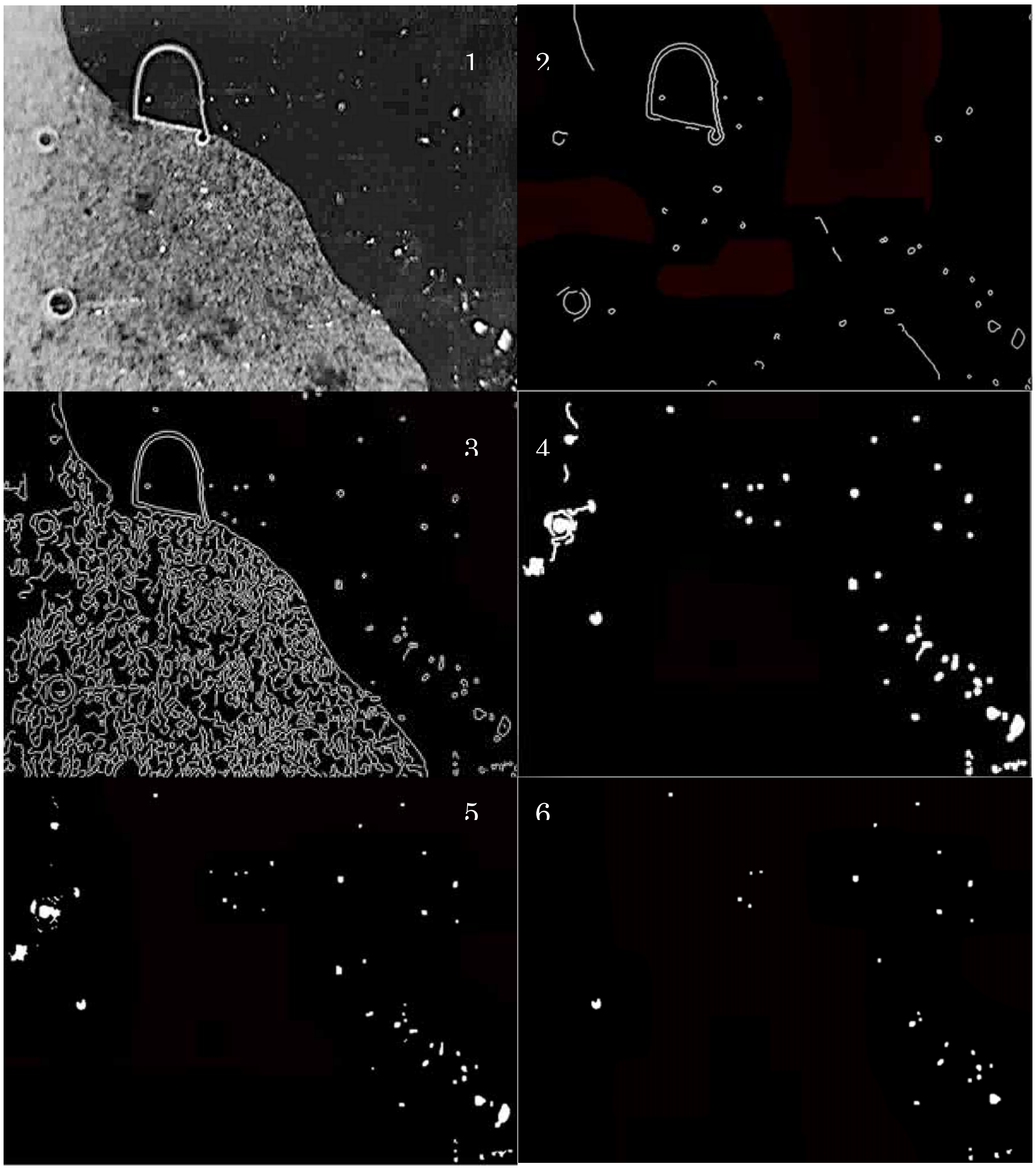


Figure 6-3: Images of segmentation sequence. Top left to bottom right are:
1. **original image, with the wax seal covering the left half**
2. **edges using the original image to calculate threshold – note missing pollen**
3. **edges using background image to calculate threshold**
4. **edges dilated, filled and the borders cleared**
5. **erosion brings blobs to original object size and smoothes edges**
6. **final objects are found with some objects removed by various tests described in the text**

The ‘fill’ checks each black pixel in the binary image and if it can not find a black pixel path to the edge of the image, it changes all black pixels in the necessarily edge-enclosed area to white, creating the blob. The edge was dilated to join the gaps but it increased pixels in all directions making the final blob larger than the original object, so an erosion is performed to reduce the blob size to about the size of the object in the original image. This also has the effect of smoothing the edges of the blobs.

After blobs are created they are tested for the likelihood of being a pollen grain. The number of pixels in a blob represents area and is used to discard blobs that are too large or small to be pollen grains. A bounding rectangle around each blob is tested for width to height ratio and if found too large or small, the blob is discarded as being too elongated in an X or Y direction. The area of the bounding rectangle compared to the area of the blob is used to deselect more objects that are elongated in a 45° direction. A convex hull is created for each blob and the ratio of its area to the area of the blob is used to eliminate any object that is essentially round but with its boundary making excursions in toward its centre.

Some more sophisticated methods of segmentation were trialled on the images, for example, watershed and Hough transform methods. A reference text used, [25], allowed an internet download of a Matlab toolbox, “DIPUM” which was useful for implementing these algorithms. It was found that the more complex or sophisticated the algorithm, the more selective or specific it was and would not as easily adapt to the variety of tasks required across a variety of slides.

A slide preparation prescription has been proposed as although segmentation is successful for pre-existing slides (examples of which are those shown in Figure 6-2), it is more robust for slides prepared in the prescribed manner (§3.2.1). The prescription is deliberately as similar to present practice as possible so the integration of AutoStage into a laboratory would be seamless.

6.3 The Segmentation of High Magnification Images

The high magnification microscope is driven to the location of each pollen grain found in the previous segmentation, so that its location is ideally central in the image. Rounding of the conversions from pixel-size to motor-step-size, and tolerances in the movement of the XY-stage, cause the pollen to appear with an offset from the centre of the image. If other objects are in the central field of view, then it can be unclear which one is the pollen target, so a second segmentation is performed. The increase of information in the higher magnification image enables the elimination of more non-pollen objects. The image size is reduced to a size within which the intended object is expected to appear given all tolerances. The size reduction reduces computation time. The size was determined empirically as 500x500 pixels, centrally located within the 1024x768 pixels of the full image. The central location of the reduced image size suggests that there is little offset from centre of the mean of the positions of all objects. This is because it is a simple matter to adjust the software to correct for any such offset.

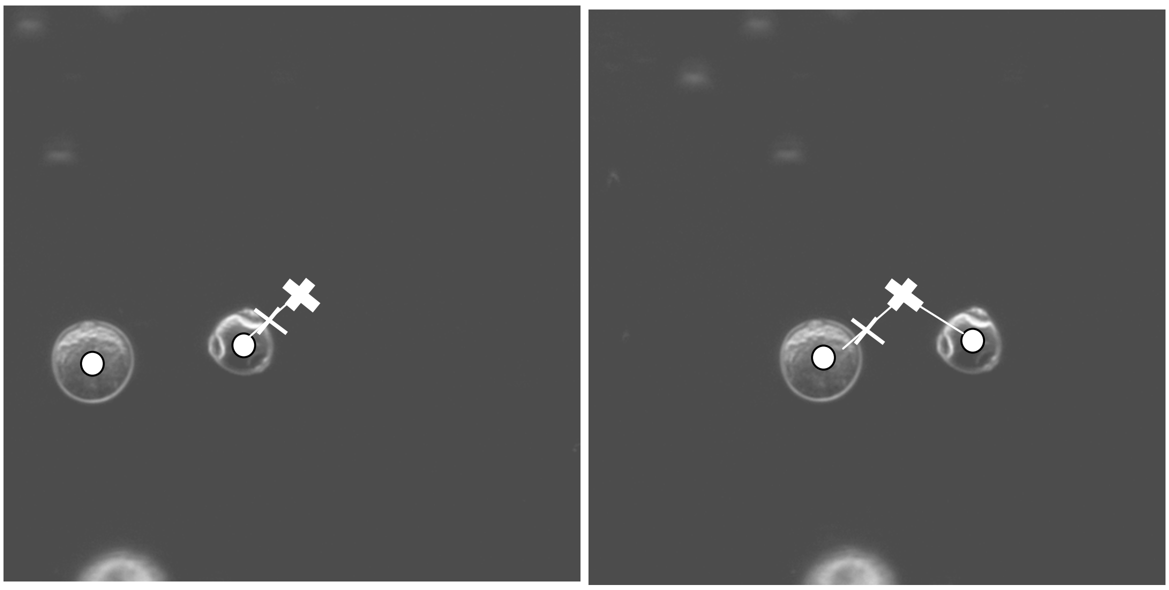


Figure 6-4: Illustration of the same pollen being selected twice. The left image is evaluated first, then the right image. The larger pollen is the target in the first image and the smaller pollen is the target in the second. The target pollen grains are tending to appear below and left of centre of the image. If the centre, or position at which a pollen is expected, is altered to be between the last found pollen and true centre (shown at the narrow cross) then the correct pollen is more likely found each time.

A pollen grain may be imaged twice if it is close to another. When the second pollen is to be imaged, its offset in position may cause the pollen already captured, to be imaged again, as indicated in Figure 6-4. To overcome this problem, two processes are implemented. The first process moves the expected position of each pollen from the centre of the image, to a pseudo-centre determined as some distance between the centre of the image and the centre of the last pollen grain found. This is effective because the offsets from image centre of successive pollen grains tend to drift slowly around the centre rather than jump randomly. In the second process, each valid pollen grain found has its global position on the slide stored and compared to the locations of all previous grains found. The pollen is discarded if its location matches any other within an adjustable tolerance. If this location match discards a pollen grain, then the segmentation algorithm continues to look for the next closest pollen to the pseudo-centre of the image. If all objects in that particular image are discarded, the slide is driven to the next location.

The "blob-finding" algorithm for high magnification images is in principal the same as for low magnification images (see §6.1) with the parameters changed to suit the differences between images.

Once shape and size have eliminated some of the blobs, the nearest to the pseudo-centre, that has a slide location not found previously is located, as described above. The final selection is cropped to slightly larger than its bounding rectangle and saved in a folder for classification.

7 Features Extraction and Classification of Pollen

Introduction

The contents of each image are represented by 43 numbers representing 43 features of that image. Classification is performed by an artificial neural network. The neural network is trained using features from images of pollen grains that have been identified by trained palynologists. Once pollen images are captured from a slide of mixed pollen types, the extracted features are classified into groups of pollen taxa by the trained neural network. The number in each group is the resulting count of the slide for each pollen taxa. The associated images are displayed on the computer screen thus each grouping can be viewed by the user and the count adjusted if required. In testing the classification process, errors ranging between three and twelve percent were found.

7.1 Pollen Features

Feature extraction is performed to reduce the data, in this case in an image, to the necessary and sufficient amount to perform the classification. Having too much data input to a classification system may reduce effectiveness [6]. Too much data invites *the curse of dimensionality* [5].

A suitable set of features was identified and reported by Zhang [74], and published in 2004 [75]. Zhang himself used empirical methods to reduce the feature set to about 12 features.

The features, and numbers of each (in parenthesis) are:

- Geometric features: area, circumference, compactness (3)
- Histogram features (2)
- Second Moment features (7)
- Grey Level Co-occurrence Matrix, GLCM features (5)
- Co-occurrence Matrix GGCM features (12)
- Gabor features (8)
- Wavelet features (6)

There are a total of 43 features. Essentially there are some shape and statistical features but the Gabor and Wavelet features are the textural representations which is considered the main method of distinguishing the pollen grains.

Textural features used are based on orthogonal wavelet decomposition and Gabor transforms [74]. The wavelet transform is a joint spatial/spatial-frequency method [45] and achieves high resolution in these domains. These transforms are consistent with recent theories of human vision [12]. Two-dimensional Gabor functions are local, spatial band-pass filters. Textural analysers implemented using Gabor transforms produce a strong correlation with human segmentation [57].

The Zhang feature set was known to have elements of redundancy, so a study was performed at Massey University by Etheridge [13], to determine the effect of a reduced set of features by comparing classification results with a Linear Discriminant Analysis (LDA) of the data and then seeing if LDA could show any possible reduction in features. Sets of highly correlated features were determined. All but one feature from the correlated sets were removed. Consequently, the full set of 39 features was reduced to 24 with discrimination errors of 10 using the full set rising to 14 errors using the reduced set. It was decided that, as the computational effort required for the full set was not excessive, the gain in computational time did not warrant even a slightly increased error rate. Another result was that the classification rate of an LDA was not better than the Multi-layer Perceptron neural network.

Principal Components Analysis (PCA) was performed to again evaluate feature set reduction and found only one or two features removable. Classification using features sets reduced using PCA results, found a reduction in accuracy not considered worth the saving in computational time, so it was decided to maintain all 43 features in the system.

Now that a complete functioning system is available and evaluated, it may be used as a benchmark for any future modifications. The performance of any modified or new feature set may be compared to the Zhang set. It is likely there will be some benefit to matching the feature set to a particular discriminatory task: for example, when classifying grass pollen that are very similar in appearance.

7.2 Classification Using an Artificial Neural Network

Zhang tested his feature set [74] using a Multi-Layer Perceptron neural network (MLP) to perform classification of images with good results. The MLP is used in this project. Support Vector machines (SVM) are considered useful for similar tasks with the advantage of not being subject to over-training [59]. Over-training is explained in the next paragraph. An SVM, with its binary decision mechanism, was trialled on two grass pollen image sets and compared to the MLP for the same two pollen image sets. The MLP scored a little over 90% correct classification while the SVM scored 80%. The MLP is used for the system as results are better than SVM. It is considered that any major work on comparisons is better done once the prototype is complete and whole-of-system trials are possible to compare new concepts.

A neural network can be configured to work optimally on a particular data type and the discussion below describes how the Netlab algorithm used is optimised for pollen image features used in AutoStage. Netlab is an MLP algorithm written by Ian Nabney [50] and available under the GNU licence and freely available on the internet. The help file is reproduced in (Appendix D).

Over-fitting and under-fitting can be a problem when trying to describe data that is a sample of some population.

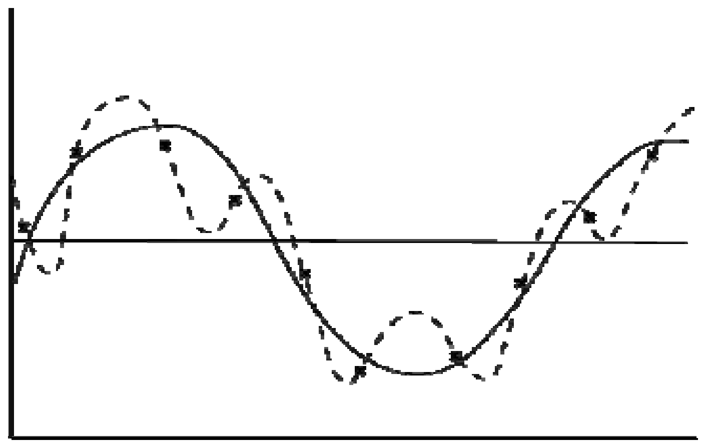


Figure 7-1: Under/over-fitting: the example data population is sinusoidal. The data can be fitted with a straight line, a sinusoid, or a polynomial of sufficient order to cut every point exactly. The polynomial models any noise present and is thus "over-fitted", as it does not represent the population data as well as the sinusoid.

In Figure 7-1, the sampled data can be represented by increasingly complex functions: a straight line, a sine wave, or a 10th order polynomial. If the data population was in fact sinusoidal, with noise causing random variation in the data, then a straight line would not contain enough information to describe the population well, a sinusoid would almost fit and a 10th order polynomial might fit the sample data exactly. The straight line would lack information and the polynomial would contain added information about the noise in the particular sample. With neural networks, repeated training causes "strengthening of the synapses", or the weights are more finely tuned to the particular training data. If training is stopped too early, the weights are not adjusted well enough. If training is continued too long, the weights can become too finely tuned and when used for recognition of new data, may reject it for being too different from the training data. So for neural networks over-fitting is a result of over-training. Under-fitting is the result of "early stopping", or under training, of the neural network. The number of hidden nodes affects the number of weights and therefore the training effort per training epoch. The number of training epochs and the number of hidden nodes need to be determined for good classification results. They are determined by verification of the neural network using exemplar data, in the case of AutoStage, this data is features extracted from images of pollen that have been identified by experts. A problem presents itself here in that the exemplar data has been chosen by humans and so our training can only be as good as the data presented as "known".

Once a reasonably sized data base of images from the AutoStage was compiled, 25% of the images were set aside for final classification tests, results of which are reported in §8.1. The remaining 75% were used for training the neural network for those tests. The 75% training image set was again split into training and verification sets for running checks to determine which parameters were optimal for the task of classifying AutoStage images. How the number of epochs, number of hidden nodes and number of training data were determined, is discussed in the next three paragraphs.

The number of network training epochs is limited in the Netlab [50] MLP algorithm when the activation error gradient reaches zero, or when a maximum training count, or 'epoch' number, is reached; whichever is first. A manual method of determining when overtraining occurs was trialled by running training loops and evaluating the classification result after each loop. More training reduces classification error until over-training occurs at which time the classification error begins to rise as shown in Figure 7-2.

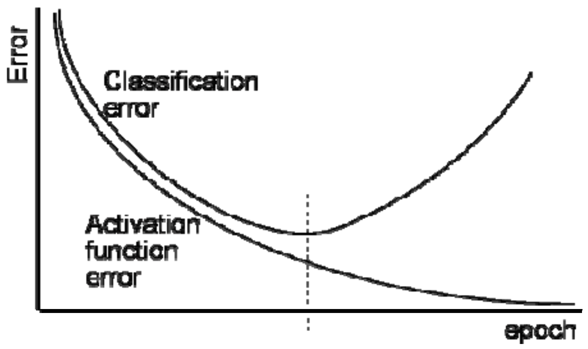


Figure 7-2: Optimisation between early-stopping and Over-fitting

After a series of trials using the type of data expected, an average overtraining point could be determined. The number of epochs required was found to vary from 80 to 150 within the range of pollen type quantities expected: 3 to 50. The variation in classification error over that range of epochs was not great, so a fixed number of epochs could be used. For the results presented in this report, the training number of epochs was fixed at 80.

The number of hidden nodes required was found by verification trials using the verification sets of data. Classification runs on 100 images using the remaining 25 for verification tests were performed varying the number of hidden nodes until the success rate was maximised.

The number of training images required was found by trials varying the numbers of images for training and checking the classification result. The data bases collected included 200 images. Of these, 50 were set aside for final testing. This left 150 for training and verification. Varying portions of the 150 were used as training and verification sets and the classification results recorded. It was found the more images used, the better the result, so 150 images were used for training the neural network for the final testing. This quantity is also required for practical use of the machine as too many training images would be a waste of resources and time. There was not a large change in accuracy when using between 120 and 140 training images so it is thought that the practical limit is close to the 150 training images used for the final tests.

The MLP neural network implemented functions optimally with the input data sets all with inputs compressed to between minus-one and one. This fits the data into the same range as the sigmoid activation function used (see Figure 2-6 in §2.4). To achieve the data compression, the features matrix has its column data normalised; that is to say transformed to have a mean of zero and standard deviation of one. This produces data marginally outside of the (-1, 1) limits. Tests showed that it produced results slightly better than fitting the data exactly into the (-1, 1) limits. The parameters required to normalise the data were saved and will be used to transform any new data. That is, if another pollen type data base is added to the present system, then the features extracted will be transformed using the saved parameters. In that way the data is very close to the values they would have been if they had all been transformed together.

7.3 AutoStage Reports

The data reported by AutoStage is simply a count of each pollen type found on a slide. The training of the neural network predetermines the pollen types expected. The neural network used does not have the ability to detect novel data, so all images are classified into one of the predetermined groups. An additional pseudo-pollen type is added by imaging common detritus found and forming another group. The pollen images are displayed in their groups as classified, and a trained operator selects any pollen grains that have obviously been misclassified and alters the count accordingly. Some incorrectly classified pollen grains stand out well amongst the others in an image matrix so the count can be improved by a quick manual adjustment. The display of image groups is useful for giving confidence to the palynologist that the classification and count is a reasonable one. As location data of images is stored, it is possible to adapt the program to move the slide back to an image of a particular pollen grain for further identification by the user.

8 Testing and Comparison with Experts

Introduction

Testing was performed by:

1) testing the classification system on various image sets. The classification system consists of the features set and neural network. It is tested by compiling three data bases of images to train and test neural networks using the features extracted from the images.

2) comparison of the complete system results with results of classification and counts by palynologists. The complete AutoStage system is verified by classifying and counting four slides four times each, and comparing the results to the classification and count of the same slides by five trained experts.

To test the system, a ‘known’ slide might have been manufactured and counted by the AutoStage to measure accuracy. Two problems with such a slide are ensuring that it is representative of many ‘real’ slides, and knowing absolutely what is on it. It is very difficult to know for certain what is on a slide at the microscopic level given the variation in counting by “experts”, as shown in the results of the experiments performed here. There is, at present, no other known method of analysing slides that would be accepted as “better”. A manufactured slide would not necessarily be representative of a prepared pollen slide and therefore not a suitable test.

It was decided that the best way forward was to compare the classification and counting of the AutoStage with that of humans. If the means of a number of counts are the same and the variances the same or smaller, then we could assume the machine would do as well as humans and the advantage would be in the time saved. Until a method of knowing the quantity of a variety of pollen on a slide is found, the aim will be to match human accuracy and do better than human variance.

8.1 Classification tests

A series of classification tests were performed on the three image sets from the three data bases of images:

1. AutoStage images
2. conventional microscope images
3. “Bangor” images from a data base used by France et al. [19].

The aim, description and results of the tests are given in each case. The AutoStage and conventional microscope images are captured from the same reference slides of known pollen. Only the capture mechanism was different. The AutoStage images were captured automatically, thus including the image background size, lighting, resolution and focus, as factors in the testing.

Results are presented as the total correctly identified pollen grains in the test, as a percentage of all pollen grains in the test. Five tests were performed in each case and the five results are shown in tabular form.

8.1.1 Comparing Stained with Unstained Pollen

8.1.1.1 Aim

Staining is common in conventional microscopy as a means to highlight features and making the pollen grain stand out against detritus that usually does not take the stain. It is not certain that it is useful for AutoStage and is an additional preparation step that might be left out. Classification is compared for conventional microscope images that are both stained and unstained. The aim is to see if the staining process may be removed from slide preparation.

8.1.1.2 Description

29 training images and 10 test images of stained pollen and of unstained pollen of the same 6 pollen types are used in the classification process. Five tests each of stained and unstained are performed.

Next a mixture of stained and unstained pollen images, of the same six types, was classified to determine if the distinction made by staining was sufficient to discriminate between the same pollen types.

8.1.1.3 Results

Stained Slide Results:

Test #	1	2	3	4	5
% correct	95	95	95	96.7	96.7

Unstained Slide Results:

Test #	1	2	3	4	5
% correct	93.3	93.3	93.3	91.7	93.3

Mixed Stained/Unstained Results:

Test #	1	2	3	4	5
% correct	74.2	69.2	72.5	70	72.5

8.1.1.4 Discussion

Comparing the stained and unstained results in the first two tables shows a consistently better result using stained pollen indicating staining has a positive effect for classification.

Regarding the third table of results above, as fifty percent of the data are stained/unstained, then we could expect a fifty percent error rate if the each set of stained/unstained image sets were otherwise randomly assigned to the two groups of the same pollen type. As there is also generally between two and ten percent error between groups of different pollen types then the expected success rate should be less than fifty percent. The seventy percent result indicates there may be some discrimination of the stained/unstained grouping. The low success rate of about 72% indicates that staining does not differentiate the pollen enough to be discriminated well by the classification system.

All other testing and evaluation exercises in this project were performed using non-stained pollen. A confirming test should be performed using AutoStage images and if the result supports the conclusion here, then the decision to use stained pollen for the system should be made.

8.1.2 Comparing Conventional Microscope and AutoStage images for classification

8.1.2.1 Aim

The aim here is to compare the discriminatory accuracy of the classification system used by AutoStage, on conventional microscope images as compared to images captured on the AutoStage. The images of all seven pollen types used in this test were captured from the same reference slides.

8.1.2.2 Description

40 training images and 10 test images of 7 pollen types were captured from a conventional microscope and the same reference slides were used to capture images using AutoStage. The images were classified and counted.

8.1.2.3 Results

Conventional Microscope Image results

Test #	1	2	3	4	5
% correct	94.3	92.9	94.3	94.3	94.3

AutoStage Image Results:

Test #	1	2	3	4	5
% correct	98.6	97.1	100	97.1	98.6

t-Test: Paired Two Sample for Means where the second sample is the result in the previous test §8.1.2.3

Excel t-test	*94.286*	*98.571*
Mean	93.92875	98.21425
Variance	0.51051025	1.87044225
Observations	4	4
Pearson Correlation	0.522188645	
Hypothesized Mean Difference	0	
Df	3	
t Stat	-7.348468778	
P(T<=t) one-tail	0.00260393	
t Critical one-tail	4.540702858	
P(T<=t) two-tail	0.00520786	
t Critical two-tail	5.840909309	

8.1.2.4 Discussion

The mean accuracy over five tests for conventional microscope images is about 94% and for AutoStage images, 98%.

For the t-test that the means are equal at the 0.01 significance level, we reject the null hypothesis that the means are equal.

The conclusion is that the AutoStage images are significantly better for classification with the system of features and neural network used here.

8.1.3 Large Pollen Type Count Using Conventional Microscope Images

8.1.3.1 Aim

The aim here is to test classification of a large pollen type group. The conventional microscope data base has more pollen types than for AutoStage images, however the AutoStage data base has more images per pollen type so both are tested separately.

8.1.3.2 Description

All pollen types available with more than fifty images available, but including only one of the seven grass pollen grains, were used to test the classification accuracy. This resulted in a test using twenty-nine pollen types. Forty training images and ten test images were used.

Usually grasses are counted by palynologists as a single type because they are very difficult to discern by eye. Grasses are tested separately in a later test.

8.1.3.3 Results

Conventional Microscope Images of Forty Pollen Types

Test #	1	2	3	4	5
% correct	78.6	77.6	78.3	80	78.3

8.1.3.4 Discussion

A 78.5% mean is not adequate for the system. In this test, the restricted number of images per pollen allows for less training images so the result will improve if 150 training images are used. Using the conclusion from AutoStage/conventional microscope comparison in §8.1.2, it can be concluded that AutoStage images will improve the result further. Previous tests indicate that this would be improved by using larger numbers of training images and AutoStage data-base images. The next test, 8.1.4, further defines AutoStage limitations.

8.1.4 Large Pollen Type Count Using AutoStage Images

8.1.4.1 Aim

The aim here is to test classification of a large pollen type group.

8.1.4.2 Description

Nineteen pollen types were tested using 150 of each type for training and 50 for testing. All available pollen types from the AutoStage data-base were used except only one of the three grasses available was included.

8.1.4.3 Results

AutoStage Images of Nineteen Pollen Types

Test #	1	2	3	4	5
% correct	89.1	89.4	88.5	89.5	88.3

8.1.4.4 Discussion

89% is possibly only just acceptable but not a great result. It can be seen that the classification system performs worse with larger numbers of pollen types. This is a subject for future work and a suggestion for overcoming this is made in §9.3.6.

8.1.5 Comparing Results from a Separate Project Data-Base

8.1.5.1 Aim

The aim is to compare the classification system using images from a data base of images used in a separate project: France et al. [19], and to compare results reported in that project using three pollen types from their data-base.

8.1.5.2 Description

Images available on the internet from a project by France et al. were classified by the AutoStage classification system. Seven pollen types using 140 training and 50 test images from the data base were used to check overall classification results of images captured outside of this project.

Then a more direct comparison of three pollen types against the results reported by France et al. [19] for the same three types. France et al., recorded results using 3 pollen types from their data base with 60/60/84 images made available on the internet. Here, 45 of each set of these images were used for training and 15 images for testing. Validation was not required as the same neural network configuration determined for AutoStage images was used.

8.1.5.3 Results

Results for Seven Pollen Types From a Separate Data-Base

Test #	1	2	3	4	5
% correct	93.4	94.3	92.9	92.9	92.3

Results for Three Pollen Types From a Separate Data-Base

Test #	1	2	3	4	5
% correct	93.3	95.6	95.6	95.6	95.6

This is compared to France et al. achieving 82%: 3% misclassified and 15% rejected by the classification scheme.

8.1.5.4 Discussion

Independent images used in an entirely separate project give reasonable results at 93% accuracy. A comparison with AutoStage images can not be made directly as the pollen types are different and may be an inherently easier or more difficult data set for the classification system. Training and test numbers however are similar as are the number of pollen types used for tests on AutoStage and conventional microscope images used in §8.1.2.

France achieved overall 82% correctly identified in the final classification stage with 3% being misclassified and 15% being rejected. The AutoStage was, on average, 95% successful in distinguishing 15 of the same images with 5% misclassification.

8.1.6 Classification of Grass Pollen

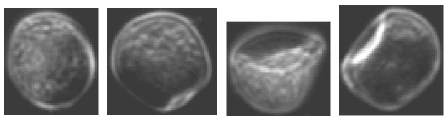
8.1.6.1 Aim

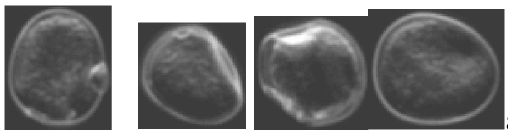
Many grasses are very difficult to distinguish by eye and often they are simply counted as one type by palynologists. The aim here is to test the classification system for accuracy in classifying different grass taxa using AutoStage and conventional microscope images. As more grass types were available in conventional microscope images than from the AutoStage data-base, both were tested.

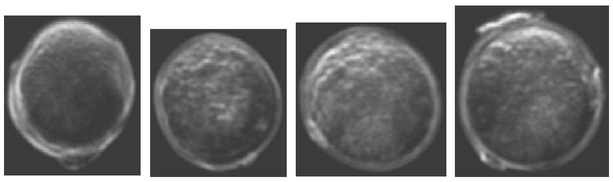
8.1.6.2 Description

Seven pollen types of grass were classified using 30 training images and 10 test images from the conventional microscope data-base.

Using AutoStage data-base images, three grass types were classified. From this data-base 150 training and 50 test images were available. Examples are shown below:


a selection of *brown-top* grass images


a selection of *cocksfoot* grass images


philaris grass images

8.1.6.3 Results

Conventional Microscope Grass Images – Seven Types

Test #	1	2	3	4	5
% correct	87.1	84.3	82.9	81.5	80

AutoStage Grass Images – Three Types

Test #	1	2	3	4	5
% correct	90	90.7	90	90	90

8.1.6.4 Discussion

The result for conventional microscope images is understandably low as the grasses are very similar in appearance and a low training number is available, however given the difficulty of the task the result is very promising.

Again, the results are very promising considering the difficult task. There remains the opportunity to develop features that are specific to classifying grass pollen. Grasses may be filtered out as one type and then reclassified separately. This would reduce numbers of types which has been shown to improve classification results.

8.2 A Complete System Test

8.2.1 Four Slides Classified by AutoStage Compared to Classification by five Experts

8.2.2 Aim

To test the complete AutoStage system, a practical test was performed operating the AutoStage as it is intended to be used in service. The aim was to operate the AutoStage as intended and verify a count of slides against counts by a method acceptable to palynologists as sufficiently accurate.

8.2.3 Description

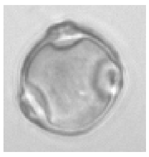
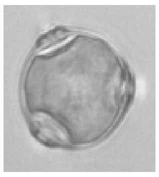
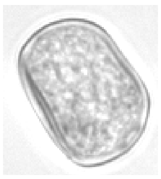
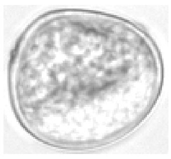
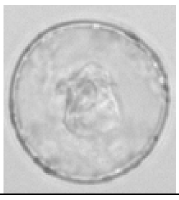
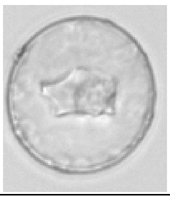
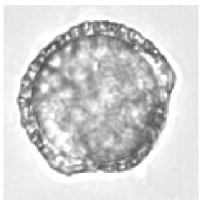
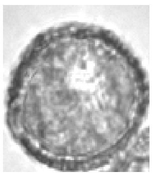
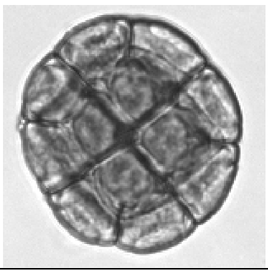
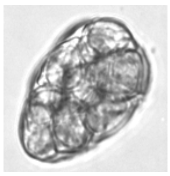
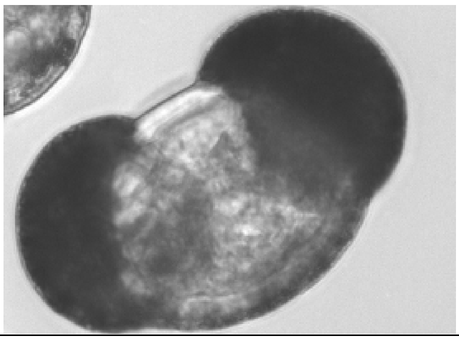
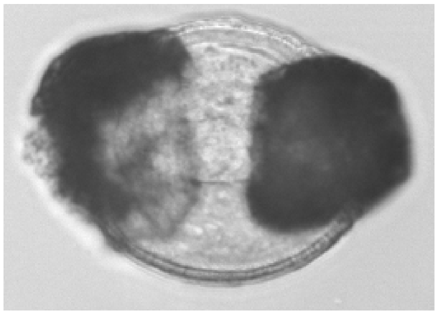
Although there are discussions regarding the efficacy of pollen counting by humans, at present there is no other known method to accurately count pollen on slides. To be statistically measurable, at least four slides, at least four counts by the machine and at least four counts by humans were proposed. Five experts were available as counters, so five human counts were performed. Two professors, a post doctoral researcher, a technician in the palynology laboratory at Massey University and an honours year student heading for a PhD involving palynology were the five selected.

The four slides were prepared by suspending six types of pollen in glycerine. Slides were placed on a warmer and a drop of the suspension placed using a pipette onto each slide. Once any air bubbles had escaped the suspension the cover slip was placed on and molten wax dropped onto the slide at a cover slip edge. The slides were removed from the warmer and left to cool, setting the wax to form a seal. The suspension was supplied by one of those who counted the slides.

The counters were given a test slide to trial and a series of light microscope images showing what the pollen on the slide looked like, along with identifying names. Two images of each of the six pollen types used appear in Table 8-1, below.

A lighting fault occurred during counting of the slides on AutoStage so further counts were performed until four were available. The results from the first four counts which included the lighting fault, made no difference to the statistical inferences and vary small differences to the general appearances of the graphs. The major effect of the lighting fault was a large count of detritus.

Table 8-1: Images of pollen types used in verification testing

					
Betula pendula (silver birch tree) *P36.*		*Dactylus glomerata* (cocksfoot grass) *P119.*		*Cupressus macrocarpa* (macrocarpa tree) P64	
					
Ligustrum lucidum (privet) P148		*Wattle acacia P147*			
					
Pinus radiata P136					

8.2.4 Results

The results are counts of each type found on each slide. The graphs below show the mean and range of each pollen type counted. Results for man and machine are side by side and in the same colour for each pollen type.

The time taken for AutoStage to count one slide was about three times longer than the time taken for a person to count the slides. The focus time of 15 seconds combined with the number of false positives identifying pollen in the low magnification segmentation due to the wax not being solid enough in some areas of some slides were causes.

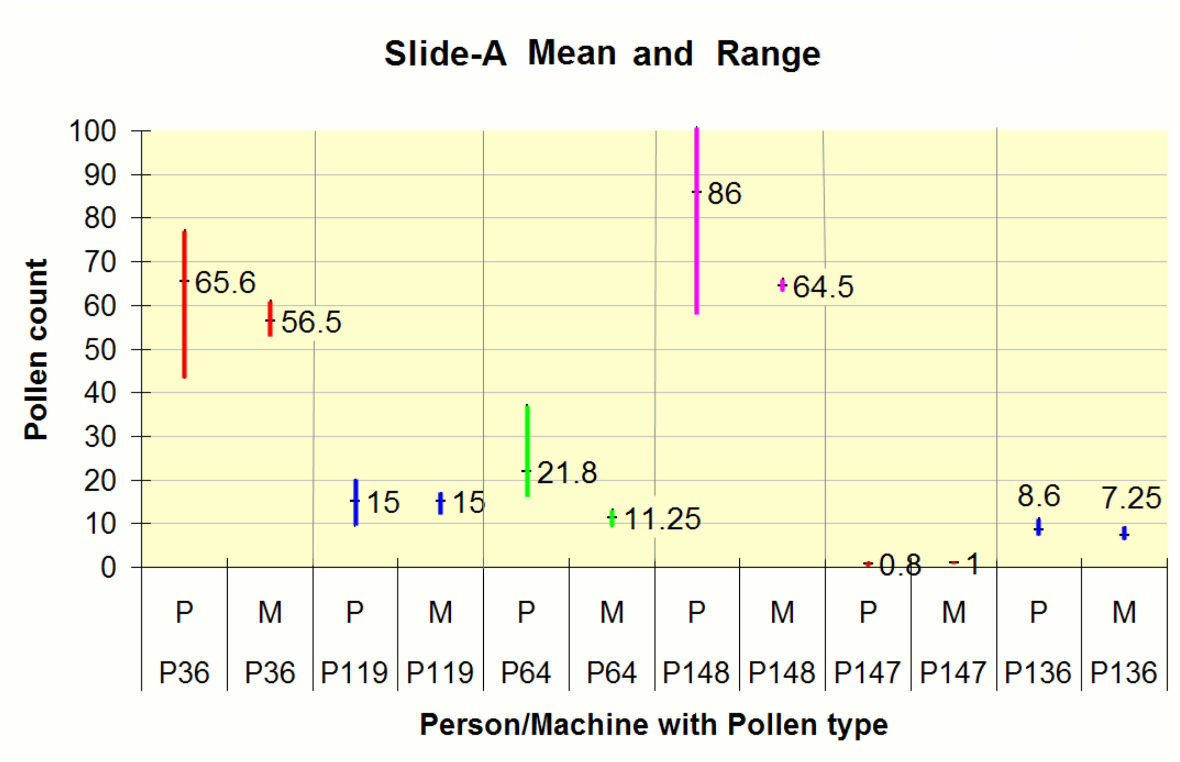


Figure 8-1: Slide-A results

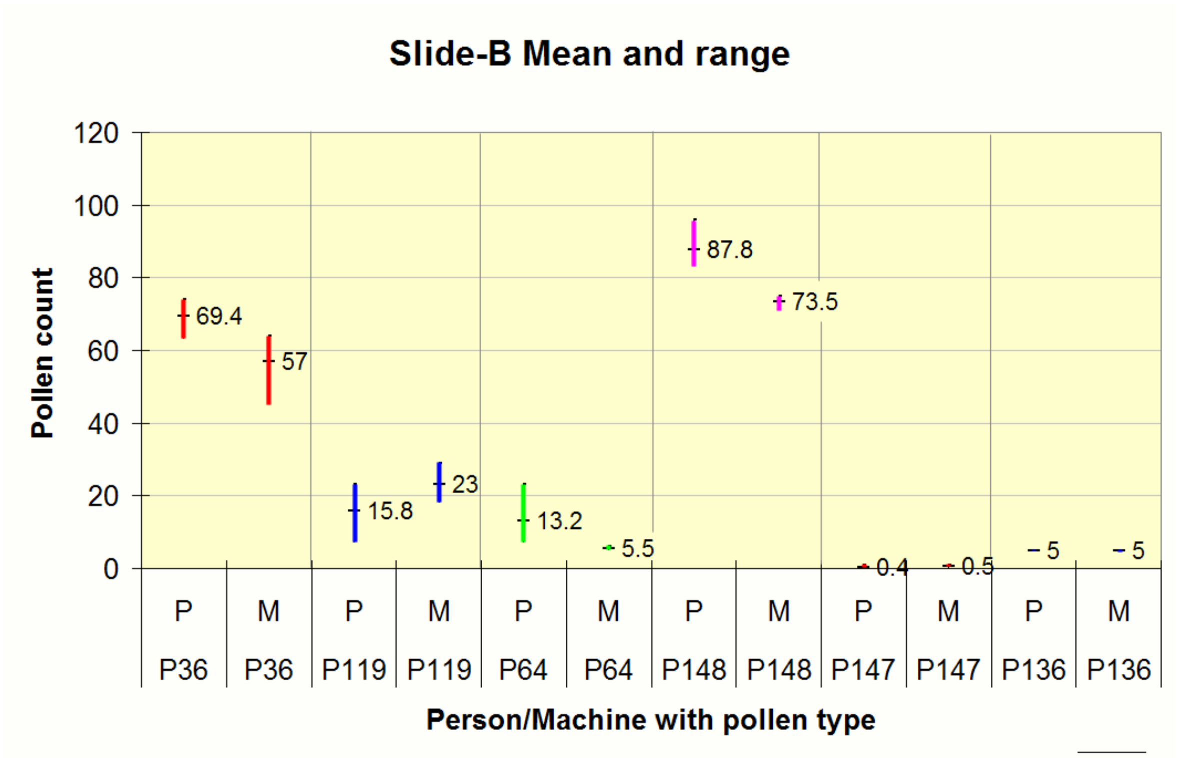


Figure 8-2: Slide-B results

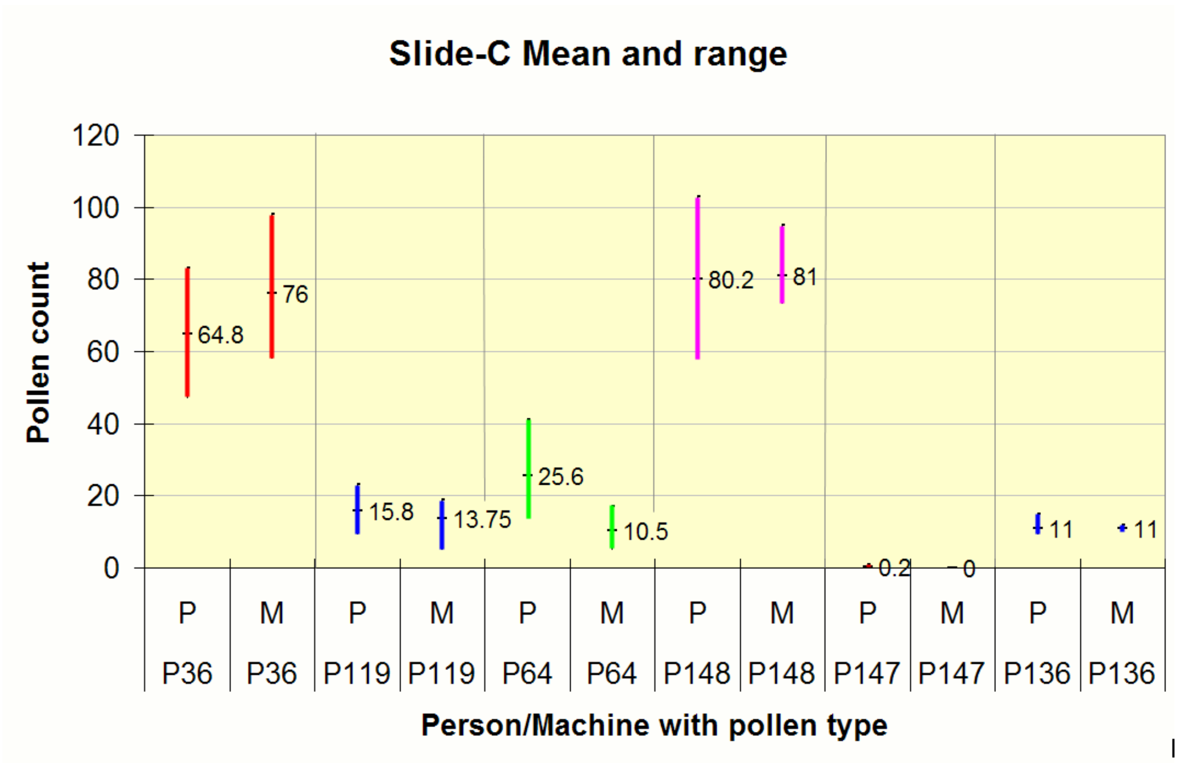


Figure 8-3: Slide-C results

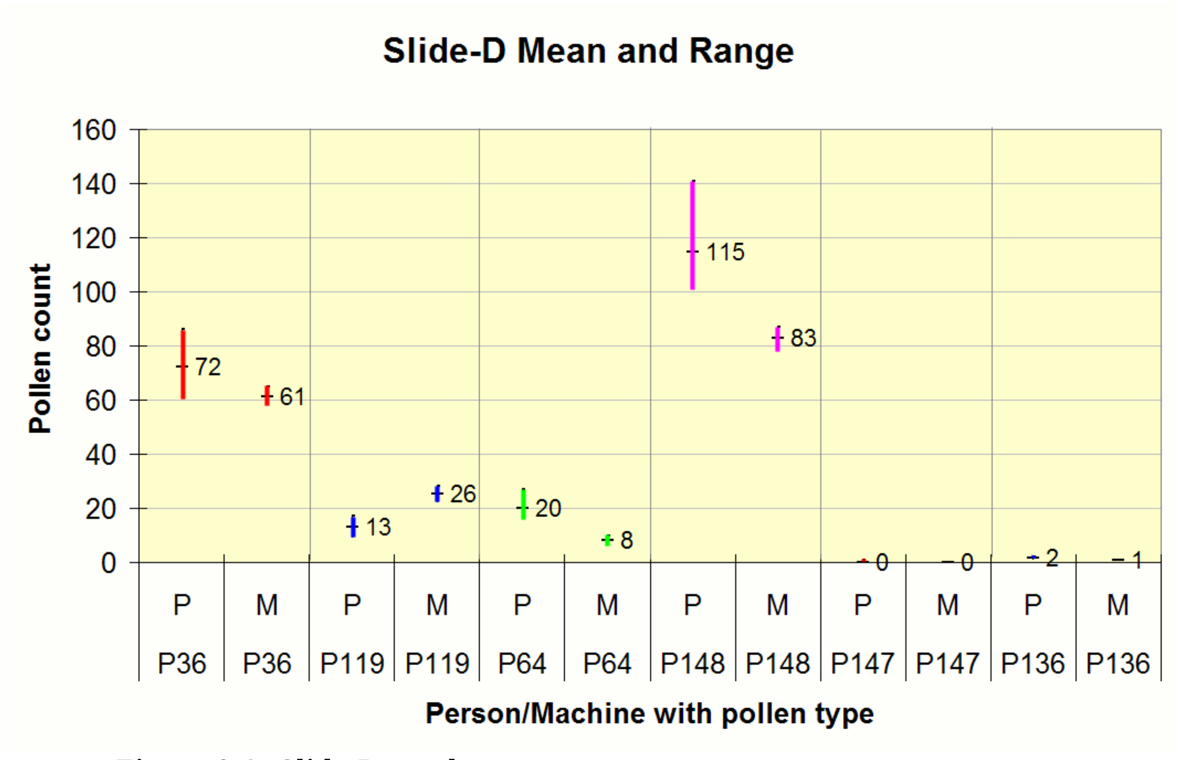


Figure 8-4: Slide-D results

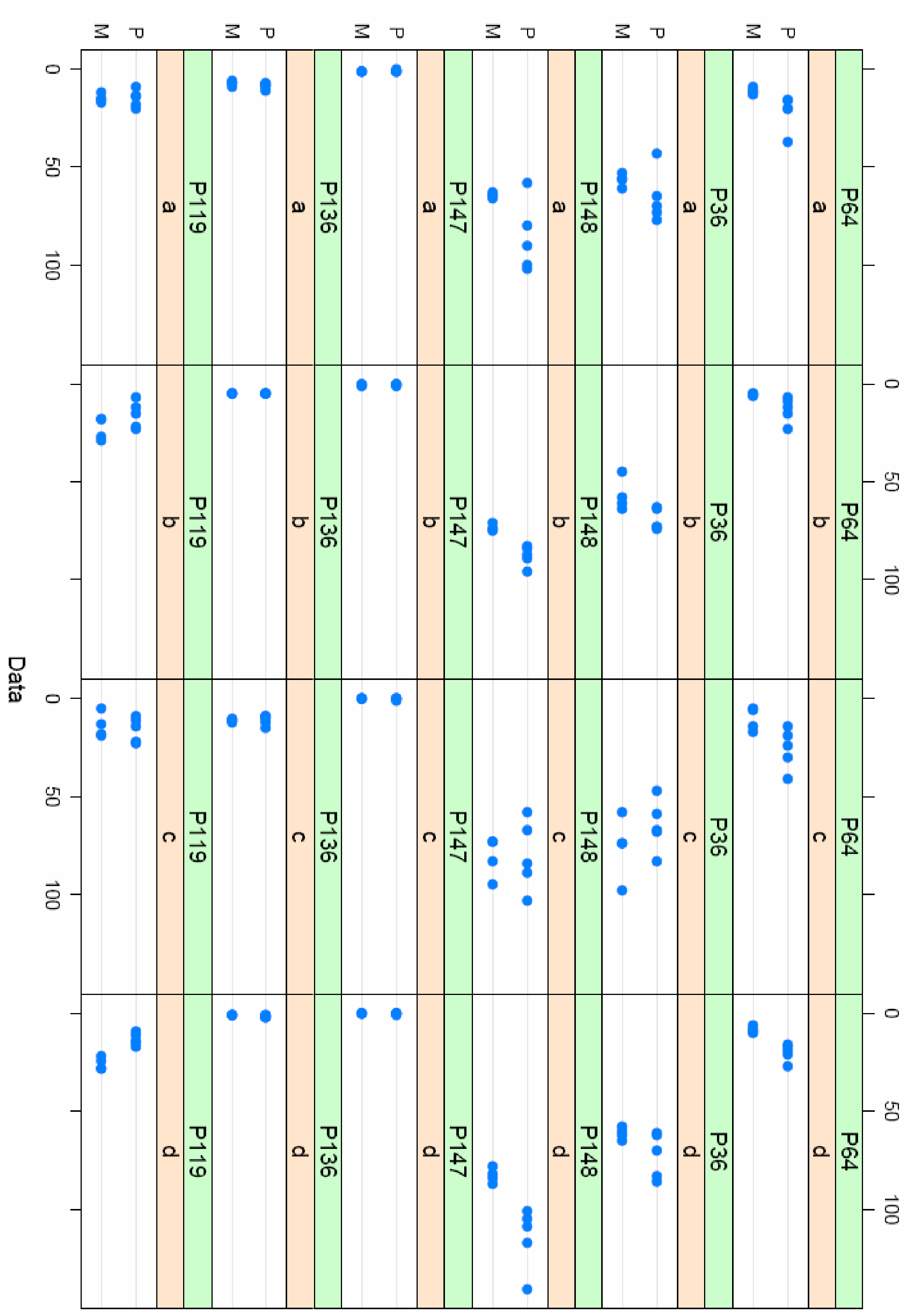


Figure 8-5: Data points for all tests. Slides a, b, c, d are in columns; pollen types Pnn are in rows; X-axis is pollen count; Y-axis is person/machine.

Figure 8-5 was produced by “R” [21], a statistical program available under the GNU general public licence on the internet. Each pair of data points are the counts of P = person and M = machine for slides: a, b, c, d and pollen type Pnnn, where Pnnn is the pollen identification shown in Table 8-1.

8.2.5 Discussion

An analysis of variance, performed also in R, determined that the overall mean of the machine counts was “significantly different” from the overall mean of the counts by experts at the 95% confidence level. The machine tends to count lower than people which is explained by clumping together of pollen grains. The relatively small variance in counting by AutoStage puts the counts mostly within the range of counts by palynologists.

Overall, the conclusion is that the task of automated counting of pollen is possible and AutoStage is capable of counting slides on a commercial scale with results similar to that of trained palynologists. For acceptance of the system it may be necessary to do side-by-side tests using counts from the system and counts by palynologists on an entire palynological project. For palynology the result here appears to be significant with implications of increasing the amount of data able to be gathered, processed and reported and an increase in consistency of reports enabling higher resolution and therefore greater information for a study than was previously possible for a given effort.

Some further work to better finding pollen grains in contact with other objects would improve the results. The system at present relies on the preparation reducing clumping to the level seen in these tests. No image segmentation solution to finding pollen grains in contact with other objects has been implemented so improvements are foreseeable.

The variation in counting is far smaller for AutoStage than for human experts. AutoStage has a major advantage in not tiring of repetitive work and is less likely to make errors of the types that humans will make. It is expected then that comparisons of studies made in quite different locations will be better compared if using an automated system such as AutoStage and standardised slide preparation is implemented.

From the classification tests we can conclude that the classification tool-set used, is an excellent starting point for the system prototype development with classification results comparable or better than those in current literature.

The cost of the system was under $NZ15,000 for prototype parts. It is likely that a unit developed for production could be sold for between $15,000 and $30,000 which would be affordable by a modest laboratory and could save that much in labour costs within its first year of operation. Compared to commercially available microscopes such as that used by Hilsenstein [32] (Figure 8-6) which is valued at about $AUS150,000, the AutoStage would be a very inexpensive option.

Figure 8-6: Olympus BX61. A commercial automatic microscope with digital camera, auto-focus, XY stage movement and slide stacker valued at about $AUS150,000

Development time for reducing the time taken for processing a slide was given second-place status to the main aim of showing the task to be possible. It could be argued that time taken is not an issue as the unit still performs the required tasks, removing the drudgery while trained personnel are engaged in more interesting work. However it is recognised that time taken will be an issue and deserves some serious attention. At present it takes 15 to 20 minutes to capture and process the low magnification images and then 15 seconds per object found to focus, process and capture. At that rate a slide of 500 pollen grains should take about 145 minutes to complete. However, false positives, due largely to the wax being too sparse in places, can double that time. This is mostly a matter of correct preparation; however segmentation improvements may be able to reduce these false positives.

Adding staining to the slide preparation process suggested here may improve results.

Tests for this project have all been performed using fresh pollen with the cellulose shell still intact and containing gamete material so their appearance therefore is different to fossil pollen. Other projects have successfully used MLP neural networks with fossil pollen [41, 43, 44]. Tests should be performed using fossil pollen on the AutoStage classification system explicitly. The segmentation algorithm has been developed by including existing slides of fossil pollen (see Figure 6-2).

The grass tests show a reasonable discrimination of pollen types that are often not done using light microscopy as it is very difficult to distinguish many grasses by eye. The results of this test shows the possibilities of discrimination of features that are not readily distinguishable by eye even under a microscope. This opens up the possibilities of developing features to distinguish other discrimination tasks found difficult by eye and to attempt the further classification of pollen by family, down to genus and perhaps down to species.

9 Thesis Conclusions and Future Work

9.1 Thesis Conclusions

All eight of the objectives of the project, as listed in §1.1, were achieved. The criterion for selection of the objectives was the development of a system sufficient to fulfil the aim.

The aim of this project is considered achieved with the completion of a system that classifies and counts slides automatically with results mostly within the range of counts by expert palynologists. The system shows such promise that the prototype could have immediate, if limited, uses within a palynological laboratory and any shortcomings are clear with evident paths to improvement, as suggested in §9.3.

Specifications have been determined so for any future modifications made, changes in specification can be compared to changes in classification results, making the prototype useful for design of a next generation AutoStage.

9.2 Final State of the Project

The project is at a significant milestone, with a working prototype that is demonstrated to be working and able to be used as a benchmark for further refinements. In summary, from the people for whom the development is aimed, two testimonials are presented.

From Professor John Flenley; palynologist:

"It seems to me that the machine is performing as well as the person, if not better. At a glance, I should not be surprised if the overall data set showed no statistical difference between the two. There may be some systematic variation, e.g. in sample P64, it appears that the person captures more images than the machine. But in P119 b and d, the reverse is the case so this may be random variation. I am particularly interested in the low number counts in P147 and P136. Think of the time saving in scanning these sparse slides by machine and then examining a matrix of images. I conclude that I would be delighted to have such a machine in my laboratory. It could transform the laborious laboratory work of palynology, and greatly increase efficiency."

From Alistair Clement, Massey scholar and palynological student:

“ [AutoStage] has great potential for easing much of the burden of the pollen counting process from palynologists practicing in any area, and is nothing less than an exciting technical achievement. Given the impressive results produced by the project in a very short period of time, I eagerly await the opportunity to go 'hands-on' with the system in the lab.”

9.3 Future Work

Introduction

As this project was to show the viability of the system as an aid to palynologists, there is a vast amount of future work, especially when considering microscopic automation for areas other than palynological endeavours. For improving the system for auto-palynology, some of the possibilities are discussed briefly below. Given software control, and classification sub-system design and training options, variations in system application are numerous.

9.3.1 Output Reports

The output at present is a simple count of pollen types found. The opportunity exists to add a pollen diagram as is often produced by palaeopalynologists showing counts of pollen types for each layer depth. Any further additions to reporting is a relatively simple matter of augmenting the software to accept addition information as inputs from the user and produce a report in almost any way desired.

9.3.2 Focal Integration Images – Improving Depth of Focus

Improving focus can be achieved by integration of focussed parts of a series of images with varying focal planes. As the depth of field is smaller than most pollen grain diameters, then parts of a grain will be more in focus than other parts. Those parts may be extracted from a series of progressively focussed images to form one image using the most focussed areas of each in the series.

Forster et al. [15], describe three different techniques: point based; neighbourhood based; and multi-resolutional based image fusion. They propose wavelets as the method to extract localised frequency information to determine the most focussed pixels from a series of images that vary in focal point through the object.

An example is shown below where the program, “combinez” , available under the GNU licence from the internet [29], using a series of nine pollen images from this project, processes them to produce the right hand image shown in Figure 9-1.

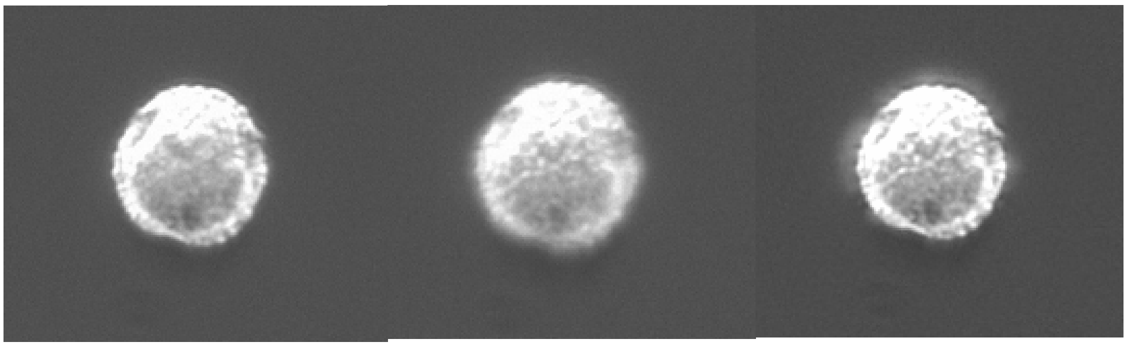


Figure 9-1: The two most in-focussed images out of nine used for focus integration (left and centre) and the result (right)

9.3.3 Focus Mechanism Improvement

The focus mechanical gear used has just sufficient resolution to capture in-focus images but with a step size that varies slightly. A finer resolution of five times the present gearing would improve the situation by allowing a half step driver. A ten times reduction would make single stepping possible and a twenty times reduction would also allow twice the present focus step resolution.

Focussing may also be improved, or automated further, by adding a distinctive mark to the surface of the slide to focus on as a starting point for the auto-focus. The mark could be found automatically and if placed in a fixed location on a standardised slide, pave the way to the addition of a slide feeder for automatic, multiple slide processing.

9.3.4 Pre-processing of final Images

The images used for classification are segmented only with no pre-processing to improve any characteristics that may improve the classifier performance. Suggestions are complete image background removal, edge enhancement and histogram stretching.

9.3.5 Texture Isolation

Images for feature extraction may be segmented further to contain only the area showing surface features and excluding all background and larger features such as culpi and pores. As the larger features can be hidden and in different locations and angles of view, then they are perhaps showing as differences between the same pollen types rather than similarities. A segmentation algorithm would need to be developed and the idea tested against current classification accuracy. In [43], 13 pollen types were classified with 100% accuracy using images containing pollen texture only.

9.3.6 Improving Results with Large Numbers of Pollen Types

By grouping the pollen that are more likely to be confused by the classification system into one group for initial classification, and then running classification on the sub groups within each group, the number of pollen types per classification task would be reduced, and with this divide and conquer technique, the overall classification rate would be improved, but at a cost of time.

9.3.7 Clumping and Clump Splitting

Clumping is only partially solved in the present system by reducing the likelihood of it occurring. Although the amount of clumping is minimal, it is still present and other pollens not used here may have stronger tendencies to stick together. The pollen grains are also found lodged next to detritus and bubbles. It would improve the system to find pollen grains in these circumstances and so improve the accuracy of the counting. There are many algorithms in the literature that would be worthwhile investigating, especially in the area of cell and tissue segmentation. Some interesting examples are [35-37]. The Hough transform, particularly the variant for detecting circular patterns, may be useful in selecting individual pollen grains in large clumps.

9.3.8 Novelty Detection

At present the pollen types to be classified must be pre-determined and the neural network trained on exemplar images of those types. Novel pollen types will be found and included in the classification results but grouped with the pollen type that most resembles it. The reliance on the palynologist to discern and separate out a new pollen type might be reduced if the classification could perform the detection. Novel detection is often handled using unsupervised networks, as used by Marsland [46], where a “grow when required” network was developed which may be useful in this area of development. Novel detection may be implemented initially by taking the groups as separated by the present neural network and further separating them using a novel detection scheme to test if there are any sub-groups, and presenting these to the palynologist for final analysis. The ability to drive the slide back for a 'live' display of the possible novel pollen, would allow the palynologist to use manual focus variation and to roll the pollen grain to view other facets, as is done presently in conventional microscopy to aid the determination of pollen type.

9.3.9 Speeding up the Process

The main delays in the operation are in the capture of images and focussing. Altering the processing of slides to ensure all pollen are as close as possible to a single plane, and increasing depth of field, may remove the requirement of automatic focussing for each high magnification image. This would speed up the process considerably. If this proves impractical, then the working on the individual elements involved, to speed each one, will produce adequate results. Matlab is designed for rapid development and not speed of execution. Well written C code, for example, is likely to improve speed of execution considerably. There is at present the requirement to add delays in the software while images are captured from the cameras. It is thought to be a property of Matlab, but it is possibly caused by the cameras themselves.

9.3.10 Spatial Sampling of Slides

Tests in this project were performed by counting an entire slide. For palaeopalynological studies, this is not how it is usually done. Sampling is performed by counting microscope field of view widths across a slide until a predetermined number of pollen have been counted or the minimum of a predetermined number of pollen and all strips across the slide are completed from edge to edge. It has been suggested that present practice is not a random selection of pollen; an assumption made for the statistics utilised to be valid [7]. To improve randomness of sampling it is proposed that it be done by AutoStage in the following manner.

The area of interest on the slide is divided into rectangular areas and random selections of these areas are imaged by the high magnification microscope. Each area is then segmented, classified and counted for each rectangular sample. A statistical analysis could then be performed to determine the slide populations of each pollen type found.

$$count = \sum x_i \times \frac{A}{na} \tag{9.1}$$

$$SE = \frac{sd(x)}{\sqrt{n}} \times \frac{A}{na} \tag{9.2}$$

$$\sum \frac{p_i}{n} = \hat{\Pi} \text{ (estimate of population)} \tag{9.3}$$

$$\frac{sd \times p_i}{\sqrt{n}} = SE\left(\hat{\Pi}\right) \tag{9.4}$$

A = area of the slide
a = area of small areas captured
n = number of small areas captured
Π = population of the slide
p_i = probability of ith species in the sample
sd = standard deviation
SE = standard error

By first running trials on slides with known populations, the standard deviation and mean could be determined and a suitable sample size, n, calculated. This should prove a better method than the present manual methods.

If this proved to be a more satisfactory method of counting a slide, then the low magnification camera might be dispensed with. The errors introduced by the low magnification imaging and segmentation would need to be weighed against the errors of sampling to determine the better technique. It may be that the system retains the low magnification microscope and the sampling is an option for certain circumstances.

9.3.11 A More Compact Microscope

To compact the structure of the AutoStage, an infinity-corrected objective lens would allow the optical tube length to be shorter but requires the addition of a suitable image forming lens.

9.3.12 Dark Field Illumination

Another simple method of dark field illumination, proposed in 1982 by Molesini et al. [47], utilises detuned interference filters. An interference filter shows a selective peak transmittance as a function of both wavelength and angle of incidence. With reference to Figure 9-2, filter-1 is defined by peak wavelength γ_1 and half bandwidth, $\Delta\gamma_1$. Filter-2 is then defined by γ_2 and $\Delta\gamma_2$. Usually, $\gamma_2 > \gamma_1$ and $\Delta\gamma_2 > \Delta\gamma_1$. Light from filter-1 can not pass through filter-2 unless the angle θ is greater than some angle depending on the difference, γ_2-γ_1. This angle is calculated to be larger than the aperture angle of the objective lens so only light scattered by the object can reach the objective lens at an acceptable angle.

The advantage of such a system over the present design is that positioning of the lighting source would not be as critical. At present the opaque light-blocks require careful shape and placement and cause some variation in lighting intensity across the image.

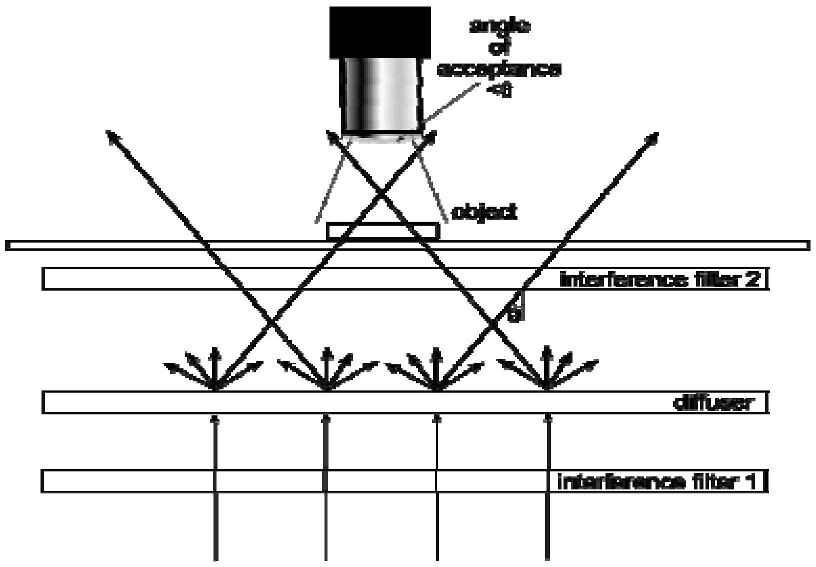


Figure 9-2: Dark field Illumination using detuned interference filters.

10 Bibliography

[1] "Digital Microscope Resource Centre," http://www.digitalmicroscope.com/, 2005.

[2] "Pollen Counting Web Sites," http://www.jrn.columbia.edu/studentwork/cns/2002-07-07/643.asp https://www.aaaai.org/nab/index.cfm?p=displaystationinfo http://www.ccairquality.org/faq/faq_pollen.html#pollen6.

[3] "Pollen dot Com," http://www.pollen.com/Pollen.com.asp, 2006.

[4] A. I. Abdel-Fattah, M. S. El-Genk, and P. W. Reimus, "On Visualization of Sub-Micron Particles with Dark-Field Light Microscopy," *Journal of Colloid and Interface Science*, vol. 246, pp. 410-412, 2002.

[5] R. Bellman, *Adaptive Control Processes: A guided Tour*. New Jersey: Princeton University Press, 1961.

[6] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[7] D. Brookes and K. W. Thomas, "The Distribution of Pollen Grains on Microscope Slides. Part 1. The Non-Randomness of the Distribution," *pollen et spores*, vol. IX, pp. 621-629, 1967.

[8] V. M. J. Bryant, *Pollen: Nature's Fingerprints of Plants*. Chicago: Encyclopedia Britannica Inc., 1990.

[9] V. M. J. Bryant and D. C. Mildenhall, "Forensic Palynology: A New Way to Catch Crooks," *New Developments in Palynomorph Sampling, Extraction and Analysis.*, vol. 33, pp. 145-155, 1998.

[10] H. Burkhardt, Q. Wang, and O. Ronneberger, "MICROBUS project," http://lmb.informatik.uni-freiburg.de/research/omnibuss/index.en.html#pub.

[11] J. Conrad, "Depth of Field in Depth," http://www.largeformatphotography.info/articles, pp. 45, 2006.

[12] J. G. Daugman, "Two-Dimensional Spectral-Analysis of Cortical Receptive-Field Profiles," *Vision Research*, vol. 20, pp. 847-856, 1980.

[13] P. Etheridge, "Discrimination of Pollen Taxa from Digital Image Feature Data," Massey University, Palmerston North, Report July 28 2005.

[14] J. R. Flenley, "The Problem of Pollen Recognition," presented at C.S.I.R.O. workshop, Canberra, Australia, 1968.

[15] B. Forster, D. Van De Ville, J. Berent, D. Sage, and M. Unser, "Complex Wavelets for Extended Depth-of-Field: A New Method for the Fusion of Multichannel Microscopy Images," *Microscopy Research and Technique*, vol. 42, pp. 65-33, 2004.

[16] M. Forster and J. R. Flenley, "Pollen Purification and Fractionation by Equilibrium Density Gradient Centrifugation," *Palynology*, vol. 17, pp. 137-155, 1993.

[17] R. M. Forster and J. R. Flenley, "The Application of Density Gradient Centrifugation to Palynology," in *Miscillaneous Series No. 35*, R. C. Ward, Ed. Hull: School of Geography & Earth Resources, University of Hull, England, 1989, pp. 19.

[18] D. W. Fountain, "Pollen and Inhalant Allergy," *Biologist*, vol. 49, pp. 5-9, 2002.

[19] I. France, A. W. G. Duller, G. A. T. Duller, and H. F. Lamb, "A new approach to automated pollen analysis," *Quaternary Science Reviews*, vol. 19, pp. 537-546, 2000.

[20] G.P.Allen, R.M.Hodgson, S.R.Marsland, G.Arnold, R.C.Flemmer, J.Flenley, and D.W.Fountain, "Automatic Recognition of Light Microscope Pollen Images.," *Image and Vision Computing New Zealand 2006*, 2006.

[21] R. Gentleman, R. Ihaka, D. Bates, J. Chambers, P. Dalgaard, K. Hornik, S. Iacus, F. Leisch, T. Lumley, M. Maechler, D. Murdoch, P. Murrell, M. Plummer, B. Ripley, D. T. Lang, L. Tierney, and S. Urbanek, "R Project," http://www.r-project.org/.

[22] J.-M. Geusebroek, F. Cornelissen, W. M. Arnold, and H. G. Smeulders, "Robust autofocusing in microscopy," *Cytometry*, vol. 39, pp. 1-9, 2000.

[23] D. J. Goldstein, *Understanding the Light Microscope - a Computer Aided Introduction*. London: Academic Press, 1999.

[24] D. J. Goldstein and M. A. Williams, "Quantitative Assesment of Radiographs by Photometric reflectance Microscopy. An Improved Method Using Polarised Light.," *Histochemical Journal*, vol. 6, pp. 223-230, 1974.

[25] R. C. Gonzalez, R. E. Woods, and S. L. Eddons, *Digital Image Processing Using Matlab*: Pearson Prentice Hall, 2004.

[26] D. G. Green, "The Ecological Interpretation of Fine Resolution Pollen Records," *New Phytologist*, vol. 94, pp. 459-477, 1983.

[27] D. G. Green, "The Environmental Challenge for Numerical Palynology," *INQUA Working Group on Data-Handling Methods*, vol. 15, pp. 3-6, 1997.

[28] F. C. A. Groen, I. T. Young, and G. Ligthart, "A comparison of different focus functions for use in autofocus algorithms," *Cytometry*, vol. 6, pp. 81-91, 1985.

[29] A. Hadley, "combinez," http://www.hadleyweb.pwp.blueyonder.co.uk/CZ5/combinez5.htm, 2006.

[30] S. Haykin, *Neural Networks A Comprehensive Foundation*, 2 ed. New Jersey: Prentice Hall Inc., 1999.

[31] D. O. Hebb, *The Organisation of Behaviour: A neuropsychological Theory*. New York: Wiley, 1949.

[32] V. Hilsenstein, "Robust Autofocusing for Automated Microscopy Imaging of Fluorescently Labelled Bacteria," 2005.

[33] R. M. Hodgson, C. A. Holdaway, Z. Yongping, D. W. Fountain, and J. R. Flenley, "Progress towards a system for the automatic recognition of

pollen using light microscope images," *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pp. 76 - 81 2005
[34] C. Holdaway, "Automation of Pollen Analysis using a Computer Microscope," Massey University, Palmerston North, Masters 2005.
[35] H. H. S. Ip and R. P. K. Yu, "Recursive splitting of active contours in multiple clump segmentation," *Electronics Letters*, vol. 32, pp. 1564-1566, 1996.
[36] X. C. Jin, S. H. Ong, and Jayasooriah, "A domain operator for binary morphological processing," *Image Processing, IEEE Transactions on*, vol. 4, pp. 1042-1046, 1995.
[37] X. C. Jin, T. T. E. Yeo, S. H. Ong, Jayasooriah, and R. Sinniah, "An automated clump decomposition system for cervical tissue sections," 1994.
[38] A. W. Jones and J. B. Hawthorn, "Towards a General Definition for Spectroscopic Resolution," presented at ASP Conference Series, 1995.
[39] N. Kehtarnavaz and H. J. Oh, "Development and real-time implementation of a rule-based auto-focus algorithm," *Real-Time Imaging*, vol. 9, pp. 197-203, 2003.
[40] M. Langford, G. E. Taylor, and J. R. Flenley, "Computerized Identification of Pollen Grains by Texture Analysis," *Review of Palaeobotany and Palynology*, vol. 64, pp. 197-203, 1990.
[41] P. Li and J. R. Flenley, "Classification and Visualisation of Pollen data Using MLP Neural Networks," *Proceedings of Image and Vision Computing New Zealand 1997*, pp. 497 - 502, 1997.
[42] P. Li and J. R. Flenley, "Pollen texture identification using neural networks," *Grana*, vol. 38, pp. 59-64, 1999.
[43] P. Li, J. R. Flenley, and L. K. Empson, "Classification of 13 types of New Zealand Pollen patterns using Neural Networks," *Proceedings of Image and Vision Computing New Zealand 1998*, pp. 120 - 123, 1998.
[44] P. Li, W. J. Treloar, J. R. Flenley, and L. Empson, "Towards automation of palynology 2: the use of texture measures and neural network analysis for automated identification of optical images of pollen grains," *Journal of Quaternary Science*, vol. 19, pp. 755-762, 2004.
[45] S. Mallat, *A wavelet tour of signal processing*. San Diego: Academic Press, 1998.
[46] S. Marsland, "On-Line Novelty Detection Through Self-Organisation, With Application To Inspection Robotics," University of Manchester Manchester, PhD 2001.
[47] G. Molesini, D. Bertani, and M. Cetica, "Dark Ground Microscopy with Detuned Interference Filters," *Opt. Eng.*, vol. 21, pp. 1061-1063, 1982.
[48] P. C. Montgomery and J. P. Fillard, "Near infrared dark-field microscopy with video for studying defects in III-V compound materials," *Measurement Science and Technology*, vol. 1, pp. 120, 1990.
[49] P. C. Montgomery and J. P. Fillard, "Study of microdefects in near-surface and interior of III-V compound wafers by dark-field transmission microscopy," *Electronics Letters*, vol. 24, pp. 789-790, 1988.

[50] I. T. Nabney, "NETLAB," Software, http://www.ncrg.aston.ac.uk/netlab/index.php, 2003.

[51] A. Narendra, "A Transform for Multiscale Image Segmentation by Integrated Edge and Region Detection," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 18, pp. 1211 - 1235, 1996.

[52] H. Nyquist, "Certain Topics in Telegraph Transmission Theory," *Transactions of the A. I. E. E*, pp. 617–644, 1928.

[53] N. R. Pal and S. K. Pal, "A Review On Image Segmentation Techniques," *Pattern Recognition*, vol. 26, pp. 1277 - 1294, 1993.

[54] J. L. Pech-Pacheco, G. Cristobal, J. Chamorro-Martinez, and J. Fernandez-Valdivia, "Diatom autofocusing in brightfield microscopy: a comparative study," 2000.

[55] F. Pedrotti and L. Pedrotti, *Introduction to Optics*. Englewood Cliffs N.J.: Prentice Hall International, 1993.

[56] C. A. Prior, J. R. Flenley, and A. Zondervan, "A New Approach to the Preparation of Pollen for AMS Dating," presented at 16th International Radiocarbon Conference, Groningen, The Netherlands, 1997.

[57] T. R. Reed and H. Wechsler, "Segmentation of textured images and Gestalt organization using spatial/spatial-frequency representations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, pp. 1-12, 1990.

[58] L. Robertson, "National Pollen and Aerobiology Research Unit," http://pollenuk.worc.ac.uk, 2004.

[59] M. Rodriguez-Damian, E. Cernadas, A. Formella, M. Fernandez-Delgado, and P. De Sa-Otero, "Automatic detection and classification of grains of pollen based on shape and texture," *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 36, pp. 531-542, 2006.

[60] P. Rosenthal and R. Tester, "VideSCope," in *Silicon Chip*, 2001.

[61] A. Santos, C. O. De Solorzano, J. J. Vaquero, J. M. Pena, N. Malpica, and F. Del Pozo, "Evaluation of autofocus functions in molecular cytogenetic analysis," *Journal of Microscopy-Oxford*, vol. 188, pp. 264-272, 1997.

[62] F. W. Sears, M. W. Zemansky, and H. D. Young, *University Physics*, 7 ed: Addison-Wesley Publishing Company Inc., 1987.

[63] C. E. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, vol. 37, pp. 10-21, 1949.

[64] E. C. Stillman and J. R. Flenley, "The needs and prospects for automation in palynology," *Quaternary Science Reviews*, vol. 15, pp. 1-5, 1996.

[65] G. Taguchi, E. Elsayed, and T. Hsiang, *Quality Engineering in Production Systems*. New York, N.Y.: McGraw Hill,, 1989.

[66] P. Tomlinson, "Ultrasonic Filtration as an aid in Pollen Analysis of Archaeological Deposits," *Circaea*, vol. 2, pp. 139-140, 1984.

[67] W. J. Treloar and J. R. Flenley, "An Investigation into the Potential of Light Microscopy for the Automatic Identification of Pollen Grains by the Analysis of their Surface Texture," presented at 9th International Palyological Congress, Houston Texas, 1996.

[68] W. J. Treloar, G. E. Taylor, and J. R. Flenley, "Towards automation of palynology 1: analysis of pollen shape and ornamentation using simple geometric measures, derived from scanning electron microscope images," *Journal of Quaternary Science*, vol. 19, pp. 745-754, 2004.

[69] S. Viller, J. Bowers, and T. Rodden, "Human factors in requirements engineering:: A survey of human sciences literature relevant to the improvement of dependable systems development processes," *Interacting with Computers*, vol. 11, pp. 665-698, 1999.

[70] R. W. Weber, "Pollen Identification," *Annals of Allergy, Asthma, and Immunology*, vol. 80, pp. 141-147, 1998.

[71] W. T. Welford, *Optics*, 2 ed. New York: Oxford University Press, 1988.

[72] H. J. L. White, "Preliminary Research into the Possibilities of Automated Pollen Counting," *Pollen Et Spores*, vol. XXX, pp. 111-124, 1988.

[73] I. T. Young, R. Zagers, L. J. Van Vliet, and J. Mullikin, "Depth-of-Focus in Microscopy," *Proceedings Of The Scandinavian Conference On Image Analysis*, vol. 1, pp. 493, 1993.

[74] Y. Zhang, "Pollen Discrimination Using Image Analysis," Massey University, Palmerston North report, 2001-2003 2003.

[75] Y. Zhang, D. W. Fountain, R. M. Hodgson, J. R. Flenley, and S. Gunetileke, "Towards automation of palynology 3: pollen pattern recognition using Gabor transforms and digital moments," *Journal of Quaternary Science*, vol. 19, pp. 763-768, 2004.

11 APPENDICES

A. Defining Depth of Field for AutoStage

The circle of confusion is to be defined (Figure 11-1) and for visual systems, this can be subjective.

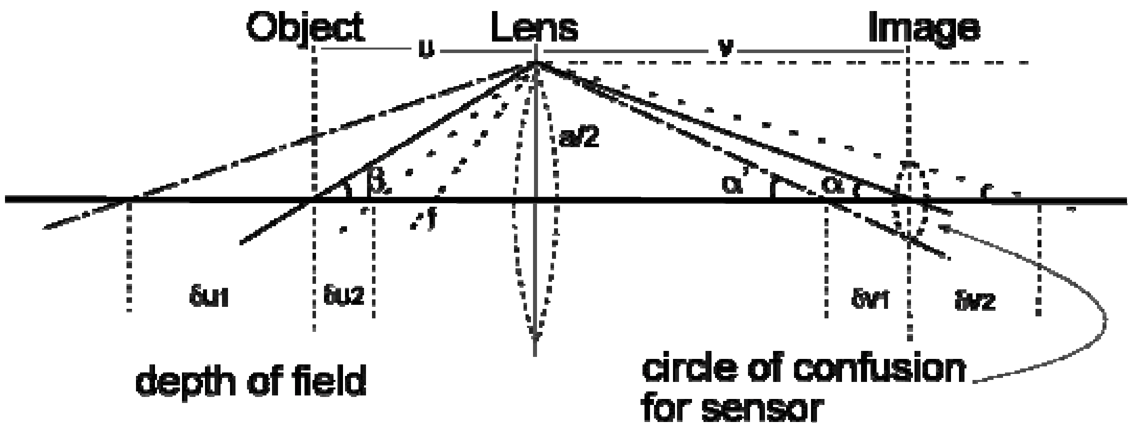


Figure 11-1: Circle of confusion and depth of field

For this digital system with sensor elements of a fixed size we can define the circle of confusion such that movement of an object point, toward or away from the lens within the depth of field limits, will not cause the light from that object point to affect a neighbouring image sensor element. The object point forms an "Airy disk" at the sensor which is calculated to be 0.61λ/a (§4.2.2). Considering the diameter of that circle and the distance across a sensor element (see Figure 11-2) the amount the image point may move before encountering the neighbouring pixel is simply the distance from centre to edge of the sensor element minus the radius of the Airy disk:

$$c = \frac{p}{2} - \frac{0.5 \times \lambda}{2 \times \sin\left(\tan^{-1}\left(\frac{a/2}{u}\right)\right)} \tag{11.1}$$

where p is the pixel, or sensor element dimension, and the Airy disk is calculated using the geometry of Figure 4-9 to find the angle β. The sensors are unable to be manufactured with sensitive elements butting exactly so elements are actually somewhat smaller, and have gaps between them making the circle of confusion defined here a conservative estimate.

Having defined the circle of confusion the depth of field is calculated.

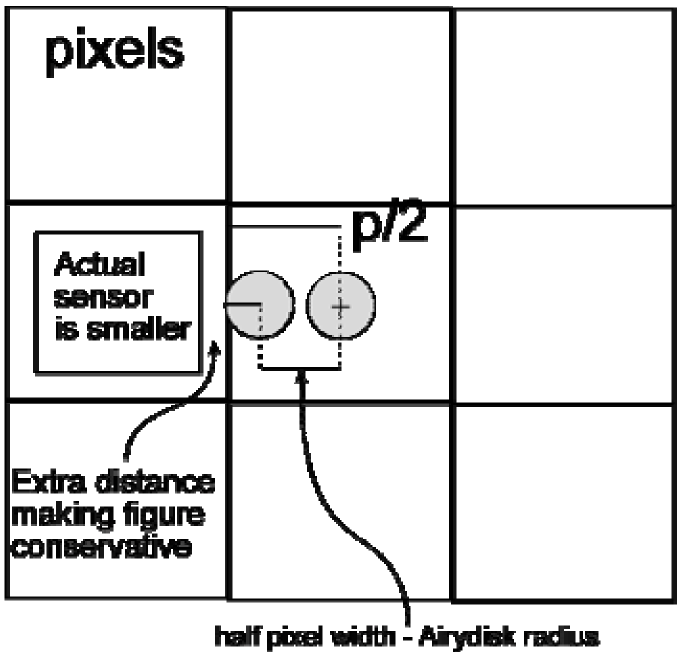

pixels
p/2
Actual sensor is smaller
Extra distance making figure conservative
half pixel width - Airydisk radius

Figure 11-2: The Airy disk and its limits of movement on the image sensor that will define a circle of confusion used for depth of field calculation.

The lens equation, rearranged to find u, is:

$$\frac{1}{v}+\frac{1}{u}=\frac{1}{f} \qquad \Rightarrow \qquad u=\frac{f\times v}{f-v} \tag{11.2}$$

To find a change in u with a change in v we differentiate:

$$\frac{du}{dv}=\frac{f}{f-v}+\frac{f\times v}{(f-v)^2} \tag{11.3}$$

And:

$$du=dv\times\left(\frac{f}{f-v}+\frac{f\times v}{(f-v)^2}\right) \tag{11.4}$$

That defines an infinitesimal change and as an approximation, we assume the same holds true for a larger change, δu and δv. Now δv = δv1 + δv2 (see Figure 4-9 and Figure 11-3) is required, and derived from geometry in Figure 11-3.

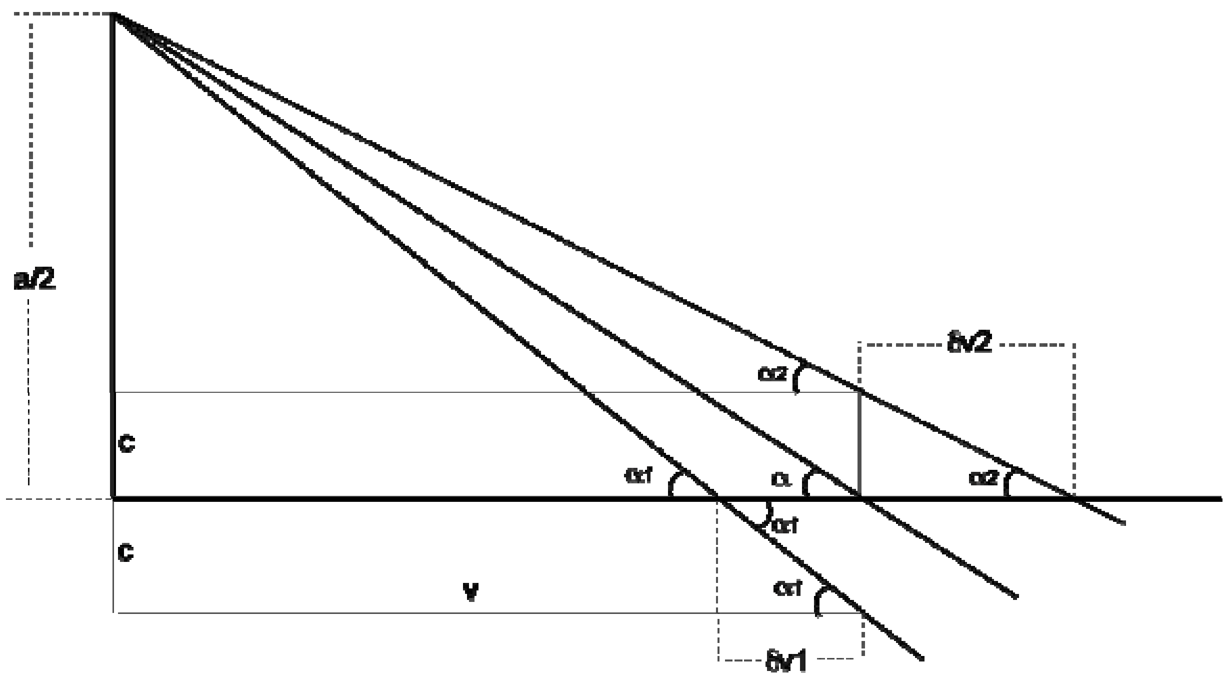


Figure 11-3: Image Side Ray Diagram showing depth of focus and circle of confusion

$\tan\alpha' = \frac{a/2+c}{v}$, where a is the aperture of the lens and a/2 is axis to aperture edge. The ray forming α' forms the same angle below the axis and intersects with the circle of confusion (CoC) at the image plane so CoC radius (c) divided by dv1 is tan(α'). So,

$$\tan\alpha' = \frac{a/2+c}{v} = \frac{c}{\delta v1} \qquad \delta v1 = \frac{c\times v}{a/2+c} \tag{11.5}$$

Similar arguments find $\delta v2 = \frac{c\times v}{a/2-c}$ (11.6)

So, depth of field is:

$$\delta u = \left(\frac{c\times v}{a/2+c} + \frac{c\times v}{a/2-c}\right)\times\left(\frac{f}{f-v} + \frac{f\times v}{(f-v)^2}\right) \tag{11.7}$$

Ignoring c in the denominators, because it has 3 orders of magnitude difference to a, the equation reduces to:

$$\frac{4cv}{a}\times\left(\frac{f}{f-v}\right)^2 \tag{11.8}$$

B. Published Paper

Automatic Recognition of Light-Microscope Pollen Images.

G.P.Allen[1], R.M.Hodgson[1], S.R.Marsland[1], G.Arnold[1], R.C.Flemmer[2], J.Flenley[3], D.W.Fountain[4]
[1] Massey University, Institute of Information Sciences and Technology.
[2] Massey University, Institute of Technology and Engineering.
[3] Massey University, Geography Programme, School of People, Environment and Planning.
[4] Massey University, Institute of Molecular BioSciences.

Email: g.p.allen@massey.ac.nz

Abstract

This paper is a progress report on a project aimed at the realization of a low-cost, automatic, trainable system "AutoStage" for recognition and counting of pollen. Previous work on image feature selection and classification has been extended by design and integration of an XY stage to allow slides to be scanned, an auto-focus system, and segmentation software. The results of a series of classification tests are reported, and verified by comparison with classification performance by expert palynologists. A number of technical issues are addressed, including pollen slide preparation and slide sampling protocols.

Keywords: pollen recognition, image processing, classification, microscopy

Introduction

Fossil pollen analysis is used to determine flora genus from which climate data, evidence of human activity and oil deposit locations, can be deduced. Honey type, and location of origin, can be indicated by the pollens found in the honey. Allergy sufferers can be advised of high pollen counts in the air. Forensic investigations can be aided by determining if an object has been in a certain general location by identifying the pollen types attached.

The need for an automated pollen counting system has been identified and detailed for many years [61]. A previous paper reported on progress toward such a system [33] and a significant milestone in that project is reached, and reported here, with the complete system designed, built and evaluated as a functioning unit.

The system will:

- reduce the massive amount of laborious counting required by highly skilled people involved in palynological endeavours (30 months in a PhD);
- increase sample quantities allowing more accurate pollen studies, especially in fine resolution sampling [26];
- increase the frequency and locations of pollen counts, which are of use to inhalant allergy and asthma sufferers.

A good description of the problems involved and requirements of a complete automated system have been described recently [19, 56]. The broad requirements are to locate pollens on a microscope slide and classify each into taxonomic categories at reasonable cost, and with a success rate at least that of a skilled person. The saving is labour, and time consumed by people with skills that could be better applied to less mundane tasks.

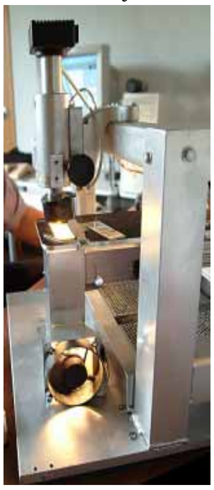


Figure 11-4: AutoStage

The steps involved in the AutoStage project are:

1. develop a set of features derived from optical images of pollen that are discriminable. [72]
2. develop a supervised classification system based on the features-set developed in step 1.

3. design a suitable low cost digital microscope [34]
4. develop an image segmentation scheme to isolate images of pollen and exclude detritus
5. develop and build an XY stage to allow slides to be scanned using transmitted or reflected light
6. develop a system to find the location of pollen on a slide and to capture in-focus images
7. integrate the system resulting from steps 1-6
8. evaluate and verify classification and count performance of the system, and compare to trained palynologists.

Steps 1-3 were completed [33]. This project is to develop and build a working microscope, build in an XY stage and focus hardware, develop working segmentation and focus algorithms: steps 4-8. We report development of the final stages and describe the completed system that takes a prepared slide and captures microscopic images from which pollen are segmented, image features extracted and pollen taxa classified and counted.

Automated System Description

The system described here finds pollen grains on a slide and captures images of them together with their location information. Image features are extracted and used for classification of pollen types, enabling a count of the number of grains of each pollen type. The classification of pollen can be manually checked.
Selection of any portion of a slide to be processed is accomplished by the user moving the camera to opposite corners of a rectangular area of interest. The current system is capable of capturing areas shaped with a pixel resolution of $^1/_2$ micron.
The system comprises:

1. a machine to capture the images (§2.1)
2. segmentation, auto-focus and classification algorithms (§2.2)
3. a computer to run the algorithms and control the hardware (§2.3)

In addition to the sub-systems, slide preparation (§2.4) and slide sampling (§2.5) are discussed.

The Machine

The 'machine', is an XY stage with attached slide holder. Two digital microscopes are solidly mounted above a filtered and cooled light source. As transmission lighting is used, the slide sits on an aperture in the XY stage positioned between the cameras and light source as in Figure 2.
There are two power supplies for lighting and stepper motors. Two motors move the XY stage to locate pollen under the microscope and a third motor adjusts the relative height of the cameras for focussing.

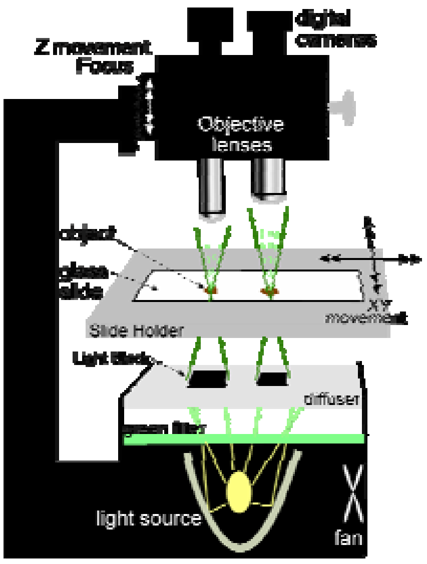


Figure 2: AutoStage elements

The Stage

The slide is held in a standard microscope holder and is moved by a commercial XY precision stage driven by two stepper motors. The motors are micro-stepped to 1/10th of their 1.8° step angle, allowing a linear movement of 2.6 microns per step (the smallest pollen of interest is about 10 microns across). The field of view of the high magnification camera is 165 x 123 steps. The speed of movement is set below maximum to about 5mm per second.

Two Microscopes

A low magnification microscope with a large field of view (FOV), locates pollen grains quickly while a high magnification microscope captures images with sufficient detail for feature extraction.
A digital camera sensor and a standard microscope objective lens placed 207mm from the camera sensor plane, forms the "high magnification" microscope with an optical magnification of 11 ·2x. Because the camera sensor elements are 4.65 microns square, the magnification that is required for a human to view the formed image occurs in translation from a 1024x768

pixels in the 6mm diagonal rectangle of the sensor, to 1024x768 pixels on a computer screen. That is about 72x, and 720x including optical magnification.
The small *optical* magnification results in a depth of field greater than for a conventional microscope with the same overall magnification.
The FOV of the main camera is less than half a millimetre square. To image an entire slide more quickly, the low magnification camera with about 1/10th the magnification, is used to more quickly cover the slide and locate potential pollen grains. A segmentation algorithm identifies most detritus and the locations of remaining objects found are stored for the high magnification camera to investigate.
Segmentation, using the high magnification camera and finding an acceptable object, produces an image slightly larger than the object bounding rectangle. The image is stored for feature extraction and classification (Figure 3).

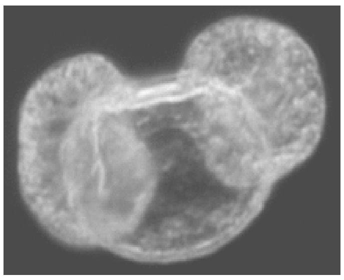


Figure 11-5: hi-mag segmented image (*Pinus radiata*. ~50μm)

The Lighting

Lighting is provided by a simple arrangement of a quartz halogen lamp directly below the cameras, with filtering, and a fan for cooling. One filter is a band-pass to reduce any chromatic aberrations caused by the objective lens. A green filter was chosen because the camera is filtered to have a maximum sensitivity in the same area of the spectrum as human vision, λ ≈ 550nm: green.
A diffusion filter is the topmost filter and has a light blocking rectangle below each camera. The diffused light therefore strikes the object oblique to the optical axis, making it a simple form of "dark field" illumination. Little of the light direct from the source enters the objective lens directly so the background is dark and objects are light with darker 'shadows' formed by the surface features. Contrast is increased over light-field transmission microscopy with one study measuring an increase from 10% to 85% contrast [44]. Sub-resolution visualisation is another property of dark-field illumination [4]. This is where objects smaller than the resolution of the optical system are indicated, but not resolved. That this has a positive or negative effect on image features extracted in this case would require further study. The dark-field effects are helpful for finding pollen in the low magnification camera and creating a better image for feature extraction.

The Algorithms

Auto-Focus

The low magnification camera is initially focussed manually at the same time the user is setting the limits for a region of interest within the total area of the slide. The auto-focus software then steps the camera through that manually set focus position, to refocus. The auto-focus operates by calculating the standard deviation of all grey levels of each image as it steps through the focal plane. The sequential values are stored as a vector and a suitable peak is located by a "local maximum" algorithm. The camera is moved back to the step where the local maximum was found. Movements of critical placement are always in the upward direction. This focus position is then used for all images taken with the low magnification camera as a high depth of field keeps pollen sufficiently in focus.
There are several focus measurement methods in the literature [22, 28, 36, 58]. After experimentation, the standard
□□□□□□□□□□□□□□□h□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□
□H□H□Ĉ□□□□□□□□□□□Z□□□□□□□□
desired smoothing effect and it is not
computationally demanding.
The high magnification camera is fixed on the same focus movement so once the low magnification camera is focussed, the high magnification camera can be moved to a near focus position. This position is used to perform an automatic refocus.
Auto-focusing is performed on each object because the pollen grains are not necessarily all within the same focal plane and depth of field is less for this microscope.

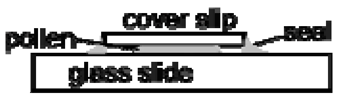


Figure 4: glass slide with cover slip
The auto-focussing algorithm used with the high magnification camera incorporates a

squared gradient measure where for each pixel, the maximum grey-scale gradient-squared, between y direction and x direction is chosen and all chosen values summed.

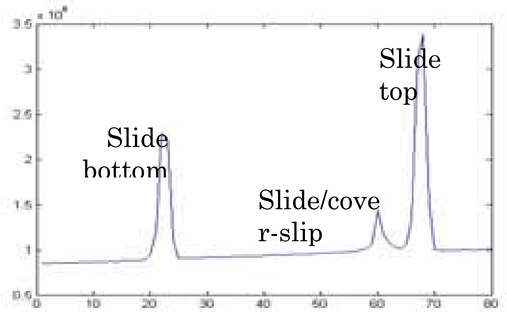


Figure 5: plot of focus image against gradient with a dirty slide giving greater focus values at the outer surfaces. Centre peak is the focus aim.

The values plotted against focus step number, results in a large 'spike' in value for 3 or 4 steps of the focus movement. To improve the auto-focus, the step size would need to be made smaller and an algorithm with greater selectivity might then be used. To reduce computation time and help ensure the object of interest is in focus, the image area is reduced to around the centre of the image where the object may be located.

It takes 15s for one complete pollen grain capture: move stage; auto-focus; capture; segmentation, save image. Auto-focus takes $^2/_3$ of that time at 10s.

Segmentation

Segmentation is difficult and often problem specific. For a review on segmentation techniques see [49].

A stored background image, taken with no slide in place, is subtracted from images captured to remove any image anomalies caused by the system. Objects are located by first finding edges using a Sobel edge operator. As pollen are small objects with well defined outlines, then the edge detection results in a mostly closed loop. Morphological operations follow: dilation, to join any broken edges; filling any closed loops to form solid 'blobs'. Erosion then reduces the blob size to be close to that of the original object.

The blob pixel counts are measured, and any blobs too small or too large to be a pollen grain are removed. The smallest pollen grains of interest (about 10 microns across) have a blob area of 5 pixels in an image from the low magnification camera. Large pollen grains, 100 microns across, are represented by a blob area of about 500 pixels.

For each blob of correct size, a bounding rectangle and its area are calculated. If the rectangle has an aspect ratio too small, or the blob area to rectangle area ratio is too small, then the blob is removed.

The area of a convex hull for each blob is calculated and if the blob area to hull area ratio is too small, the object is removed.

The centres of remaining blobs are found and their positions on the slide calculated and stored. The high magnification camera is moved to each of those positions and performs a segmentation process to find a valid object nearest the centre of the image. Tolerances in movements cause the object to appear with a variable offset.

Classification

To perform taxonomic classification, image features extraction and a multi-layer perceptron [46] are used in line with [40]. The features used are those identified in [71] consisting of 43 shape and texture features.

Texture features are represented by a series of Wavelet transforms that measure localised spatial/spatial-frequency content using Gabor and Orthogonal Wavelet transforms. Orientation sensitivity is reduced by averaging the results corresponding to different directions [72]. Other textural features used are Grey Level Co-occurrence Matrix, and Grey Gradient Co-occurrence Matrix. Shape features are geometric, histogram and second moment.

Linear Discriminant Analysis, together with Principal Components Analysis, were employed to compare discrimination and check for any redundant features [13]. No reduction of feature-set size was found useful. A Support Vector Machine algorithm, with its binary classification capability, was used to discriminate two grass pollens and found to be less effective than the multi-layer Perceptron.

The Computer

The computer used is a PC with a 2.6GHz processor and 1Gbytes of RAM running Windows XP professional. All the code is written in Matlab including: image acquisition via USB and IEEE1394 (FireWire); control of the stepper motors via a serial port; and the auto-focus, segmentation, and classification algorithms.

Slide Preparation

To improve the efficacy of the system the slides should be prepared in a prescribed and suitable manner. It is important this should be similar to current practice. Auto-focus can be adversely affected by objects on surfaces other than the top of the slide and the bottom of the cover-slip. The segmentation algorithms could be compromised and images captured would be degraded if dust or oil were present, even if they were out-of-focus.
The prescription proposed is for the pollen samples to be suspended in some setting gel. Silicon oil is suitable and may be desirable if the slides are to be checked on a conventional microscope, as are agar or glycerol if an aqueous medium is required. The suspension should have a concentration that results in no more than 500 pollen grains per slide to reduce clumping. The sample medium volume and viscosity is such that when dropped onto the slide and the cover slip is placed on top, the medium does not travel past the outer edges of the cover slip.
The slide is placed on a warmer to allow air bubbles to escape the gel. Wax is dropped onto the slide at the edge of the cover slip to 'wick' under the cover slip to seal the pollen suspension in, and hold the cover slip firmly in place. The slide surfaces can now be cleaned without moving the pollen grains within the slide. Adding detergent to a last rinse will help reduce clumping.

Spatial Sampling of Slides

If sampling the slide is applicable, the high magnification camera only might be utilised. It may perform sampling better than in the current methods of manual counting.
It is proposed that the area of interest of the slide be divided up into rectangles, a sample of those rectangles randomly selected, and that the camera capture an image of each selected rectangle. The images would be segmented, classified and counted for each rectangular sample. A statistical analysis would estimate the slide populations of each pollen type.
By running trials on slides with known populations, a suitable sample size could be calculated.
This should prove a better method than the present manual methods, as the randomness of the present slide sampling approach is suspect [7].

Experiments and Results

Three image data bases were compiled:

1. CM: captured using a conventional microscope
2. AS: captured using AutoStage
3. BR: images used by France et al. [19]
   A selection of the data base images was made of 50% for training, 25% for validation and 25% for the final tests reported here. The validation set was used with the training set to adjust neural net parameters for optimum results and verify the system working. The training and validation sets were then combined for training and the test set used for the final test. The feature sets extracted from the images, were presented in random order to the classification software. Results are expressed as total correctly classified pollens as a percentage of all pollens, and the means and standard deviations over 5 tests recorded.

Compare AS with CM

The aim of this experiment is to compare classification results using images taken from the same slides by AutoStage and by a conventional microscope.
Test description: Take 40 training, 10 test and 7 types of images from AS and CM data bases. Classify both sets and compare mean results and check for difference with a Students t test.
Results: The AS mean was **98%** correct (sd = 1.2) and the CM mean was **94%** correct (sd = 0.6). Using a 95% confidence t-test, the means are significantly different.

Classification of Grass Pollens

The aim of this experiment is to check performance of the AutoStage when classifying grass pollens which are commonly counted as one type as they are very difficult to distinguish manually under a light microscope.
Test description: take 3 grass pollen image sets from the AS data base, using 150 training and 50 test images. Classify the sets.
Results: Mean = **90%** correct (sd = 0.3).

Large Pollen Type Count

The aim of this experiment is to check the performance of the AutoStage using a wider range of pollen types in a single test.
Test description: 19 types were used for the experiment including all types available, however 2 of the 3 grass pollens were excluded. 150 training and 50 test images were used.
Results: Mean = **89%** correct (sd = 0.5).

AS Compared With another Project

The aim of this experiment is to compare AS classification results, to results recorded by France et al [19].
Test description: France, recorded results using 3 pollen types with 60/60/84 images made available on the internet. Here, 45 of each set of these images were used for training and 15 images for testing. Validation was not done as the neural network configuration and weights were not altered from other tests.
Results: France achieved overall **82%** correctly identified in the final classification stage with 3% being misclassified and 15% being rejected. The AS was, on average, **95%** successful in distinguishing 15 of the same images with 5% misclassification.

AS Compared with Experts

The aim of this experiment is to compare the total process of pollen counting from a slide by the AutoStage, with the count of the same slide by experts.
Test description: A slide with 6 pollen types is prepared. Five 'experts' including two professors, a post doctoral student, a technician working in palynology and an honours student, count the slide. The AutoStage then counts the slide.
Result. The table below shows statistics of the human count and one AutoStage count.

Pollen type	**5 People**			**AutoStage**
	Mean	StdDev	Range	Raw Count
1	65.6	13.4	43 - 77	64
2	14.2	4.8	9 - 20	13
3	21.8	8.7	16 - 37	18
4	86	17.9	58 - 102	75
5	0.8	0.4	0 - 1	1
6	8.6	1.5	7 - 11	7

Table 1: The performance of AutoStage was compared to five human experts.

Conclusions

1. Most importantly, for a *complete working system* and functional test described in §3.5, AutoStage has matched the result of experts. The *variability* of AutoStage has yet to be determined with multiple counts by AutoStage on more slides and a comprehensive statistical analysis.
2. The AutoStage system is giving classification results improved upon known published results.
3. The system is completed, functions well with promises of the ability to meet the requirements to be useful to a palynologist.
4. Images from the AutoStage used for classification performed better than images from a conventional microscope.
5. The lighting system described gives images of excellent contrast.
6. The auto-focus system performs well. The digital microscope, having a greater depth of field than a conventional microscope, makes focussing less critical.
7. The XY stage, with movement limits larger than a slide, a repeatability of position of 20 microns, speed in excess of 10mm per second, and a spatial resolution of 2.6 microns, would be satisfactory for a manufactured product.
8. The component costs of the prototype system were under $NZ15,000 including the computer.

Acknowledgements

Many thanks to Steve Denby and crew at the mechanical workshop of the Institute of Fundamental Sciences, Massey University, who built the AutoStage. Thanks also to, Xiuying Zou for conventional image capture of reference pollens.

References

[1] E. C. Stillman and J. R. Flenley, "The needs and prospects for automation in palynology," *Quaternary Science Reviews*, vol. 15, pp. 1-5, 1996.
[2] R. M. Hodgson, C. A. Holdaway, Z. Yongping, D. W. Fountain, and J. R. Flenley, "Progress towards a

system for the automatic recognition of pollen using light microscope images," *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pp. 76 - 81 2005

[3] D. G. Green, "The Ecological Interpretation of Fine Resolution Pollen Records," *New Phytologist*, vol. 94, pp. 459-477, 1983.

[4] I. France, A. W. G. Duller, G. A. T. Duller, and H. F. Lamb, "A new approach to automated pollen analysis," *Quaternary Science Reviews*, vol. 19, pp. 537-546, 2000.

[5] M. Rodriguez-Damian, E. Cernadas, A. Formella, M. Fernandez-Delgado, and P. De Sa-Otero, "Automatic detection and classification of grains of pollen based on shape and texture," *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 36, pp. 531-542, 2006.

[6] Y. Zhang, D. W. Fountain, R. M. Hodgson, J. R. Flenley, and S. Gunetileke, "Towards automation of palynology 3: pollen pattern recognition using Gabor transforms and digital moments," *Journal of Quaternary Science*, vol. 19, pp. 763-768, 2004.

[7] C. Holdaway, "Automation of Pollen Analysis using a Computer Microscope," vol. Masters. Palmerston North: Massey University, 2005, pp. 125.

[8] P. C. Montgomery and J. P. Fillard, "Study of microdefects in near-surface and interior of III-V compound wafers by dark-field transmission microscopy," *Electronics Letters*, vol. 24, pp. 789-790, 1988.

[9] A. I. Abdel-Fattah, M. S. El-Genk, and P. W. Reimus, "On Visualization of Sub-Micron Particles with Dark-Field Light Microscopy," *Journal of Colloid and Interface Science*, vol. 246, pp. 410-412, 2002.

[10] F. C. A. Groen, I. T. Young, and G. Ligthart, "A comparison of different focus functions for use in autofocus algorithms," *Cytometry*, vol. 6, pp. 81-91, 1985.

[11] N. Kehtarnavaz and H. J. Oh, "Development and real-time implementation of a rule-based auto-focus algorithm," *Real-Time Imaging*, vol. 9, pp. 197-203, 2003.

[12] A. Santos, C. O. De Solorzano, J. J. Vaquero, J. M. Pena, N. Malpica, and F. Del Pozo, "Evaluation of autofocus functions in molecular cytogenetic analysis," *Journal of Microscopy-Oxford*, vol. 188, pp. 264-272, 1997.

[13] J.-M. Geusebroek, F. Cornelissen, W. M. Arnold, and H. G. Smeulders, "Robust autofocusing in microscopy," *Cytometry*, vol. 39, pp. 1-9, 2000.

[14] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277 1294, 1993.

[15] I. T. Nabney, "NETLAB," Software, http://www.ncrg.aston.ac.uk/netlab/index.php, 2003.

[16] P. Li and J. R. Flenley, "Pollen texture identification using neural networks," *Grana*, vol. 38, pp. 59-64, 1999.

[17] Y. Zhang, "Pollen Discrimination Using Image Analysis," Massey University, Palmerston North report, 2001-2003 2003.

[18] P. Etheridge, "Discrimination of Pollen Taxa from Digital Image Feature Data," Massey University, Palmerston North, Report July 28 2005.

[19] D. Brookes and K. W. Thomas, "the distribution of pollen grains on microscope slides. Part 1. The non-randomness of the distribution," *pollen et spores*, vol. IX, pp. 621-629, 1967.

C. Data Sheets

These data sheets are embedded in PDF format and will only be viewable from within the soft copy of this thesis; an accompanying CD.

11.1.1 High Magnification Camera

Adobe Acrobat 7.0 Document

Micro-Pix technical reference manual

Adobe Acrobat 7.0 Document

MicroPix specification

11.1.2 High Magnification camera sensor

Adobe Acrobat 7.0 Document

MicroPix sensor manual – Sony ICX204AL

11.1.3 Stepper Motor Drivers and RS422 Controller

Adobe Acrobat 7.0 Document

mStep-407 hardware manual with SIN-11 serial line converter

Adobe Acrobat 7.0 Document

Stepper driver, SMC-40 (v1.07) Software Guide

11.1.4 Power Supplies

Adobe Acrobat 7.0 Document

Power supplies data sheets

11.1.5 Linear Stage Movement

Adobe Acrobat 7.0 Document

Linear Product Manual

D. Raw Data From Verification Testing

The raw data as collected appears in Table 11-1. There was a change noticed in the lighting which affected the data slightly so more collections were made once the problem was fixed. The results are not markedly different as the noticeable change was in the count of detritus and some additional errors in classification were fixed by the user adjustment.

Table 11-1: Raw data from verification testing.

		raw count						user adjusted						
	Detritus	P36	P119	P64	P148	P147	P136	P36	P119	P64	P148	P147	P136	
SLIDE-a	22	58	7	12	61	2	7	56	16	12	63	1	7	
	17	59	13	11	62	2	6	53	17	9	64	1	6	
	22	61	17	19	68	7	8	61	15	11	66	1	7	
	25	64	18	13	66	0	8	56	12	13	65	1	9	
SLIDE-b	20	37	60	26	29	6	6	54	22	5	68	0	5	lighting
	25	58	41	16	39	8	6	61	23	4	68	0	5	lighting
	26	36	68	29	33	5	5	45	18	5	74	0	5	
	40	90	20	7	75	4	7	74	22	5	80	1	5	lighting??
	20	51	38	8	64	8	5	58	29	6	71	0	5	
	18	62	34	10	68	5	7	61	27	6	75	1	5	
	16	49	36	12	53	5	9	64	18	5	74	1	5	
SLIDE-c	212	132	19	30	77	0	13	98	5	17	73	0	10	lighting??
	172	95	31	18	69	20	14	74	13	14	73	0	12	lighting??
	79	97	24	10	98	9	27	74	19	5	95	0	11	
	57	62	28	9	85	3	13	58	18	6	83	0	11	
SLIDE-d	101	81	21	3	112	0	6			not counted				lighting
	100	95	22	7	106	3	5	78	18	7	86	0	0	lighting
	46	70	52	10	75	0	3	52	27	9	81	0	1	lighting??
	52	51	48	34	72	4	2	58	28	9	82	0	1	
	66	62	40	12	77	4	2	62	28	6	87	0	1	
	33	49	56	17	47	0	2	60	24	10	84	0	1	
	26	46	77	28	36	23	12	65	22	8	78	0	1	

E. mlpfwd – NETLAB help file

11.1.6 Purpose

Forward propagation through 2-layer network.

11.1.7 Synopsis

```
y = mlpfwd(net, x)
[y, z] = mlpfwd(net, x)
[y, z, a] = mlpfwd(net, x)
```

11.1.8 Description

`y = mlpfwd(net, x)` takes a network data structure `net` together with a matrix `x` of input vectors, and forward propagates the inputs through the network to generate a matrix `y` of output vectors. Each row of `x` corresponds to one input vector and each row of `y` corresponds to one output vector.

`[y, z] = mlpfwd(net, x)` also generates a matrix `z` of the hidden unit activations where each row corresponds to one pattern.

`[y, z, a] = mlpfwd(net, x)` also returns a matrix `a` giving the summed inputs to each output unit, where each row corresponds to one pattern.

F. MLP - NETLAB help file

11.1.9 Purpose

Create a 2-layer feedforward network.

11.1.10 Synopsis

```
net = mlp(nin, nhidden, nout, func)
net = mlp(nin, nhidden, nout, func, prior)
net = mlp(nin, nhidden, nout, func, prior, beta)
```

11.1.11 Description

`net = mlp(nin, nhidden, nout, func)` takes the number of inputs, hidden units and output units for a 2-layer feed-forward network, together with a string `func` which specifies the output unit activation function, and returns a data structure `net`. The weights are drawn from a zero mean, unit variance isotropic Gaussian, with varianced scaled by the fan-in of the hidden or output units as appropriate. This makes use of the Matlab function `randn` and so the seed for the random weight initialization can be set using `randn('state', s)` where `s` is the seed value. The hidden units use the `tanh` activation function.

The fields in `net` are

```
  type = 'mlp'
  nin = number of inputs
  nhidden = number of hidden units
  nout = number of outputs
  nwts = total number of weights and biases
  actfn = string describing the output unit activation function:
      'linear'
      'logistic
      'softmax'
  w1 = first-layer weight matrix
  b1 = first-layer bias vector
  w2 = second-layer weight matrix
  b2 = second-layer bias vector
```

Here `w1` has dimensions `nin` times `nhidden`, `b1` has dimensions `1` times `nhidden`, `w2` has dimensions `nhidden` times `nout`, and `b2` has dimensions `1` times `nout`.

`net = mlp(nin, nhidden, nout, func, prior)`, in which `prior` is a scalar, allows the field `net.alpha` in the data structure `net` to be set, corresponding to a zero-mean isotropic Gaussian prior with inverse variance with value `prior`. Alternatively, `prior` can consist of a data structure with fields `alpha` and `index`, allowing individual Gaussian priors to be set over groups of weights in the network. Here `alpha` is a column vector in which each element corresponds to a separate group of weights, which need not be mutually exclusive. The membership of the groups is defined by the matrix `indx` in which the columns correspond to the elements of `alpha`. Each column has one element for each weight in the matrix, in the order defined by the function `mlppak`, and each element is 1 or 0 according to whether the weight is a member of the corresponding group or not. A utility function `mlpprior` is provided to help in setting up the `prior` data structure.

`net = mlp(nin, nhidden, nout, func, prior, beta)` also sets the additional field `net.beta` in the data structure `net`, where beta corresponds to the inverse noise variance.

G. Software Description

Introduction

The AutoStage control software is written in Matlab in one ‘m’ file but is separated into functions within the same file. Matlab facilitates development by having many pre-written functions and uses an interpreted scripting that is quick to implement. The control of both cameras was performed by an image acquisition toolbox which could control them directly from within the Matlab program.

The software described here controls the AutoStage to capture images and save them to a directory. It asks the user for features from a library of known pollen features, trains a neural network and classifies the saved images into the groups as selected by the user. This is the complete set of operations as performed for final all-of-system testing performed.

11.1.12 The System Control Software

Named mainStage.m, the Matlab ‘m’ file contains seven functions of which one is the main function. The other, sub-functions, are: movestage; stepstage; focus; segim1; segim2; loclmax, featXtract and trainNN. These functions: move the stage (XY stepper motors); focus (Z stepper motor) segment captured images from cameras one and two; find the peaks or local maximums in a vector, extract features from image files and train neural networks with supplied feature data (see Sub-Functions).

11.1.12.1 Main Function

The main function finds the stepper motors on the RS232 serial link (converted to RS422 by a device supplied with the motor drivers to run all, 3 in this case, on the same bus), finds cameras; one on the USB serial link and one on the IEEE1394 or “FireWire” serial link.

The cameras’ parameters (contrast, exposure, etc) are setup.

A dialog box allows the user to select a major folder for file storage and each session is stored in a folder below that, in a uniquely (based on time and date) named folder.

MoveStage is used to allow the user to move to the two extreme corners of the area of interest. At a place between where there is some pollen, an auto-focus is performed and the ‘z’ position set to zero. Once accepted as focussed by the user, the second corner is marked and the area is divided into camera view-sized rectangles and an image taken at each position. The images are saved, named to indicate their row/column position in the area of interest.

Once all images are taken they are all segmented and those files are saved with “segment” added to the original images name.

Image names and location vectors are added to a data file stored in the folder.

The high-magnification camera now moves to the first object found and focuses. A course focus followed by a narrow and fine focus is used for the first position and just the fine focus after that. If the software detects a bad focus, it goes back to doing a course focus first. Each focus starts at the zero focus position as found by the focus done with lo-res camera, but adjusted up for the hi-res camera.

After focussing, the hi-res camera takes its image which is immediately segmented. If it fails to find a ‘good’ object, the image is not saved and its name is added to a ‘bad image’ file list in the data file (“data.mat”). The segmentation function returns an image that is closely cropped around the valid object found, which is saved into a folder below the lo-res camera folder labelled “hiRes”. The whole image and the closely cropped image are again both saved together.

11.1.12.2 Sub-Functions

11.1.12.2.1 Setting the Stage in Motion - Movestage

This function allows movement of the stage by the user controlling it from the keyboard. Using the numbers with arrows on them, the stage can be moved a constant speed in any of the four directions. Two may be used to move diagonally. Any other key stops all movement.

11.1.12.2.2 Stepping the Stage by a Fixed Amount - Stepstage

Stepstage moves the stage by a given xy co-ordinate amount from its present location.

11.1.12.2.3 Segmentation in High Magnification Images - SegIm1

Segmentation of images from camera 1 (hi-res camera) is performed.

A ‘canny’ edge detector finds edges, producing a two level, Black and White image. The edges are dilated to join any small breaks in outlines of objects. A fill is performed make a solid “blob” of any enclosed shapes. Any white pixels near the edge, and all white neighbours of those pixels, are removed. Erosion then reduces the size of any (white) “blobs” back to about the same size as the object in the original image. Bounding rectangle, area of blob, and convex hull are defined.

The area of the blob is used to determine if the object is the right size to be a pollen grain.

Bounding rectangle aspect ratio is used to determine if it is elongated in X or Y directions and the ratio of pollen area to bounding rectangle eliminates any elongated angled blobs.

The ratio of blob area to Convex Hull area indicates non-regular shapes. Pine pollen do have concavities in their shape so once a method of determining the presence of a pine pollen by other means during segmentation is found, then the Hull can be set to be more effective in eliminating other irregular shapes.

The centre of the object is found, measures to the outer edges of the object determined and that portion of the original image is saved as the final image to go toward feature extraction and classification. A variable sets the area calculation to add a small amount of image area so that the object sits within the borders of the image, rather than touching.

The image is only accepted if the objects centre is within a certain radius of the centre of the image. This allows for inaccuracies and resultant offsets in placement of the pollen under high magnification microscope. To eliminate multiple images of the same pollen, each pollen found has its location in steps from a fixed location on the slide recorded and each subsequent pollen found is checked against the list. If a pollen is found that has already had its location recorded then the next nearest pollen is considered.

11.1.12.2.4 Segmentation in Low Magnification Images - SegIm2

This algorithm is essentially the same as segIm1 but with values changed to suit the lower magnification image and smaller sizes of objects expected. It returns only the centre locations of the ‘blobs’ found so they can be found again by the high magnification camera. The original captured image and its segmented version is saved as a by-product for later evaluation.

Finding Local Maximum - LclMax

This is associated with focussing. A series of focus images produces a series, or vector, of numbers, each with a value that indicates how “focussed” the associated image is. LclMax takes that vector and returns the values of local peaks, or maximums, within the vector and the index to those values in the original vector. The peaks are in order of local height so outside of the function, the maximum peaks, the maximum value and the order of peaks in the series can be extracted. The latter is done by reordering the indices (ascending or descending).

This function runs along the input vector and if it finds a low1-high-low2 value it sets it as a peak and calculates the height of the peak as: high-(low1+low2)/2.

LclMax also returns the maximum value overall and indicates if that value is at either end of the vector.

11.1.12.2.5 Focussing - focus

Focus is used for both cameras. The focus data output is used in each case in an algorithm in the main function to suit the camera and situation.

This function moves the cameras up from input arguments that determine number-of-images and number-of-steps between images.

There are 11 focus algorithms to choose from when calling the function. They are: standard deviation; variance; normalised variance; gradient (maximum of X and Y directions at each point used); vollath4 (so named by the developer) vollath5; derivative; histogram; power; entropy; Fourier Transform.

A region of interest can be defined that reduces the image to some portion of the original. This reduces computation time of the focus algorithm. It is set to the area in which the object is expected to be (given errors in the system in finding the objects between cameras). This makes it more likely to find focus on the object and not be influenced by surrounding objects that may be in a different focal plane.

A background image is subtracted from, and its mean pixel value added back to, the focus image.

11.1.12.2.6 Neural Network Training – trainNN

trainNN asks the user for files of image features of known pollen types to use to train the neural network. Each time the system runs, the expected types of pollen can be selected from a library of pollen features, including detritus and bubbles.

11.1.12.2.7 Feature Extraction - featXtract

Feature extraction is a series of “get-feature” m-files written by Zhang[74] as implementations of the features selected for use with pollen discrimination.

featXtract takes all images in the current folder and calls each get-feature algorithm until a complete set of features (43) has been compiled. It is stored in a matrix with features in columns and images in rows. The data is then normalised using the same parameters that were used to normalise the data used for the trained network and then presented to the trained neural network for classification

H. Software Source Code – Matlab M files

NetLab is required as is a slightly modified version of the confusion matrix viewing algorithms. These are found on the accompanying CD.

11.1.13 Main Code Module – mainStage.m

```
function stage()
%Program to control STAGE for Pollen Project at Massey University
%Allows user to define a region of interest and moves camera across the
%region taking overlapping images, segments pollen in images and determines
%their position within the region of interest.  Then moves the main camera
%to each identified pollen, takes its image, cuts the rectangle
%containing only that pollen and stores it for classification. Classifies
%and counts pollens and flags those that are not certainly identifiable for
%optional classification by an operator.
close all;
clear all;
closepreview;
% Written by Gary Allen 2006

data=[];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
maxTarget = 100000;% select maximum number of pollen to collect
folderName = 'TSTd7-';%adds date-time stamp to this name to ensure uniqueness

%motors give 2.6um per step @ 1/10th microstepping
%Camera2 is 8mmx6mm FOV.
C2FOV=[8E-3, 6E-3]; %field of view - not used
C1FOV=[4.3E-4, 3.23E-4]; %field of view
C1rc = [768;1024];
C1xy = ['172'; '127']; %Was 165 123 step length of image - must be same length string

C2xy = ['672'; '900']; %672,904;WAS 670;877(673;904)(;)  step length of image - must be same length strings
overLap = ['0040'; '0040'];%Border not considered in Image 20 steps is 1/2 of 100 micron pollen

mPerStep = 2.6E-6; %meters per step
C2toC1=['13020'; '00122';'00345']; %was 13010;00122;00345- must be same length strings
stepsPerPixelr=1.454;% used for cam2 - pixels dont seem to be square
stepsPerPixelc=1.41; % so the Y direction has a different value

C1exposure=1023;
C1contrast=350;%325
C1brightness=250;
C2exposure=38;%was 35  was 30


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% get path to save files
topPath=uigetdir('D:\','Select Folder. Files saved in created subfolders');
cd(topPath);
nuDir=datestr(clock);
nuDir = strrep(nuDir, ' ', '');
nuDir = strrep(nuDir, ':', '');
nuDir = strrep(nuDir, '-', '');
nuDir=strcat(folderName,nuDir);
```

```
mkdir(nuDir);
cd (nuDir);
%%add folder for Hi-res, LoRes images and data.Folder above is for final
%%pollen images so classification SW can find them
mkdir('data');
cd ('data');
% cd (topPath);

%+++open port
serPort=instrfind('Port','COM1');
if ~isempty(serPort)
fclose(serPort);
end
serPort = serial('COM1','BaudRate',9600,'DataBits',8);
fopen(serPort);

%+++set stepper drivers to party mode and retrieve x & y handles

 fprintf(serPort,'%s','&','async');
 pause(8);
 ret=fscanf(serPort,'%s');
if isequal(ret,'xyz')==1
else
   fprintf(serPort,char(13),'async');
   pause(1);
   ret=fscanf(serPort,'%s');
   pause(1);
   if isequal(ret,'#')==1
   else
   error('stagerr1:stageCamFocMSy', 'connection suspect\n');
   end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%% IMAGING SET UPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
cams=imaqhwinfo('winvideo');
camNum = size(cams.DeviceIDs,2);
if camNum==2
   if (isequal(strtrim(char(cams.DeviceInfo(1).DeviceName)), '1394 Streaming Digital Camera')) &&
(isequal(strtrim(char(cams.DeviceInfo(2).DeviceName)), 'VGA USB Camera'))
      T.cam2in=videoinput('winvideo',2, 'RGB24_640x480' );
      T.setcam2=set(T.cam2in);
      T.cam2s=getselectedsource(T.cam2in);
      T.setcam2s=set(T.cam2s);
      pause(.1);
      T.cam1in=videoinput('winvideo',1,'Y800_1024x768' );
      T.setcam1=set(T.cam1in);%set common camera proerties
      T.cam1s=getselectedsource(T.cam1in);
      T.setcam1s=set(T.cam1s);%set camera source properties, unique to camera
   elseif  (isequal(strtrim(char(cams.DeviceInfo(2).DeviceName)),  '1394 Streaming Digital Camera')) &&
(isequal(strtrim(char(cams.DeviceInfo(1).DeviceName)), 'VGA USB Camera'))
      T.cam2in=videoinput('winvideo',1, 'RGB24_640x480' );
      T.setcam2=set(T.cam2in);
      T.cam2s=getselectedsource(T.cam2in);

      T.setcam2s=set(T.cam2s);
      pause(.1);
      T.cam1in=videoinput('winvideo',2,'Y800_1024x768'  );
      T.setcam1=set(T.cam1in);%set common camera proerties
```

```
        T.cam1s=getselectedsource(T.cam1in);
        T.setcam1s=set(T.cam1s);%set camera source properties, unique to camera
    end
else error('errCamerAcq2','Must be Two cameras attached\n');
end

set(T.cam1s,'BacklightCompensationMode','auto');
set(T.cam1s,'BacklightCompensation','on');
set(T.cam1s,'Brightness',C1brightness);
set(T.cam1s,'ContrastMode','manual');
set(T.cam1s,'Contrast',C1contrast);
set(T.cam1s,'ExposureMode','manual');
set(T.cam1s,'Exposure',C1exposure);

set(T.cam2s,'Brightness',0);
set(T.cam2s,'ColorEnable','off');
set(T.cam2s,'ExposureMode','manual');
set(T.cam2s,'Exposure',C2exposure);
set(T.cam2s,'FrameRate','30.0000');
set(T.cam2s,'HorizontalFlip','off');
set(T.cam2s,'Hue',0);
set(T.cam2s,'Saturation',13);
set(T.cam2s,'Sharpness',6);
set(T.cam2s,'VerticalFlip','off');


% get(T.cam1in,'source') %source should be one only and set to on
% get(T.cam2in,'source') %modify here if problems later
%  T.cam1in.SelectedSourceName = 'input1'; %to change from source 0 to 1
% preview, stoppreview, closepreview (T.cam1in)
%  start,  stop(T.cam1in)
%delete(T.cam1in)

%set trigger
triggerconfig(T.cam1in, 'manual');
triggerconfig(T.cam2in, 'manual');
triggerconfig(T.cam1in, 'Immediate');
triggerconfig(T.cam2in, 'Immediate');
set(T.cam1in,'TriggerFrameDelay',0);
set(T.cam2in,'TriggerFrameDelay',0);
set(T.cam1in,'FramesPerTrigger',1);
set(T.cam2in,'FramesPerTrigger',1);


%Focus Motor set up (should perhaps be set up in driver memory)

fprintf(serPort,'zE254','async');%hold motor on fully for
    pause(0.2);                    %YEnnn ms before hold current
    fscanf(serPort,'%s');

cam1Bgnd = imread('D:\Acode\cam1Bgnd','tif');%read in background images
cam2Bgnd = imread('D:\Acode\cam2Bgnd','tif');% these may be read in as part of
                                             % each user set up - ?

%% END IMG set ups
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf(serPort,'zI56','async'); %56->
    pause(1);
    fscanf(serPort,'%s');

    fprintf(serPort,'zV56','async');%56->
    pause(1);

    fscanf(serPort,'%s');

    fprintf(serPort,'zK0 0','async');
    pause(1);
    fscanf(serPort,'%s');

    fprintf(serPort,'zE250','async');%hold motor on for nnn<255=n.nns until hold current reduces
    pause(1);
    fscanf(serPort,'%s');

%% Set up stage coordinates
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%+++move stage to lower left corner of area of interest and set
%co-ordinates to zero
preview(T.cam2in);

fprintf('move camera2 to the bottom right corner of the section to be scanned \n');
movestage(serPort);
fprintf(serPort,'xO','async');%mark position as X's origin
xhome = '';
while isempty(xhome);
     pause(.4);
     xhome=fscanf(serPort,'%s');
     xhome=strtrim(xhome);
     if ~isempty(xhome)
         if xhome(1)=='?'
             error('error in xhome sending xO0 to serPort \n');
         end
     end
    % fprintf('xhome in while=%s \n',xhome); %need error message here?
end

fprintf(serPort,'yO');%mark posistion as Y's origin
yhome = '';
while  isempty(yhome);
    pause(.4);
    yhome=fscanf(serPort);
    yhome=strtrim(yhome);
    if ~isempty(yhome)
         if yhome(1)=='?'
             error('error in yhome sending yO0 to serPort /n');
         end
     end
    %fprintf('yhome in while= %s\n',yhome);%need error message here?
end
xhome=xhome(2:size(xhome,2));
yhome=yhome(2:size(yhome,2));
fprintf('serPort set at zero position %s, %s \n',xhome,yhome);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%


% set z axis to zero
fprintf(serPort,'zO');%mark posistion as z's origin
pause(1);
zPos = fscanf(serPort);
fprintf('User Set Focus set to zero: %s\n',zPos);
```

```
%
initFocus = 'f';
%Initial Cam2 Focus repeat until user satisfied
while ~(isequal(initFocus,'c') || isequal(initFocus,'C'))
  fprintf('move camera2 to some central area where there is lots of detail \n');
  movestage(serPort);
  goBack = 30;
  xtra = 30;
  fprintf(serPort,strcat('z-',num2str(goBack+xtra)),'async');
  pause(0.2);
  fscanf(serPort);
  fprintf(serPort,strcat('z+',num2str(xtra)),'async');
  pause(0.2);
  fscanf(serPort);

   %
  stepSize = 2;
  stepNum = ceil(goBack*2/stepSize);
  algorithm = 'sDev';            %%ADJUST vars for focus on cam2

  % FIND ROI BY SELECTING AREA WITH MOST EDGES...???
  ROIcam2 = [100 100 440 280];
    [steps,useMax,focThold] = focus(serPort, T.cam2in, stepNum, stepSize, ROIcam2,
algorithm,3,cam2Bgnd);
  %While not case 1 else, add to go back checking not greater than max dist to slide...

  switch size(steps,2)
    case 4
      if useMax == 1
        step2 = steps(1,1);%max value
      else
        step2 = steps(1,2);%steepest peak
      end
    case 3
      if useMax == 1
        step2 = steps(1,1);%max value(Consider first peak here)
      else
        step2 = steps(1,2);%steepest peak
      end
    case 2
      if useMax == 1
        step2 = steps(1,1);%max value
      else
        step2 = steps(1,2);%steepest peak
      end
    case 1
      if useMax == 1
        step2 = steps(1,1);%max value
      else
        fprintf('no peaks and max is at one end or other \n');
        step2 = steps(1,1);%steepest peak
      end
    otherwise
      error('woops1');
  end%switch


  focusSteps = (step2)*stepSize;
  fprintf(serPort,strcat('z+',num2str(focusSteps)),'async');%move back to focus point
  pause(0.2);
  fscanf(serPort);
```

```
 initFocus = input('Press "r" to refocus, else press "c" to continue','s' );
end %if initFocus

fprintf(serPort,'zZ','async');
pause(1);                    %check FOCUS position
autoFoc = fscanf(serPort);% => zZ z-nnn so later take from 5th position only...

fprintf('First Focus: %s',autoFoc);%print position
%% END 1st FOCUS


%+++move stage to one corner of area of interest and get co-ordinates
fprintf('Move camera2 to the top left corner of the section to be scanned \n');
movestage(serPort);

fprintf(serPort,'xZ','async');
xfar = '';
while isempty(xfar);
    pause(1);
    xfar=fscanf(serPort);
    xfar=strtrim(xfar);
    if ~isempty(xfar)
       if xfar(1)=='?'
          error('error in xfar sending xZ to serPort \n');
       end
    end
  % fprintf('xfar in while= %s \n',xfar);%need error message here?
end

fprintf(serPort,'yZ','async');
yfar = '';
while isempty(yfar);
    pause(1);
    yfar=fscanf(serPort);
    yfar=strtrim(yfar);
    if ~isempty(yfar)
       if yfar(1)=='?'
          error('error in yfar sending yZ to serPort');
       end
    end
   %fprintf('yfar in while= %s \n',yfar);%need error message here?
end
xfar=xfar(5:size(xfar,2));
yfar=yfar(5:size(yfar,2));
data.regionSize={xfar; yfar};
%fprintf('serPort set at far corner of Area Of Interest. %s, %s \n',xfar,yfar);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% have coordinates of  Area Of Interest, focussed - now start capturing on Camera2.


OlapAdjX = (str2double(C2xy(1,:))-(str2double(overLap(1,:))));
OlapAdjY = (str2double(C2xy(2,:))-(str2double(overLap(2,:))));

imRows = ceil(abs(str2double(yfar)/(OlapAdjY)))+1; %calc number of rows to capture
imCols = ceil(abs(str2double(xfar)/(OlapAdjX)))+1; %calc number of columns to capture
imNames = {};

for x = 1:imRows
```

```
  y=1;
  pause(0.1);
  closepreview;
  preview(T.cam2in);
  pic2 = getsnapshot(T.cam2in);
  pic2 = pic2 - cam2Bgnd + mean2(cam2Bgnd);
  pause(0.5);
  imwrite(pic2,strcat(num2str(x),'-',num2str(y),'.tif'),'TIFF');
  imNames(x,y)={strcat(num2str(x),'-',num2str(y),'.tif')};
  for y = 2:imCols
    stepStage(serPort,num2str(OlapAdjX),'0');
    pause(4);%stage move time
    closepreview;
    preview(T.cam2in);
    pic2 = getsnapshot(T.cam2in);
    pic2 = pic2 - cam2Bgnd + mean2(cam2Bgnd);
    pause(0.2);%stage stabilisation time
    imwrite(pic2,strcat(num2str(x),'-',num2str(y),'.tif'),'TIFF');
    imNames(x,y)={strcat(num2str(x),'-',num2str(y),'.tif')};
  end
  if x ~= imRows
    fprintf(serPort,strcat('xR',xfar),'async');%move to position 'xfar' relative to origin
    fscanf(serPort,'%s');
    stepStage(serPort,'0',num2str(OlapAdjY));
    pause(5);%stage move time
  else
     data.imNames = imNames;
  end
end
closepreview;%cam2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Have images on disk, segment images

zz=0;
ROILocs=[];
%  p and q are rows and columns respectively of  Cam2 images because imNames is saved
%  as rows/cols in a matrix the same pattern as the images were taken.

for p=1:size(imNames,1) %row
  for q=1:size(imNames,2) %col
    zz=zz+1;
    imName=imNames(p,q); %get name back
    tempim = imread(char(imName)); %read image
    [imOut,pollLocs] = segIm2(tempim,cam2Bgnd);%segim2 function
    %rename file to indicate it is segmented
    imName = strrep(imName,'.tif','');
    segImName = char(strcat(imName,'seg.tif')); %add 'seg' to image name
    imwrite(imOut,segImName,'TIF'); %write image
    data.(strcat('Loc',num2str(zz)))=pollLocs;
      imCentreX = str2double(xfar) + ((q-1)*(str2double(C2xy(1,:))-str2double(overLap(1,:))));
      imCentreY = str2double(yfar) + ((p-1)*(str2double(C2xy(2,:))-str2double(overLap(2,:)))); %FIX stage is -
ve in Y direction
      %imCornerX = imCentreX + (round(str2double(C2xy(1,:))/2) - str2double(overLap(1,:)));
      %imCornerY = imCentreY - (round(str2double(C2xy(2,:))/2) + str2double(overLap(2,:))); %FIX stage is
-ve in Y direction
      imCornerX = imCentreX + round(str2double(C2xy(1,:))/2);
      imCornerY = imCentreY - round(str2double(C2xy(2,:))/2); %FIX stage is -ve in Y direction

    for r = 1:size(pollLocs,1) %for each pollen found in an image, calculate the location within ROI
      pX = imCornerX - round(pollLocs(r,1)*stepsPerPixelr);
      pY = imCornerY + round(pollLocs(r,2)*stepsPerPixelc);
```

```
            ROILocs = [ROILocs; [pX, pY]];  %Pollen locations x,y steps in region of interest on slide
        end
    end
end

%Add indicators of locations found that are close
%this code not used yet - maybe used to check if two images are taken of
%the same pollen
closeEnough = 90;%90
for d = 1:size(ROILocs,1)-1
    for e = (d+1):size(pollLocs,1)
        tooClose = (abs(ROILocs(e,1)-ROILocs(d,1))<closeEnough) && (abs(ROILocs(e,2)-ROILocs(d,2))<
closeEnough);
        if tooClose
            ROILocs(d,3)= double(tooClose);
            ROILocs(e,3)= tooClose;
            indx = find(ROILocs(e,:));
            indx = indx(size(indx,2))+1;
            ROILocs(e,indx)= d;
        end%if
    end%for e
end%for d

data.ROILocs = ROILocs;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Have ROI locations list as xy step numbers for total ROI - now move cam1
% to each and take image:
preview(T.cam1in);
%move CAM1 to focus position relative to CAM2,
    fprintf(serPort,strcat('z+',(C2toC1(3,:))),'async');
    pause(0.3);         %move to cam1 focus (from cam2)
    fscanf(serPort);



%move cam1 to origin using stage measures and reset origin
fprintf(serPort,strcat('xR',C2toC1(1,:)),'async');
pause(1);
fscanf(serPort);
fprintf(serPort,strcat('yR',C2toC1(2,:)),'async'); %move cam2 to origin...
pause(15); %%??? find better way to determine drive has ended
fscanf(serPort);
fprintf(serPort,'yO','async');
pause(0.2);
fscanf(serPort);
fprintf(serPort,'xO','async'); %And mark it as new origin for stage
pause(0.2);
fscanf(serPort);


badimg=[];%save bad image numbers - where segIm1 finds nothing
%input('Moved to focus from cam2 to Cam1 focussed?\n');

%focHome = str2double(C2toC1(3,:));
autoFoc = autoFoc(5:size(autoFoc,2));
wideFoc = 0;
found = [0,0];
finalLoc=[];
%MAIN LOOP
for s = 1:min(maxTarget,size(ROILocs,1))%maxTarget, alter in code at top
```

```
 %drive focus to user focussed cam2 adjusted for cam1
 fprintf(serPort,strcat('zR+',num2str(str2double(autoFoc)+str2double(C2toC1(3,:)))),'async');
 pause(1);
 focPos = fscanf(serPort);

 picLocX = ROILocs(s,1);
 picLocY = ROILocs(s,2);
 fprintf(serPort,strcat('xR',num2str(picLocX)),'async');%move to position relative to origin
 pause(.4);
 fscanf(serPort);
 fprintf(serPort,strcat('yR',num2str(picLocY)),'async');%move to position relative to origin
 pause(.4);
 fscanf(serPort);
 pause(.4);
 focusing = 1;
 ROIcam1 = [337 209 350 350]; %made square
 ROIcam1Plus = [262 134 500 500];%[302 174 420 420];%[322 224 380 380]bigger to make a useful circle for
near centre limit in segIm1
 % make Bground image same size as reduced image ready for subtraction
 % inside loop
 roiA = ROIcam1Plus(1,2)+1;
 %b=ROI(1,2) + ROI(1,4);
 roiC = ROIcam1Plus(1,1)+1;
 %d = ROI(1,1) + ROI(1,3);
 cam1BgndIn = cam1Bgnd(roiA:size(cam1Bgnd,1),roiC:size(cam1Bgnd,2));



 %FOCUS PRELIM
 if s == 1 %always do a coarse focus on first 'pollen' found
   wideFoc = 0;%Turn off wide focus with 0 turn on with 1
 end

 while focusing == 1;
   if wideFoc == 1
     wideFoc = 0;
     %add reset to focus here??

       fprintf(serPort,strcat('zR+',num2str(str2double(autoFoc)+str2double(C2toC1(3,:)))),'async')
       pause(0.2);      %move to cam1 focus (from cam2) and below for refocus upward
       fscanf(serPort);

       xtra = 30;
       goBack = 100; %go back half amount to refocus; fine resolution
       %fprintf('goback should be 100: %s and: %d \n',num2str(goBack),goBack);%ok
       fprintf(serPort,strcat('z-',num2str(goBack+xtra)),'async');
       pause(0.2);      %move to cam1 focus (from cam2) and below for refocus upward
       fscanf(serPort);
       fprintf(serPort,strcat('z+',num2str(xtra)),'async');
       pause(0.2);
       fscanf(serPort);
       % END FOCUS PRELIM


       stepSize = 10;
       stepNum = goBack*2/stepSize;
       algorithm = 'deriv';             %%ADJUST vars for focus on cam1
       %ROIcam1 = [320 230 380 300];%now back above while statement
         [steps, useMax, focThold] = focus(serPort, T.cam1in, stepNum, stepSize, ROIcam1,
algorithm,3,cam1Bgnd);
```

```
            switch size(steps,2)
                case 4
                    if useMax == 1
                        step3 = steps(1,1);%max value
                    else
                        stepSteep = steps(1,2);%steepest peak
                        [valSort, sortIndx] = sort(steps(2,2:4),'descend');%get highest peaks in desc order
                        sortIndx = sortIndx+1;%account for 1st being max value - sorting LocalMax values
                        stepHigh = steps(1,sortIndx);%get back to index numbers of largest values in order
                        %stepHigh = steps(1,ndxSort(1));%Highest peak
                        if stepHigh >= stepSteep
                            step3 = stepSteep;
                        else
                            step3 = stepHigh;
                        end
                    end
                case 3
                    if useMax == 1
                        step3 = steps(1,1);%max value
                    else
                        step3 = steps(1,2);%steepest peak
                    end
                case 2
                    if useMax == 1
                        step3 = steps(1,1);%max value
                    else
                        step3 = steps(1,2);%only peak
                    end
                case 1
                    if useMax == 1
                        step3 = steps(1,1);%max value
                    else
                        wideFoc = 1;
                        fprintf(serPort,'zZ','async');
                        pause(1);
                        zPos = fscanf(serPort);
                        fprintf('Focus-Bad FOCUS @ cam1-1st: %s',zPos);
                        fprintf('no peaks and max is at one end or other \n');
                        step3 = steps(1,1);% NOT GOOD NEED ANOTHER FOCUS
                        fprintf('NEED ANOTHER FOCUS @ Cam1-1st!! /n');
                    end
                otherwise
                    error('woops3');
            end%switch

        focusSteps = step3*stepSize;%
        fprintf(serPort,strcat('z+',num2str(focusSteps)),'async');%
        pause(0.3);
        fscanf(serPort);

%
    %input('Finished first focus on CAM1 \n');

        else %if wideFoc == 1
            goBack = 18;
            xtra = 20;
            fprintf(serPort,strcat('z-',num2str(goBack+xtra)),'async');
            pause(0.5);       %
            fscanf(serPort);
            fprintf(serPort,strcat('z+',num2str(xtra)),'async');
            pause(0.1);       %
            fscanf(serPort);
        stepNum = 25; %goBack*2;
        stepSize = 1;
        algorithm = 'gradMax';              %%ADJUST vars for focus on cam1
        %ROIcam1 = [320 230 380 300]; %change image size if required

        [steps, useMax, focThold] = focus(serPort, T.cam1in, stepNum, stepSize, ROIcam1, algorithm,
2,cam1Bgnd);
         focusing = 0;

         switch size(steps,2)

            case 3
%                    step4 = steps(1,2);%highest peak
                  [minVal minNdx] = min(steps(1,2:3));
                  [maxVal maxNdx] = max(steps(1,2:3));
               if  steps(2,minNdx+1) > focThold;
                  step4 = minVal;% first peak
               elseif steps(2,maxNdx+1) > focThold;
                  step4 = maxVal;%1st peak is lower than Thold 2nd peak is highest
               else
                  step4 = steps(1,1);%max value
               end
            case 2
               if useMax == 1
                  step4 = steps(1,1);%max value
               else
                  step4 = steps(1,2);%only peak
               end
            case 1
               if useMax == 1
                  step4 = steps(1,1);%max value
               else
                  focusing = 1;
                  fprintf(serPort,'zZ','async');
                  pause(1);
                  zPos = fscanf(serPort);
                  fprintf('Focus-Bad FOCUS @ Cam1-2nd: %s',zPos);
                  fprintf('no peaks and max is at one end or other \n');
                  %step4 = steps(1,1);% NOT GOOD NEED ANOTHER FOCUS
                  %fprintf(serPort,strcat('zR',num2str(C2toC1(1,3))),'async');
               end
            otherwise
               error('woops3');
          end%switch
        if focusing ~= 1
        focusSteps = (step4)*stepSize;
        fprintf(serPort,strcat('z+',num2str(focusSteps)),'async');%
        %fprintf('STEP4 = %d \n focusSteps = %d \n stepSize = %d \n',step4,focusSteps,stepSize);
     %input('Finished SECOND focus on CAM1 \n');
        pause(0.5);
        fscanf(serPort);

        end%if focusing
      end % if wideFoc == 1
    end %while focusing ==1



        %END FOCUS**
    closepreview;
```

```
   set(T.cam1in,'ROIposition',ROIcam1Plus);
   preview(T.cam1in);
   pic1 = getsnapshot(T.cam1in);
      %make back ground image same size as captured image
      cam1BgndIn = cam1BgndIn(1:size(pic1,1),1:size(pic1,2));
   pic1 = pic1 - cam1BgndIn + mean2(cam1BgndIn);
   pause(0.5);
   imwrite(pic1,strcat(num2str(s),'.tif'),'TIFF');%can remove this later
   pause(0.5);
   [pic11, found, foundLoc] = segIm1(pic1,ROIcam1,ROIcam1Plus,cam1Bgnd,found,finalLoc,ROILocs(s,1:2));
% pic11 is a small closely bordered image of a segmented pollen grain.

   if ~isempty(foundLoc)
   finalLoc = [finalLoc;foundLoc];%list of pollen found to eliminate finding same one again
   end%if
      if size(pic11)~=[0,0]
         cd ..;
         imwrite(pic11,strcat('pollen',num2str(s),'.tif'),'TIFF');
         pause(0.5);
         cd('data')
      else
         badimg = [badimg; {strcat('pollen',num2str(s),'.tif')}];
      end %if size(pic11)
end % s

data.badimg = badimg;
data.finalLoc = finalLoc;

%%TEMP move cameras back
fprintf(serPort,strcat('xR-',C2toC1(1,:)),'async');
pause(.5);
fscanf(serPort);
fprintf(serPort,'yR40','async');
%pause(15);
fscanf(serPort);

closepreview;
%%END TEMP

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CLEAN UP FROM IMAGING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save data data;
%closepreview(T.cam1in);
clear tempim;
%%!!! Clean up Cameras
%delete(T.cam1in);
%delete(T.cam2in);
%clear T;
% Close serial ports
fclose(serPort);
%clear all;
%% END CLEAN UP  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% IMAGES ON FILE, NOW CLASSIFY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%cd ..;%move to nuDir folder with images
%[name,imagePath]=uigetfile;
```

```
%parameters from main data base to convert mean and stdDev to same as main
%data base.
normParams =
[5030.134980307913,231.3034371643394,11.549145573441029,0.293437972241872,0.227959697576128,0.074001360
388508,0.000198687762285,0.000016941121767,0.000006335590846,0.000000000008587,0.00000001362427,-
0.000000000044924,0.440962893954756,0.220468737572616,0.955566969815511,6825.828280073598,117.263694951
66487,0.77756426949119,2485182.0254206937,1497.7238042250021,38.56475676785264,1101240.540736761,140219
3.7644309185,1954679.6813661144,4063279.210944027,3872055.6853244323,1537042871.0196767,1460751068.2200
918,15240603486338705000,0.008659205087587,0.005295770481277,0.004894526646877,0.004182615161962,0.0032
47431079919,0.003328716734365,0.002790500456022,0.003369855374229,99.09191585246701,0.908084147533,96.7
5340994541313,0.618414164599578,91.66260346632902,0.392911669629661;3460.333605618244,80.9061667604875
5,2.236268994033373,0.061566809328527,0.045626694361407,0.014966302126472,0.000453335049101,0.000028902
27307,0.000008106466735,0.000000000173313,0.000000092755801,0.000000000221719,0.187331184360503,0.05791
0192164941,0.059807099483805,5224.615706898697,86.89454714842559,0.221293882750971,5284786.335172836,34
17.2158562600184,23.568266770203916,2407543.8788592117,2893574.3608193165,3784126.675981967,7594492.505
268886,7092315.602556042,3913485349.063464,3633766697.0870805,77015982694969260000,0.003071396507139,0.
00164390615226,0.001663863105227,0.001319338601508,0.000996686773119,0.001058083429597,0.0008079262294
12,0.001098144523812,0.383358513513111,0.383358513513083,1.227158269430299,0.188171754579054,2.50855563
7726779,0.096577306785522;];
cd ..;
[normData, imList] = featXtract(normParams);
%imList is list in order of images used to present to MLP

%**********************************************************************
   %%%Commented parts below untested
   %%%uncomment if the data base of features requires a trained net on
   %%%certain selection of pollen types.
[trainedNet,randOrderImages] = trainNN();

%OR
% [netName,netPath] = uigetfile('*.*','select a trained net');
% trainedNet = load(strcat(netPath,netName));

%OR
%trainedNet = load('D:\Acode\trainedNet');
%classList is a cell of names of nornFeature files used to train net: the
%class or pollen type trained to.
classList = trainedNet.classList;
trainedNet = trainedNet.trainedNet;
%**********************************************************************

%classify normData from unknown images wiht trained net
classifNet = mlpfwd(trainedNet,normData);
%result is classifNet, matrix of imList x classList

%[images, classes]=size(classifNet);
%determine winning class from MLP result
[mxm, imClass] = max(classifNet, [], 2);


%RESULT TO PRINT: matrix of imageNames, classified type
classRes = imList;
classRes = {classRes, classList(imClass)};
close all;
%get a series of matrices of images of each classified type
for p = 1:size(classList,1)
   postn = find(imClass==p);
   tmp = imList(postn,:);
   %classSorted.(char(strcat(int2str(q),classList(q))))= imList(:,q);
   %classSorted.(char(strcat(classList(p),'class')))= imList(:,p);
   %figure(p), title (classList(p)), hold on;
   plotSize= 72;
   rLimt = ceil(size(tmp,1) / plotSize);
   %qLimt = size(tmp,1);
   if rLimt > 0
      for r = 1:rLimt
         if r == rLimt
            qLimt = mod(size(tmp,1),plotSize);
            if qLimt == 0
               qLimt = plotSize;
            end
         else
            qLimt = plotSize;
         end%if r
            figure(str2num(strcat(num2str(r),num2str(p)))), hold on;

         qLimt
         for q = 1:qLimt%plot each series as matrix plot
            imTemp = imread(char(tmp((r-1)*plotSize+q,1)));
            %subplot(floor(sqrt(size(tmp,1))) , ceil(size(tmp,1) / (floor(sqrt(size(tmp,1))))),q),imshow(imTemp);
            subplot(floor(sqrt(qLimt)) ,ceil(qLimt / (floor(sqrt(qLimt)))), q),imshow(imTemp),
title(strcat(classList(p),'-',num2str(r)))
         end%for q
      end %for r
   else
      figure(str2num(strcat('NILL',num2str(p)))),title(classList(p));
   end%if rLimt
   truesize;
   hold off;
   hgsave(strcat('fig-',num2str(p)));
   %save(class);
end%for p


%%#####################################################
%%  FUNCTIONS
%% ____________________________________________________
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% movestage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   function movestage(serialport)
%function to allow manual positioning of stage by user

uin='9';
fprintf('USE ARROWS ON KEYPAD WITH NUMLOCK ON \n');
while ~isequal(uin,'x')
   uin = input('"8"= +X,  "6"= +Y,  "4"= -X,  "2"= -Y,  \n Any other key stops movement  \n Type "x"  when
camera positioned \n','s');
   switch uin
      case '6'
         %fprintf('you typed %s \n',uin);
         fprintf(serialport,'xM0','async');
         fscanf(serialport);
         fprintf(serialport,'xM+1000','async'); %move until stopped
      case '2'
         %fprintf('you typed %s \n',uin);
         fprintf(serialport,'yM0','async');
         fscanf(serialport);
         fprintf(serialport,'yM+1000','async'); %FIX should change hwen reverse Y motor direction
```

```
        case  '4'
          % fprintf('you typed %s \n',uin);
           fprintf(serialport,'xM0','async');
           fscanf(serialport);
           fprintf(serialport,'xM-1000','async');
        case  '8'
           %fprintf('you typed %s \n',uin);
           fprintf(serialport,'yM0','async');
           fscanf(serialport);
           fprintf(serialport,'yM-1000','async'); %FIX should change hwen reverse Y motor direction
        case '0'
           %fprintf('x?? You typed %s !! \n',uin);
            fprintf(serialport,'*M0','async');
        otherwise
          % fprintf('other?? You typed %s !! \n',uin);
           fprintf(serialport,'*M0','async');
    end
  ret =  fscanf(serialport);
%   pause(0.2);
%   ret =  fscanf(serialport);
  ret=strtrim(ret);
  if ~isempty(ret)
     if ret(1) == '?'
         error('movestage1:stageCamFoc', 'key entry failure \n');
     end
  else
     fprintf('Check your input key usage \n');
     ret='9';
  end
  %fprintf('uin is: %s \n',uin);
end
%% END movestage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% stepStage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function stepStage(serialport,x,y)
% moves stage x and y steps using serialport object and string inputs
% representing x and/or y direction step numbers

if ~isequal(x,'0');
    retx='';
    while isempty(retx)
        if str2double(x)>=0
           fprintf(serialport,strcat('x+',x),'async');
        else
           fprintf(serialport,strcat('x',x),'async');
        end
        pause(0.2);
        retx = fscanf(serialport,'%s');
        if ~isempty(retx);
           if isequal(retx(1),'?');
               error('stepStage error in x');
           end
        end
    end
end


if ~isequal(y,'0');
    rety='';
    while isempty(rety)
        if str2double(x)>=0
           fprintf(serialport,strcat('y+',y),'async');
        else
            fprintf(serialport,strcat('y',y),'async');
        end
        pause(0.2);
        rety = fscanf(serialport,'%s');
        if ~isempty(rety);
            if isequal(rety(1),'?');
                error('stepStage error in y');
            end
        end
    end
end

%% END stepStage %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SEGMIM2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        function [imOut, pollLocs] = segIm2(img,bgnd)
% function [imOut, pollLocs] = segIm2(img,bgnd)
%
% Takes in an image and segments the pollen grains.
% Returns a matrix:
% Rows are of objects found
% Columns (2) are of row / col co-ordinates for object centres in the
% input image.
% Also returns a B&W image where white blobs are objects found


K=img(1:size(img,1),1:size(img,2));%remove third dimension if exists

%figure, imshow(K), title('original');
%L = edge(K, 'sobel', (graythresh(K) * .25));            %edge Thold ALTER
bgnd=bgnd(1:size(bgnd,1),1:size(bgnd,2));
Thrshold = graythresh(bgnd);%normalised value in range [0,1]
if Thrshold == 0
    Thrshold =0.0001;
end

L = edge(K,'canny',[Thrshold*0.1,Thrshold*1.7]);
 %figure, imshow(L), title('Edges');

se90 = strel('line', 2, 90);    % try alter middle param: original=3 ALTER
se0 = strel('line', 2, 0);      % try alter middle param: original=3 ALTER
M = imdilate(L, [se90 se0]);
N = imfill(M, 'holes');
P = imclearborder(N, 8);
 %figure, imshow(P), title('dilate, fill and clear border');
seD = strel('diamond',1);%original=1,
Q = imerode(P,seD);
Q = imerode(Q,seD);
%figure, imshow(Q), title('erodex2');
[L,n] = bwlabeln(Q,4);
regTholdlo = 5.5;    % 5.47=>10um diameter  <size of smallest pollen ALTER
regTholdhi = 5470; %5470=> 100um diameter <size of largest pollen ALTER
%imAvg = mean2(K);

for a=1:n
    [r,c] = find(L==a);
    nm = find(L==a);
    Sr = size(r,1);
```

```
  %Tim = mean(K(nm));
  maxR = max(r);
  minR = min(r);
  maxC = max(c);
  minC = min(c);

  bgndAdd = 20;%add to object size to estimate background ADJUST
  %Test if object is too near border and limit to border
  if maxR+bgndAdd > size(K,1)
    addR = size(K,1);
  else
    addR = maxR + bgndAdd;
  end

  if maxC+bgndAdd > size(K,2)
    addC = size(K,2);
  else
    addC = maxC + bgndAdd;
  end

  if minR-bgndAdd < 1
    minusR = 1;
  else
    minusR = minR - bgndAdd;
  end

  if minC-bgndAdd < 1
    minusC = 1;
  else
    minusC = minC - bgndAdd;
  end
  AvimObjPlus = mean2(K(minusR:addR,minusC:addC));%avg of area larger than object
  AvimObject = mean2(K(nm)); %avg of object
  AvimLocBgnd = (2*AvimObjPlus) - AvimObject;%avg of area that is not object (local bckground)
  if AvimObject > AvimLocBgnd %if object is darker than local background, don't count it.
      LocR = maxR-minR+1;%get some dimesions
      LocC = maxC-minC+1;%add 1 to avoid 0 if used as divisor
      LocBoundArea = (LocR * LocC);
      LocFit = Sr/LocBoundArea;
      LocSkew = (min(LocR,LocC))/(max(LocR,LocC));
    if(Sr > regTholdhi) || (Sr < regTholdlo);%compare with Thold
      Q(nm) = 0;%remove large or small
    elseif LocFit < 0.55 | LocSkew < 0.65;%pine set it at 0.55 | 0.65
      Q(nm) = 0;%remove long and thin
    else
      %eIm = zeros(size(Q));
      %eIm(nm) = 1;
      %figure, imshow(eIm), title('edges');hold on;
      %[re,ce] = find(eIm==1);
      [CH, CHa] = convhull(r,c,[]);
      LocHull = (Sr/CHa);%ratio of object area to localHull area
      %plot(ce(CH),re(CH),'r-');
      if LocHull < 0.97 %pine sets at 0.97
        Q(nm) = 0;%remove areas with large concaves
      end %if LocHull
    end %if Sr
  else
    Q(nm) = 0;
  end %if Tim >
end
```

```
pollLocs=[];
[L,n] = bwlabeln(Q,8);
for a=1:n
   [r1,c1] = find(L==a);
   pollLocs(a,:)=[round(((max(r1)-min(r1))/2) + min(r1)), round(((max(c1)-min(c1))/2) + min(c1))];
end
%figure, imshow(Q), title('small objects removed');
R=zeros(size(Q));
for b=1:size(pollLocs)
   R(pollLocs(b,1),pollLocs(b,2))=1;
end

 imOut = Q;
return;
%% END SEGIM2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SEGIM1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  function [imOut, found, foundLoc]=segIm1(img, ROI,ROIplus, bgnd,found,finalLoc,ROILoc)
% function [imOut]=segIm1(img, ROI, bgnd)
%
% Takes in an image and a region of interest being used for focus and
% bacground image for subtraction
% found is co-ordinates from *centre* of image of found object
% segments the pollen grains if they fall in that region.
% Returns an image that just surrounds the pollen grain nearest the centre
% of the input image (with limits to distance from centre)
imgsize = size(img);
K=img(1:imgsize(1),1:imgsize(2));%remove third dimension if exists
bgnd=bgnd(1:size(bgnd,1),1:size(bgnd,2));
%L = edge(K, 'sobel', (graythresh(K) * .083));
Thrshold = graythresh(bgnd);%normalised value in range [0,1]
if Thrshold == 0
   Thrshold =0.0001;
end

L = edge(K,'canny',[Thrshold*0.09,Thrshold*0.3] );           %edge Thold ALTER
  %figure, imshow(L), title('EdgesCanny');

se90 = strel('line', 5, 90);              % try alter middle param: original=3 ALTER
se0 = strel('line', 5, 0);                % try alter middle param: original=3 ALTER
se45 = strel('line', 3, 45);
se135 = strel('line', 3, 135);
M = imdilate(L, [se0 se90 se45 se135]);
N = imfill(M,8, 'holes');
P = imclearborder(N, 8);
 %figure, imshow(P), title(' fill');%was diamond,1

seD = strel('diamond',2);%original=1,

Q = imerode(P,seD);
Q = imerode(Q,seD);
Q = imerode(Q,seD);
   %figure, imshow(Q), title('erodex2');
   %figure, imshow(K), title('Imdilate + Hough1');hold on


[L,n] = bwlabel(Q,4);
regTholdlo = 455;              %455=>10um diameter size of smallest pollen ALTER
```

```
regTholdhi = 45600;              %45600=>100um diameter size of largest pollen ALTER
for a=1:n
   [r,c] = find(L==a);
   Sr = size(r,1);

      LocR = max(r)-min(r)+1;%get some dimesions
      LocC = max(c)-min(c)+1;%add 1 to avoid 0 if used as divisor
      LocBoundArea = (LocR * LocC);
      LocFit = Sr/LocBoundArea;
      LocSkew = (min(LocR,LocC))/(max(LocR,LocC));
   if(Sr > regTholdhi) || (Sr < regTholdlo);%compare with Thold
      for jj = 1:Sr
      Q(r(jj),c(jj)) = 0;%remove large or small
      end
   elseif LocFit < 0.55 | LocSkew < 0.65;%pine set at 0.55 | 0.65
      for jj = 1:Sr
      Q(r(jj),c(jj)) = 0;%remove long and thin
      end
   else
      eIm = zeros(size(Q));
      for jj = 1:Sr
      eIm(r(jj),c(jj)) = 1;
      end
      [re,ce] = find(eIm==1);
      [CH, CHa] = convhull(re,ce,[]);
      LocHull = (Sr/CHa);
      if LocHull < 0.92 %0.92
         for jj = 1:Sr
         Q(r(jj),c(jj)) = 0;
         end
      end %if LocHull
   end %if Sr
end%for a

%%%find centre of all regions found
pollLocs = [];
[L,n] = bwlabel(Q,8);
for a = 1:n
   [r1,c1] = find(L==a);
   pollLocs(a,:)=[round(((max(r1)-min(r1))/2) + min(r1)), round(((max(c1)-min(c1))/2) + min(c1))];
end
%figure, imshow(Q), title('small objects removed');

%%%makle image with centre of regions marked
% R = zeros(size(Q));
% for b = 1:size(pollLocs,1)
%     R(pollLocs(b,1),pollLocs(b,2)) = 1;
% end
enlarge = 1.8; %2 makes a border just around object. <2 increases image size around object
partn = 0.75;%portion of img centre to last found object
%distance in *steps* of pollen centre to be considered the same pollen centre captured later
samePol = 25;
centreImg = round(imgsize./2);
foundLoc=[];
if size(pollLocs)~= [0,0];
   %limt = round(ROI(1,3)/2);%using circle inside ROI
   %distance from centre
   %distns = sqrt((abs(round(imgsize(1,1)/2)-pollLocs(:,1))).^2 + (abs(round(imgsize(1,2)/2)-pollLocs(:,2))).^2
);
   %distance from between centre and last found object.
   distns = sqrt((abs(round((imgsize(1,1)/2)+found(1,1)*partn)-pollLocs(:,1))).^2 +
(abs(round((imgsize(1,2)/2)+found(1,2)*partn)-pollLocs(:,2))).^2 );
```

```
   [a,b] = sort(distns);
   %[a,b] = min(distns); %find minimum
   for dd = 1:size(distns,2)
      %calculate foundLoc location in steps on entire slide so multiples
      %can be detected. Send foundLoc back to calling prog which adds to
      %finalLoc and sends back in here to use as check for closeness to
      %current found pollen
      augR = round((pollLocs(b(dd),1) - (ROIplus(1,4)/2))/6.215);%6.215 is Cam1 pixels per step
      augC = round((pollLocs(b(dd),2) - (ROIplus(1,3)/2))/6.215);%6.215 is Cam1 pixels per step
      foundLoc = ROILoc + [augC, -augR];%Changing Y sense? then change sense of AugR !!!!
      repPol = 0;
      for ee = 1:size(finalLoc,1)
         if ((finalLoc(ee,1) < (foundLoc(1,1)+samePol))&&(finalLoc(ee,1) > (foundLoc(1,1)-samePol))) &&
((finalLoc(ee,2) < (foundLoc(1,2)+samePol))&&(finalLoc(ee,2) > (foundLoc(1,2)-samePol)))
            repPol = 1
            dd
            foundLoc = [];
            imOut=[];
            break
         end% if
      end%for ee


      if repPol == 0
      LocOut = pollLocs(b(dd),1:2); %reference that back to polLocs to get r/c co-ordinates of region
      [r2,c2] = find(L==b(dd)); %get list of pixels of pollen image
      %Define smallest region around pollen and extract it as an image
      %lastFound is location in image of last pollen as a clue to where
      %next one is more likely, so change target from centre accordingly
      %actually by 'partn'times distance from centre to location. used in distns above
      lastFound = [round(((max(r2)-min(r2))/2) + min(r2)), round(((max(c2)-min(c2))/2) + min(c2))];
      found = [lastFound(1)-centreImg(1), lastFound(2)-centreImg(2)];%subtraction order
      imOut = img(max(1,round(LocOut(1,1)-((max(r2)-
min(r2))/enlarge))):min(size(img,1),round(LocOut(1,1)+((max(r2)-min(r2))/enlarge))) ,
max(1,round(LocOut(1,2)-((max(c2)-min(c2))/enlarge))):min(size(img,2),round(LocOut(1,2)+((max(c2)-
min(c2))/enlarge))) );
      break
      end%if repPol
   end
else
   imOut = [];
end %if size(pollLocs)
%% END SEGIM1%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% AUTO-FOCUS%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   function [steps, useMax, focThold] = focus(serialPort, camera, stepNum, stepSize, ROI, algorithm, pkNum,
bgimg)
% function [69] = focus(serialPort, camera);
%
% STEPS: indices to their position in focus images,
%   of array of max value, plus local maxima (up to pkNum)
% SERIALPORT is a serial port for comms to steppers (Z in this cse)
% CAMERA is an imaging device, for stageCam is cam1in or cam2in
% STEPSIZE: number  of steps per image taken
% STEPNUM: number of images taken
% ROI: region for focus [Row Column RowOffset ColumnOffset]
% ALGORITHM: see switch case options for 'algorithm'
% pkNum: number of local maxima to return in 'steps'(after maximum value
% index)


camROI = get(camera,'ROIposition');
set(camera,'ROIposition',ROI);

imFoc = getsnapshot(camera);
focusData = [];
  for ii = 1:stepNum
    imFoc=[];
    imFoc = getsnapshot(camera);
    imFoc = imFoc(1:size(imFoc,1),1:size(imFoc,2));
    imFoc = double(imFoc);
    if isequal(imFoc,[])
      fprintf('snap missed getting an image in function FOCUS\n');
    end
    if ii ~= stepNum
      fprintf(serialPort,strcat('z+',num2str(stepSize)),'async');
    end %if

    a = ROI(1,2)+1;
    %b=ROI(1,2) + ROI(1,4);
    c = ROI(1,1)+1;
    %d = ROI(1,1) + ROI(1,3);
    bgimgIn = bgimg(a:size(bgimg,1),c:size(bgimg,2));
    bgimgIn = double(bgimgIn);
    bgimgIn = bgimgIn(1:size(imFoc,1),1:size(imFoc,2));
    if ~isequal(bgimgIn,[])
      imFoc = imFoc - bgimgIn + mean2(bgimgIn);
    end %if %bgimgIn

    %imwrite(imFoc,strcat(num2str(ii),'.tif'),'tiff');

    switch algorithm
      case 'sDev'
        imVal = std2(imFoc);
      case 'vars'
        imVal = sqrt((mean(diag(cov(imFoc)))^2) + (mean(diag(cov(imFoc')))^2));
      case 'gradMax'
        [gradX gradY] = gradient(imFoc);
        imVal = sum(sum((max(gradX,gradY)).^2));
        clear gradX gradY;
      case 'vollath4'
        tempImageShiftX = imFoc(:,2:size(imFoc,2)); %X shifted image
        tempImageShiftX2 = imFoc(:,3:size(imFoc,2)); %X shifted 2pixels
        imageMult1 = tempImageShiftX(1:size(tempImageShiftX,1))*imFoc(:,1:size(tempImageShiftX,2));
        imageMult2 = tempImageShiftX2(1:size(tempImageShiftX2,1))*imFoc(:,1:size(tempImageShiftX2,2));
        imVal = sum(sum(imageMult1(size(imageMult2,2))))-sum(sum(imageMult2));
        clear tempImageShiftX tempImageShiftX2 imageMult1 imageMult2;
      case 'vollath5'
        tempImageShiftX = imFoc(:,2:size(imFoc,2)); %X shifted image
        imageMult1 = tempImageShiftX(1:size(tempImageShiftX,1))*imFoc(:,1:size(tempImageShiftX,2));
        imVal = sum(sum(imageMult1))-(size(imageMult1,1)*size(imageMult1,2)*((mean2(imFoc))^2));
      case 'deriv'
        der1X = diff(imFoc,1,1);
        der1Y = diff(imFoc,1,2);
        der2X = imFoc(1:size(der1X,1),:).^2.*der1X.^2;
        der2Y = imFoc(:,1:size(der1Y,2)).^2 .* der1Y.^2;
        imVal = sum(sum(der2X(:,1:size(der2Y,2)) + der2Y(1:size(der2X,1),:)));
        clear der1X der1Y der2X der2Y;
      case 'pwr'
        meen = mean2(imFoc);
```

```
            thold = meen*1; %%ADJUST
            tHoldImage = (imFoc > thold).*imFoc;
            imVal = sqrt(sum(sum(tHoldImage.^2)))/(size(imFoc,1)*size(imFoc,2));
            %Fpower2 = sqrt(sum(sum(tHoldImage(:,1:size(imFoc,1))^2)))/(size(imFoc,1)*size(imFoc,2));
            clear tHoldImage thold meen;
         case 'normVar'
            meen = mean2(imFoc);
            imVal = sum(sum((imFoc-meen).^2))/(size(imFoc,1)*size(imFoc,2)*meen);
            clear meen;
         case 'hGram'
            L = hist(reshape(imFoc,1,[]),1:255);
            L=sort(L,'descend');
            imVal = mean(L(1:25)); %ADJUST
            clear L;
         case 'entrpy'
            L = imFoc(imFoc>150); %Thold
            L = hist(reshape(L,1,[]),1:255);
            imVal = -sum((L+1).*log2(L+1));
            clear L;
         case 'fTrans'
            Fr = sort(reshape(real(fft2(imFoc)),1,[]),'descend');
            imVal = mean(Fr(:,1:1000)); %%ADJUST
            clear Fr;
         otherwise
            error('algorithm not recognised');
      end%switch
      focusData = [focusData; imVal];
      if ii ~= stepNum
         fscanf(serialPort);%required for fprintf, just above switch
%           pause(0.1);
%           pause(0.1);
%           pause(0.1);
%           pause(0.1);
           pause(0.5);
      end %if
   end%for ii

if isequal(algorithm, 'hGram') || isequal(algorithm,'entrpy')
   focusData = max(focusData) - focusData; %inverse for these two algorithms
end %if algorithm

[maxVal, maxind] = max(focusData);
[lmVal,pkIndx] = loclmax(focusData,pkNum);
steps = [maxind-1,pkIndx-1];
steps = [steps; maxVal, lmVal];


if steps(1,1)== 0 || steps(1,1)==stepNum-1
                useMax = 0;

else
                useMax = 1;
end%if

tholdData = focusData(2:size(focusData)-1);
minThold = min(tholdData);
maxThold = max(tholdData);
%find a level above noise floor, below which the peak is too low level to be of value

   %focThold = minVal+((maxVal-minVal)/3);
focThold = minThold+((maxThold-minThold)/3);
```

```
%figure, plot(1:size(focusData,1),focThold,'r-',pkIndx,focusData(pkIndx),'rx',1:size(focusData,1),focusData,'b-
');

 fprintf(serialPort,strcat('z-',num2str(stepSize*(stepNum-1))),'async');%move camera back
 pause(0.1);
 fscanf(serialPort);
set(camera,'ROIposition',camROI);%set camera back
imFoc = getsnapshot(camera);
%% END AUTO-FOCUS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% LOCAL MAXIMUM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   function [maxval,ndx]=loclmax(vector,points)
%LMAX              [maxval, ndx]=loclmax(vector,points).
%   Find local maxima in vector XX,where
%          MAXVAL is the output vector with maxima values,
%   NDX  is the corresponding indeces,
%   VECTOR is the input vector to in which to find local maxima
%          POINTS is the number of points (from 'highest' down) to detect

x=vector;
len_x = length(x);
maxval=[]; ndx=[]; hght = [];
i=2;                    % start at second data point in time series

   while i < len_x,

            if x(i) > x(i-1)
              if x(i) > x(i+1)     % definite max
maxval =[maxval, x(i)];
ndx = [ ndx i];
hght = [hght, (x(i)-((x(i-1) + x(i+1))/2))];
              elseif (i ~= len_x-1) && (x(i)==x(i+1)) && (x(i)==x(i+2))          % 'long' flat spot
i = i + 2;                          % skip 2 points
              elseif x(i)==x(i+1)           % 'short' flat spot
i = i + 1;              % skip one point
              end
            end
            i = i + 1;
   end

if nargin == 1
   points = size(hght,2);
end
dd = min(points,size(hght,2));
if ~isempty(hght)
   [htval indx] = sort(hght,'descend');
   indx = indx(1:dd);%index of peak heights in descending order
   %indx = sort(indx);%reorder to image captured order (moving up thru focus)
   maxval = maxval(indx);%vector input values in peak height descending order
   ndx = ndx(indx);%index to input vector = motor steps taken to get to that point/stepSize
end%if

% uncomment for a graph of the original with found points:
% figure, plot(ndx,vector(ndx),'rx',1:size(vector,1),vector,'b-')
%% END LOCAL MAXIMUM  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% FEATURE EXTRACTION AND DATA NORMALISATION %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [normFeatures, imList] = featXtract(normParams)
```

```
%NORMPARAMS is a set of normalisation parameters used to normalise library data and again here
%NORMFEATURES is sets of normalised feature data named using the folder
%name from which the images were found.
%IMLIST is a list of images
%  Program extracts features, normalises, and saves a mat file for each
%  image set of features in the top folder,
%  non-image files are filtered out and not used.


% extract features from images in folder
%imList=dir(topPath);
%cd(imPath);
ims = dir;
ims = ims(3:size(ims,1));%first two are '.' & '..'
features=[];
normFeatures=[];
imgCnt=0;
for j = 1:size(ims,1);
   if strcmp(finfo(ims(j,1).name), 'im')== 1;
      imgCnt = imgCnt+1;
      thisPath = ims(j,1).name;
      features = [features; getAllFeatures(thisPath)];
      imList(imgCnt,1) = {ims(j,1).name};
   end
end%for j = 1:

if ~isequal(features,[]) & isequal(size(features,2), size(normParams,2))
  save('imageFeatures.mat','features');

   %% Normalise using the main database parameters for transforming data to
   %% mean=0 and stdDev = 1 (parameters in featParams)
   for j = 1:size(features,2)
   normFeatures(:,j) = (features(:,j)-normParams(1,j))./normParams(2,j);
   end

  save('normFeatures.mat','normFeatures','imList');
else
   error('feature file is empty or wrong sized normParams');
end%if ~isequal
%cd(tempPath);
return
%% END FEATURE EXTRACTION AND NORMALISATION %%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TRAIN ANN FROM DATA BASE of normFeatures  %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [trainedNet, randOrderImages] = trainNN()
%train a set of normFeature sets from data base.
% TRAINEDNET a trained neural network representing images from the data base of
% pollen types and a classList indicating pollen types and order of training
% RANDORDERIMAGES is an index to the random ordering performed so original order can be determined

NFset=[];
[netName,netPath] = uigetfile('*.*','select a set of normFeatures - nf???.mat ','nf*','MultiSelect','on');
for k=1:size(netName,2);
   tempStr = load(strcat(netPath,char(netName(k))));
   NFset.(strrep(char(netName(1,k)),'.mat',''))= tempStr;
end

classList = fieldnames(NFset);
numSets = size(classList,1);
imagNumPerSet = [];
```

```
featureTrain = [];

for i=1:numSets
   tempNorm=NFset.(char(classList(i))).normFeatures;
  %tempNorm = tempNorm(randperm(size(tempNorm,1)),:);
   imagNumPerSet = vertcat(imagNumPerSet,size(tempNorm,1));%[imagNumPerSet; size(tempNorm,1)];
   featureTrain = [featureTrain; tempNorm];
end
clear tempNorm;
options = foptions; %defaults
options(1) = 0; %Display parameter (Default:0). 1 displays some results.
options(14) = 80;%round(3.78*(numSets)+11.1); %Maximum number of function evaluations.
nHidden = size(featureTrain,2)*2+1;
nInputs = size(featureTrain,2);
nOutputs = numSets;
randOrderImages = randperm(size(featureTrain,1))';
%featureTrain = featureTrain(randOrderImages,:);

targets=[];
adj=0;
%fill targets depending on number of pollen types and number
%of images used for each pollen type
for i = 1:nOutputs      %fill targets
   if i == 1; else adj = adj+imagNumPerSet(i-1);end
   for j = 1:imagNumPerSet(i);
      targets(adj+j,i)=1;
   end
end

%targets = targets(randOrderImages,:);
netSoft = mlp(nInputs,nHidden,nOutputs,'softmax');
trainedNet = netopt(netSoft,options, featureTrain, targets,'scg');
classList = classList;
save('trainedNet','trainedNet','classList');
tempNet.trainedNet = trainedNet;
tempNet.classList = classList;
trainedNet = tempNet;
clear tempNet;

%% END  TRAIN ANN FROM DATA BASE of normFeatures %%%%%%%%%%%%%%%%%%%%

%% END MAINSTAGE CODE
```

11.1.14 Feature Extraction featXtract.m

```
function [normFeatures, imList] = featXtract(topPath)
%TOPPATH is a path to folder containing images for
%which feature extraction is required.
%NORMFEATURES is sets of normalised feature data named using the folder
%name from which the images were found.
%  Program extracts features, normalises, and saves a mat file for each
%  image set of features in the top folder,
%  non-image files are filtered out and not used.

normParams =
[5030.134980307913,231.3034371643394,11.549145573441029,0.293437972241872,0.227959697576128,0.074001360
388508,0.000198687762285,0.000016941121767,0.000006335590846,0.000000000008587,0.00000001362427,-
0.000000000044924,0.440962893954756,0.220468737572616,0.955566969815511,6825.828280073598,117.263694951
66487,0.77756426949119,2485182.0254206937,1497.7238042250021,38.56475676785264,1101240.540736761,140219
3.7644309185,1954679.6813661144,4063279.210944027,3872055.6853244323,1537042871.0196767,1460751068.2200
918,15240603486338705000,0.008659205087587,0.005295770481277,0.004894526646877,0.004182615161962,0.0032
47431079919,0.003328716734365,0.002790500456022,0.003369855374229,99.09191585246701,0.908084147533,96.7
5340994541313,0.618414164599578,91.66260346632902,0.392911669629661;3460.333605618244,80.9061667604875
5,2.236268994033373,0.061566809328527,0.045626694361407,0.014966302126472,0.000453335049101,0.000028902
27307,0.000008106466735,0.000000000173313,0.000000092755801,0.000000000221719,0.187331184360503,0.05791
0192164941,0.059807099483805,5224.615706898697,86.89454714842559,0.221293882750971,5284786.335172836,34
17.2158562600184,23.568266770203916,2407543.8788592117,2893574.3608193165,3784126.675981967,7594492.505
268886,7092315.602556042,3913485349.063464,3633766697.0870805,77015982694969260000,0.003071396507139,0.
00164390615226,0.001663863105227,0.001319338601508,0.000996686773119,0.001058083429597,0.0008079262294
12,0.001098144523812,0.383358513513111,0.383358513513083,1.227158269430299,0.188171754579054,2.50855563
7726779,0.096577306785522;];

%% extract features from images in folder
imList=dir(topPath);
imList = imList(3:size(imList,1));%first two are '.' & '..'
features=[];
normFeatures=[];
for j = 1:size(imList,1);
   if strcmp(finfo(strcat(imPath,'\',imList(j,1).name)), 'im')== 1;
      thisPath = strcat(imPath,'\',imList(j,1).name);
      features = [features; getAllFeatures(thisPath)];
   end
end%for j = 3:

if ~isequal(features,[])
  save(strcat(topPath,'\','imageFeatures.mat'),'features');

   %% Normalise using the main database parameters for transforming data to
   %% mean=0 and stdDev = 1 (parameters in featParams)
   for j = 1:size(features,2)
   normFeatures(:,j) = (normFeatures(:,j)-normParams(1,j))./normParams(2,j);
   end

  save(strcat(topPath,'\','normFeatures.mat'),'normFeatures');
else
   error('feature file is empty. No normFeatures created');
end%if ~isequal

return
```

11.1.15 Classify a set of images – classify.m

```
[out] = classify(trainedNet, featureSet, targets);

%classsify takes in a trained neural net (by trainNet) and feature sets
%saved by FXtract. The output is the sorted feature sets.


meen=mean(featureSet);
featureSetNorm = (featureSetNorm-ones(size(featureSetNorm,1),1)*meen);

netResultSoft = mlpfwd(B,featureSetPrune);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% Generate Expected result (as for 'target' in training)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
nOutputs=size(imagNumPerSet,1);
targetsIm=[];%create targets
fix=0;
```

```
%fill targets depending on number of pollen types and number
%of images used for each pollen type

if nargin = 3
   for i = 1:nOutputs      %fill targets
      if i == 1; else fix = fix+imagNumPerSet(i-1);end
      for j = 1:imagNumPerSet(i);
         targetsIm(fix+j,i)=1;
      end
   end


   targetsPC=zeros(size(featurePC,1),pollenTypeNum);
   for y = 1:pollenTypeNum
      targetsPC((((y*imagPrune)-imagPrune)+1):(y*imagPrune),y)=1;
   end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
   % check with expected result

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

   %display confusion matrix in a graph
   cmSoft = conffig( netResultSoft,targetsIm);

    % confusionMatrix 'C' and performance rate 'rate'
    %from predictions 'y' and targets
   %  [c,rate]=confmat(netResultSoft,targets);
end %if nargin
```

11.1.16 Final classification testing – testFinal.m

```
%Two image sets are previously saved, one for training and verification and one for final
%testing.  Use this for the final classification tests.  Each set is in a different
%subfolder
% A file is saved with data and a results file with results of each of the 5(testrepeatNum) tests

clear;
%clc;
close all hidden;

nHidden=87;
epochs = 100;
testrepeatNum = 5;
resultMat = [];
featPrune = (1:43);

loopCount=0;
resultFile=[];

[trainFileNames,trnPath]=uigetfile('*.*','Select all training files','*','MultiSelect','on');
[testFileNames,tstPath]=uigetfile('*.*','Select all testing files','*','MultiSelect','on');
cd(tstPath);
%trainFileNames=cellstr(trainFileNames);
%testFileNames=cellstr(testFileNames);
```

```
pollenTypeNum=size(testFileNames,2);


for ii = 1:testrepeatNum

      imagNumPerTrSet=[];
      imagNumPerTstSet=[];
      featureTrain=[];
      featureTst=[];
      netSoft=[];
      dataMean=[];
loopCount=loopCount+1;
   sort(testFileNames);
   sort(trainFileNames);
   namesPerm = randperm(size(testFileNames,2));
   trainFileNames = trainFileNames(namesPerm);
   testFileNames = testFileNames(namesPerm);

         fprintf('image order used to train NET: \n');%print image file names
      for i = 1:size(testFileNames,2)
         testFileNames(i) = strrep(testFileNames(i),'.mat','');
         fprintf('%s \n',char(testFileNames(i)));
         fprintf('%s \n',char(trainFileNames(i)));
      end;


      for i=1:pollenTypeNum
         trnData=load(char(strcat(trnPath,trainFileNames(1,i))));
         tstData=load(char(strcat(tstPath,testFileNames(1,i))));
         trnData=trnData.trnFile;
         tstData=tstData.tstFile;
    tstData = tstData(randperm(size(tstData,1)),:);
    trnData = trnData(randperm(size(trnData,1)),:);
         imagNumPerTstSet = vertcat(imagNumPerTstSet,size(tstData,1));
         imagNumPerTrSet = vertcat(imagNumPerTrSet,size(trnData,1));
         featureTst = [featureTst; tstData];
         featureTrain = [featureTrain; trnData];
      end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         % randomise images (targets follow by using the same random permutation)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       randOrderImages = randperm(size(featureTrain,1))';
       featureTrain = featureTrain(randOrderImages,:);
       size(featureTrain)
       featureTrain = featureTrain(:,featPrune);
         %size(featureTrain)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         % build an MLP neural network

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %variables:
         %1) inputs are number of features(columns) extracted from images(rows)
         %2) One hidden layer node numbers to be determined empirically
```

```
        %3) outputs depend on number of pollen types to be identified
          %e.g. non-airborne Vs airborne has two outputs

      %        %Nothing proved well but 'appears' to do better when odd
      %    if mod(size(featureFileNames,2)*5,2)==0
      %        hidden = size(featureFileNames,2)+1
      %    else
      %        hidden = size(featureFileNames,2)
      %    end %if mod

        nInputs=size(featureTrain,2); %inputs to neuralNet same as feature number

        nOutputs=pollenTypeNum;%numer of files selected indicating
                                %number of image types

        netSoft = mlp(nInputs,nHidden,nOutputs,'softmax');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % train network

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%

        options = foptions; %defaults
        options(1) = 1; %Display parameter (Default:0). 1 displays some results.
        options(14) = epochs; %Maximum number of function evaluations.
                    %Vary to find best (when result is very near zero)
                    %compromise compute time Vs nearer zero result.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % create targets

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        targets=[];
        adj=0;
        %fill targets depending on number of pollen types and number
        %of images used for each pollen type
        for i = 1:nOutputs      %fill targets
          if i == 1; else adj = adj+imagNumPerTrSet(i-1);end
          for j = 1:imagNumPerTrSet(i);
             targets(adj+j,i)=1;
          end
        end
   targets = targets(randOrderImages,:);
      %size(targets)
        %pack;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % train NET

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%


        netSoft = netopt(netSoft,options, featureTrain, targets,'scg');
        %netSoft = mlptrain(netSoft, featureTrain, targets, epochs);
        %net.pruneFeat=pruneFeat;%save feature prune info


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % Generate Expected result (as for 'target' in training)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%


        targetTst=[];%create targets
        fix=0;
        %fill targets depending on number of pollen types and number
        %of images used for each pollen type
        for i = 1:nOutputs      %fill targets
           if i == 1; else fix = fix+imagNumPerTstSet(i-1);end
           for j = 1:imagNumPerTstSet(i);
              targetTst(fix+j,i)=1;
           end
        end

        %randomise test image features and targets together
   randOrderTst = randperm(size(featureTst,1))';
   featureTst = featureTst(randOrderTst,:);
   featureTst = featureTst(:,featPrune);
   targetTst = targetTst(randOrderTst,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % Forward Propogation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        net = mlpfwd(netSoft,featureTst);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
        % check with expected result

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%


         %confusionMatrix 'C' and performance rate 'rate'
         %from predictions 'y' and targets
         %display confusion matrix in a graph
        [c, rate] = confmatPer(net, targetTst,testFileNames);
        confMatrx = conffigPer( net,targetTst,testFileNames);
trNum = size(featureTrain,1) / pollenTypeNum;
tstNum = size(featureTst,1) / pollenTypeNum;
        hgsave(strcat(num2str(ii),'figure'));
        save(strcat(trnPath,num2str(ii),'netTrained','-
','netSoft'),'netSoft','trNum','tstNum','trainFileNames','testFileNames','nHidden','epochs','featureTst','targetTst')
;
resultFile(1,loopCount) = loopCount;
resultFile(2,loopCount) = rate(1);

end %for ii =
save('resultFile','resultFile');
%END
```

11.1.17 Training a Neural Net – trainNet.m

```
[trainedNet, targets, randOrder] = trainNet(filesPath)

%Train an MLP neural net using all normFeatures (in filesPath)
%and return trained net.
%normFeatures is a set of files produced by featXtract.
%featXtract produces feature files, each one of a single type of pollen.
%FILESPATH path to normFeatures files
%TRAINEDNET returned trained neural net
%TARGETS key to image feature vectors derived from different(separate)files
%RANDORDER numbers used to randomise the input vectors and targets

featCat=[];
imNames = {};
imNumPerSet = [];
pruneNum=8; %<<*****ADJUST ******<<remove images from set for later testing
fileList=dir(strcat(filesPath,'normFeatures*.mat'));

for ii = 1:size(fileList,1);
   tempData = load(strcat(filesPath,fileList(ii).name));
   tempFeats = tempData.normFeatures;%remove from structure.
   featCat = [featCat; tempFeats];
   imNumPerSet = [imNumPerSet; size(tempFeats,1)];
   %image names is derived from names of folders holding images used by featXtract
   imNames = [imNames; strrep(strrep(fileList(ii).name,'normFeatures',''),'.mat','')];
   clear tempFeats tempData
end %for ii


featCatNorm = featCat;
meen = mean(featCatNorm);
featCatNorm = (featCatNorm-ones(size(featCatNorm,1),1)*meen);
%featCat=featureCatNorm; %Switch normalise on ????

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% build an MLP neural network
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%variables:
%1) inputs are number of features(columns) extracted from images(rows)
%2) One hidden layer node numbers to be determined empirically
%3) outputs depend on number of pollen types to be identified
   %e.g. non-airborne Vs airborne has two outputs
nInputs=size(featCat,2); %inputs to neuralNet same as feature number
nHidden=22; %18  %nInputs*2+1;  %suggested in text - use for now
nOutputs=size(imNames,1);%numer of files selected indicating
                          %number of image types
%netLog = mlp(nInputs,nHidden,nOutputs,'logistic');
netSoft = mlp(nInputs,nHidden,nOutputs,'softmax');
%netLin = mlp(nInputs,nHidden,nOutputs,'linear');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% train network
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
options = foptions; %defaults
options(1) = 1; %Display parameter (Default:0). 1 displays some results.
options(14) = 150; %Maximum number of function evaluations.
```

```
          %Vary to find best (when result is very near zero)
          %compromise compute time Vs nearer zero result.

targets=[];%create targets
adj=0;
%fill targets depending on number of pollen types and number
%of images used for each pollen type
for i = 1:nOutputs     %fill targets
  if i == 1; else adj = adj+imNumPerSet(i-1);end
  for j = 1:imNumPerSet(i);
    targets(adj+j,i)=1;
  end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% randomise images (targets follow change)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
randOrder = randperm(size(featCat,1))';
randFeatCat = featCat(randOrder,:);
randTargets = targets(randOrder,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
% train NET
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
%netLog = netopt(netLog,options, randFeatureCatPrune, randTargets,'quasinew');
trainedNet = netopt(netSoft,options, randFeatCat, randTargets,'quasinew');
%netLin = netopt(netLin,options, randFeatureCatPrune, randTargets,'quasinew');

%net.pruneFeat=pruneFeat;%save feature prune info
save(strcat(filesPath,'netTrained'),'featCat','trainedNet','pruneNum','imNames','randOrder');

%END
```