Intelligent Remote Chromatic Processing

by

Saad Uddin Ahmed

library copy

May 1994

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy.

Dedicated to my family:

"Allah is the light of the Heavens and the Earth.
The parable of his light is as if there was a niche;
And within it a lamp.
The lamp is enclosed in glass;
The glass as it were a brilliant star lit from a blessed tree;
An olive neither of the east nor of the west.
Whose oil is well-nigh luminous though fire scarce touched it:
light within light."

***Quran** Sura 24 v 34-36.*

Acknowledgements.

I would like to thank Prof. G. R. Jones who provided the necessary sources of funding for this project. He has been a source of inspiration, a guide and I have enjoyed working under his supervision. Thanks also to Dr. P. Russell, my supervisor and friend who has provided help and constructive criticism whenever it was needed.

Thanks are also due to all my fellow colleagues in GO3 during my three years working in the group who provided a friendly atmosphere for work and discussion, they include: J. Humphries, P. Henderson, I. Khandakar, J. Cosgrave, J.Chaaraoui, D. Alston, R., R. and S. Smith, A. Tranter, E. Glavas, I. West, N. Pilling, Li and G. Jia. I wish them success for the future.

The support of the technicians, specially F. Ormesher, the electrical and mechanical workshop are much appreciated. They, and the people associated with the Department of Electrical Engineering, are all responsible for the smooth progression of this work.

Finally, I thank my mother, my sisters and my wife, Amina, who was patient and supportive throughout my final years of my work. I give them all my love.

Abstract

This thesis presents the development of a general purpose remote chromatic monitoring system for addressing parameter sensitive transducers without contact. The system comprises a digital CCD colour camera linked to a computer and sensor elements which change their chromaticity in response to variations in a measureand. Image processing techniques are incorporated to allow automated object location, tracking and monitoring. A number of specific systems are investigated to demonstrate the versatility of the system:

- Remote temperature monitoring using thermochromic materials.
- Pressure monitoring using a photo elastic diapragm barometer.
- Speckle monitoring for intrinsic fibre sensors.
- Plasma monitoring for plasma condition and etch rate monitoring
- Chemical indicator analysis for medical and industrial applications.

The analysis of chromatic images obtained from a number of real world situations are complex and highly dependent on external surrounding conditions. To incorporate the effects of all these parameters into a monitoring system is difficult. Alternative methods using artificial neural networks are presented and the usefulness of such an approach in instrumentation applications is discussed.

The tests on the systems described above have demonstrated that the remote chromatic processing system provides a cost effective, flexible and elegant solution to a wide range of industrial monitoring needs and is capable of providing the required resolution and dynamic range compatible with industrial needs.

Contents

CHAPTER 1
Introduction

1.1 Introduction to Remote Optical Image Processing

The use of remote optical techniques for monitoring physical parameters is a relatively old technology first finding application in early NASA space industry for meteorology [Green 1983]. However this technology has taken a long time to progress and meet application demands in the general industry. The driving force behind remote optical monitoring research has mainly been space and military programmes where costs and demands were favourably matched. Indeed there were few who could afford the computational power and technology costs to consider remote sensing as a possible route for addressing their instrumentation requirements. It is in this environment that remote optical monitoring was established. The service it provided initially was primarily to niche industrial needs where conventional instrumentation could not be used or did not exist. These areas included explosive environments, radioactive areas, plasmas and situations where non contact monitoring was of main consideration [Henderson 1989]. Competing and surviving against competition from conventional instrumentation provided a solid foundation and established consumer confidence in this technology.

With the major advancements in the electronics industry and the subsequent reduction in cost of high computational power and optical monitoring systems, this technology could begin to play a significant role in addressing more general commercial needs. Presently remote optical monitoring has been suitably implemented in a wide and diverse range of applications [AVC88 Conference 1988]. These include:

- Traffic surveillance,
- Industrial plant monitoring,
- Medical Imaging,
- Nuclear industrial process monitoring,
- Metrology and weather forecasting,
- Monitoring of astronomical data.

In most cases images are obtained by digital cameras and processed digitally. In general, the image processing tends to be monochromatic in nature, i.e. the image colour is not processed. This is sufficient for most remote monitoring applications and so a majority of processing tasks have been developed to efficiently handle normal black and white images. The advantages of using grey scale images are the simplicity of algorithms necessary for analysis, processing and interpretation. However, the disadvantages of such systems are that they are highly dependent on the image intensity and also loose a significant amount of information that is available from chromatic image data.

One example of a chromatic remote monitoring system is the human eye. The eye itself is a relatively simple organ. However, with the availability of a few simple detector types it is capable of providing the scope of identifying millions of colours. The brain has a major role to play in the human's ability to perceive but, irrespective of the brain's capability, the data is still obtained through the eye and depends on the resolution of these detectors. Also the brain's ability to compensate for lighting levels, its ability to extract colour information depends a great deal on the nature of these receptors in the eye.

Because humans have always been able to see in colour and needed techniques of quantifying them, it is not surprising that colour science has long been established. The first impact was made when Isaac Newton divided white light to obtain a rainbow of colours using a prism in his historic experiment in 1666. Since then the science developed extensively with the establishment of an International body of standards in France called the Commission Internationale de Eclairge [Hunt 1987]. This body was responsible for defining all the terms and properties of colour. The Commercial industry demanded suitable quantification methods and these have been developed to a high degree of accuracy and proficiency by initially using spectrophotometers and later with tri-colour detectors.

An optical image contains a vast amount of information. Firstly the spatial distribution of the optical signal gives the image its shape [Ballard, Brown 1982]. Secondly each of these optical signal points has a spectral distribution which is the power distribution over the wavelength range being monitored. This spectral distribution provides colour information. Monochromatic images represent this whole spectral distribution as a single intensity value and thus contains spatial information but no spectral information. In some cases this is sufficient for remote sensing applications like identifying an object in the field of view or monitoring objects that provide a total spectral power change with measurand. However, recently the need for higher accuracy and the desire to monitor colour changes of objects have led to the development of remote colour monitoring systems that use both spatial and spectral information. One example of this is in NMR spectroscopic imaging where magnetic images obtained contained the spatial and spectral information not only to segregate different parts of the object but also provide a means of identifying the elementary composition of a given object [Chmeurny et. al. 1986]. One problem associated with spectroscopic images is that the amount of information available is too large to manipulate easily and requires a huge amount of computational resources to store and interpret.

The desire to extract spectral information and to process it has led to the development of two photo-detector and three photo-detector systems where the detectors operate with different parts of the optical spectrum. The spectral characteristics of these detectors usually tend to be similar to that of the human eye detector responses [Hunt 1987]. The reasons for this are:

- Aesthetic in that visual inspection of this colour image provides a representation of the object similar to direct observation of the object.

- The detectors are spatially well separated in the visible spectrum giving information about spectral changes that can be conceptually visualised (e.g. We may say as the red intensity increases that the object is becoming redder).

- The great success in the human eye's ability with processing colours suggests that this combination will be successful in monitoring colour changes that can be visually identified.

The advantage of tri-stimulus detection (three detector systems) is that it provides the scope for limited spectral information from an optical image using only three parameters and the use of simple detector systems and avoids the use of the full spectrum data that can be produced by a spectrophotometer [Hunt 1987].

This thesis presents work done in the acquisition and processing of images obtained by a colour CCD camera and processed using a PC computer. The aims of this specific work and the thesis outline will be further discussed in the following sections.

1.2 Aims of Work.

A major aim of this work was the development of a general remote chromatic monitoring system using the CCD camera and an IBM-PC clone computer with suitable interfacing facilities. This system would form the test platform for a number of different chromatic sensors. It would identify the spectral range and wavelength shifts that a chromatic sensor produces and indicate suitable information about instrumentation needs and demodulation techniques for extracting measurand information. However, the success of this work has meant that the system was developed further to address specific sensor systems and provide remote sensing solutions to a number of chromatic systems namely:

- Photo-elastic Barometer
- Thermochromic Temperature Monitoring
- Chromatic Speckle Analysis
- Plasma Monitoring
- Chemical Indicators Analysis

A number of real world problems have been identified that directly affect chromatic monitoring such as glare, lighting level changes and system drift. These effects are quantified on the chromatic system and suitable methods of compensation presented. In particular neural networks have been exploited to provide a base for system intelligence to process complex chromatic information. As the system developed, the need for a fully automated monitoring system arose and image processing facilities have been incorporated into the system to automatically locate and track sensors in the field of view; again a number of different techniques have been implemented to provide scope for intelligent object identification by determining a value of confidence for the image data.

1.3 Summary of Work

This work represents the merging of three distinct technologies namely, chromatic processing, neural networks and image processing to produce a system which addresses some of the general and specific needs of remote chromatic processing. One of the major problems associated with processing visual data is the high degree of dimensionality in the data, that is, the spatial distribution of the data and also its spectral content. Each of these dimensions adds exponentially to the quantity of data that needs to be processed. Image processing techniques are employed as a method of reducing the dimensionality by firstly parameterising the image and then selecting regions of interest to the monitoring application.

Image processing techniques considers primarily the spatial distribution of the data rather than its spectral content.

Chromatic processing is the second stage in reducing the dimensionality of the problem. The use of detectors that have wide band overlapping response characteristics allows the continuous spectral distribution information to be reduced into three components namely Red, Green, and Blue. These three parameters may be reduced further into two components, x and y, which are intensity compensated functions these three parameters. This reduction in dimensionality of the problem often leads to a complex non-linear relationship between the sensing parameter and the chromatic data. The dependence of the acquired data on external effects like lighting level changes and glare mean that conventional processing methods of parameter mapping provides an inadequate solution to the problems associated with remote image processing and more intelligent processing methods are required.

Neural networks have been used to perform the required mapping between chromatic data and the parameter being sensed. This intelligent non-linear processing allows methods of noise reduction by its generalisation and interpolation characteristics.

A number of specific chromatic systems are considered as examples of applications for the technology and these have been presented in this thesis. Experiments based on the thermochromic strip utilised a system designed to automatically locate a sensor of known characteristics and perform real time tracking to monitor it. In this case a specific set of general purpose modules were integrated to address the specific needs for sensor identification and tracking. The complex chromatic data obtained from the thermochromic system was processed using a neural network to relate the changes in sensor chromaticities to temperature. The dependence of noise on the network learning was investigated using a fifth order polynomial fitting algorithm to smooth the data.

A non-contact Photoelastic barometer was designed in the project and this was monitored using two dimensional analysis software. As the object was fixed in terms of image position, no image processing techniques were used in the location of the sensor. However the data was reduced by considering mean chromaticities of 10 concentric annular regions of the surface of the barometer which were of equal area. This method of analysis produced complex chromaticity variations with measurement and neural networks were utilised to process the data to relate these ring chromaticities with pressure.

A system for launching two laser sources operating at different wavelengths into a multi-mode fibre was developed as a part of this work. This system was used to preferentially excite different propagating modes of the fibre by launching the laser and different angles incident into the fibre. The chromatic speckle images were analysed statistically using a similar technique as used for the photoelastic barometer experiments.

Other chromatic systems that were investigated in this thesis included plasma monitoring system for determining plasma power and end point analysis and medical indicators which exhibit chromatic changes with concentrations of different components of urine. These systems were implemented to demonstrate the versatility of the system in terms of being used as a general purpose remote chromatic monitoring system.

1.4 Thesis Layout.

Chapter 2 presents the theory of colour and the principles of chromatic processing. It introduces the concept of computer vision in relation to human vision. A number of different methods of quantifying colour are presented. Finally this section presents a number of chromatic systems that are introduced as possible areas of application for a remote chromatic monitoring system.

Chapter 3 presents the theory of digital image processing. It provides an overview of the related theories and techniques. The section is compiled such that it demonstrates the approach used in developing a more specific image processing tool based on a collection of more general image processing modules. The techniques of image identification and tracking are discussed in this chapter.

Chapter 4 discusses the different techniques used for data processing. In particular it presents the theory behind neural networks that have been extensively employed in the development of remote thermochromic and barometer calibrations.

The details of the practical monitoring system are presented in chapter 5. This section describes in detail the features of the hardware of the system and also presents an outline of the various sensor systems investigated.

The experimental methodology is presented in section 6. This chapter describes all the various experiments performed and the apparatus and conditions that were implemented in the acquisition of test data.

Chapter 7 presents the results of all tests performed. These tests include tests on the remote chromatic monitoring system itself as well as the tests of specific system configurations. The results of the analysis of data using neural networks are also presented here.

The results are discussed in chapter 8.

Finally the conclusions in chapter 9 summarise the results and provides an indication of future work that may be useful in the development of this technique.

CHAPTER 2
Colour

2.1. Introduction

The human perceives the world in colour. The interpretation of this colour is subjective and depends on the physical characteristics of the human eye and psycophysical characteristics of the eye and the brain [Grand 1968]. Historically the first step to the understanding of colour was the famous experiments performed by Isaac Newton in 1666. In these experiments Newton used prisms and light to separate pure colours from white light. This work provided the foundation for the understanding of the nature of visible light and its position in the electromagnetic spectrum. It was later discovered that only three primary colours (red, blue and green), rather than the seven spectral colours that Newton described as pure colours, could be mixed together to provide all the colours for human perception. Once this tristimulus theory was understood it found implementation in colour photography and television. The first noteworthy study of the perception of colour was due to Helson [Helson 1943] and Judd [Judd 1940]. Their equations can be used to predict the colours observed by a subject as well as he/she is able to describe it. To show the adaptive and complex nature of the human vision system Land [Land 1959] showed that only two of these primary colours could be used to give full three colour perception with his famous two colour projection demonstrations. Because of the subjective nature of colour, quantitative measurements are difficult. The Commission Internationale d'Eclairge (CIE) was set up to standardise colour and light for measurement. This body is responsible for formal definition of terms of colour measurement and human perception.

The next section provides an introduction to the propagation of light through a fibre and presents some of the basic properties of light. This is followed by a basic definition of some of the terms used in this thesis before a detailed study of colour is presented.

2.2. Properties of Light

Light, in common with all wave phenomena, shares the properties of amplitude, wavelength and phase. As it propagates in a transverse fashion it also has properties of polarisation. Optical monitoring systems consider the modulation of one, or a number of these properties of light with measurand.

Light is a form of electromagnetic radiation, as is the case for X- rays, radio waves and radar. The electromagnetic radiation propagate through different mediums at different speeds. The refractive index n, for a non absorbing medium is given as the ratio of the velocity of light in vacuum c to that in the sample v [Wolf 1981].

$$n = \frac{c}{v} \qquad .. (2.1)$$

The specific property of the electromagnetic spectra that gives them their characteristics is their wavelength. Visible light occupies a wavelength region from about 380nm (Violet) to 780nm (Red). A range of wavelengths is referred to as a spectrum and on either side of the visible spectrum lies the ultra violet and infra red regions (see figure 2.1).

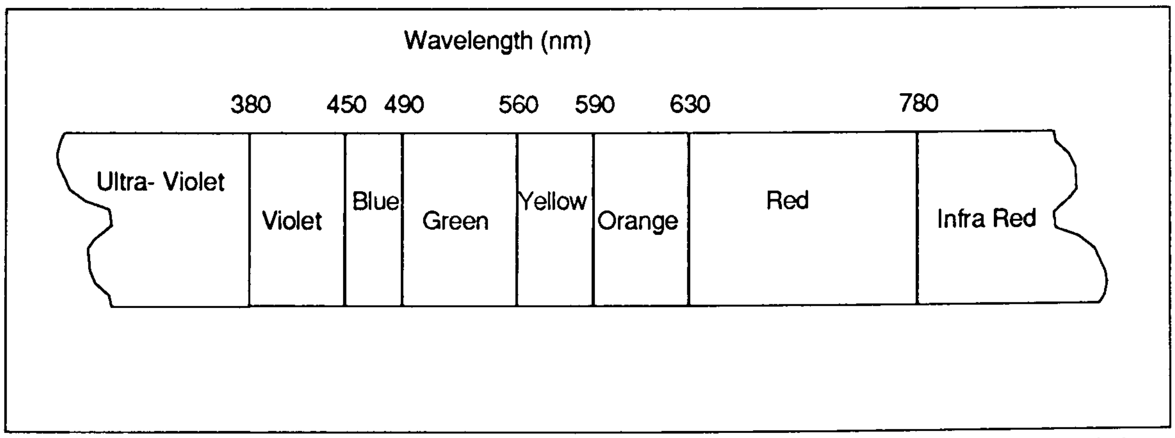


Figure 2.1. Schematic showing a part of the electromagnetic spectrum containing the visible spectrum.

The power spectrum of the electromagnetic waves for a signal describes the distribution of energy of that signal in terms of wavelength. This distribution is referred to as the colour or chromaticity of the spectrum. If a signal contains energy distributed in only a narrow band of wavelengths then the signal is referred to as monochromatic. However, if energy is distributed over a wide range of wavelengths the signal is referred to as polychromatic. Figure 2.2 shows examples of monochromatic and polychromatic spectra as graphs of wavelength against intensity at that wavelength.

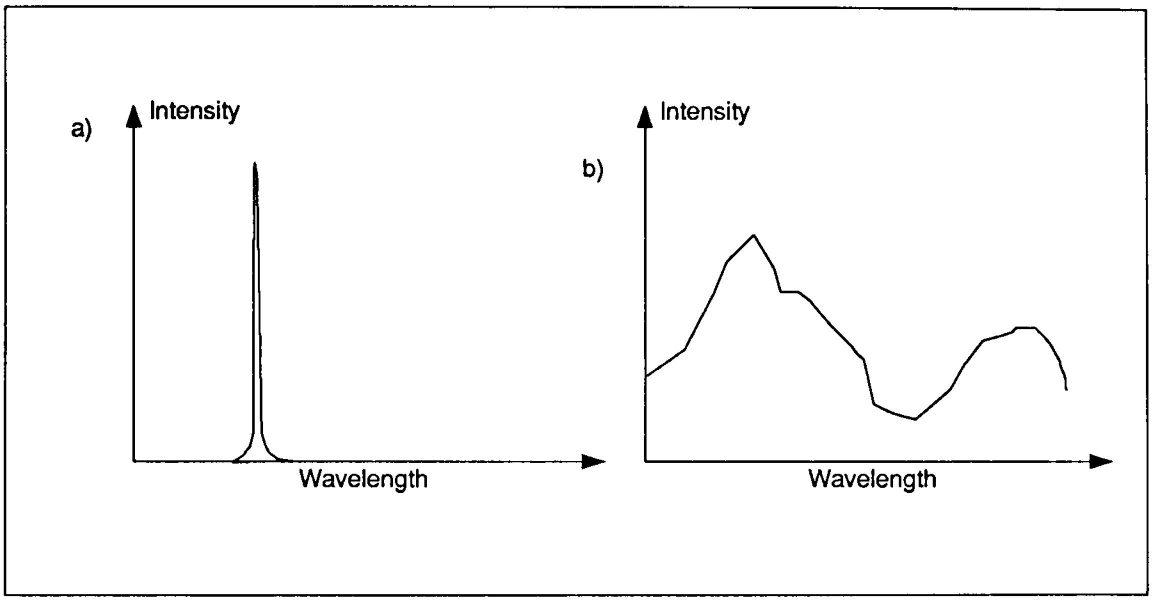


Figure 2.2 Diagram showing a) spectrum of a monochromatic signal and b) spectrum of a polychromatic signal.

The phase of a light source is the relationship between one propagating electromagnetic wave with another. If a source produces electromagnetic waves without any phase relationship between them then the signal is said to be incoherent. This is the more common form of optical signals. However if all waves propagate in synchrony then the signal is said to be coherent. One example of an optical coherent source is the laser. Coherent sources readily show effects of interference where waves interfere constructively and destructively to produce light and dark regions.

The polarisation of a given electromagnetic wave indicates the manner in with the waves oscillates. In vector terms electromagnetic waves may be resolved into two orthogonal components [Murphy 1991]. In linearly polarised light the two components are in phase whereas in elliptically polarised light the two components have a phase difference between them. Hence a given light may be represented as [Murphy 1991]:

$$E(t) = \left(i\cos\omega t + j\cos(\omega t + \phi)\right)E_0 \qquad ..(2.2)$$

Where i and j are unit vectors, ω is the angular frequency of light, E_0 is the amplitude of the E field and φ is the phase difference between the two components. If $\phi = 0$ then the state of polarisation is referred to as linear and if $\phi \neq 0$ then the state of polarisation is elliptical. Circular polarisation is a special case of elliptical polarisation where $\phi = \pi/2$.

2.2.1 Introduction to Optical Fibre Propagation of Light

Optical fibres provide a convenient medium for guiding light through convoluted routes. They consist of a cylindrical region of high refractive index dielectric surrounded by an annular layer of lower index dielectric. Incident light is confined by a series of total internal reflections taking place at the boundary of the two different dielectric media (Figure 2.3).

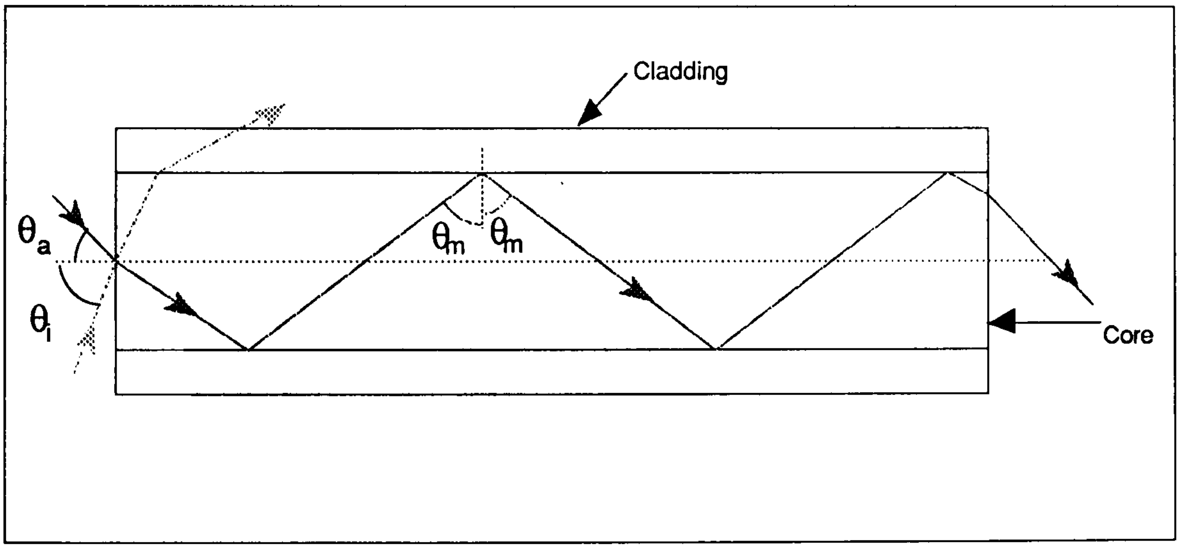


Figure 2.3 Diagram showing propagation of rays of light through a fibre.

As a beam of light encounters the interface between these two dielectrics the relative intensities of the reflected and transmitted waves are a function of the angle of incidence as given by the Fresnel relations. If the beam travels from a region of high refractive index to a region of lower refractive index, then after a certain critical angle all the energy in the incident beam is reflected; This phenomenon is known as total internal reflection. An optical fibre consists of a core of high refractive index cladded by a dielectric of lower refractive index. In such a system a beam of light may undergo a series of total internal reflections and so propagate along the axis of the fibre. The critical angle is dependent on the difference between the core and cladding refractive indices n_1, n_2 respectively. Light propagation within a fibre (figure 2.1) will be confined in the high refractive index region by total internal reflection if the angle of incidence θ_m is: [Hect & Zajac 1974]

$$\theta_m \geq \arcsin\frac{n_2}{n_1} \qquad \text{.. (2.3)}$$

From this equation it can be shown that there is a maximum angle of acceptance beyond which any incident ray will not be guided. In figure 2.3. it can be seen that for angles of acceptance greater than θ_a (e.g. θ_i) the rays leak out of the fibre since the conditions for total internal reflection are not satisfied. The maximum acceptance angle, θ_a, is related to the difference in the core and cladding refractive indices by the equation [Hect, Zajac 1974]

$$n_0 \sin\theta_a = \sqrt{\left(n_1^2 - n_2^2\right)} = NA \qquad \text{... (2.4)}$$

This angle is related to the Numeric Aperture of the fibre, NA, and is characteristic of the fibre itself. Here n_0 is the refractive index of air.

Dispersion occurs in materials used for the fabrication of waveguides. For such materials the Cauchy Dispersion theory shows that the refractive index is wavelength dependent: [Born, Wolf 1970]

$$n = A + \frac{B}{\lambda^2} + \frac{C}{\lambda^4} + \ldots \qquad \text{... (2.5)}$$

Where A, B and C are material constants (normally only A and B are significant). Since the coefficient A and B differ for the dielectrics in the core and cladding equation 2.3 suggests that the fibre birefringence (n1-n2) is wavelength dependent. The dispersion of the core is greater than that of the cladding, this in conjunction with equation 2.5 enable us to deduce that shorter wavelengths are better confined than longer wavelengths. As the critical angle is approached, shorter wavelengths are retained in preference to longer wavelengths. Hence the spectrum of a higher order mode is biased towards the shorter wavelengths.

Although simple Gaussian optics is a convenient way of deducing the quantitative behaviour of light in waveguides, it is a purely geometric method incapable of analysing effects dependent on the wave nature of light. For a fuller understanding of the optical characteristics of fibres, the theory of waveguides may be applied. In principle this is done by solving the Maxwell equations [Feynman 1969] within boundary conditions defined by the fibre geometry. For waveguides with a high degree of symmetry the mode structure of the confined electromagnetic field can be described by analytical functions. In real optical fibres irregularities in the core cladding interface and imperfections in the core structure induces loss of symmetry thereby causing a redistribution of modal energy, so at best waveguide theory yields equations not susceptible to precise analysis. However an approximate answer can be obtained from perturbation theory but this is beyond the scope of this thesis.

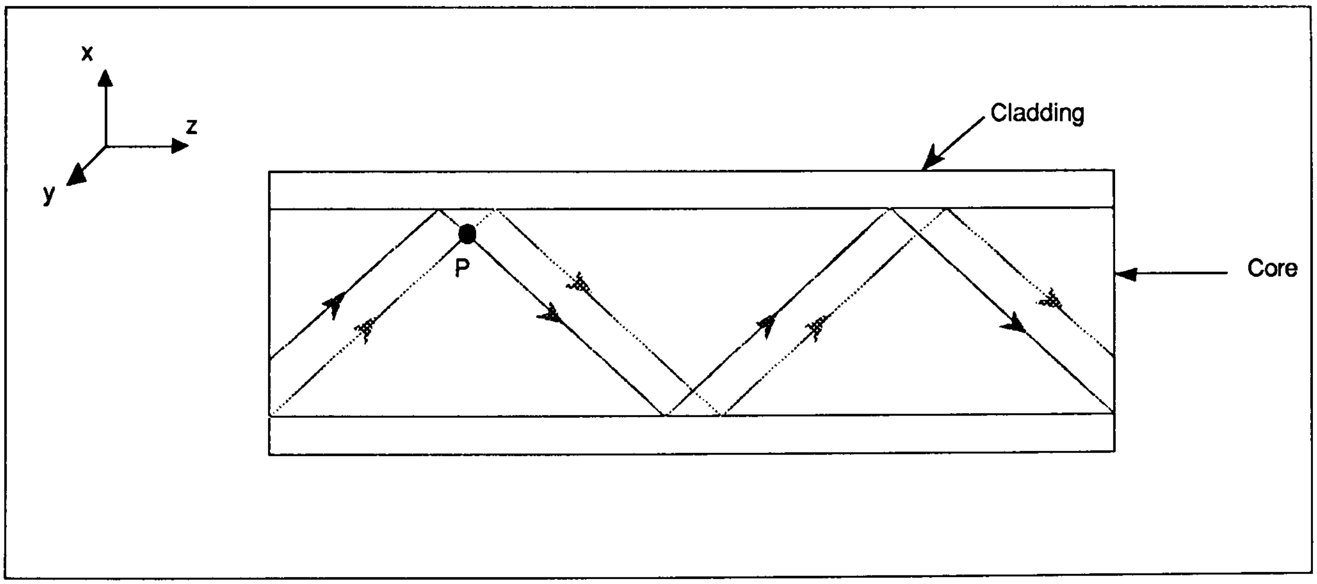


Figure 2.4 Diagram illustrating the effect of superposition of light rays to form modes in a rectangular waveguide.

Waveguide theory considers the propagation of light through the core of the fibre in a number of modes of different order. Consider a rectangular step index waveguide [Trickler 1989] (Figure 2.4). At any point P there will be a superposition of electric fields of upward and downward travelling waves. Then for $\hat{E}$ perpendicular to the paper, the net field at P is:

$$\hat{E} = 2E_0\left(\sin K_y Y\right)e^{i(wt-K_z Z)} \qquad \text{.. (2.6)}$$

which is the sum of two travelling waves of amplitude E_0 That gives rise to standing wave described in equation (2.6). Ky, Kz are the propagation constants in the y and z direction.

$$K_y = \frac{2\pi}{\lambda_y} \text{ and } K_z = \frac{2\pi}{\lambda_z} \qquad \text{... (2.7)}$$

The boundary conditions for the electric field $\hat{E}$ require that their net amplitudes to be zero at the core cladding interface (by analogy with the zero amplitude of oscillation of a string at its supports). Substituting this condition into equation (for $\hat{E}$):

$$\sin K_y A = 0 \qquad \text{... (2.8)}$$

$$K_y A = N\pi \qquad \text{... (2.9)}$$

$$\frac{2\pi A}{\lambda_y} = N\pi \qquad \text{... (2.10)}$$

$$A = \frac{N\lambda_y}{2} \qquad \text{... (2.11)}$$

Where the value of A indicates the number of modes supported by the fibre and N is an integer.

The order of the mode increases from the direction parallel with the fibre axis up to a maximum angle representing the numerical aperture of the fibre [Unger 1977]. The higher order modes are reflected more often from the core-cladding interface than the lower order modes and should be influenced more due to imperfections within the core cladding interface. Thus energy will be transferred from one set of modes to another and some modes will be more susceptible to loss due to waveguide imperfections and external forces such as bending.

2.3. Basic Definitions.

When describing colour there are three basic attributes that can be used. The Hue is an attribute of a visual sensation according to which an area looks similar to one, or a proportion of two, of the perceived colours red, green and blue [Hunt 1987]. When something is said to be green or blue then we are referring to the hue of the colour. The Brightness of a colour is that attribute of a visual sensation according to which an area appears to exhibit more or less light. When an object colour is referred to as bright or dim we refer to the brightness. Colourfulness is an attribute whereby an object appears to exhibit more or less of its hue. Colours that are devoid of hue are called achromatic colours. Words that describe these colours are white, black, grey neutral and colourless. On the other hand colours that exhibit hue are referred to as chromatic colours. The Saturation of a colour is the Colourfulness judged in proportion to its brightness.

2.4. Optical Systems

In this section two kinds of optical vision systems will be described. Firstly the human eye will be discussed to form a base for comparison with computer vision systems. Secondly a brief discussion of a computer based system will be presented. The principles of colour are based greatly on the nature of human perception and this will be further explained in section 2.4.

2.4.1. Human Vision

The human eye is the image acquisition mechanism in the body. It contains a focusing lens and a photo receptive area called the retina (Figure 2.5.).

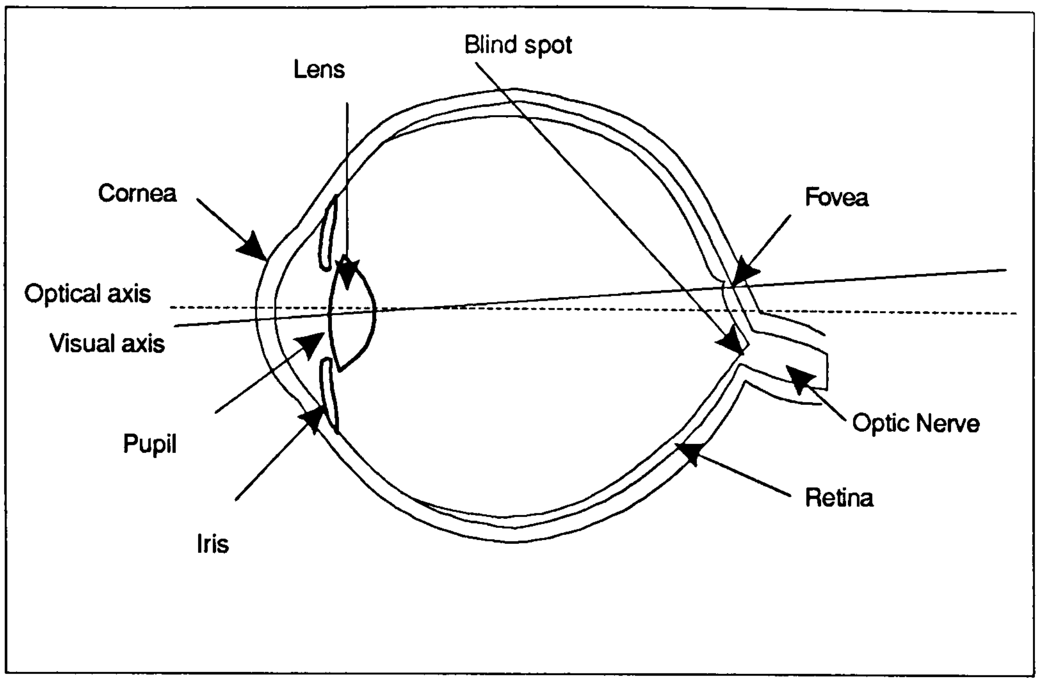


Figure 2.5. Schematic diagram showing a cross-sectional view of the human eye.

The retina of the human eye contains two groups of photoreceptors called rods and cones, The function of the rods is to provide monochromatic vision under low levels of illumination [Grand 1968]. The function of the cones on the other hand is to provide colour vision at normal levels of illumination (several candelas per square meter). The cones can be subdivided into three types, individually sensitive to the red, green and blue regions of the visible spectrum [MacAdam 1942][Brown 1949]. Figure 2.6 shows the spectral response of these three detector types. From the graph it can be seen that the spectral range for human vision is between 400nm and 700nm. This part of the electromagnetic spectrum is referred to as "the visible spectrum ". Figure 2.6 also shows that the cones in the human eye have wide-band overlapping responses and, by utilising information provided by all these detectors, the human is capable of distinguishing a wide range of colours. It is estimated that the human eye can distinguish around 10 million colours [Judd, Wyszecki 1975].

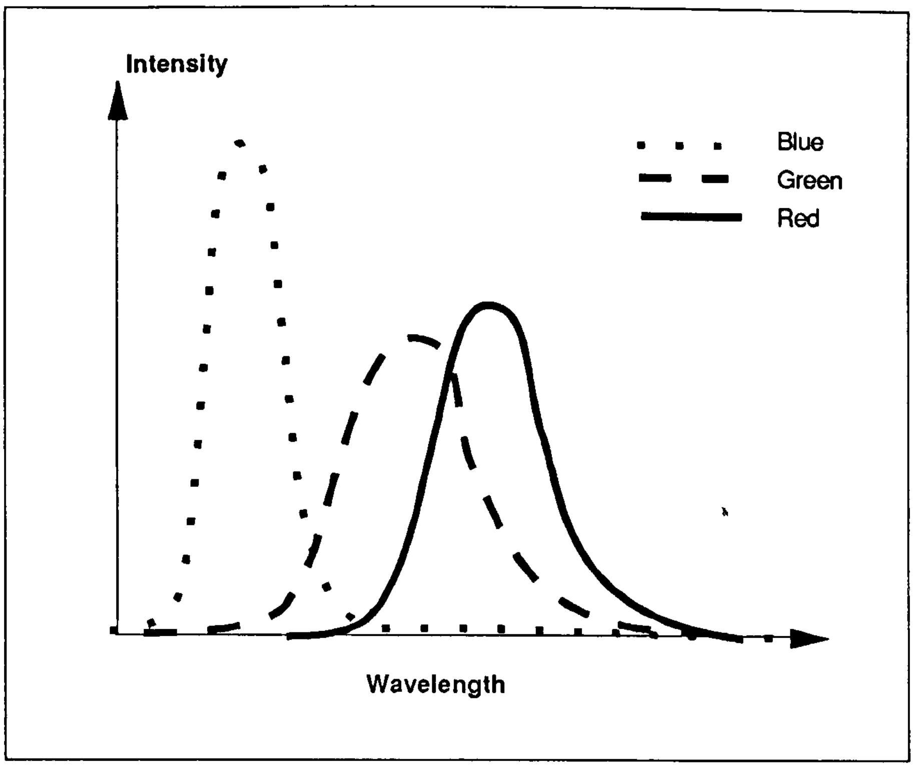


Figure 2.6 Graph Showing spectral response of three detector cone types in the human eye

2.4.2. Computer Vision

The computer vision system can be broken down into two distinct elements namely, Camera (the eye) and the Frame Grabber Card and the Computer (the brain). These two elements will be discussed briefly in the section below.

2.4.2.1. The Camera

The CCD camera is an analogue of the human eye (Figure 2.7). It forms the method of acquiring real world visual information and converting it into an electrical form for display and processing. The CCD element of the camera consists of an array of photodetectors that convert the spatially distributed optical information into electrical signals [Boyle 1970]. This serves much the same purpose as the retina of the eye. For a colour CCD array there are three layers of such detectors that detect three overlapping bands of the visual spectra. These bands or channels correspond to the red, green and blue parts of the spectra. Each of these elements provides an analogue voltage output that varies as a function of the incident light intensity and its spectral profile. The array size determines the spatial resolution of the image and for the standard commercial camera used the array size is 512 x 496 per channel.

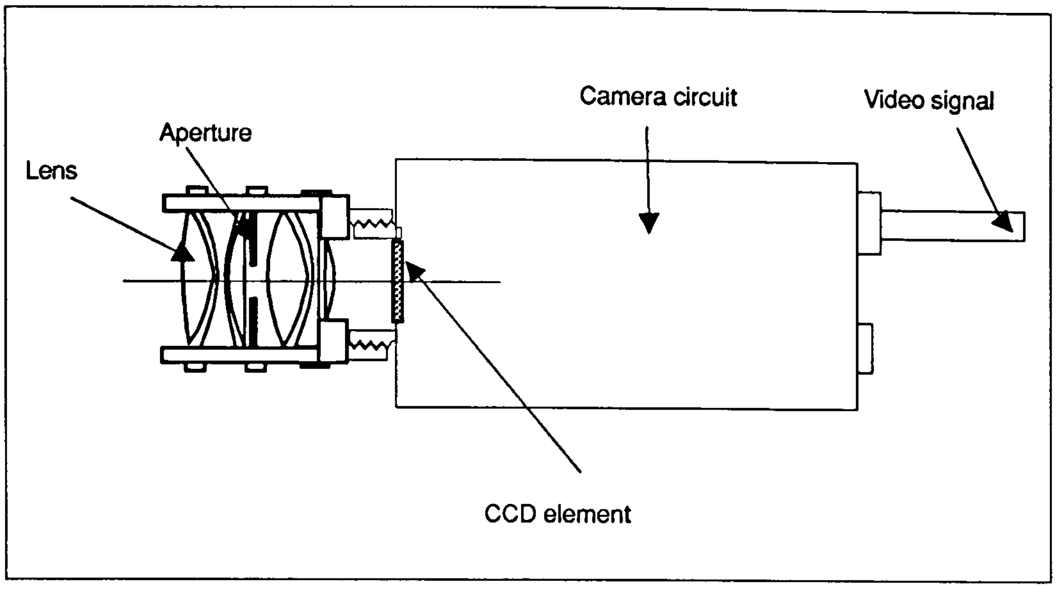


Figure 2.7 Diagram showing principle components of a CCD camera

The optical image falling on the CCD array is scanned from the top left hand corner, to the bottom right hand corner. The image is scanned as a series of horizontal lines, starting from the left of the image going across to the right. Once one line has been completely scanned the scanning is repeated for the next line vertically below. The camera takes 20 ms to completely scan an image.

2.4.2.2. The Frame Grabber Card

The Frame Grabber Card (FGC) is the interface between the computer system and the camera. As input it accepts the analogue image from the camera and stores it in a digital form in its frame store memory. The FGC process and stores each colour channel information separately. The resolution of the FGC is determined by the resolution of the analogue to digital converter (ADC). A commercial colour FGC may have a resolution of 8 bits per channel that means that each colour picture element (pixel) may have one of 256 different possible values for each of its red, green and blue channels. The different combination of values of red, green and blue can produce a total of 16777216 (2^24) unique colours. The FGC uses the frame synchronisation and line synchronisation information to time the ADC and place the digital data in form that can be meaningfully displayed and analysed. The FGC not only allows the host computer to access the framestore memory but provides facilities for controlling the FGC and also performs simple processing of the image.

2.4.2.3. The Computer System.

The computer system is the control platform for the FGC and processing of the image data. The computer system should be able to cope with the speed of the FGC and be able to process the image data sufficiently fast to meet the specifications of the project. In essence the colour of the image is presented in red green and blue (RGB) form. For the purpose of measurement the computer system needs to translate this RGB data to a number of alternative colour forms that provide meaningful interpretation of colour.

2.5. Colour Measurement.

The determination of colour using three detectors is called tristimulus detection. The human eye and the computer vision system described above are examples of tristimulus detection systems. Tristimulus detection is not reliant on the detection of a complete spectral power distribution. It is sufficient to measure three attributes of the spectral power distribution namely the hue, saturation and brightness. Before considering the principles of trichromatic measurement it is important to discuss the most specific method of representing the spectral characteristics of light namely Spectrometry.

2.5.1. Optical Spectrophotometry.

Spectrophotometry is a technique for determining the spectral energy distribution of an electromagnetic wave. This distribution is represented as intensity by a function of wavelength. An example of such a signal is shown in figure 2.8. To achieve this the comparison of intensities is performed using narrow bands of non overlapping filters or detectors centred on wavelengths situated at regular intervals throughout the spectrum (section 2.2). At each wavelength, a comparison is made between the incident light and a working standard. There are a large number of designs for spectral photometry which provide varying degrees of resolution and accuracy. However, they all require a light source, a method of dispersing the light into a spectrum, and a means of detecting and comparing the intensities from the sample and from the standard beams.

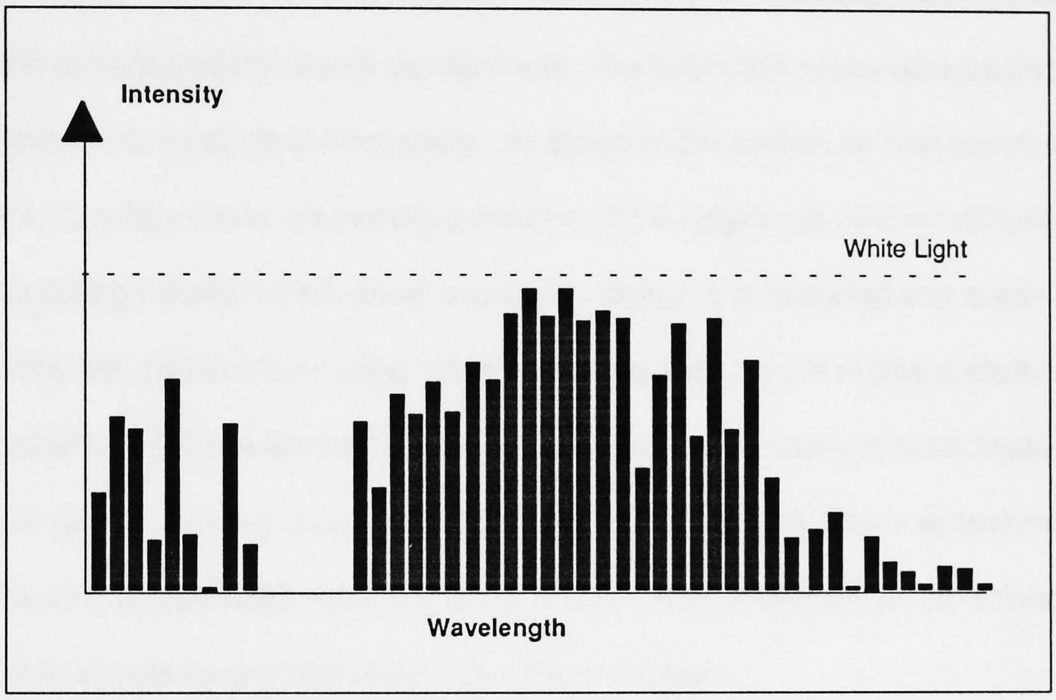


Figure 2.8 Diagram showing example spectro-photometric output

Spectrophotometry is the most precise method of describing the spectral profile and thus the measurement of colour of the incident light. However, the interpretation of this spectral data is complex and, together with the high cost of spectrophotometers, makes it an unattractive technique for general colour measurement. For colour measurement it is necessary to reduce this data to only a few parameters that still provided spectral profile information and also gives a perceptual correlation between the colour of light detected and the observer. This is called colour matching and is described in the section 2.5.2 below.

2.5.2. Colour Matching.

Colour matching is important both for commercial and scientific purposes. Its importance lies in the reproduction of perceived colour, for example, in photography and television. The standard method of colour matching is by using a human observer presented with a colour sample together with three controllable colour sources matched to the human eye response as shown in figure 2.9. The observer adjusts the intensities of the sources until no distinction between the sample and the source can be made. The intensities of the sources then, forms the basic colour components of the sample. As shown in the section on human vision (section 2.4.1), colour vision is basically a function of the responses of three different spectral weighting functions (red, green and blue) . Hence, it is expected that colour can be extracted from the spectral data using similar functions. The advent of cheap electronic detector systems that have spectral responses similar to the human eye have made this approach to colour matching a preferred technique for colorometry (the measurement of colour). The CIE is responsible for most of the definitions of standards for this science and some of these standards are presented in the section below.

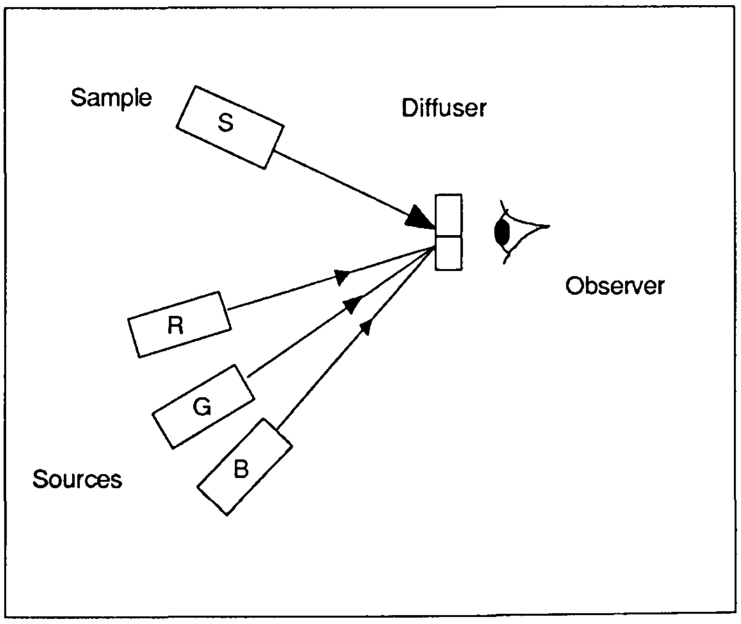


Figure 2.9. Schematic showing method of standard colour matching

2.5.2.1. The Red Green Blue (RGB) Cube

The simplest method of specifying tristimulus colour is the RGB cube (figure 2.10). It is related to the spectrometric approach in that it provides intensity information of the spectrum by considering bands of wavelengths. However, there are a large number of differences which make this technique more useful for colour measurement. Firstly RGB detection is tristimulus. That is, only three areas of the spectrum are considered. More important, however, is that these areas overlap and also have a spectral weighting function. The equations then representing the parameters R, G and B are described below as:

$$R = \int_{-\infty}^{\infty} P(\omega) \times F_{red}(\omega) d\omega \quad ...(2.12)$$

$$G = \int_{-\infty}^{\infty} P(\omega) \times F_{green}(\omega) d\omega \quad ...(2.13)$$

$$B = \int_{-\infty}^{\infty} P(\omega) \times F_{blue}(\omega) d\omega \quad ...(2.14)$$

Where $F_{red}(\omega)$,$F_{blue}(\omega)$ and $F_{green}(\omega)$ represents the spectral weighting functions for the three detectors and $P(\omega)$ is the incident spectral signal. The weighting functions have been standardised by the CIE to correspond closely with the response of the human visual system. This RGB data may be provided by the computer system described above (section 2.4.2.1) or by other systems utilising three photodetectors with overlapping spectral responses. In all trichromatic translations the representation of spectral information in this RGB form is the first step for colour representation. In the raw form the three parameters R, G and B can only give an indication of the spectral content of the light in the specified bands and may be presented as a cube plotting the intensity of the spectrum in the red, green and blue regions. It should be noted that the space occupied, within the cube, by the possible RGB values will depend on the degree of overlap between the detectors. The disadvantage of the RGB colour scheme is that the hue, saturation and brightness of the colour and not readily apparent. Alternative techniques can be used to translate this RGB cube representation into a more satisfactory method of specifying colour. These translations are discussed in the sections below.

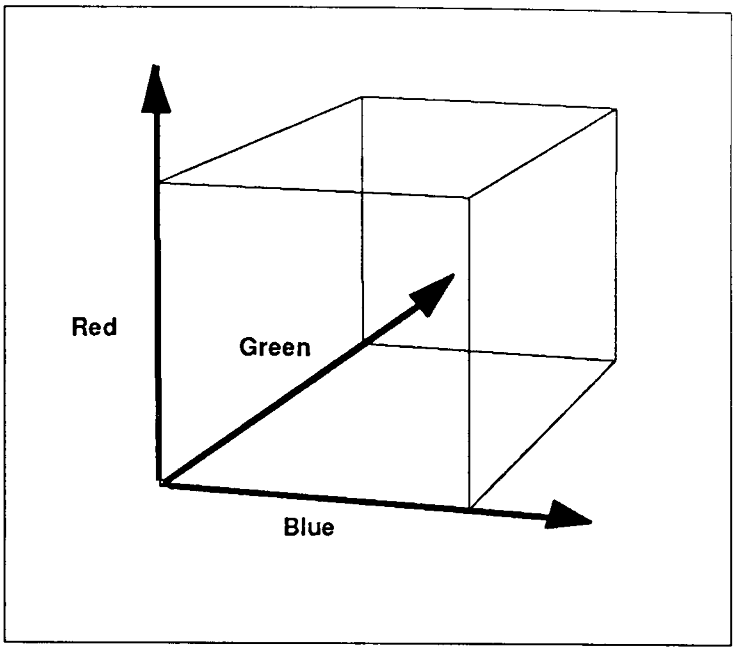


Figure 2.10. Diagram showing RGB Cube representation of colour.

2.5.2.2. The Lxy Colour Scheme

The CIE agreed to define colour in terms of the responses obtained from three idealised photodetectors sensitive to stimuli in the red R, green G and blue B (Section 2.5.2.1 above). From these photodetectors three spectral characteristics can be obtained: The brightness defined as L is the sum of R, G and B and the chromatic co-ordinates x, y and z given by the equations:

$$L = R + G + B \qquad ...(2.15)$$

$$x = \frac{R}{R+G+B} \qquad ...(2.16)$$

$$y = \frac{G}{R+G+B} \qquad ...(2.17)$$

$$z = \frac{B}{R+G+B} \qquad ...(2.18)$$

The three parameters x y and z can be referred to as normalised colour co-ordinates. It is clear from the way that x, y and z are calculated that:

$$x+y+z = 1 \qquad ...(2.19)$$

Hence given two parameters, say x and y, the third can be calculated and so is redundant. Therefore, two co-ordinates can be used to plot a chromaticity diagram. This diagram provides a colour map on which chromaticities of all colours can be plotted (Figure 2.11). The point S is called the white light point and represents the equi-energy stimulus point where x=y=z=1/3. All the achromatic colours map on to this point. The curved line represents the spectral locus and is the boundary for the colour map. Colours that map onto the boundary represent saturated or monochromatic light. One important point to note about the Lxy system is that generation of the x and y values from RGB data are simple and simple electronic techniques may be employed to realise a chromatic system of this nature. Furthermore the chromaticity parameters x and y are independent of the intensities of the individual R, G and B values as they are averaged over the total intensity (R+G+B). This technique then can inherently eliminate effects that change the total intensity of the incident light.

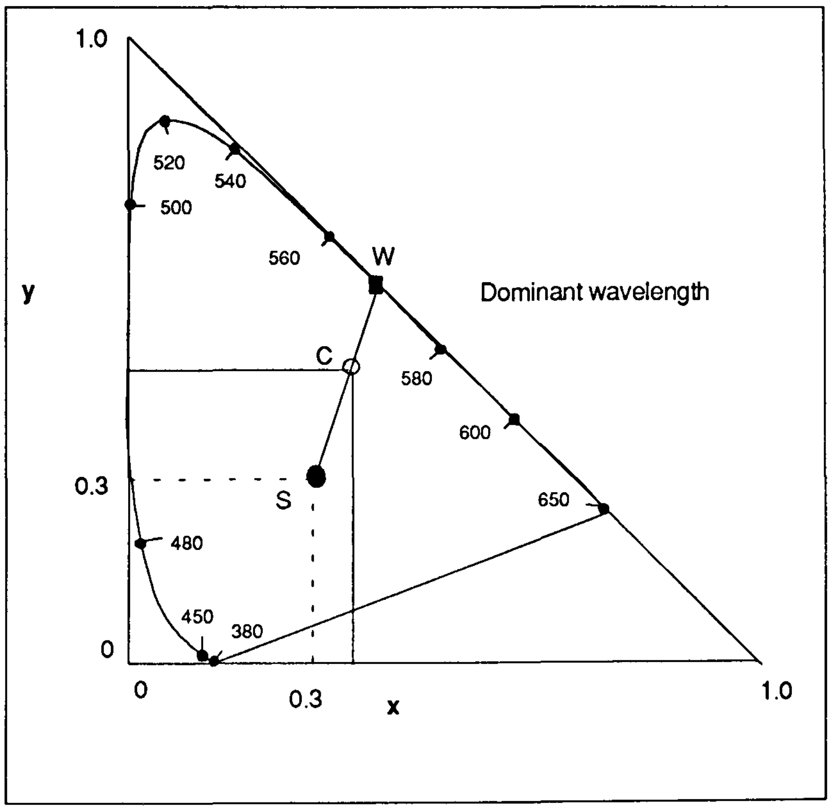


Figure 2.11 A CIE plot of the chromaticity of a sample C showing how dominant wavelength is calculated.

Visualisation of the hue and saturation of a colour using CIE Lxy is easier than the RGB cube. A colour C can be mapped onto the Lxy diagram. The hue is determined as a dominant wavelength W. The dominant wavelength is defined as the point W of intersection of a line with the spectral locus which passes through the source chromaticity, C, the achromatic point, S, and the chromatic boundary. From the CIE Lxy diagram it can be seen that this dominant wavelength does not map uniformly along the spectral locus even though it is monotonic in nature. A measure of the hue can still be provided by using the hue angle H_{xy} defined as:

$$H_{xy} = \arctan\left(\frac{x - x_S}{y - y_S}\right) \qquad ...(2.20)$$

The saturation is described as the proximity of W to the achromatic point. It may be represented as the excitation purity, P_E which is the ratio of the lines SC and SW.

One of the major disadvantages of the CIE Lxy colour mapping system is that the distribution of colour is very non uniform. This can be seen from the mapping of dominant wavelengths on the spectral locus. Alternative systems that use extensions of the Lxy have been defined by the CIE which stretch the CIE diagram to provide a more uniform mapping like Lu'v' and L*u*v* [CIE 1979] . Even if these provide some improvements they still suffer from the same problems in extracting dominant wavelength and saturation information in a simple and direct method. Another colour mapping system proposed was the CIE LAB system which will be discussed in the section below.

2.5.2.3. CIE LAB colour system (CIE 1976)

The CIE L*a*b* was another proposed standard that had the advantage of providing a uniform colour space for mapping. It moved significantly away from the Lxy system mentioned above (section 2.5.2.2) and provided a simple visualisation of the hue and chroma of the light being measured. The L*a*b* colour space is produced by plotting along three perpendicular axes the quantities:

$$L^{*} = 25\left(\frac{100R}{R_{SE}}\right)^{\frac{1}{3}} - 16 \qquad ...(2.21)$$

$$a^{*} = 500\left[\left(\frac{R}{R_{SE}}\right)^{\frac{1}{3}} - \left(\frac{G}{G_{SE}}\right)^{\frac{1}{3}}\right] \qquad ...(2.22)$$

$$b^{*} = 200\left[\left(\frac{G}{G_{SE}}\right)^{\frac{1}{3}} - \left(\frac{B}{B_{SE}}\right)^{\frac{1}{3}}\right] \qquad ...(2.23)$$

Where R_{SE}, G_{SE} and B_{SE} correspond to the reference white light co-ordinate. The constants used provide a close correlation with the Munsell renotation scheme of colour [Hunt 1987]. The ratios R/R_{SE}, $G/G_{SE}$and B/B_{SE} are used for normalisation of the light.

Because the tristimulus values are incorporated as cube roots there is no chromaticity diagram associated with the CIE LAB space and also no correlation with saturation. However, hue H_{ab} and chroma C^{*}_{ab} may be easily extracted from it using similar functions to the Lxy systems

$$H_{ab} = \arctan\left(\frac{b^{*}}{a^{*}}\right) \qquad ...(2.24)$$

$$C^{*}_{ab} = \sqrt{a^{*2} + b^{*2}} \qquad ...(2.25)$$

The LAB system has pleasing graphical representation of colour but suffers from having a more complex method of derivation and can be realised only by the use of computation or complicated electronic systems.

2.5.2.4. Hue, Lightness and Saturation (HLS) Colour Scheme

A colour scheme developed for computers and television systems for colour matching is the HLS colour scheme [Niblack 1986] . This technique moves away from the CIE standards described earlier but has important benefits, in that it provides a direct mapping of hue lightness and saturation of a colour from the original red green and blue data. The hue H_{HLS}

is calculated first by finding the minimum of the three stimuli R, G and B. Then the three parameters are reduced to two by subtracting out the minimum response:

$$R' = R - \min(R,G,B) \qquad \text{...(2.26)}$$

$$G' = G - \min(R,G,B) \qquad \text{...(2.27)}$$

$$B' = B - \min(R,G,B) \qquad \text{...(2.28)}$$

So if blue was the minimum response then B' would be zero. Then a set of conditional functions are implemented to provide the hue value:

$$\text{if } B' = 0: \; H_{HLS} = 120\left(\frac{G'}{R'+G'}\right) \qquad \text{...(2.29)}$$

$$\text{if } R' = 0: \; H_{HLS} = 120\left(1+\frac{B'}{G'+B'}\right) \qquad \text{...(2.30)}$$

$$\text{if } G' = 0: \; H_{HLS} = 120\left(2+\frac{R'}{B'+R'}\right) \qquad \text{...(2.31)}$$

The hue value is presented as an angle ranging from 0º to 360º and segments the full circle into three sectors representing hue ranges red to green, green to blue and blue to red respectively.

The principle behind this approach is to first remove the white light component from the incident light (by subtracting out the minimum value) and then performing simple distimulus detection techniques for calculating hue angle.

Lightness L_{HLS} is calculated simply as:

$$L_{HLS} = \frac{\max(R,G,B)+\min(R,G,B)}{2} \qquad \text{...(2.32)}$$

and the Saturation $S_{HLS}$is:

$$S_{HLS} = 1-\frac{\max(R,G,B)-\min(R+G+B)}{\max(R,G,B)+\min(R+G+B)} \qquad \text{...(2.33)}$$

where max(R,G,B) represents the maximum value between R, G and B.

The importance of this technique is that it does not try to match colour according to the human perception but provides a meaningful quantitative representation of hue, lightness and saturation. The algorithms are not over simplistic like the Lxy colour scheme or as complicated as the L*a*b* scheme. They are intended mainly for computational purposes. However, they may also be realisable using suitable electronic techniques.

So far the techniques for colour measurement have been discussed. The importance of this theory for this work lies in its application to sensing. The principles of such sensing systems are presented in the following sections.

2.6. Chromatic Sensors

When we look at an apple and by its colour determine its ripeness, we are performing the task of chromatic sensing. We base our decision on the colour of the apple, the lighting conditions and our experience. Similarly, techniques for automated chromatic sensing should employ some of these basic principles. The main requirement of chromatic sensing is that the object being monitored varies its chromaticity with the measurand. Two types of chromatic sensing systems can then be realised. Firstly, systems that show intrinsic changes in the spectral content with specific parameters. These systems can be monitored directly without the need of a specific sensor or external light source. Examples of intrinsic chromatic systems are the spectral emissions from plasmas, flames and speckle patterns. The second type of chromatic systems relies on the conversion of the required parameter to a chromatic change by the use of a suitably selected sensor. These extrinsic systems usually require external light sources and the location of a sensor on, or in, the system being monitored. Examples of extrinsic chromatic systems are Thermochromic strip temperature monitoring, Photo-elastic stress monitoring and various chemical indicators.

The sections below present a brief overview of the principles involved in the various chromatic systems mentioned above.

2.6.1. Chromatic Stress Sensing

Chromatic stress sensing is performed using materials that exhibit a chromatic change when stressed. Such materials are called photoelastic materials. Materials which exhibit photo elasticity belong to a class of materials which are optically anisotropic [Read 1981]. These media have optical parameters (like refractive index (section 2.2)) which depend on the direction of light propagation. Such materials exhibit double refraction or birefringence. Birefringence is observed when a ray of light enters a medium and is split into two separate rays called the ordinary and the extraordinary rays [Jessop, Harris 1960]. The refractive index for the ordinary wave is constant and therefore the ordinary rays propagate at a constant velocity. However, the refractive index for the extraordinary rays varies according to the direction of propagation and hence so does its velocity. The ordinary rays and the extraordinary rays are both plane polarised (section 2.2) in mutually perpendicular planes. When polarised light is incident onto such a material (Figure 2.12) it is split into two components and because of the differences in their velocities through the material they have a phase difference. The resultant emerging light is the result of interference between these two out of phase components. If this light is observed using a second polariser oriented orthogonally with respect to the first then the intensity obtained will be [Zandman 1977].

$$I = K\sin^2(2\theta)\sin^2(\Delta/2)\cos^2(\omega t) \qquad \text{...(2.34)}$$

Here K is a constant of proportionality, θ is the angle of the polariser relative to the extraordinary axis of the material and Δ is a function of thickness *h*, wavelength λ and the birefringence of the material,δ_B[Jessop, Harris 1960]:

$$\Delta = \frac{2\pi h\delta_B}{\lambda} \qquad \text{...(2.35)}$$

Because this intensity function is wavelength dependent certain wavelengths are preferentially extinguished and this results in the emergence of coloured light from such a system. Photoelastic materials show a proportional change in birefringence with stress. If such a material is differentially stressed coloured fringes are observed when viewed through orthogonal polariser arrangements.

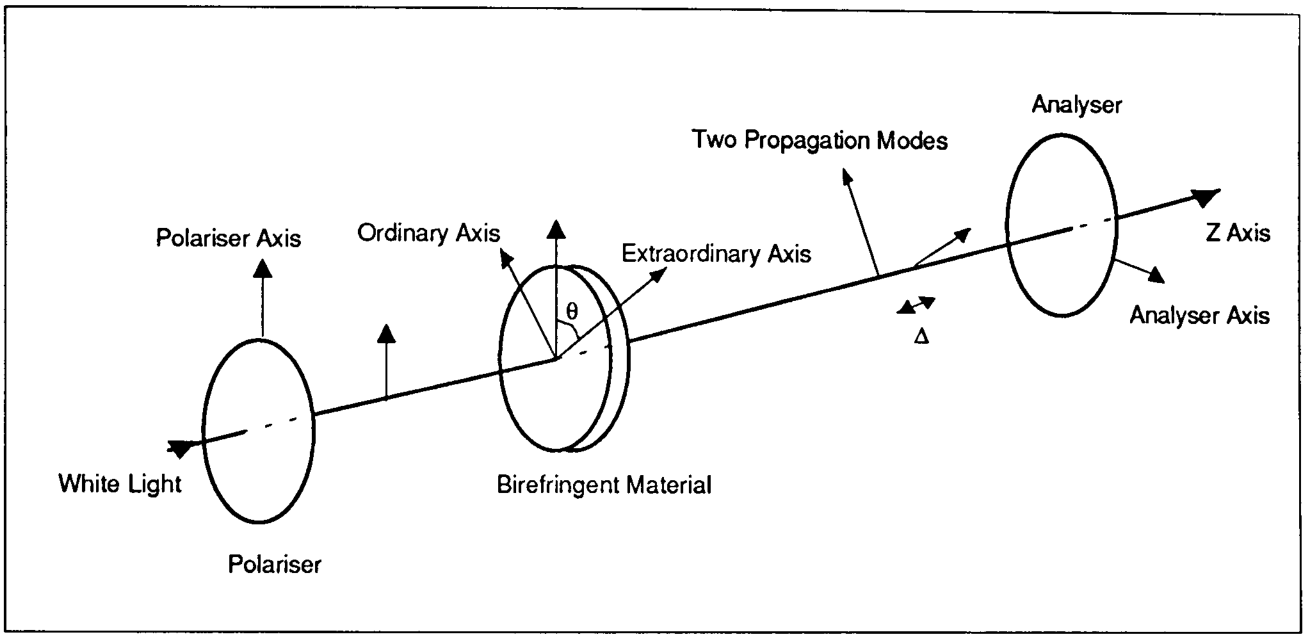


Figure 2.12 Diagram showing propagation of polarised light through a birefringence material.

2.6.2. Thermochromic Liquid Crystals

Thermo-optic effects in liquid crystals occur when the temperature of the liquid crystal is varied in the vicinity of a phase transition[Raynes 1983]. One group of liquid crystals that exhibit thermochromic effects are cholesteric materials. Cholesteric liquid crystals are helically ordered (figure 2.13). In a well planer sample, Bragg like reflections of one sense of polarised light occur from the helical plane at a wavelength related at normal incidence to the average reflective index $\bar{n}$ and pitch P by [Elser, Ennulant 1976]:

$$\lambda = \bar{n}P \qquad ...(2.36)$$

For thermochromic effects the pitch of the sample and the displacement angle can vary with temperature to produce changes in the surface chromaticity of the material (figure 2.14). These chromatic changes can therefore be directly related to temperature.

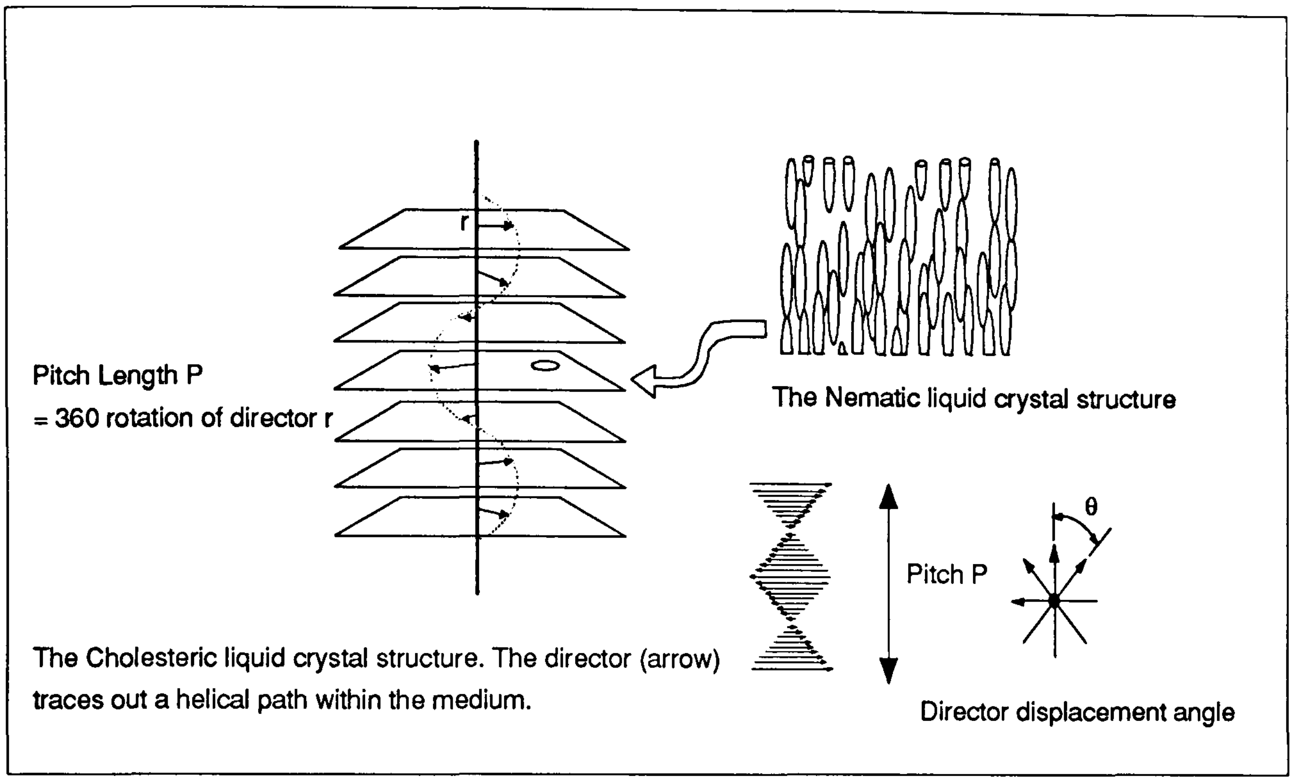

r
Pitch Length P
= 360 rotation of director r
The Nematic liquid crystal structure
Pitch P
θ
The Cholesteric liquid crystal structure. The director (arrow) traces out a helical path within the medium.
Director displacement angle

Figure 2.13 Diagram showing structure of Cholesteric Liquid Crystals in Thermochromic materials.

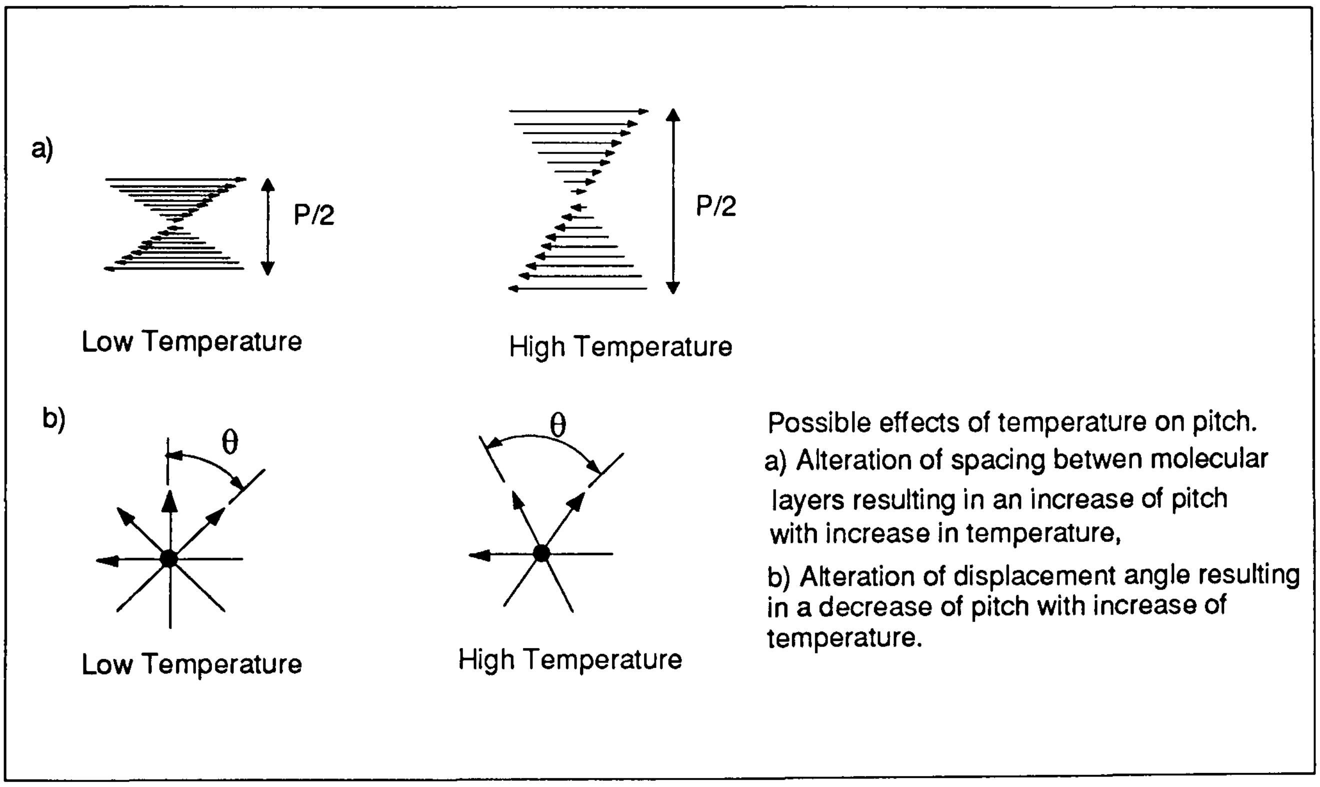

a)
P/2
Low Temperature
P/2
High Temperature
b)
θ
Low Temperature
θ
High Temperature
Possible effects of temperature on pitch.
a) Alteration of spacing betwen molecular layers resulting in an increase of pitch with increase in temperature,
b) Alteration of displacement angle resulting in a decrease of pitch with increase of temperature.

Figure 2.14 Diagram showing How temperature can affect optical properties of Thermochromic Liquid crystals.

Thermochromic devices are commonly fabricated using one of two methods. Firstly, they may be in the form of an ink where tiny droplets of liquid crystals are micro encapsulated in polymer shells (diameters 5 -50 μm). The ink is formed from a slurry of these capsules in an appropriate base. This method is used in commercially available liquid thermo-chromic materials. Secondly, a dispersion of the liquid crystal droplets in a binder polymer film can be used directly without the need for encapsulation. This is a more common from of thermo-chromic materials and is usually presented as plastic strips or pads. By altering the constitution of these crystals a wide range of temperatures may be addressed with varying temperature resolutions.

2.6.3. Chemical Indicators

Chemical indicators form the basis for a large number of systems for monitoring a wide range of parameters. Perhaps the commonest form of chemical indicator is litmus Paper for monitoring pH. All chemical indicators show an observable change in their optical property with measurand. Ideally they are independent of other parameters that system possesses. There exists a wide range of indicators that exhibit optical change with measurand. There are a number of effects that may cause such an optical change. Firstly, an indicator may consist of two or more optically different components. The introduction of a sample for measurement causes a change in the ratio of these indicator components and thus an optical change is observed. Alternatively the introduction of a sample may be used to induce a change in the absorption characteristics of the indicator. Usually chemical indicators that show optical changes are designed so that they can be identified by human inspection. Some of the indicators that have been investigated are presented in the sections below.

2.6.3.1 Nephur™ Medical Indicators:

Combur™ and Nephur™ [Myllia 1991] strips are the basis for tests for various medical conditions. They consist of a number of different indicator patches, each sensitive to a specific component of a urine sample from a patient and are used for approximate medical diagnosis. The presence of various chemicals in the urine changes the colour of the pads on the strips to vary with respect to a "normal" standard also provided with the strip. The strips are provided with a chart showing the relationship of the colour changes observed on the strips with the measurand in the urine sample. The user would use this chart to match the colours on the test strip and be able to obtain some quantitative information of urine sample (figure 2.15). This colour matching is a subjective matter and so provides only a crude method of quantification.

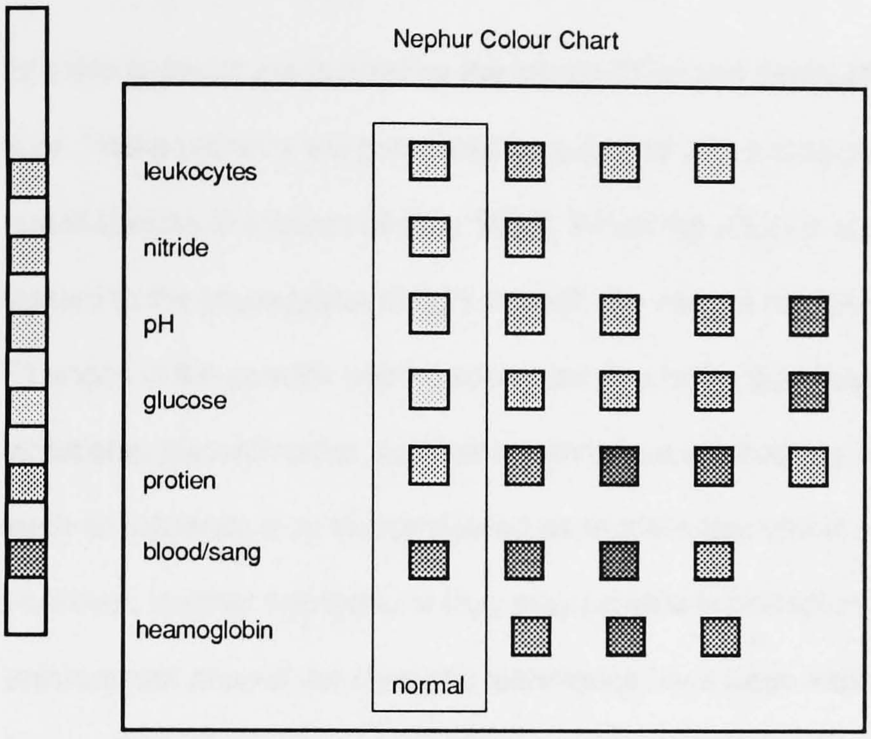


Figure 2.15. Diagram showing Nephur colour chart for comparison with strip for quantifying urine composition.

2.6.3.2 Oil Monitoring.

Oils are used in an industrial environment where mechanical and high power system need cooling and/or lubrication. The efficient performance of such equipment depends a great deal on the quality of the oil which degrades with use. Electrical instrumentation for monitoring such degradation is often difficult to implement because of the demanding environments in which they need to function and because of a lack of suitable sensor. In such instances the oils are changed at regular intervals regardless of the oil condition. The problems associated with dead time in servicing the equipment and changing the oil either too early or too late is obvious. Optical techniques provide an alternative solution to the problems associated with oil quality monitoring, providing a non-electrical (so safe) and intrinsic sensor (the changes in the oil's optical characteristics) design. Development of such a system has been performed using optical fibre chromatic techniques at the University of Liverpool and has shown good performance characteristics [Khandakar et al. 1993].

2.6.4. Speckle Modulation

Speckle patterns are formed by the constructive and destructive interference of coherent light. These patterns are generated by a source with a long coherent length and narrow spectral width like lasers [Senior 1985]. When the effect is observed in an optical fibre it is related to the propagation of light through the various modes of the fibre [Ahmed 1990]. Changes in the speckle pattern can occur due to the disturbances along the fibre such as vibrations, discontinuities, connectors and source/detector coupling. In some applications such an influence may be considered as modal noise and is an undesirable feature. However, in other applications they may provide important information about the environment around the fibre and techniques have been implemented that utilise speckle patterns for sensing purposes [Ahmed et. al. 1991][Cosgrave 1992]. Both intensity and chromatic techniques may be employed to interrogate the speckle pattern and extract usable information [Moore 1990] .

For fibres that accommodate a large number of modes (section 2.2.1). (A property associated with the cross-sectional diameter of the fibre and its numeric aperture) the analysis of the speckle pattern itself is very difficult. However, a statistical analysis of the whole speckle image is possible and provides information about the modal propagation through the fibre by considering ringed regions of the speckle [Smith et. al. 1992]. First order statistical analysis of speckle formed by a coherent light may be expressed as a complex Gaussian process [Wolf 1976]. The joint probability function of the real and imaginary parts of the field, denoted by A_R and A_I respectively is given by [Wolf 1976]:

$$p(A_R, A_I) = \frac{1}{\pi S^2} \exp\left(\frac{-\left(A_R^2 + A_I^2\right)}{S^2}\right) \qquad ...(2.37)$$

where s is the standard deviation of the random distribution. If the real and imaginary parts of the field have joint Gaussian distribution, then the modulus of these functions has a Rayleigh distribution [Wolf 1976] with the probability density function of the intensity expressed as [Wolf 1976]:

$$p(I) = \frac{I}{\langle I \rangle} \exp\left(\frac{-I}{\langle I \rangle}\right) \qquad \text{for } I \geq 0 \qquad \ldots(2.38)$$

$$= 0 \qquad \text{for } I < 0 \qquad \ldots(2.39)$$

Where I is the Intensity and $\langle I \rangle = s^2$ is the mean intensity. This suggests that the probability distribution function is a negative exponential distribution and as this function has maximum entropy [Wolf 1976] it indicates that a normal speckle pattern is totally random. First order statistical analysis shows the randomness of the speckle image but is insufficient for describing the nature of speckle observed emergent from a fibre with certain modes preferentially excited [see R. Smith et. al. 1992]. If the coherent light is injected normal to the principle axis of a multi mode fibre then all modes are excited equally. However, if light is injected at an angle (Figure 2.16) then the higher order modes are excited preferentially. This can be observed as a halo of speckle with few inner modes excited.

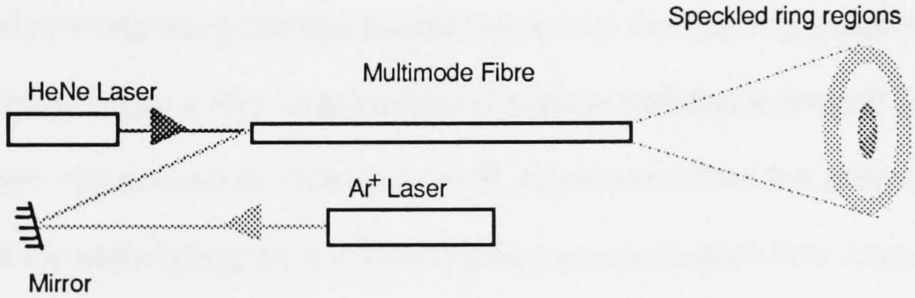


Figure 2.16. Diagram showing generation of chromatic speckle patterns and the formation of rings.

The use of two or more of such sources functioning at different wavelengths chromatic information can be obtained. The chromaticity of a two source system at a specific point on a speckle pattern is the ratio of the intensities of the two wavelength components at that point. There are a number of different effects that can change the transmissive properties of a fibre which include microbending and radiation [R. Smith 1993]. They tend to differentially affect the modes and can be monitored in the speckle pattern profile. In this way the fibre propagating the light can act as an intrinsic detector for monitoring these specific parameters.

2.6.5. Plasma Monitoring

An example of Electric Plasma monitoring is the measurement of the optical emissions from glow discharges [Khandaker 1994]. Like speckle monitoring (section 2.6.4) no sensor or auxiliary light source is required. Under normal conditions gasses are electrically insulating in nature. However, if a gas is suitably energised e.g. by the application of a high power radio frequency field its atoms becomes ionised [Russell et. al. 1990] and the gas becomes electrically conducting. If the gas concentration of the electrons and ions are approximately equal and uniformly distributed the resulting medium is known as a plasma. The highly energetic state of ions and electrons leads to the emission of energy in the form of light so that the plasma appears luminous. There are a number of effects that cause radiation from a plasma. They include braking radiation caused by elastic collisions of electrons and ions, radiation of photons caused by the electrons moving from a high to a low energy state and absorption and re-emission of photons by different ions in the plasmas [Beaven 1989]. As this emission occurs from a very large number of plasma particles a spectral signature of the plasma provides vast amount of statistically valid information about the state of the plasma. An example of the use of plasmas is in neon lighting where electron flow creates a plasma glow discharge in a transparent tube. In the Semiconductor industry low pressure plasmas are used for the processing of semiconductor devices [Beaven 1989] where a high degree of precision in etching and deposition is required. In this field the careful monitoring and control of the plasma are essential for good process operation. Until recently the work of controlling the plasmas was essentially a black art where experts controlled the multitude of plasma properties according to experience and observation after trial and error runs [R.V. Smith 1992]. However, with the introduction of artificial expert systems like Intelligent knowledge based systems (IKBS) [R.V. Smith 1992] the interpretation and processing of various plasma properties allows scope for automatic plasma control. One major problem that exists in the monitoring of plasmas is that electronic monitoring systems are not suitable because of the difficult nature of the environment. Not only is the electrical and thermal environment within a plasma hostile to electronic devices but also the introduction of any such monitoring system would itself affect the plasma and degrade process performance. Therefore remote monitoring methods become attractive for such applications. Thus analysis of the optical signal from the plasma may provide information about the plasma pressure, sustaining electrical power and the etching and deposition rate of the materials being treated. Although the use of spectrophotometric methods is widespread they are usually slow and provide too large a data set making information processing difficult. A compromise is therefore to use chromatic techniques. Preliminary tests have shown such an approach to be useful for such purposes [Khandakar et. al. 1994].

2.7. Summary

The fundamental and relevant optical principles have been discussed. The basic concept of colour has been introduced and the various techniques for colour quantification are presented. One of the simplest techniques for measuring colour is the use of chromaticity x and y parameters form the Lxy colour matching scheme. It is shown that these chromaticity values are intensity independent.

Examples of systems that exhibit chromatic change have been presented. Their related theory has been discussed to identify the means whereby chromatic changes are generated.

The next chapter discusses the manner in which image processing can be applied for identifying objects for further manipulation.

CHAPTER 3
Digital Image Processing

3.1 Overview of Digital Image Processing

Digital Image Processing is the computer analysis of images stored in a digital form. It is a well established technology and has been available for over 3 decades. The most visible origin of this technology has been the NASA unmanned space program [Green 1983]. The great development in the production of fast and application specific computers have made digital image processing an affordable technology that can be used to solve practical problems in the real world.

The advantages offered by Digital Image Processing are [Niblack 1986]:

- Digital imaging systems are capable of acquiring images from a wider spectral range of sources than the human eye.
- Digital images can present a large amount of information in a compact and easily interpreted form. A Digital image may contain several million bits of information.
- Digital Image Processing can manipulate images that cannot readily be manipulated by non-digital techniques.

The availability of such a powerful technology has made possible a wide range of new technologies that otherwise would have been impractical. The areas in which digital image processing has found application include [Judd, Wyszecki 1975]:

- Meteorology
- Astronomy
- Particle Physics.
- Magnetic Resonance Imaging.

- Ultrasonic CT scanning.
- X-ray Imaging.
- Electron Microscopy.
- Surveillance Systems.

In all of these areas most specific image processing tasks are accomplished using a mixture of a number of smaller well defined image processing modules. The availability of a large number of such modules 'off the shelf' means that the user needs only to choose the strategy of connecting together these modules to accomplish the task at hand.

This chapter presents the techniques necessary to solve a specific problem; that of identifying a sensing region in the image and being able to track it. It is assumed that the sensor characteristics are already known and are simple in design. Before proceeding to the sections specific to Image processing it would be pertinent to present some basic definition of image processing terms that will be used in the sections below.

3.2 Basic Definitions:

A digital Image can be considered as an array of numbers. The location of these numbers in the array relates directly to the spatial information of the image data. Each elementary number is called a pixel [Boyle 1988] and the value of the pixel contains the image data itself. The location of a pixel in the image array is represented in parametric form (i, j). Where i and j represents the pixel column and row from the top left hand side of the image. Each pixel is composed of a discrete number of elements called Bits. These can take a logical value of 1 or 0. The number of bits used to represent one pixel determines the resolution of the Image. Thus if 8 bits were used to represent the pixel of an image then a pixel can be in only one of 2^8 (256) possible states. This can be stated as 8 bit resolution or 256 grey levels. Figure 3.1. shows the relationship between a digital image, a pixel and a bit. A digital window is a small defined area of pixels inside the image array. A template is an array that is used to interrogate the pixel data in a window. If a number of images are acquired over a period of time then each image is referred to as a frame.

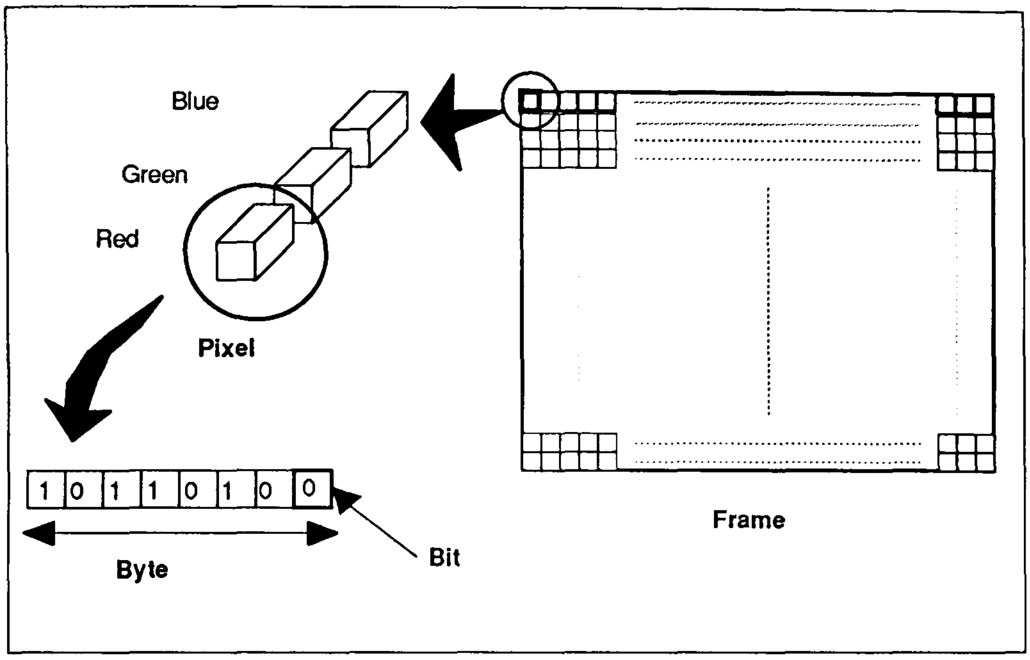


Figure 3.1 Diagram showing relationship between Frame, Pixels, Bytes and Bits.

3.3 Scope of this Chapter:

There is a wide and varied range of different techniques for image processing. The aim of this section is to present the underlying techniques used for the present application for identifying and tracking the image and for passing co-ordinates of the sensor location to the data analysis module (Chapter 4). The following sections are arranged in the order of processing performed on the image to extract and track the sensor in the field of view.

3.4 Image Processing Solutions to Sensor Location & Tracking:

Given an image there are a large number of routes that can be taken to extract the desired information. The wide array of computational algorithms available of varying degrees of complexity and the scope of cascading these processes means that a designer needs only to find one specific combination of these modules to achieve the desired goal. However, for real world practical applications, a number of important factors have to be considered. These are:

- The computational resources that are available.
- The speed of the operations that is necessary
- The flexibility of the system.
- and The complexity of the image and the object features.

The flexibility of the system is a parameter over which the designer has the most control and it determines that efficiency and performance of the system in real world image processing. Poor consideration of the system flexibility may mean that the object may not be identified if rules for identification are too stringent and alternately non-objects are being tagged as objects if the system is slack.

3.4.1 Image Data Acquisition

This section deals with the process by which the digital images are acquired for processing. The main components for this operation are the camera and the frame grabber card. These two modules are described in the section below.

3.4.1.1 The CCD Camera.

The images processed by the computer need first to store information in a digital from suitable for processing. The acquisition of visual information is achieved using a Charge Coupled Device (CCD) array Camera. The CCD array can be considered as a matrix of photodetectors that convert the optical image into an analogue voltage signal. The voltages of each of the elements in the CCD array are transmitted sequentially working from the top left to the bottom right and sampling across the image horizontally for a frame.(figure 3.2). The time duration between consecutive frames is known as the frame period. For a colour CCD array three detectors with different spectral characteristics are used for each of the picture elements (pixels). The array provides an indication of the picture quality. The greater the array size the higher the quality of the camera image. The CCD camera used in this project consisted of 768x512 tri-colour elements with a frame frequency of 50Hz. The camera output signal was three raw analogue signals (channels) which represented the three pieces of colour information of the image. This analogue signal was not directly interrogateable by the computer. The interface between the camera and the computer was the Frame Grabber Card (FGC) and this will be described in the section below. A more detailed description of the CCD camera is presented in section 5.3.1.

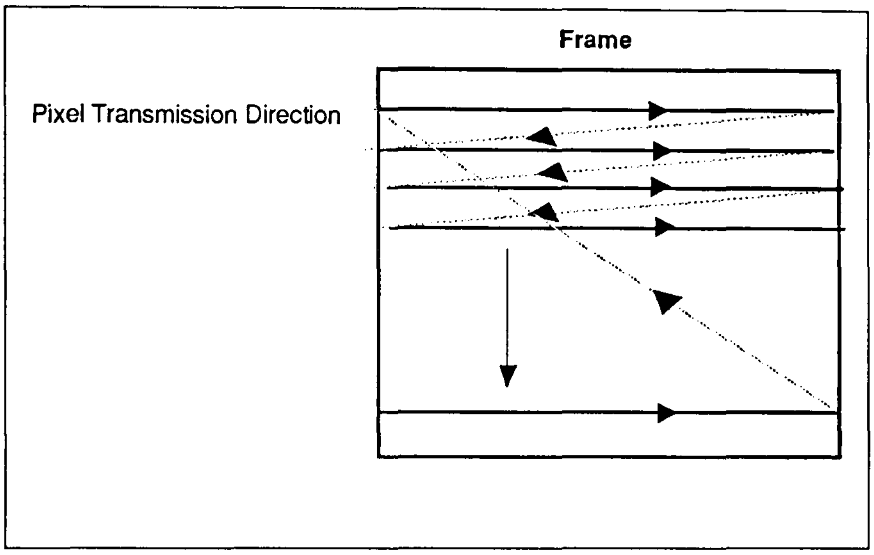


Figure 3.2. Diagram showing the order in which pixel data is transmitted from a frame.

3.4.1.2 The Frame Grabber Card.

A Frame Grabber Card (FGC) is used to convert the camera signal into a digital value and then store this in its internal memory. The FGC must be carefully chosen so as to be able to process the image data provided by the camera. The FGC is a three channel system operating at a frame acquisition frequency of 50Hz with a pixel resolution of 8 bits per colour channel and a frame memory size of 768 x 512 per channel. The FGC can be controlled by the computer to perform different tasks of frame acquisition and capture. Also it has the capability of performing simple image processing tasks by the use of internal look up tables. A detailed description of the image acquisition system used in the present work is given in section 5.3.

3.4.2 Image Data Reduction.

To enhance the speed of processing of a single frame the image needs to be reduced. This reduction is important for two reasons. Firstly it reduces the memory demands on the processing system and secondly it increases the speed of other processing tasks that are heavily dependent on the image size. Two forms of reductions can be performed on an image. The image array size can be reduced or the pixel size can be reduced. The original image is a matrix of size 768 x 512 with a pixel size of 24 Bits (3*8 bits per RGB Channel) . An averaging technique is used to reduce this image to a 96 x 64 image. This technique takes an 8 x 8 window in the image and averages over this area to provide the mean pixel value and then creates a new image data. The process is repeated for the next 8 x 8 pixel area of the original image until the whole image is reduced. Apart from the obvious advantages of a reduced matrix size, this technique also reduces the noise in the image. The Disadvantage of this step is that it reduces the detail of the resultant image. However, if the object of interest occupies a large area in the image the reduction in detail is not restrictive [Davies 1984].

Reducing the pixel size is achieved by stripping the image of colour information. Since the image processing techniques described below perform well with grey scale intensities the 24 Bit colour information can be reduced to an 8 Bit Grey scale. The equation used to perform this transformation for a pixel (i, j) is [Grand 1968]:

$$\text{Grayscale}_{i,j} = \frac{B_{i,j} + 3G_{i,j} + 2R_{i,j}}{6} \qquad \text{...(3.1)}$$

This equation approximates to the human eye response, which takes the brightest colour to be yellow (Red and Green) and the darkest colour to be deep blue. This equation provides a visually reasonable representation of a colour image and also retains some colour information in the resultant grey scale image.

If both these processes are completed the Image data is reduced by a factor of 192 (8 x 8 x 3). This data reduction process is a time-consuming operation, taking approximately 5 seconds for the given system.

3.4.3 Edge detection Filtering:

The theory of digital image filtering has origins in linear systems theory [Boyle 1988]. Edge detection is a method of identifying the boundaries of an image [Lunscher 1986]. This is achieved by considering the rate of change of intensity around a given pixel

3.4.3.1 Robert's Operator:

On of the simplest edge detection technique is the Robert's operator [Roberts 1965]. Two templates as shown in figure 3.3 are used to give a measure of intensity change in two orthogonal directions.

$$\Delta_1 = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array} \qquad \Delta_2 = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline\end{array}$$

Figure 3.3 Templates used for Robert's Operators.

For any pixel (i, j) The Robert's operator yields two numbers.

$$\Delta_1 = f_{i,j} - f_{i+1,j+1} \qquad \text{...(3.2)}$$

$$\Delta_2 = f_{i+1,j} - f_{i,j+1} \qquad \text{...(3.3)}$$

These two numbers represent the gradient between two diagonally related pixels.

The gradient magnitude is:

$$S_{i,j} = \sqrt{\Delta_1^2 - \Delta_2^2} \qquad \text{...(3.4)}$$

$S_{i,j}$ provides a measure of the rate of change of intensity near a pixel (i, j). Thus if a suitably chosen threshold T is used then:

S >= T indicates edge present

S < T indicates no Edge.

The choice of T is subjective and is a critical step in the edge detection process. The choice of T depends upon the image sharpness and contrast. Using too small a value of T would reduce the sharpness of an edge and, at worst identifies non-edge pixels as edges. On the other hand using too large a value for T would provide very faint and unconnected edge data and at worst fail to identify significant edges in the image.

Figure 3.4 shows an image and the effect of applying Robert's operator to the system. The edge detection algorithm described by Robert's operator provides no indication of the direction of the gradient in the image. A Borland™ C++ program listing of the Roberts transform is given in the appendix A1 pp46-48.

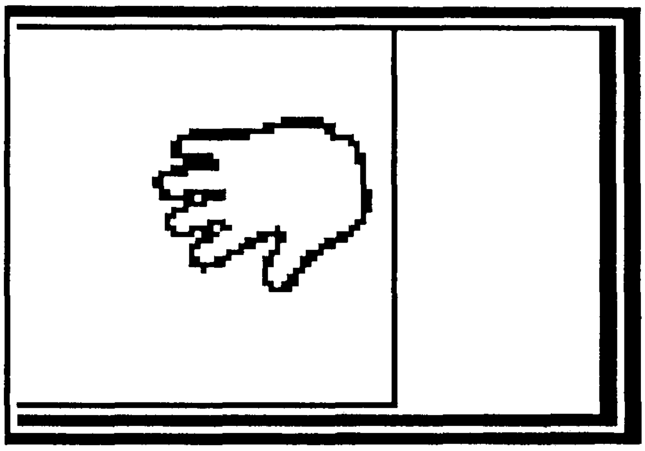


Figure 3.4 The results obtained from using Robert's operators on an Image.

Another Edge detection technique which preserves gradient information is the Sobel Operator [Ballard , Brown 1982], This will be described in the section below.

3.4.3.2 Sobel Operator:

The Sobel Operator is an edge detection template that also provides information on the direction of the edge. The Templates of the Sobel operator is shown in figure 3.5. For a pixel (i,j) The Sobel Operator generates:

$$\Delta_1 = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \Delta_2 = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Figure 3.5 Templates used for Sobel Operators.

$$\Delta_1 = f_{i+1,j-1} - f_{i-1,j-1} + 2(f_{i+1,j} - f_{i-1,j}) + f_{i+1,j+1} - f_{i-1,j+1} \qquad ...(3.5)$$

$$\Delta_2 = f_{i+1,j+1} - f_{i-1,j-1} + 2(f_{i,j+1} - f_{i,j-1}) + f_{i+1,j+1} - f_{i+1,j-1} \qquad ...(3.6)$$

The gradient magnitude is calculated in the same way as the Robert's operator (Eqn. 3.4). In this case an approximation for the gradient direction in the neighbourhood of pixel (i, j) can be calculated:

$$\theta_{i,j} = \tan^{-1}\left(\frac{\Delta_2}{\Delta_1}\right) \qquad ...(3.7)$$

Figure 3.6 shows an example of applying Sobel transformation to an image. A Borland™ C++ program listing of the Sobel transform is given in the appendix A1 pp62-64.The Intensity of the Sobel edge gives an indication of the direction of the edge. The advantage of the Sobel technique over the Robert's method is that it provides a method of generating a parametric representation of the lines of edges. This will be discussed in the section below.

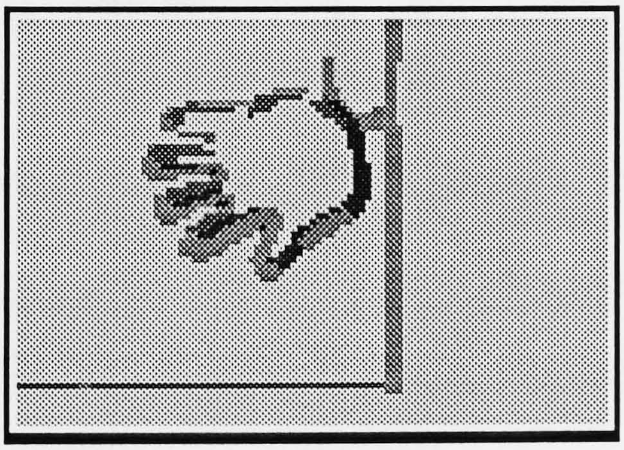


Figure 3.6 Result of using Sobel transformation on an Image.

3.4.4 Segmentation Techniques:

The Sobel Operator described above forms the basis of an image segmentation processing technique. The aim of segmentation is to split the image into regions within an image [Boyle, Thomas 1988]. The identification of distinct surfaces simplifies the task of object identification and further processing.

3.4.4.1 Extracting Line Equations from Sobel:

Once the Sobel Operators have been applied to the image, the parametric equations of the edges can be extracted. Each pixel on the Sobel Image data contains two important pieces of information. Firstly, its spatial location indicates the position of the edge (i, j) and secondly, its intensity represents the angle of the edge (θ) (section 3.4.3.2) [Lowe 1982]. It is possible to translate this representation to a θ, σ space where θ represents the angle and σ represents the distance of a line that intersects the point (i, j) and has a gradient θ from the corner (figure 3.7). σ is calculated as:

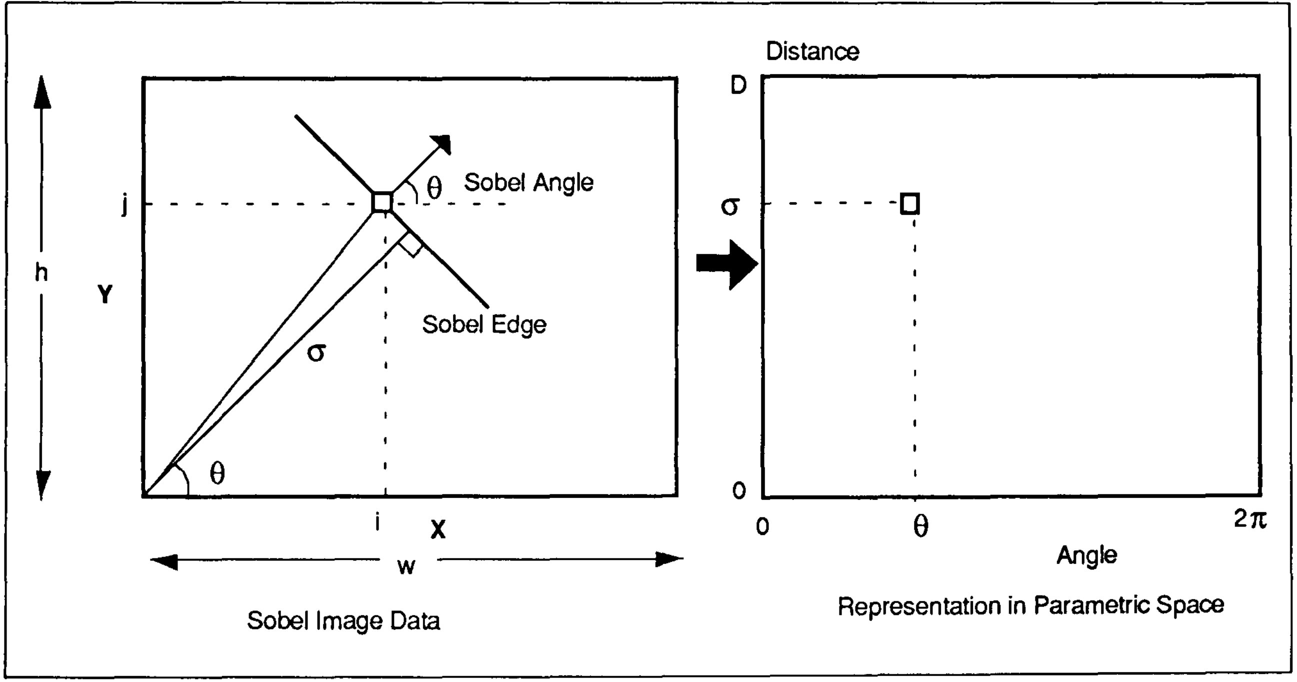


Figure 3.7 Diagram showing the transformation technique of converting Sobel image data to a parametric space.

$$\sigma = \sqrt{i^2 + j^2}\cos\left[\tan^{-1}\left(\frac{j}{i}\right) - \theta\right] \qquad ...(3.8)$$

This equation allows the definition of a new space for the mapping of image edges obtained from the Sobel Operation (section 3.4.3.2). The range of θ is between 0 and 2π and the range of σ is from 0 to $\sqrt{w^2 \times h^2}$ where w and h represent the width and height of the image array. An incremental array can be used for storing the θ, σ Sobel data. As each Sobel point is processed the generated location in the θ,σ array is incremented. The result of this processing is a collection of points about specific regions in the θ,σ space that represent the line locations on the original image [Ballard, Brown 1982]. Usually there are a spread of points in a region caused by:

- Approximation of edge gradient using Sobel Masks.
- Quantisation noise in the equation to array transformation.
- Image noise creating an uneven edge.

The determination of the actual image edge equation has to take into account this spread. Simple thresholding is not the solution for such a problem because images of varying edge complexity and size are not suitably accounted for. A technique of selection and elimination is usually adopted where the maximum array element is accepted and then it and all neighbours are eliminated as being associated to the maximum value [Shanmogam 1979]. This process is repeated for the next maximum array point and is iterated until no further points are left to process. This technique is suitable for system in which the object is of regular shape with edges that produce distinct locations in the θ,σ space. Once unique edge points are located the image edge equations can be regenerated using equation:

$$y = \tan(90+\theta)x + \frac{d}{\sin\theta} \qquad \text{...(3.9)}$$

The actual edge data can be reconstructed using this parametric representation by replotting on an incremental array. A Borland™ C++ program listing of this segmetation technique is given in the appendix A1 pp56-59.

The next step is to determine possible corners of the objects, by considering the points of intersection between the different lines. The next section describes a number of corner detection algorithms.

3.4.4.2 Connected Components:

Connected components is a technique for grouping different parts of an image into distinct objects [Boyle, Thomas 1988]. The technique considers each pixel in the image and determines if it is related to its neighbour. A pixel is said to be related to its neighbour if the difference between them is within a defined threshold. One of the many algorithms for identifying connected components is Region Growing [Boyle, Thomas 1988], which is a simple and efficient grouping method.

The region growing algorithm starts by considering a single pixel F and allocating it a unique value. It then checks the neighbouring pixel N and determines if it is associated with F. It does this by defining a threshold level T and checking if N falls within the threshold bound: If $N < F+T$ and $N > F - T$ then N is a part of F. N is then assigned the same value as F. This is done for all the Neighbours of F. Once completed F is marked as visited and the process is repeated for its neighbouring pixels. This algorithm creates the effect of a growing system and hence the name "Region Growing." After one region is tagged (assigned a unique value) , another unvisited pixel is considered and the process repeated until all the image is completely tagged. Provided edge boundaries are well defined, region growing can cope with intensity differences over the object and maintain grouping integrity. This is true because the acceptance of a neighbouring pixel is determined by the value of the pixel itself and not by a global threshold level as in Robert's and Sobel Operators. One other advantage of region growing is that it provides an accurate presentation of areas of object. Figure 5.8 shows the result obtained from using region growing on a sample image. A Borland™ C++ program listing of the region growing method used in this work is given in the appendix A1 pp42-46.

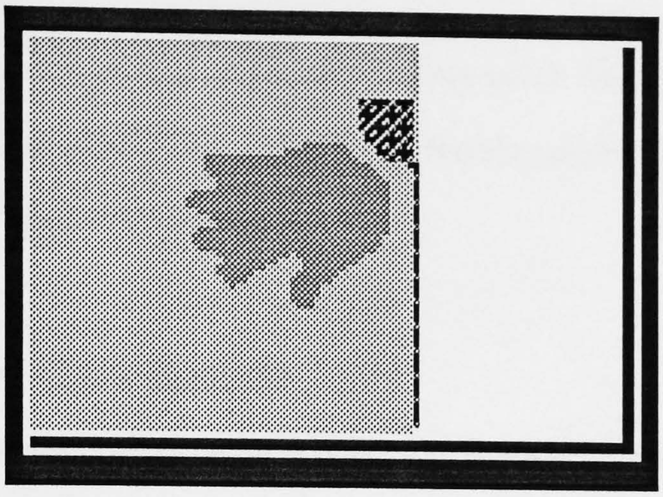


Figure 5.8. Result of region growing on a sample image.

3.4.5 Corner Identification:

Corner Identification is the next step in image location [Niblack 1986]. Thus far, an image has been reduced and deprived of colour information(grey scaling), the edges extracted (Robert's and Sobel Operations) and then edge lines generated from these data (Sobel). The image was grouped into different regions and tagged uniquely. Next the information is reduced down to locations of the corners of the object. Once this is accomplished objects will have been suitably parameterised, providing information to allow object location and tracking.

A number of different approaches for corner identification have been developed and these are described below.

3.4.5.1 Corners from Sobel Operators

Corners can be readily obtained from Sobel images (Section 3.4.3.2) by considering the regions where a significant change in the angle θ, of a Sobel edge occurs (see figure 3.6). A method that can be implemented is one that considers the Sobel angle $\theta_{i,j}$ with its neighbours: $\theta_{i+1,j}$, and $\theta_{i,j-1}$ [Ballard, Brown 1982]. The algorithm implemented can test if its neighbours are also edges and if they have a significant difference in their angles. A threshold value can be selected for this difference above which the point i,j is attributed as an edge. A Borland™ C++ program listing of the metod of extracting corners from Sobel operator is given in the appendix A1 pp56-59.

3.4.5.2 Corners from Line Equations.

Corners from line equations (section 3.4.4.1) are generated by identifying the intersection of lines with other lines (see figure 3.8) Because the line reconstruction array is incremental (section 3.4.4.1) , corners are identified by considering points that have a value greater than 1. The disadvantage of this technique is that it also locates a number of points that are not corners, as they are intersection of lines identified from other objects in the image. However, it does positively identify those corners indicated by the Sobel method (section 3.4.5.1) and can be used to validate these points.

3.4.5.3 Corners from Connected Components.

Corners can be obtained from connected components (section 3.4.4.2) by considering the locations of the extremities of the object. One way of achieving this is to consider an object and locate the first and last elements of that object by scanning in two orthogonal directions (see figure 3.9). The disadvantage of this technique is that it can not perform well with complex or curved objects. However, it performs well with locating edges of simple quadrilateral objects. Again this information can be utilised in conjunction with other techniques of corner identification to validate edges of quadrilateral shapes. A Borland™ C++ program listing of the method used for extracting corners from connected components is given in the appendix A1 pp32-35.

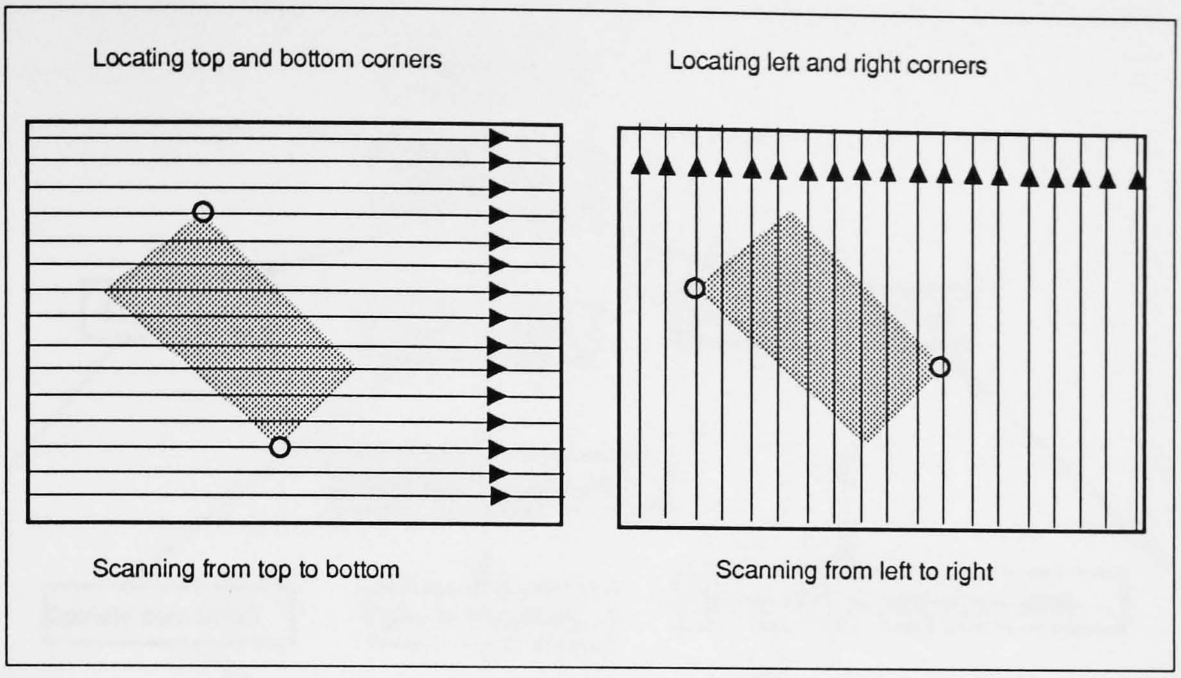

Locating top and bottom corners
Locating left and right corners
Scanning from top to bottom
Scanning from left to right

Figure 3.9 Diagram showing method of extracting corners from connected components.

3.4.6 Object Identification and Selection:

Object identification and selection is the final stage of an image processing sequence. Once different methods have extracted important features from the image, the data must be used in a meaningful manner to locate the specific object and pass the required parameters for selection. The normal method of selection is to define a window in the image and specify that the object is bound by this window [Boyle Thomas 1988]. Selection is performed by first defining those features which describe an object. These features may include segmentation information (providing region areas), boundary information (describing bound regions), shape information (specifying the location of specific shapes in the image) and masking techniques (to identify known object shapes) [Weymouth 1986]. Usually a number of different routes of image identification are necessary to increase the confidence in a given selection. Also if a number of candidate objects are identified by a particular image processing technique then different techniques may identify a common candidate as the object required. The diagram below (figure 3.10) shows a possible route that an object identification process may follow.

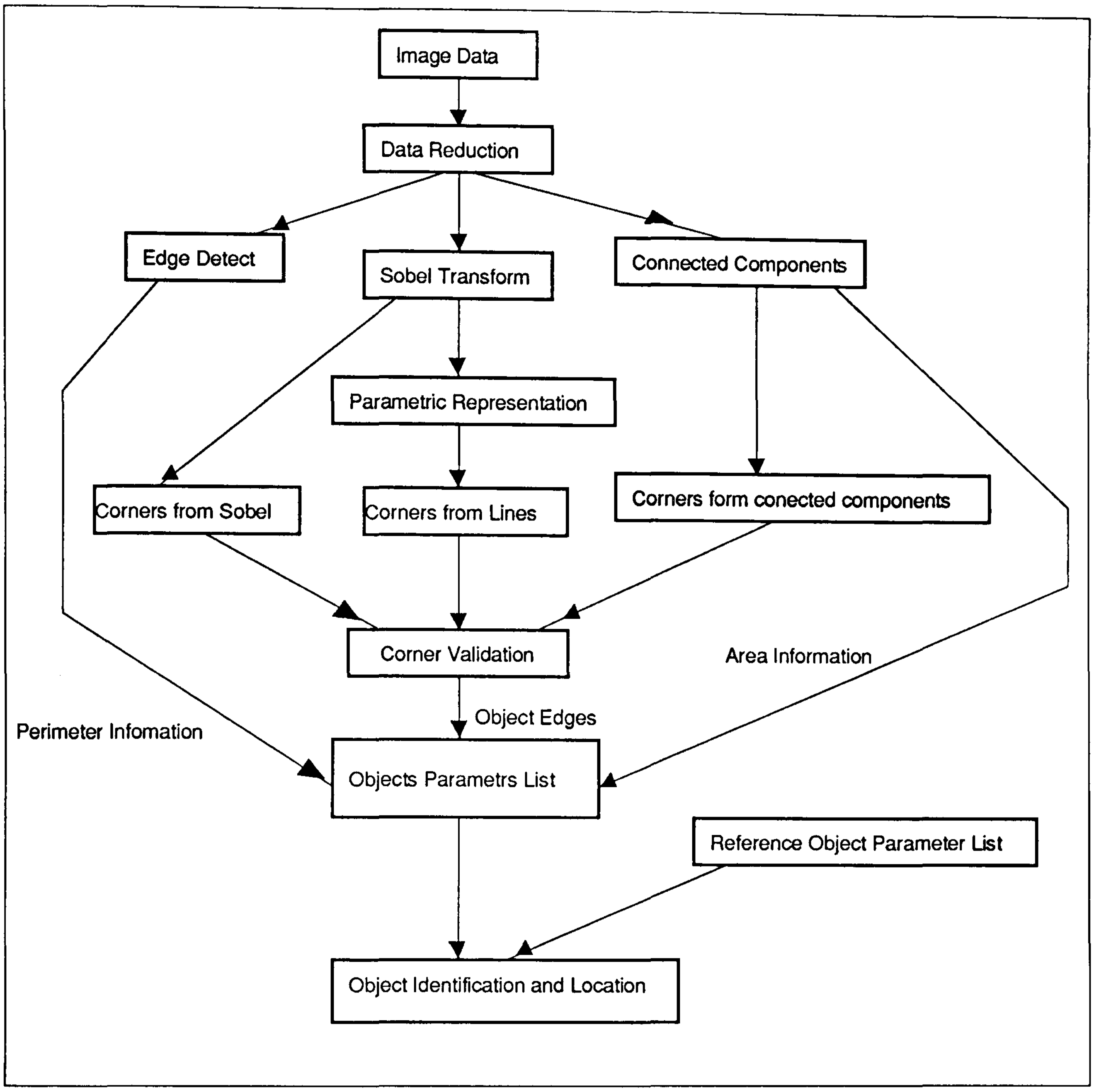


Figure 3.10. Diagram showing the process flow for an object identification and location process.

3.4.7 Object Confidence:

Once an object has been selected it is important to identify how confident the system is in the identification. The confidence figure is a ratio of the detected candidate object attributes and the known parameters of the object. If the object confidence is below a threshold value then the system will not make a positive identification of the candidate object and the object identification process is repeated. If this confidence figure is above the threshold value then the object will be said to be positively identified. Another possibility would be if a number of candidate objects were identified in the field of view. The system would then select the one with the highest confidence figure. It is important that during the image tracking and the data analysis operations the object confidence be provided.

3.4.8 Image Tracking:

Once the object has been successfully located it has to be tracked to detect small movements of the object in the field of view. As the object is already located in the image frame the tracking algorithm need only concentrate on the smaller image area. The resulting algorithm should, therefore, be fast enough to track the object in real time. Unlike the image processing stage, which is only used a few times in a process, the image tracking algorithm needs to be performed after every data processing cycle. Thus the image tracking algorithm is identified as a potential bottleneck in the process. An extensive effort in the production of a fast and efficient algorithm was considered to be necessary. The algorithm chosen for this work was a repeated centre approximation system [Green 1983], which locates the image boundary and then redefines the centre point. This iterative process works well with the simple object's design used here but also with more complex objects as is shown in figure 3.11. A Borland™ C++ program listing of the tracking algorithm used is given in the appendix A1 pp37-42.

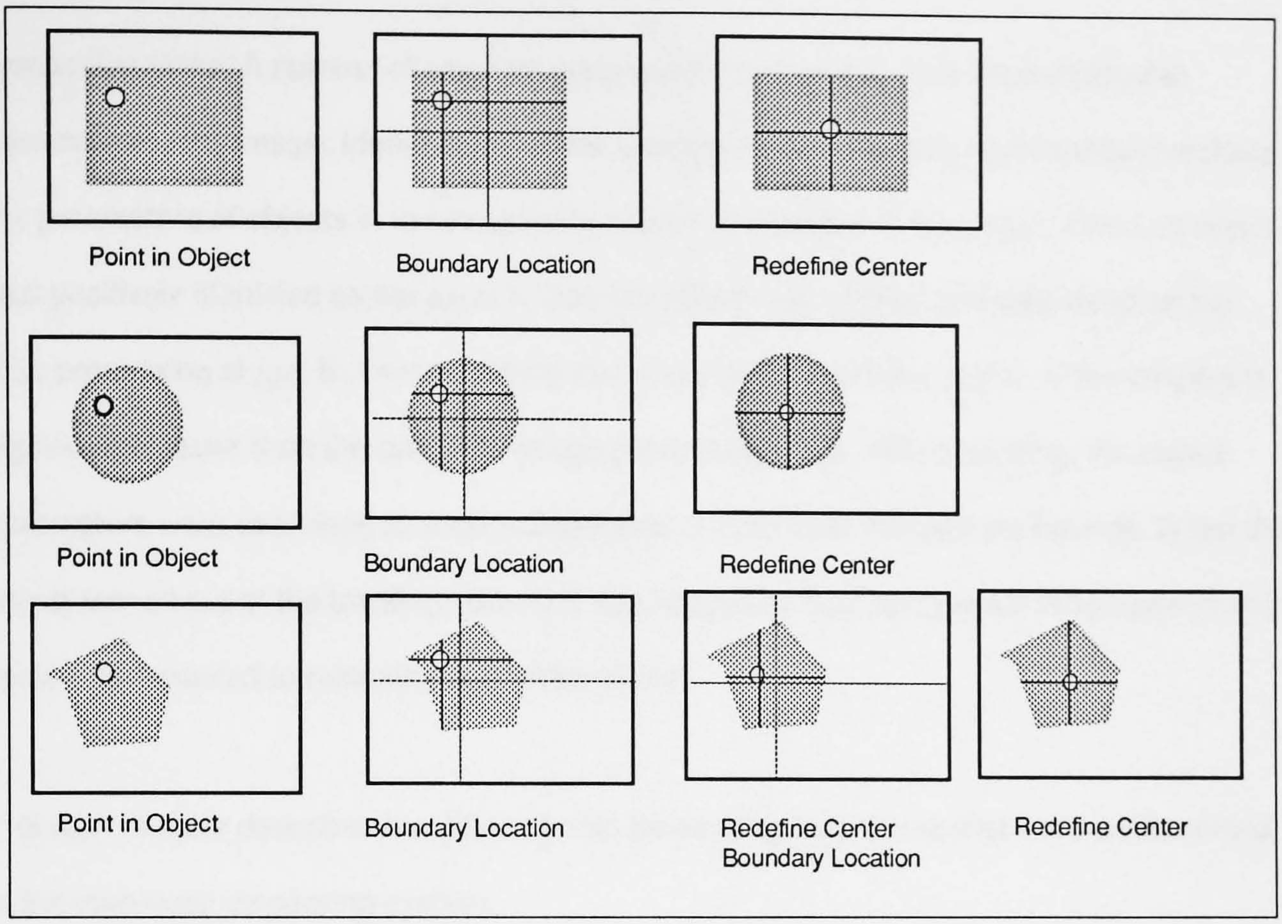


Figure 3.11 Diagram showing how centre approximation method can track objects.

3.4.9 Object lost:

During the process of tracking, the image may move rapidly enough for it to fall outside the tracking range. If this occurs then the object is said to be lost and the system has to restart the process of object identification to relocate it. One of two criteria must be fulfilled for the system to detect that the object is lost from tracking bound:

1) The change in this size of object boundary is greater than a threshold value,

2) The colour of the centre has changed by above a threshold value.

Once the object is lost then the process must return to the object identification and location stage to again find the object.

3.5 Summary

This chapter introduced the techniques that needed to be implemented to perform image processing tasks in the remote monitoring system. The image was first stripped of colour information and reduced to increase the speed and reduce the complexity of the image processing tasks. A number of small image processing modules were implemented to parameterize the image. Identification of the sensing region was accomplished by matching the parameters of objects in the image with known parameters of the object. Once an object was positively identified as the sensor, then the object was tracked and monitored for the data processing stage. Because tracking concentrates on a smaller region of the image it is significantly faster than the complete image processing stage. While tracking, the object parameters were examined to ensure that it was not lost from the tracking bounds. When the object moved out of the tracking bounds it was tagged as 'lost' and the full image processing cycle was repeated to relocate the desired object.

The next chapter describes the different data processing techniques that were implemented in the chromatic monitoring system.

CHAPTER 4

Data Processing

4.1 Introduction.

Data processing involves the analysis and interpretation of information in a meaningful way. It is perhaps the most demanding part of a process as it directly relates to the accuracy and performance of the system as a whole. This is especially true for remote sensing applications as the system needs to be more resilient to external parameters that corrupt the received data. Like image processing, linear data processing is a well established science that can be readily applied [Grand 1968]. Linear data processing can be considered in a modular fashion where blocks of processing can be connected together to achieve the desired goal [Benjamin 1976]. However, the solutions of non-linear problems are not easy to implement using standard linear signal processing techniques unless piecewise linear approximations can be used. In situations where chromatic processing is required it is often difficult to form linearly separable components. This is due to the complex variations in the chromaticity of an object and the measurand. Also, the existence of a number of other external phenomena, such as glare and lighting level fluctuations makes it difficult to implement methods that employ linear data processing alone. There is therefore a need for non-linear processing to be used. Such techniques include look-up tables, algorithmic techniques and neural networks. Look-up tables, whereby a map is made between the input data and the output data, perhaps form the simplest form of non-linear processing. Algorithmic techniques on the other hand attempt to parameterise the characteristic behaviour of a system using a number of non-linear and linear equations. The implementation of neural networks is also feasible for the solution of processing non-linear data because they are themselves non-linear elements [Minsky 1988].

With regard to neural network option, there exists a range of different types of neural networks each of which is better suited to different forms of data processing needs. Some, Like Binary Array Memory (BAM) are essentially digital in nature and have little application for analogue data processing [Lisboa 1992]. The Error Back Propagation Multi Layered Perceptron (MLP) is a well established network structure and is able to process analogue data [Willis et. al. 1991]. Even then, there are a wide range of variations in the design of this class of networks many of which are still in the experimental stages of their development.

The remainder of this chapter presents a brief overview of the various methods that are available for data processing, section 4.2 deals with the Neural Networking methods, section 4.3 deals with Look up tables and section 4.4 with Algorithmic Techniques.
This chapter presents a brief overview of the different forms of data processing. The following section presents an overview of neural networks as a method of data processing. It also introduces look up table and algorithm techniques for handling non-linear problems.

4.2 Introduction to Neural Network Processing.

Neural networks originated from the study of artificial intelligence [Feldman, Yakimovsky 1974]. The processing method used by neural networks is totally different to conventional computational techniques. Most present day computational techniques rely upon numerical or logical methods of problem solving. However, unlike the human brain, conventional techniques are poor at processing patterns or manipulating convoluted or non-linear data. Neural networks are a simplified model of the neural organisation in the human brain. By their intrinsic architecture they are able to 'learn' a mapping between a set of inputs and output patterns. More interestingly they are able to exhibit features of generalisation and data abstraction that is uncommon in other processing methods.

One practical form of a neural network system is the multi-layered perceptron (MLP) [Minsky et. al. 1988]. This consists of a network of interconnected processing units with input and output cells connected directly with the outside world. When an input is applied, the output is determined by the internal attributes of the system. These attributes may be altered such that a mapping is formed between the input and the output patterns. This is the basis of learning for an artificial neural network. The development of efficient learning algorithms has made neural network technology more attractive in the solutions of problems where normal algorithmic techniques are difficult or impossible. Neural networks have found application in image processing, speech processing, and a vast range of specialised industrial and commercial data processing needs [Willis et. al. 1991][Sutton et. al 1992]. Interest in neural networks has lead to the development of a large number of network designs and ideologies to handle a wide range of processing needs. A discussion of all these various techniques is beyond the scope of this work but the theory of a specific neural network is presented since it was primarily used in the present work.

4.2.1 Theory of the Multi Layered Perceptron (MLP).

The MLP consists of a matrix of interconnected layers of identical processing elements called nodes or neurones. The layer arrangement consists of input, output and internal, or hidden layers, which allow connectivity between the input and the output (figure 4.1). The convention used in referring to a network with a specific topology follows that adopted by Bremmerman and Anderson [Anderson 1988]. For example, a MLP with three input nodes, two hidden layers with twenty and ten nodes respectively, and one output node is referred to as a (3,20,10,1) network.

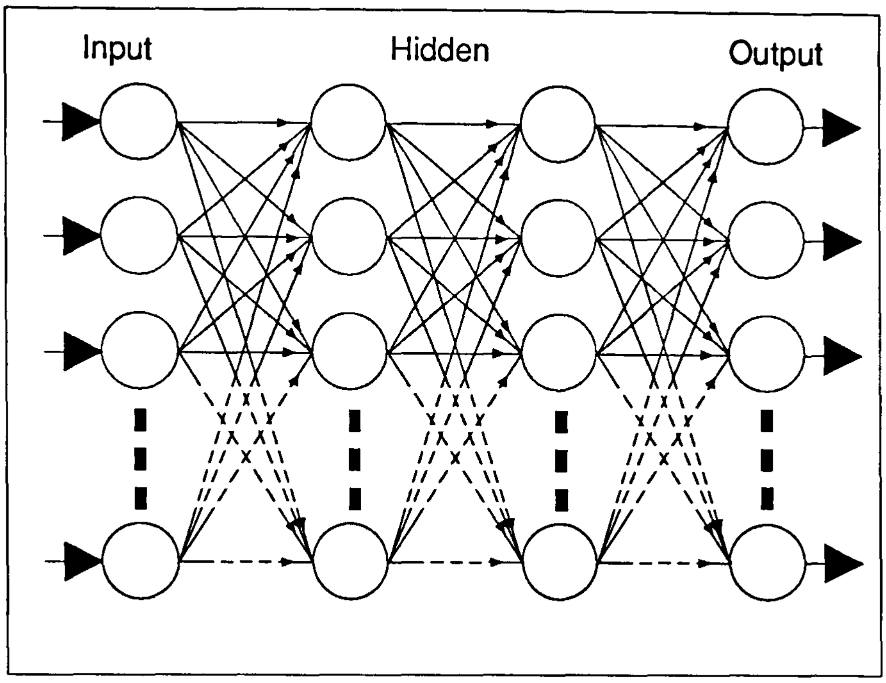


Figure 4.1. Schematic of a Multi Layered Perceptron

The input data is presented to the input nodes of the network and the information propagates through the network to the output. The number of hidden layers used is arbitrary but it has been shown that two hidden layers are sufficient for most applications.

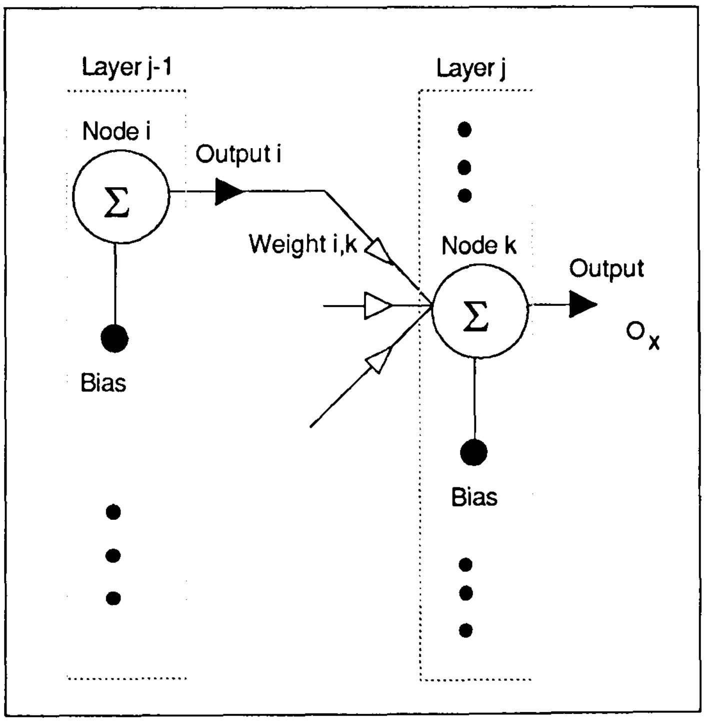


Figure 4.2. Diagram showing relationship between layers, nodes, weights and bias.

Apart from the input nodes (which are only used to provide a method of presenting scaled data to the network), the input to each node is the weighted sum of the outputs from the nodes in the previous layer (figure 4.2). For example, if the information from the ith node in the j-1th layer, to the kth node in the jth layer is $O_{j-1,i}$, then the total input to the kth node in the jth layer is given by:

$$NET_{j,k} = B_{j,k} + \sum_{i=1}^{n} W_{j-1,i,k} O_{j-1,i} \qquad ...(4.1)$$

where B is a bias term associated with each node and W is the weight associated with the node connection i to k . The output of each node is obtained by passing the weighted sum NET through a non-linear operator. This is typically a sigmoid function, the simplest of which has the mathematical description:

$$O_x = \frac{1}{1+e^{-NET_{j,k}}} \qquad ...(4.2)$$

The response characteristic is shown in figure 4.3. It is noteworthy that the sigmoidal non linearity has been observed in human neurone behaviour [RummelHart, McClelland 1986]. This sigmoid function allows the mapping of non linear functions. The output of each node is propagated through the network, working from the input layer to the output layer. The results of the output node are related to the weights and biases of the network and the values applied to the input nodes. Learning is achieved by adjusting the weights and biases such that an input set gives the correct output. One of the most common learning algorithm was originally proposed by Rummelhart [Rummelhart et al. 1986] and is referred to as the error back propagation algorithm. This is described below.

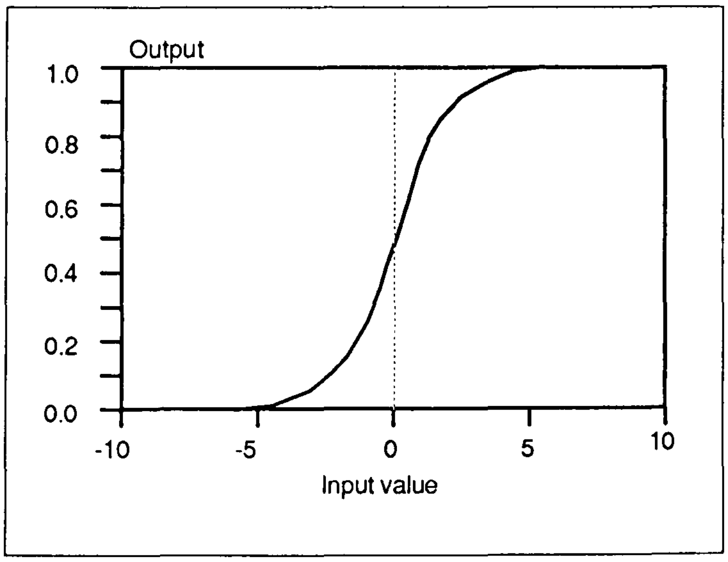


Figure 4.3. Sigmoid function relationship between input and output of a node.

4.2.2. Back Propagation Algorithm for MLPs

During learning the output of the system is observed for an input set. This is referred to as the forward pass. During this stage the outputs of all nodes in the system are stored. As the title suggests, the error between the output and a target output for that particular input is propagated back through the network to minimise this error. The error, E_i for a node in the output layer, between the target value for that node T_i and the output of the node, O_i is:

$$E_i=T_i-O_i \qquad ...(4.3)$$

This error is propagated back to adjust the weights accordingly. The change in the weight between the output and the hidden layer below,$\Delta OW_{i,j}$ is given by the equation:

$$\Delta OW_{i,j} = \eta \times E_i \times O_i(1-O_i) \times O_j + \alpha \times \Delta LOW_{i,j} \qquad ...(4.4)$$

where O_j is the output of the hidden layer nodes, α and η are acceleration and learning rate constants respectively. These parameters determine the rate of learning and the ability of the network to locate global minima and avoid points of local minima that may exist for a given data set. This can be visualised as a ball (the error vector) moving over an area that has peaks and troughs (the weight space). The learning rate is then visualised as the smoothness of the surface and acceleration as gravity. $\Delta LOW_{i,j}$ represents the last weight change for that particular node. The bias associated with that node is also changed using the equation:

$$\Delta OB_i = \eta \times E_i \times O_i(1-O_i) + \alpha \times \Delta LOB_{i,j} \qquad ...(4.5)$$

where ΔLOBi is the last bias change value.

Weight and bias changes for subsequent layers are calculated by:

$$\Delta HW_{j,k} = \eta \times \sum_i E_i O_i(1-O_i) \times O_k \times O_j(1-O_j) + \alpha \times \Delta LHW_{j,k} \qquad ...(4.6)$$

$$\Delta HB_j = \eta \times \sum_i E_i O_i(1-O_i) \times O_j \times (1-O_j) + \alpha \times \Delta LHB_j \qquad ...(4.7)$$

where $\Delta LHW_{j,k}$ and ΔLHB_j are the last weight change and bias changes for the connections concerned and B_j is the bias value at that node. All other symbols have their usual meaning.

Using the above process it is possible to describe the changes in weights and biases for a one input-output data set pair. This process is repeated for all input-output data sets and the weight and bias changes are stored cumulatively. Once all inputs and outputs have been processed the cumulative change is implemented through the system. This completes one training cycle or iteration. An indication of how close the target and the output compare is called the costing, υ. This is the Euclidean distance between the two sets, and is expressed as:

$$\upsilon = \frac{1}{2}\sum_{m=0}^{m}\sum_{i=0}^{i} E_i^2 \qquad \text{...(4.8)}$$

where m represents the number of input output data sets and i is the number of output nodes. The iterations are continued to reduce this costing to a level specifying the desired accuracy between the output and the target.

The information learned by the network is stored as a set of weights and biases for the network. The capacity of information storage depends on the complexity of the input data, the network size and topology. The theoretical determination of the best network architecture is still not well understood and usually, the best approach is experimental trial of various network topologies, selecting the best system in terms of learning rate, accuracy and resolution. However, recent work [Lisboa1992] has indicated that it is possible to develop procedures which aid topology specification.

4.3 Look Up Table Selection (LUT)

Look up tables provide a simple method of mapping between inputs and the desired output. The dimensions of the look up table are determined by the number of inputs that are necessary for the calculations [Kershaw 1991]. Figure 4.4 shows how a LUT can be used to relate a two input system with a single output. The LUT holds the result space for all possible values of inputs in an array. The inputs presented to the LUT are then converted to addresses to interrogate the LUT array. The problems associated with the technique are that the accuracy and resolution of the system are limited by the size of the array. The size of the array also increases as a square of the number of inputs required. However, techniques of data interpolation and nearest neighbour determination may be used as post and pre-processing methods to control the array size and the resolution.

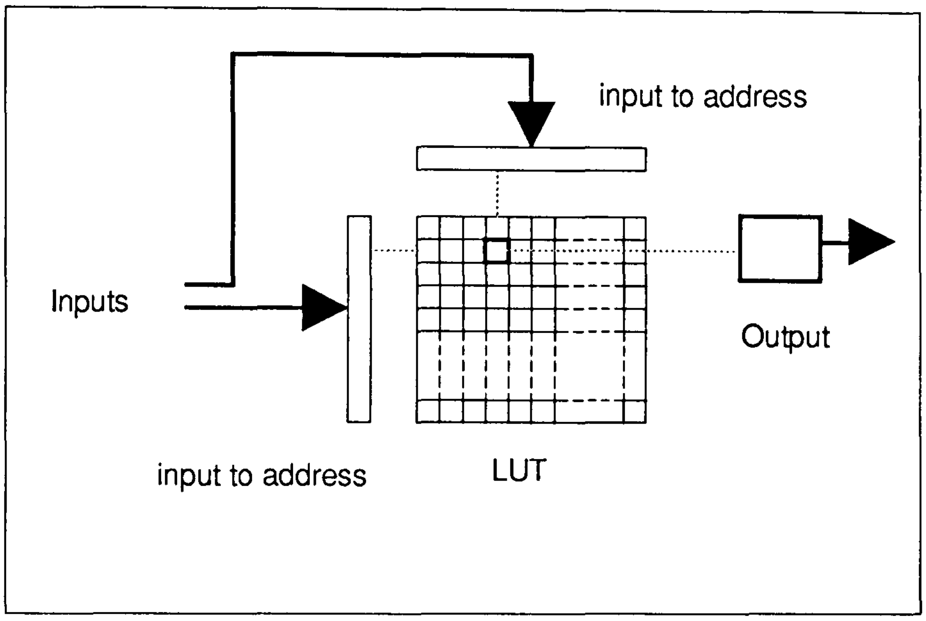


Figure 4.4. Diagram showing method of using LUTs for mapping inputs with output.

Nearest neighbour identification identifies the data in the LUT that has the least error compared with the input data. Interpolation is an extension of nearest neighbour location process where two neighbours are identified and the input data lies between them. Interpolation between the two neighbouring points by a ratio of their errors with respect to the input data and adjusting the output in a similar way can provide a better approximation of the result.

4.4 Algorithmic Techniques.

Algorithmic techniques rely upon the formulation of a set of routines to relate the input to the output [Murphy 1991]. The development of suitable algorithms relies on the identification of the relationship between the input and the desired result. In some applications this may require the representation of the input into high order parametric forms. The algorithmic approach works best with linear data processing problems and with functions that may be reducible to a linear form. Algorithmic methods are poor at handling complex and convoluted data that are non-monotonic in nature and with a high degree of degeneracy. Unfortunately the processing of data that does not have a clear relationship with the output is difficult, and, possible solutions may require the implementation of some of the other techniques mentioned above.

4.4 Conclusions.

Each of the three different techniques mentioned above, namely algorithmic, LUT and neural network approaches have a number of different advantages and disadvantages. In the implementation of these techniques consideration must be given to how these limitations may affect system performances.

The advantage offered by algorithmic techniques is that they provide an accurate mapping between the input and output data. The accuracy is limited by how accurately the data is parameterised and how well the system has been modelled. However, the disadvantages of this technique are the complexity involved in considering all the system parameters that may affect the data. These techniques also provide poor results when the input data is slightly corrupted and may even produce nonsensical results in the worse case. Finally they tend to be difficult to update and modify when new parameters are considered.

LUTs are more flexible than algorithmic techniques and are by far the simplest technique to implement. With the aid of interpolation and nearest neighbour techniques they have the scope of dealing with corrupted data. However they perform well only with a small number of inputs as increasing the number of inputs increases the dimensionality of the LUT and thus the LUT size. Even though LUT size can be reduced by pre and post-processing of data they often tend to be large especially if a large number of inputs are used. The size of the LUT often determines the resolution and accuracy of the system.

Neural networks are capable of processing non-linear data because they are constructed using a network of non-linear components that are adjusted iteratively. They can model non-linear systems well and can learn to any precision value required (dependent on the network topology). They can be adapted to account for new conditions not considered at the initial learning stage. They also have a limited scope for generalisation (or guessing) with corrupted data [Shepherd, Krough]. However, learning is usually slow and dependent on computational resources. Also the accuracy of the system with corrupted data can not be defined but only checked by experimentation. Perhaps the reason for most reluctance in the acceptance of this technology is the lack of understanding of the nature of neural networks and proper rigorous methodologies in the design of network topologies [Denker 1987].

CHAPTER 5
The Experimental Monitoring System

5.1. Introduction

5.1.1. System Requirements

A major aim of this project was to develop a system to remotely monitor the chromaticity of a sensing element and to relate this chromaticity to the corresponding measurand. The sensor to be monitored could be one that would be responsive to one of the several measurands. The system should be able to operate in a real world situation and therefore be able to tolerate the spurious effects of external parameters that might affect the chromaticity of the sensor. These parameters would include glare, lighting level changes and shadowing. Another important requirement was that the system should be capable of performing the task of acquiring the image of the sensor and process it in a realistic time period. The system should be general in functionality and be flexible enough to be easily tailored to monitor a specific sensor or a number of different sensors if necessary simultaneously.

This chapter presents detailed information about the system that has been developed in the monitoring phase and discusses those parameters that affect the system performance.

5.2. Overview of the Monitoring System

The monitoring system employed can be separated into two areas. The first area includes the hardware, comprising of sensors, light sources and image capturing equipment that presents data in an interpretable form. The second area involves the software that performs the interpretation of the image data. Both of these combined provide the basis of the remote monitoring system and will be discussed separately.

The hardware module consists of the camera, the frame grabber card and the computer. These were introduced in section 2.3.2 as an analogue to the human vision system. The Camera is responsible for acquiring the image and presenting it in a suitable electrical form for the Frame Grabber Card (FGC). The FGC is a card that resides inside the computer and is used in essence to form the interface between the camera and the computer. The computer used was a PC workstation and had the task of providing a platform for the software and a control source for the FGC.

The software to accompany this hardware was specially developed for this project work. It, together with the computer, determines the speed, resolution and limitations of the system. A number of different software systems were developed and a general overview of the specifications is presented.

The next section presents a detailed explanation of the hardware used in the project.

5.3. Hardware Specification

The monitoring system hardware consisted of a colour CCD camera, the frame grabber card and a PC workstation. Each of these modules can be purchased off the shelf and matched to handle general or specific tasks. The Sections below describe the computer vision system that was used for the general remote chromatic monitoring systems.

5.3.1. The CCD Camera.

The Camera used was an iEC800CC which is a commercially available colour camera using a 768x512 array charge coupled device (CCD) detector [iEC800 man. 1989]. This has already been introduced in section 3.4.1. The CCD array consists of an array of optically sensitive field-effect transistors (FET's) with their drain and sources connected in series and the drain capacitively coupled to the gate in the arrangement of a digital shift register (figure 5.1). The incident light causes a charge to accumulate in the capacitors associated with the FETs. This charge represents the intensity received by a particular CCD pixel. By applying a clocking pulse to the bases as shown in the diagram, the charges in the capacitors are made to ripple through the FETs and are collected at the signal output node for amplification and transmission. In the camera used for this project, the CCD array element contains three arrays of 768 x 512 FETs arranged in this fashion, where each array addresses different overlapping wavelength ranges of the visible spectrum and so provides data streams for the three colour channels. CCD arrays can operate at high frequencies (up to 10 MHz) [Kaplan 1992] and technology has advanced significantly to provide a reliable and cost effective method of obtaining optical images.

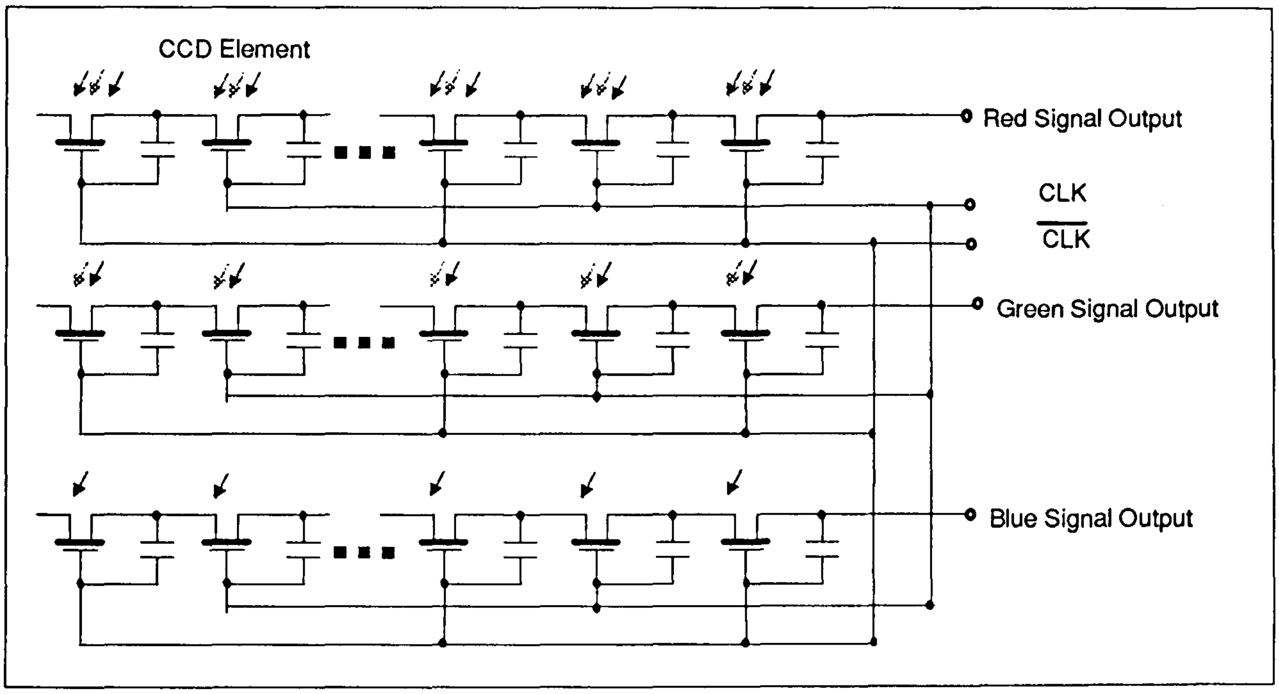


Figure 5.1. Diagram showing arrangement of elementary components in a CCD array.

Attached to the camera is a removable lens with focus and aperture control. The focal range for the lens is between 1m and infinity. The aperture controls the incident image intensity onto the CCD array and allows control of the aperture from fully open to fully closed. Other features of the camera include Gamma Correction options and video output signal.

Gamma correction is a non-linear function applied to the R, G, B detector responses. The gamma correction is similar in action to the pupil of the human eye. Its non-linear function provides high amplification for low intensity light and reduced gain for high intensity light. The advantage of this technique is that it provides a larger intensity bandwidth for a given aperture setting.

Gamma correction is a facility available on most CCD cameras and is applied to all detector inputs. The colour calculations however, are still intensity independent. Gamma correction does affect the chromatic values obtained from equations 2.5 and 2.6 in that different values of x and y will be obtained when gamma compensation is activated and when it is not.

A significant part of the camera electronics is dedicated to converting the electrical information into a form suitable for transmission to external devices such as a monitor or a frame grabber card. The output of the camera follows a standard called NTSC (National Television Standards Commission), containing separate channels for red green and blue information and a sync pulse for picture synchronisation to an external device.

The analogue video signal contains both video and timing information. The timing information is transmitted between each horizontal line scan. The NTSC defines an "interlaced raster scanning" technique that eliminates flicker when the image is displayed. The horizontal scan lines that make up a frame are divided into two fields [Sproson 1983]. The even fields consist of all the even numbered scan lines in the frame, The odd fields contain all the odd-numbered scan lines in a frame. Interlaced scanning results in the display of one line at a time. First all the even fields of a line are displayed and then the odd-numbered fields are displayed. Interlacing takes advantage of the human eye's characteristic of persistence of vision to eliminate flicker [Sproson 1983]. This sequence repeats continuously to provide the raster image.

5.3.2 The Frame Grabber Card.

The frame store used in this work was an Imaging frame grabber card. It provided the interface between the colour camera and the computer. The input to the FGC is the RGB sync signal from the camera and control information from the computer. The frame grabber card is responsible for digitising the image data and some software control over this digitisation is available. The image is stored in the Frame grabbers internal memory. This allows images to be captured and stored for processing. The frame grabber takes 50 frames per second and provides direct access to frame data during acquisition.

A detailed description of all the advanced features of the FGC is not necessary for the work presented in this thesis, but a closer look at the input stage of the FGC is necessary and is presented in detail in the section below.

5.3.2.1 Input ADC stage for the FGC.

The input ADC stage is responsible for converting the analogue camera data to digital data for storage. Three identical stages exist, one for each of the red, green and blue channels. They work in parallel to convert the analogue camera frame data into a digital form for storage in the frame memory. The block diagram of the input interface of the FGC is shown in figure 5.2.

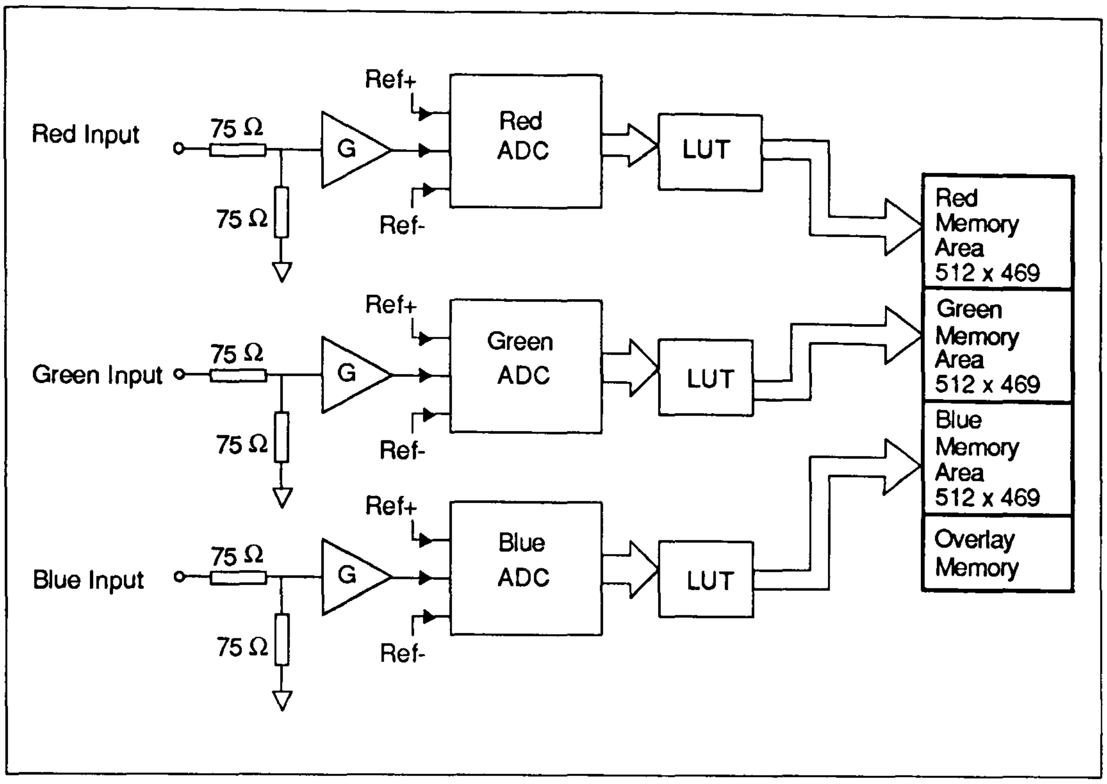


Figure 5.2 Block diagram of the input stage of the frame grabber card.

The ADC employs an 8 bit flash converter for fast digitisation of the video signal. In this method the input signal voltage is applied to one input of each of 256 comparators with the other end connected to 256 equally spaced reference voltages. A priority encoder generates a digital output corresponding to the highest comparator activated by the comparator. A ladder network is utilised (figure 5.3) to provide the reference voltages for the comparisons. This is the fastest technique for analogue to digital conversion [Howritsz, Hill 1990].

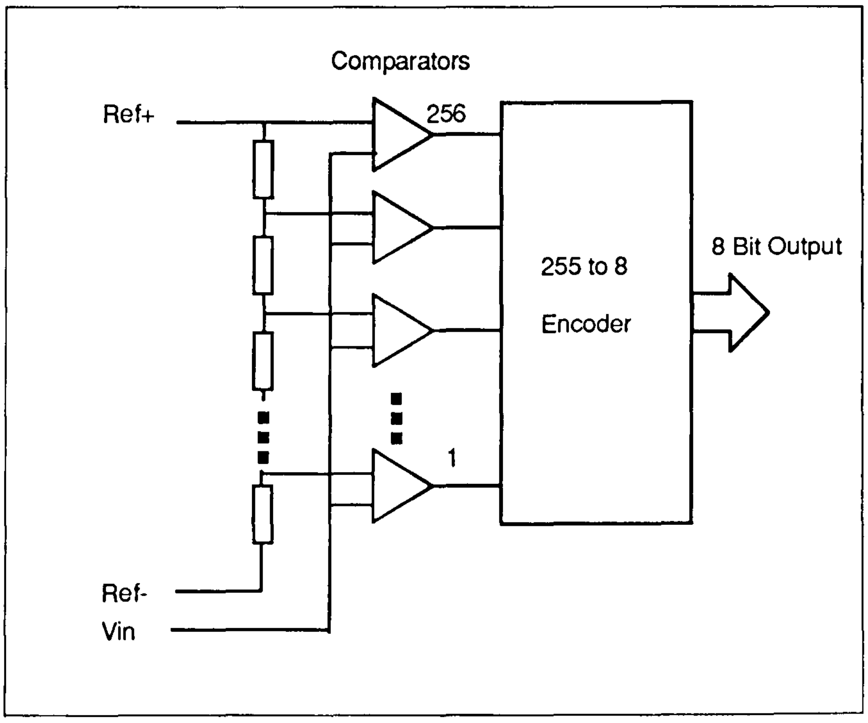


Figure 5.3. Diagram showing ladder network used in a fast ADC circuit.

The zero and full scale levels of each of the ADCs are programmable. This feature allows increased flexibility when digitising video sources; smaller sections of the video signals can be digitised over the full grey-scale range of 256 bits. Control of the ADC range also provides methods of hardware compensation of colour.

The offset voltage (REF -)level is adjustable from 0 to 1.47 Volts in 64 steps (23 mV/step). The full-scale (REF +) can be adjusted from 0 to 1.97 Volts also in 64 steps (31mV/step). The factory default settings for these parameters are 0V and 1.7V for the REF- and REF+ respectively. The input voltage range is the difference between the two reference voltages: REF+ - REF-. These values set the positive and negative values of the ladder network in the input ADC (figure 5.3). Output data from the ADC is corrupted if the input signal presented is outside this range.

The digitised ADC outputs are passed through Input Look Up Tables (LUT) which convert the ADC output to a new form. For normal processing the LUT is programmed as a ramp which provides no translation of the data. As well as the input LUTs there are output LUTs. These may be used to perform simple transformations of the image data, such as, inverting the image, providing a logarithmic output, etc. However, these features were not used in this project.

5.3.3 The Light Source.

The Light source is an important part of a chromatic system. It affects the colour that is observed from the image and thus the parameters being measured. Ideally the light source should be white. The formal definition of a white light source is one which produces an equal intensity for all wavelengths [Hect Zajac, 1974]. Practical white light sources vary significantly from this ideal characteristic primarily because of the nature by which the light is produced. The most common artificial light source is the tungsten filament lamp. Tungsten filament lamps are widely used because they are inexpensive and compact. The usefulness of this form of lighting, specifically for chromaticity measurement is that the spectral power distribution is entirely dependent on the temperature of the filament and correlates well with black body radiators as defined by Planck's law [Henderson 1989]. However there are problems of short life and variability of output with age with such simple filament lamps. Improved life and performance can be achieved with tungsten halogen lamps.

The tungsten halogen lamp that was used in this work was a THORN M4S lamp with a partially reflecting back-plate driven by a stabilised dc 12V supply. This arrangement was used because of the stability and optical power of the light source. Improved light source designs include temperature sensing for the feedback control of light colour. The tungsten halogen lamp needs to be the primary source of image illumination, mainly because other light sources such as room lighting and fluorescent lightings are driven by the mains ac. supply and show flicker at 50Hz. Together with the response of the camera, this could cause flickering and poor image quality.

The CCD elements in the camera are designed to provide an equal output on each of the colour channels for a Planckian radiator at 3200 K, as defined by CIE standard illuminant A [CIE 1931] which is defined as the spectral power distribution caused by a black body radiator at 3200 Kelvin.

5.3.3.1 Other Light Sources

The tungsten halogen lamp is the most suitable wide band spectral source and is recommended by Jones [Jones et al. 1993] for chromatic based system. Other light sources that may be used include light emitting diodes (LED) and lasers. Unfortunately these sources produce narrow band and monochromatic optical output respectively which are not so easily used for chromatic modulation as individual sources. LEDs are available in a wide range of output colours and they vary in size and optical power [Wolf 1981]. The advantage of such systems is the reliability, compactness and mechanical stability. However, these systems have low optical power and poor temperature stability. Lasers are monochromatic sources (section 2.2) which have good wavelength stability. They can very in size from a few centimetres square, for a laser diode source to a few meters for industrial grade welding lasers. The main advantage of lasers is that they can provide coherent optical radiation (Section 2.2). This means that interference patterns called speckle noise may be observed. The interference, or speckle, from more than one laser of different wavelengths when analysed using chromatic techniques, can provide important information about light propagation through a medium.

5.3.4. Computer System.

For image processing applications the demands on computer processing power and capacity are high. However, computer power has to be balanced with system costs for practical system design and implementation [Pauk 1992]. An IBM-PC compatible system was preferred for this work for a number of reasons, These were:

- adequate software resources
- suitable hardware interfaces
- Established industrial and commercial usage
- Reasonable pricing with adequate support
- Scope for high processing power

The specifications for the computer used throughout the project are:

IBM-PC compatible

Intel 80386-DX processor working at 33 MHz

80387 Maths Co-processor

4 MB Ram 70ns access time

110 MB HD space 15ms access time

VGA monitor

Apart from the other standard PC components the base unit also housed the FGC card which is described in section 5.4.2. The computer was the platform on which all the software was developed.

5.4. Software Description

The language used for software development was Borland C/C++[Kernighan 1978]. This language was chosen because it produced good interfacing functions and can generate programs that make good use of the system resources [Ezeel 1992]. C/C++ is a well established language that allows scope for making the programs more transportable (transferable from one operating system to another) and modular for simple updating and checking. The software module may be subdivided into four main sections. These are:

- Control functions for the FGC.
- Image processing.
- Data processing.
- User interface.

The Control functions for the FGC form the link between the FGC and the Computer. These are responsible for all the basic operations involved with handling image data that is stored in the frame grabber memory. The speed of performing frame operations depends on the speed of both the computer and the FGC. C++ object oriented structures allowed the development of simple objects that performed all the housekeeping operations for the FGC control. The most time consuming operation within the control function block was that of dumping a whole image from the FGC memory to the computer and the reverse process of restoring an image from the computer to the FGC memory. This time period was approximately 10 seconds.

Most other operations could be considered fast enough for real time processing applications [Alford 1977], performing tasks at a few MHz.

The speed of the image processing tasks were mainly computer dependent and maximum use of the computer memory was made to ensure that the highest processing speeds were obtained. The purpose of image processing in this project was to locate a sensor in the image. Once the sensor was located then the chromatic information could be extracted and processed. Object identification required a large amount of the total time required for image processing. Starting from the acquisition of the image and ending with the positive identification of the sensor, the time required is 10 to 12 seconds depending on the image quality and its complexity. During this "dead time" the sensor is assumed to be stationary in the camera field of view. Once identification has been completed the data processing and image tracking can operate at a much higher speed, acquiring a number of data readings in a second. A fuller description of the image processing methods has been given in section 3.4.

The data processing operation required a selection of different methods to be implemented on the chromatic information. These methods included algorithmic techniques, neural network processes, and look-up table approaches. The speed of these operations is computer dependent.

The user interface provides the front end of the program. It provides the presentation of the outputs provided by the system and some control over the operations of the system. A number of graphical user interfaces (GUI) have been developed during the course of this work addressing specific system requirements.

5.5. Sensors

As well as forming the means for translating a measurand, the sensors are also an important part of the image processing system. Sensors may be either extrinsic or intrinsic chromatic changes in an object may be used. The form of the chromatic element may be either the light source itself (e.g. plasma emissions, speckle patterns) or a transducing element (e.g. thermochromic and photoelastic devices). To simplify the tasks of image processing the sensor shape and design is of importance and ideally should be fixed. This would allow dimensional checks on the object to confirm the identification of the object (section 3.4.5). For data processing it is advantageous to have a reference colour area on the sensor itself to allow for referencing and compensation. The design of such a sensor is shown in figure 5.4. A fuller explanation of the sensors that were used throughout this project is given in section 2.6.

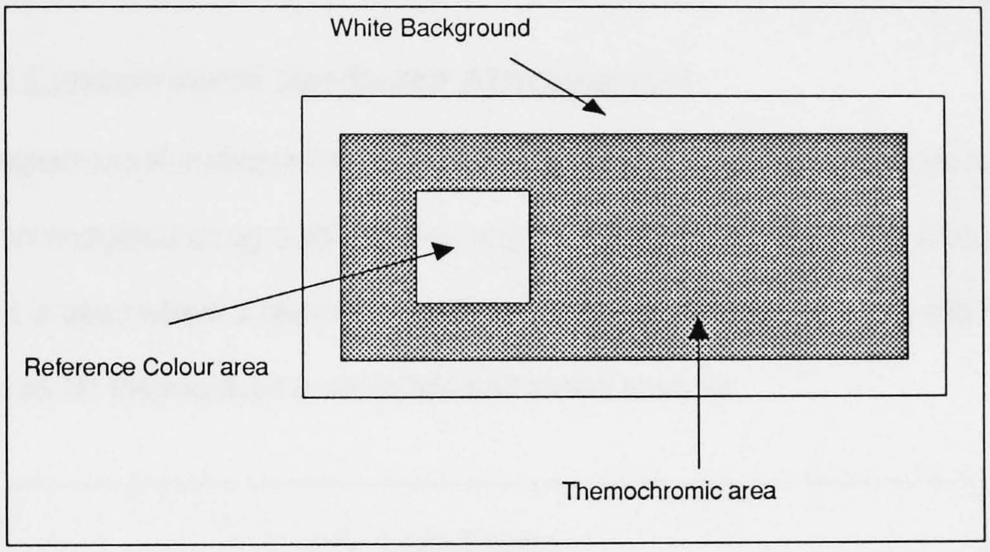


Figure 5.4 Diagram showing design of a sensor to aid image processing tasks.

5.6. Parameters That Affect System Performance

A number of different real world effects are identified as possible sources of data corruption. These are listed below:

- Drift
- Lighting Level
- Glare
- Saturation

Drift is the change with time observed in the steady state output of the system. The sources of drift can be; light source variation, sensor ageing and degradation.

Lighting level changes may be caused by the shadowing of the light source, the presence of secondary sources and by the ageing of the light source.

Glare can cause saturation of the input data because of direct reflection of light from the source. If a partially reflective object is illuminated by a bright source, then at angles equal to the angle of incidence from the normal of the object the light is directly reflected back.

Saturation is where the incident light energy is bright enough to saturate the detectors. This undesirable effect is caused by poor control of the camera aperture and source intensity.

5.7. Basic Experimental Hardware Arrangement

The basic experimental arrangement is shown in figure 5.5. The diagram shows how a sensor was interrogated using a light source and a camera connected to the computer. This arrangement is used where a textural chromatic change is observed on the surface of the sensor such as for thermochromic materials and stress analysis.

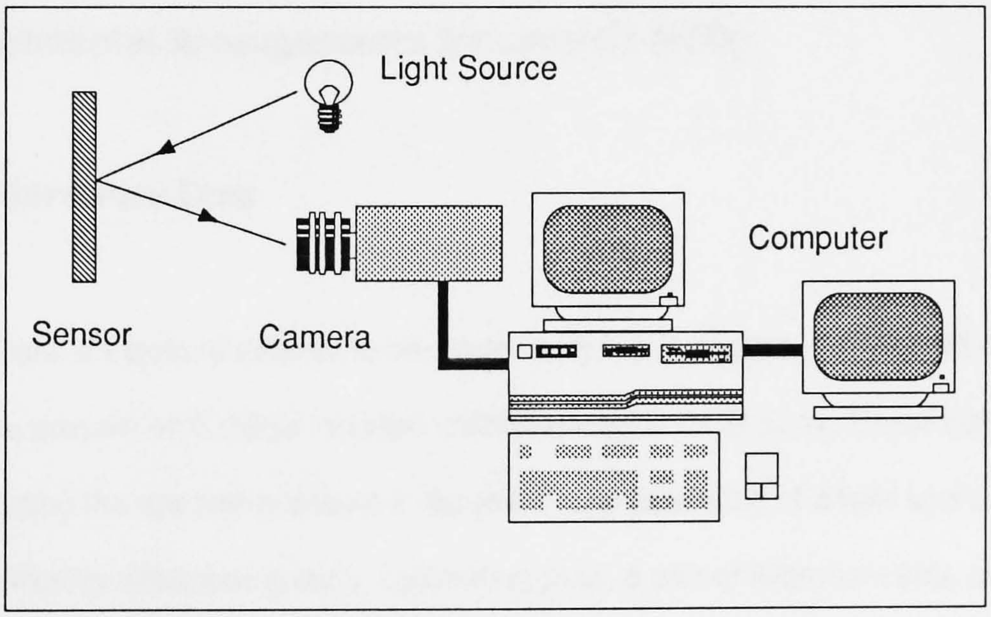


Figure 5.5. Diagram showing basic system arrangement used for reflective monitoring.

If the sensor is translucent and observation of the system is through the material itself then a transmissive variation of the system is employed (figure 5.6). This arrangement is used for oil monitoring and chemical indicator tests.

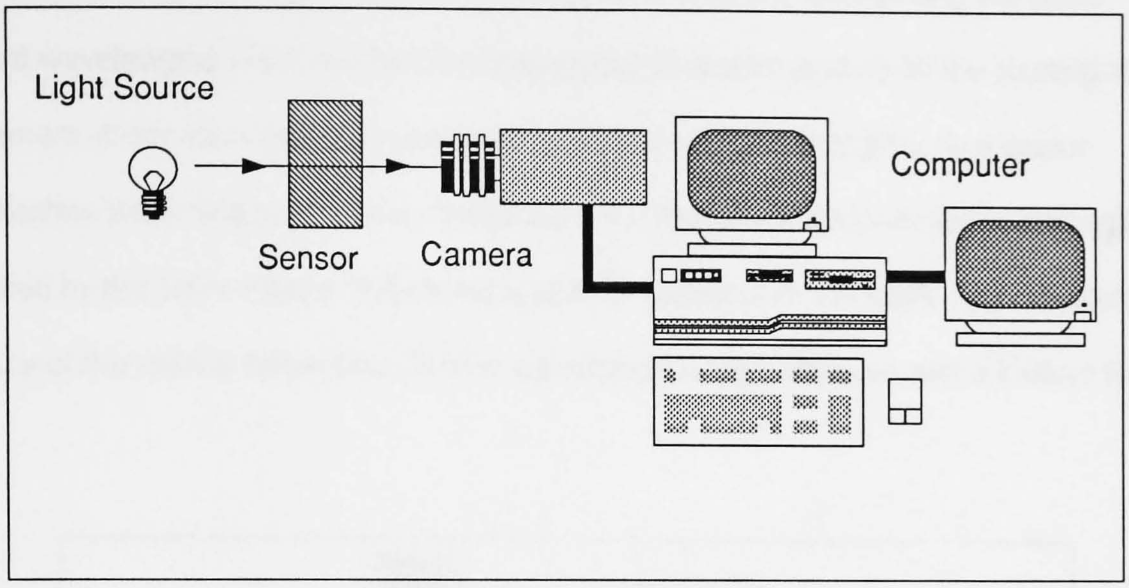


Figure 5.6. Diagram showing basic system arrangement used for transmissive monitoring.

In experiments where the chromatic change being monitored is of the optical source itself the system is much simpler not requiring source and sensor. This arrangement was used for plasma monitoring.

5.8. Experimental Arrangements for specific tests

5.8.1. Hardware Tests

Hardware tests are tests to determine the performance of the camera, FGC and computer only. For the present work these included calibration of the CCD array. These tests were performed using the apparatus shown in figure 5.7 and consisting of a light source, a concave reflecting diffraction grating, collimating slots, a pair of alignment slits, an optical spectrum analyser (OSA), a rotateable stand for the grating and an oscilloscope for recording the output from the OSA. The diffraction grating provided a spectrum of coloured fringes when illuminated by the white light source. A specific part of the spectrum was selected by the alignment slits. The light was passed through a half silvered mirror for monitoring simultaneously by both the camera and the OSA. The oscilloscope displayed information of the light intensity as a function of wavelength incident onto the camera and the OSA. Different wavelengths were selected by rotating the diffraction grating on the rotating stand. The camera information was then processed using the computer to provide a colour quantification according to schemes of section 2.4.2 that correlated with the wavelength monitored by the OSA. Figure 5.8 shows a sample output from the OSA. The wavelength accuracy of the OSA is better than 5nm in wavelength once calibrated with a Helium Neon Laser.

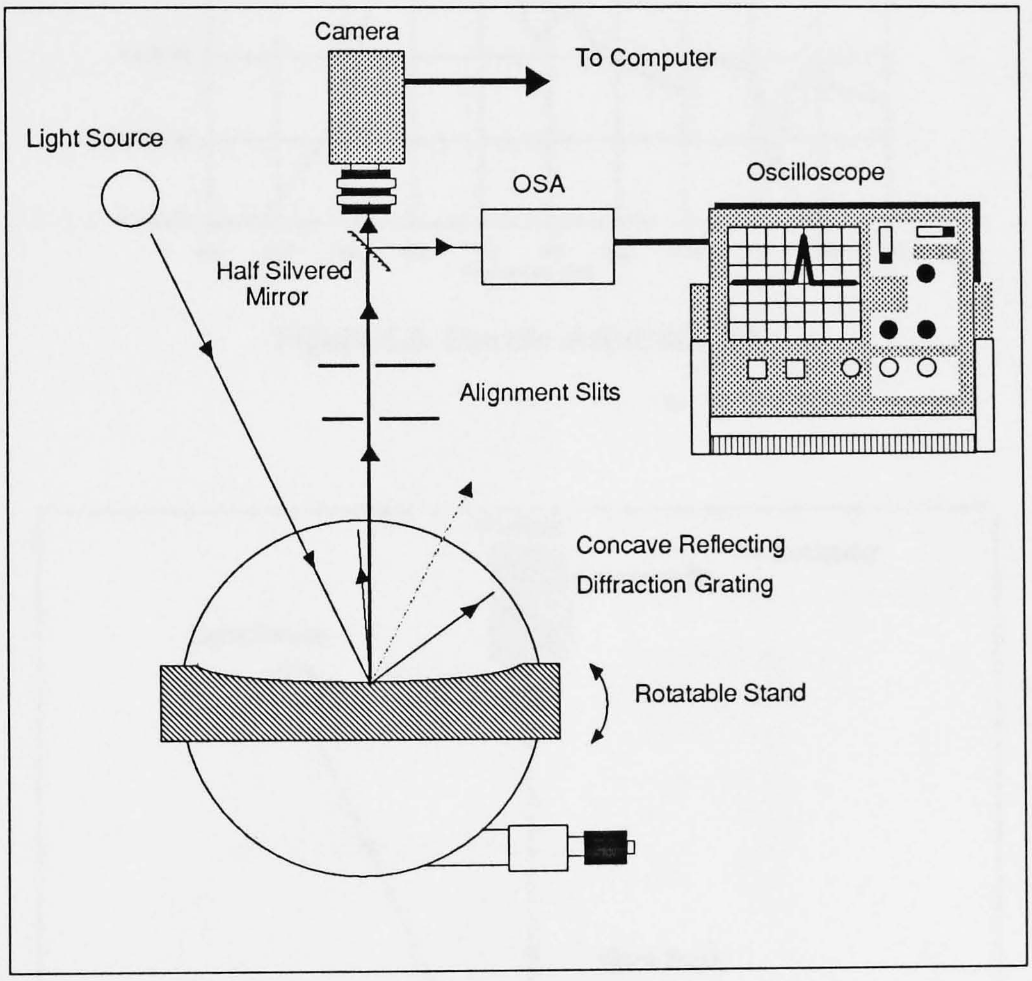


Figure 5.7. Diagram showing arrangement used for calibration of camera.

In this manner the output of the camera system could be calibrated under ideal operating conditions. Similar tests could be performed with this apparatus to measure also the effects of external spurious effects like glare. For such purposes the diffraction grating was replaced by a thermochromic strip (Figure 5.9) This was chosen as the colour producing element since it could provide a stable colour sample by controlling the latter's temperature. By rotating the thermochromic strip on the rotating stand the amount of glare on the sample could be varied and the results measured with the camera. The rotateable stand had graduations that allowed the angle of rotations to measure to an accuracy of one degree.

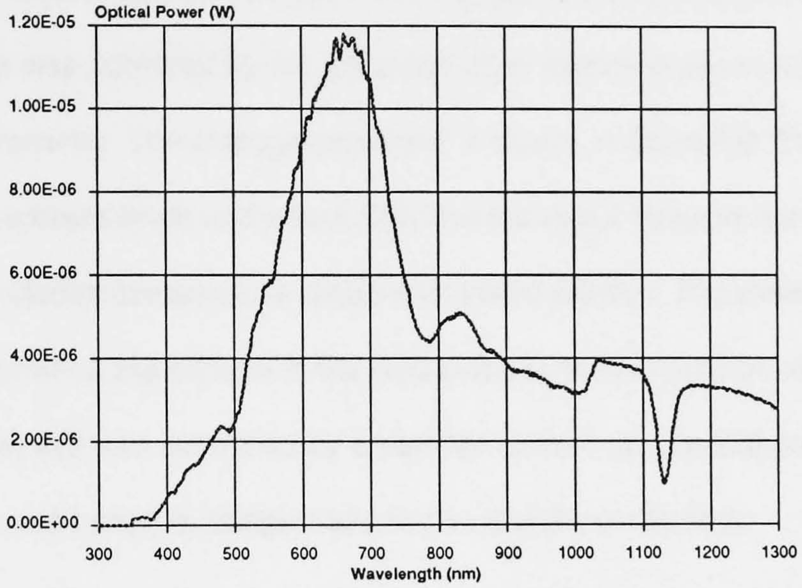


Figure 5.8. Sample output of OSA

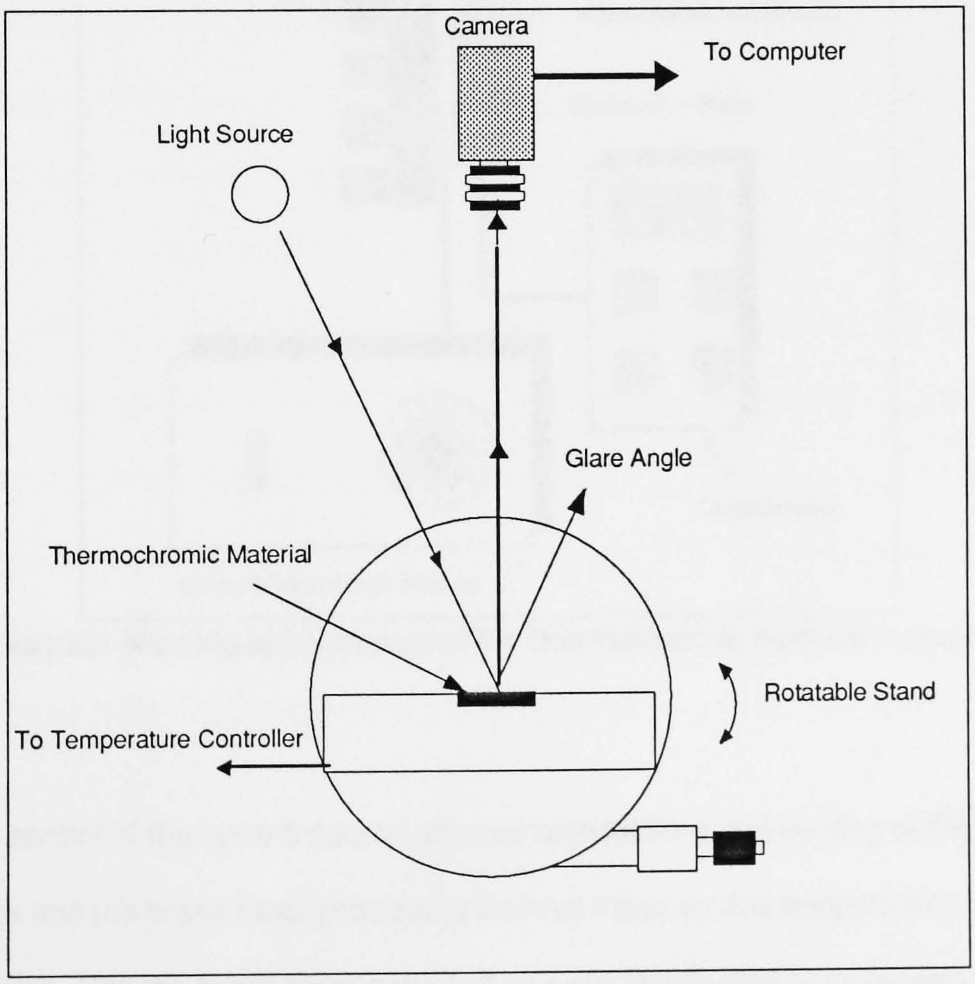


Figure 5.9 Diagram showing arrangement for monitoring glare effects.

5.8.2. Thermochromic Temperature Tests

Calibration tests on thermochromic strips were performed using the basic hardware arrangement described in section 5.7. Careful temperature monitoring and control was also necessary and this was achieved by the utilisation of an electronically controlled Peltier cell and a digital thermometer. The arrangement used is shown in figure 5.9. The Peltier cell was clamped between a brass block and a heat sink and the temperature of the brass plate was measured with an electric temperature probe embedded within it. The thermochromic element was mounted on the surface of the bras block so that it could be viewed by the CCD camera. The Peltier cell was controlled by a variable current source that provided the heating of the brass block and hence its temperature to be carefully controlled.

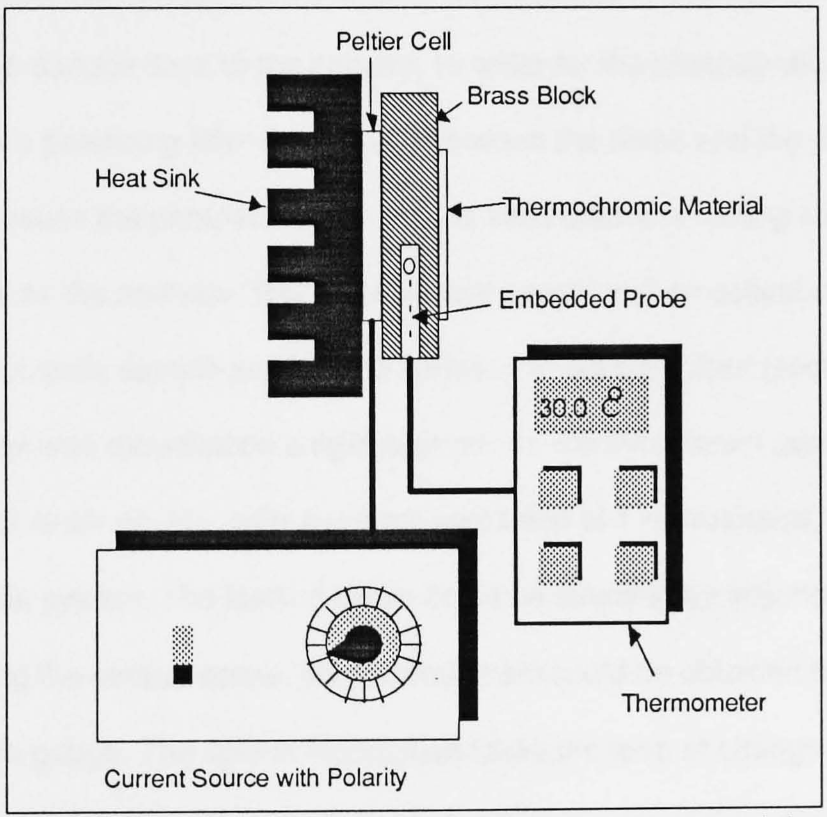


Figure 5.9 Diagram showing apparatus used for Thermochromic material temperature control.

The polarity control of the current source allowed both heating and cooling of the Peltier cell. The heat sink and the brass block provided a thermal mass so that temperature fluctuations were minimised. Also the brass block provided an even distribution of temperature over the surface of the thermochromic material and a location for embedding the temperature probe for accurate temperature readings. The digital thermometer provided temperature readings during the tests with an accuracy of 0.1°C. The current source with the Peltier cell provided control of temperatures in the range from 20 to 80°C.

5.8.2 Photoelastic Strain Monitoring

The photoelastic strain experiments were based upon two sensor configurations. One configuration was used for single point strain experimentation and the other was used for two dimensional barometer type measurements. The apparatus for the single point measurement is shown in the figure 5.10. It consisted of a photoelastic plastic material that was interrogated using optical fibre transmission. One fibre was used to transmit light from a source to the photoelastic surface and the other was used as to transmit the light modulated by the photoelastic surface back to the camera. In order for the photoelastic material to modulate the light a polarising filter was placed between the fibres and the photoelastic material itself. Because the photoelastic material is fixed onto a reflecting surface the polariser acts also as the analyser.This arrangement constituted an optical strain gauge since straining the photoelastic sample produced a variation in optical output (section 2.6.1). The optical strain gauge was mounted on a rigid support. An electrical strain gauge mounted close to the optical strain gauge, with a system resolution of 1 microstrains, was used to calibrate the optical system. The loading beam could be stressed by adjustment of a central screw. By adjusting the central screw, the desired strain could be obtained by monitoring with the electrical strain gauge. The optical modulation takes the form of changes in polychromatic fringes produced by the strained photoelastic material (section 2.6.1).The colour of the modulated fringe patterns for incremental strain setting was monitored using the CCD camera.

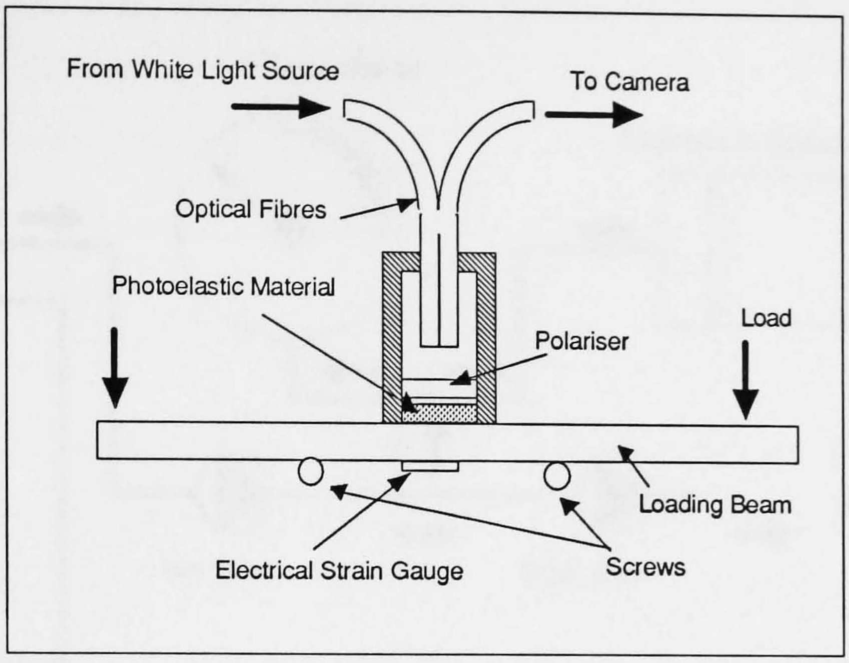


Figure 5.10. Apparatus for single point stress analysis

For the photoelastic barometer experiments the apparatus shown in figure 5.11 was used. This consisted of a pressurised gas canister to provide a pressurised air source of the gas control system used in conjunction with a gas delivery system, a Bourden pressure gauge and a photoelastic element based on a pressure sensitive membrane. The flow of pressurised gas was controlled by adjusting the inlet and outlet valve. The controlled pressurised air was then expound to both the photoelastic barometer and a standard mechanical pressure gauge with a range from 0 to 120 lb/in^2 with an accuracy of 2 lb/in^2. Details of the construction of the photoelastic barometer are shown in figure 5.12.

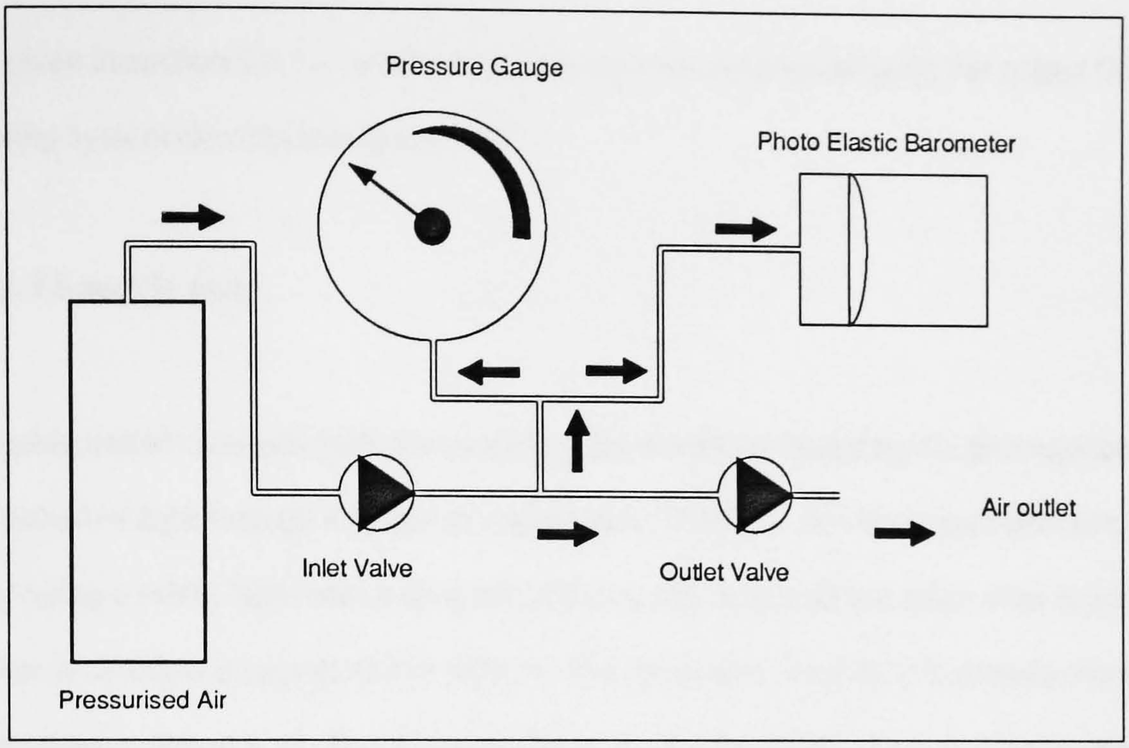


Figure 5.11 Diagram showing Barometer experimental apparatus

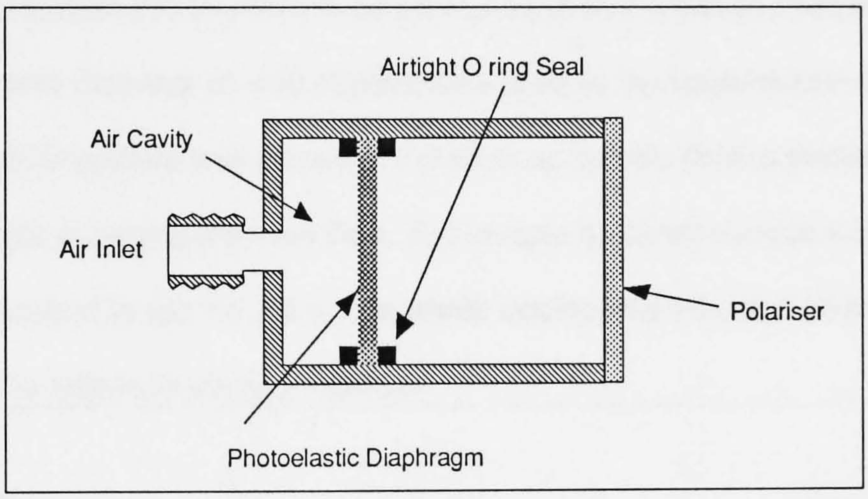


Figure 5.12 Diagram showing construction of the photoelastic diaphragm.

This was constructed using a cylindrical container that held a reflecting photoelastic diaphragm with air tight O ring seals. The air inlet allowed air to enter into an air tight cavity behind the photoelastic diaphragm. The air pressure behind the diaphragm induces mechanical stresses over its surface. A polariser placed in front of the photoelastic diaphragm allowed it to be interrogated when illuminated with a white light source. As the diaphragm was stressed coloured fringe patterns are observed in accordance with the basic theory given in section 2.6.1. These fringe patterns were monitored using the colour CCD monitoring system described in section 5.7.

5.8.3 Speckle tests.

For speckle pattern analysis tests the speckle pattern was produced by the propagation of highly coherent light through a length of optical fibre. The fibre was energised with two lasers, the first being a HeNe laser with a spectral output at 632.8nm and the second an Argon ion laser with a tuneable output at 450 or 550nm. The apparatus used for the speckle monitoring tests is shown in figure 5.13. The diagram shows the configuration of the two lasers and the method used to launch the laser light from one laser into the fibre at different angles of acceptance by the fibre. Neutral density filters placed in front of the laser outputs allowed the intensity of light incident into the fibre to be controlled. A step indexed multi mode fibre of 1 m length with a core diameter of 400 microns was used as the transmission medium for the laser light. The CCD camera was placed at the other end of the fibre to monitor the speckle patterns in the light emerging from the fibre. The images obtained were processed using the computer as described in section 2.5.4. The whole system was mounted on an optical bench and fixed rigidly to eliminate vibration effects.

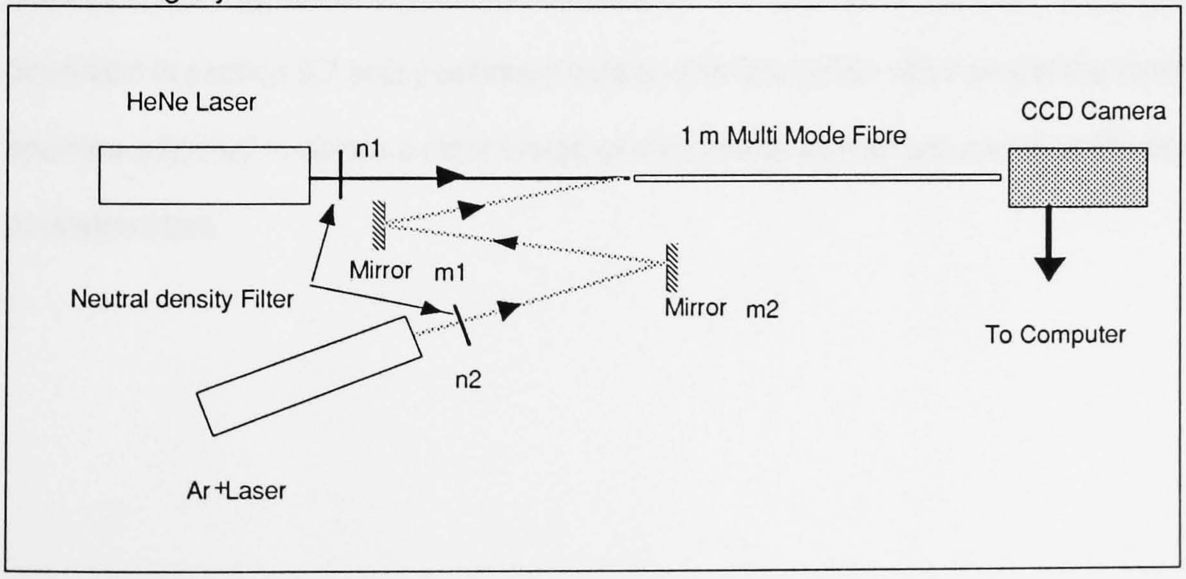


Figure 5.13 Diagram showing arrangement used in chromatic speckle monitoring.

The HeNe laser was arranged so that it injected light along the principal axis of the fibre. The Ar^+ laser was arranged to inject light at a shallow angle into the fibre. The angle of incidence of the Ar^+ laser could be adjusted by moving the two mirrors m_1 and m_2. The lens and aperture module which was a component of the camera unit was not used but instead light emerging from the laser was directly incident on the CCD element of the camera. The intensity of the laser light was adjusted by the use of the neutral density filters m_1 and m_2.

5.8.4 Plasma Experiments.

The plasma experiments required the use of a commercial plasma chamber provided by CHELL Instruments. The plasma chamber required a process control module for controlling the various plasma parameters such as chamber pressure, radio frequency power and gas flow controls (section 2.6.5). A schematic of the system is given in figure 5.14. This shows the r.f. plasma chamber that housed the semiconductor wafer to be processed with the plasma. The reactant gasses (CF4 and Nitrogen) flowed through the chamber via an inlet and outlet pipes as shown in figure 5.14. The operation of the plasma chamber (gas flow and plasma pressure, r.f. power, etc.) was under computer control via a module produced by Chell Instruments Ltd. A port hole on the side of the plasma chamber with a glass window allowed visual inspection of the chamber interior. The camera of the monitoring system described in section 5.7 was positioned outside this inspection window and the camera aperture adjusted to obtain a clear image of the plasma without saturation of the camera photodetectors.

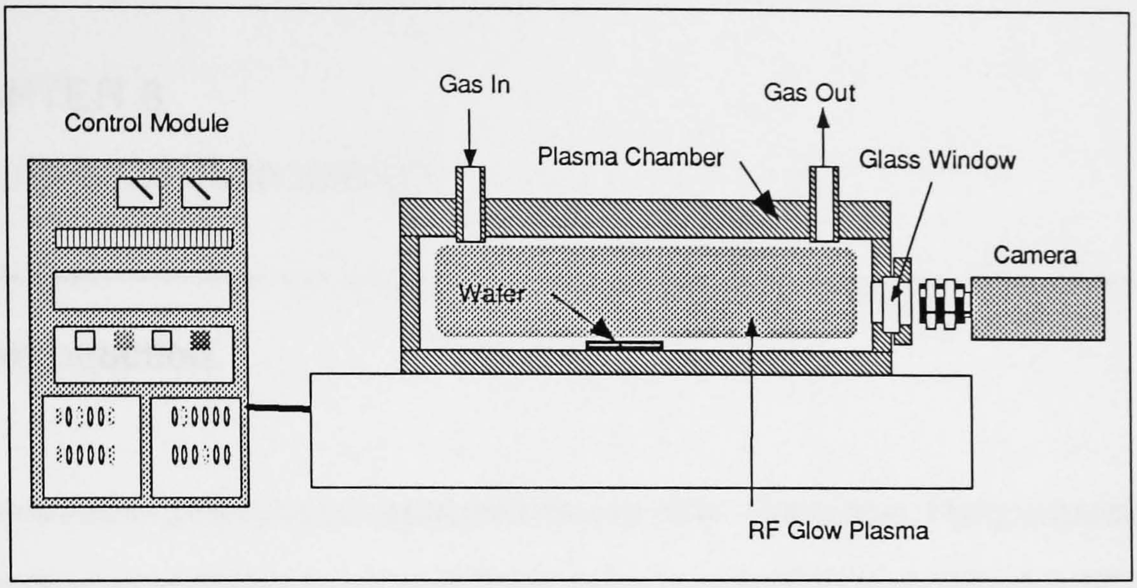


Figure 5.14. Apparatus used for Plasma Monitoring.

5.9. Summary

This chapter described the system proposed for processing remotely acquired chromatic signals. The principles of operations of the various components of the system have been discussed and the particular embodiment of these systems for tests with different chromatic modulation methods have been described.

The next chapter presents the experimental methods employed for the various tests before the results are presented in chapter 7.

CHAPTER 6
Experimental Methodology.

6.1. Introduction.

The experiments performed can be divided into two distinct categories. Firstly experiments were performed on the hardware itself. This included camera tests and calibration, FGC adjustments, ADC control for referencing, drift and saturation. These experiments provided information on the system capabilities and its limitations. The second kind of experiment dealt with tests on a number of different chromatic sensing devices. These devices included photo-elastic barometer sensors, thermochromic strips, speckle patterns, plasmas and chemical indicators. The objectives of these experiments were to investigate the performance of the remote chromatic processing system for specific applications.

6.2. Hardware tests.

Hardware tests were performed in order to characterise the system performance independent of the sensor or the environment. These tests were divided into three categories, namely:

a) Camera tests

b) FGC Tests

c) General Hardware Tests

These sections will be dealt with independently.

6.2.1. Camera Tests

1) CCD Array Response.

The optical frequency response of the CCD element was measured using the apparatus described in section 5.8.1. The system was configured so that a specific wavelength was observed with the camera and the OSA. The OSA connected to the oscilloscope provided intensity and wavelength information of the incident light. Calibration of the OSA was performed with a single point calibration from a HeNe Laser (632.8nm). The grating provided monochromatic light with a bandwidth of 25nm and measurements were taken at 25nm intervals. The focus and aperture lens were not used in the experiment and the light from the grating was incident directly on the CCD array. Gamma correction was not used on the camera and the FGC system was initialised with normal default values (section 5.3.2.1). Software written to acquire the data monitored an 8x8 pixel area of the image and provided a mean value of the R, G, B intensities. The tests were performed in a dark environment to eliminate the effects of external lighting and also repeated for different areas of the CCD array.

2) Aperture Effects.

The aperture of a camera provides a control for the intensity of light incident into the camera. It is similar in nature to the pupil of the eye. Ideally aperture adjustments should have no effect on the chromaticity of the image. However, because of chromatic aberrations caused by the lens and the unequal sensitivities of the detectors, practical apertures for cameras are known to affect the chromaticities [Hect Zajac 1974]. The extent to which the aperture affects image chromaticities needs therefore to be investigated.

The apparatus was arranged as shown in figure 5.4 and described in section 5.7. The sensor here was replaced with a white card to provide a source of diffused white light. Light source intensity was adjusted using neutral density filters such that the image was illuminated to the point of saturation when the aperture was fully open. The Gamma correction factor was switched off and the output measured using the FGC and the computer system running similar software to that used in the CCD array response experiment. Aperture values were read directly from the aperture dial of the lens and provided a scale for the measurements. Output RGB values were obtained for periodic intervals of aperture settings. The experiment was performed in a dark environment to avoid external lighting effects.

3). Gamma Correction

Gamma correction is an option available on the camera to provide a non linear gain on the CCD array input amplifier (section 5.3.1). The effect of gamma compensation was tested using the same apparatus as described in the aperture tests. Here the RGB intensities were obtained for different aperture settings with the gamma compensation on and then repeating it with gamma compensation switched off (linear gain).

4) Dark Noise.

Dark noise is an undesirable feature of the camera output that is defined as the output voltage of the camera when no light is incident on the CCD array. To monitor this the FGC ADC could not be used directly. Instead the output of the camera was monitored directly with an oscilloscope. The oscilloscope was made to trigger with the frame sync pulse from the camera and line data was observed. The output signal from the camera when the aperture was closed was monitored on the oscilloscope.

6.2.2. FGC Tests.

The FGC tests concentrated on the analogue to digital conversion of data obtained from the camera. Tests on the linearity of the ADC were performed and also the effects of positive and negative reference adjustments considered. These tests not only helped to determine the hardware limitations of the system, due to the FGC but also indicated possibilities of using reference levels in a compensation technique.

1) ADC Linearity.

For ADC linearity tests the same experimental apparatus as the aperture tests were utilised. An oscilloscope was used to monitor the input camera voltage levels and software was used to interrogate the resulting FGC data. The reference voltages for the ADC were set to the factory default settings (section 5.3.2.1). Values of ADC output were obtained for the various input signal voltages obtained from the camera. The signal produced from the camera was controlled using the camera aperture and by monitoring signal voltages directly on the oscilloscope.

2) Reference Adjustments.

The FGC allows software control of the positive and negative reference voltages for each of the input ADCs. This was found to provide a means of chromatic reference adjustment. A series of tests were performed to determine the limitations of this reference control. The experimental arrangement was identical to that used for the ADC Linearity test. As the RGB channels exhibited identical characteristics only one channel was investigated. Firstly the ADC outputs were monitored for various input voltages using different positive reference voltage settings and keeping the negative voltages constant. The experiments were then repeated with the positive reference voltage constant and the negative reference voltages were varied.

6.2.3. General Hardware Tests.

General hardware tests dealt with the general performance of the data acquisition system, taking into account the effects of light source, camera and the FGC. These tests included system noise, glare and drift tests.

1) System Noise.

System noise tests determine the noise on the data provided by the system which include light source flicker, camera signal noise and FGC resolution. For system noise tests the monitoring system was used to observe a uniform grey area which provided an RGB response in the middle of the range of the system. Initially a single pixel was monitored and then an 8x8 pixel area was observed and logged over a one minute interval. This allowed quantification of system noise.

2) Glare Tests

Glare tests were performed to determine the extent to which glare affects the chromaticity values obtained from a sensor. To perform these tests the experimental arrangement shown in figure 5.8 was used (section 5.8.1). A thermochromic strip with a temperature control system was used as the target sensor. This was used primarily because the thermochromic strip provided a suitably reflective surface to observe the effects of glare and also because a variety of different colour values could be tested by adjusting the temperature of the thermochromic strip. Colour RGB values were obtained from the system for various glare angles of the thermochromic strip at a range of temperatures.

3) Drift Tests.

The long term stability of the chromatic system was investigated by the monitoring of a diffused grey surface illuminated with a stabilised light source. The system was made to log data for a period of 2 days obtaining hourly data points. This test took into account the drift in both the light source and the camera system.

6.3. Sensor Experiments

6.3.1. Thermochromic Materials

The Thermochromic tests employed a strip of thermochromic material produced by Liquid Crystal Devices (LCD). This material exhibited visible chromatic change over a temperature range of 20 to 50^{O}C. The specified dynamic range allowed easy control of the strip temperature which was implemented using the apparatus described in section 5.8.2. The tests that were performed on the system included temperature calibration tests and temperature cycling tests.

One of the main objectives of the thermochromic experiments was to determine the remote sensing capabilities of the system and this work included a number of software tests. Software tests required a more specific definition of a sensor that contained a reference colour area and a predefined sensor dimension. This was shown in figure 5.4 (section 5.5). The tests mainly concentrated on the capability of the system to identify the sensor and implement tracking and data extraction techniques. A more detailed review of the experimental methodology is presented in the sections below.

1) Temperature Calibration

The variation of observed chromaticity with temperature of the sensor was obtained using the apparatus described in section 5.8.2. Care was taken to eliminate glare by changing the angle of the sensor face from the camera and the aperture was adjusted to provide a bright picture for the full range of the sensor chromaticities. Camera Gamma correction was not used and the FGC reference values were set at the default settings (section 5.3.2.1). The temperature of the sensor was measured using the digital thermometer and chromatic data was taken for 0.5^{O}C increments in temperature over the 20 to 50^{O}C Range. Data was monitored using a fixed 8x8 area on the sensor surface and the average RGB values for that area were logged along with sensor temperature. This data set was then used to perform a number of off line tests on the development of suitable calibration techniques.

6.3.2. Photoelastic materials

Photoelastic materials that exhibit a chromatic change with applied stress may be purchased in the form of plastic sheets of various thicknesses and can be cut to a desired shape. They require the use of polarisers for their operations. A number of different sensor designs are proposed for non-contact monitoring.

1) Single Point Analysis.

Single Point Analysis utilised optical fibre based test apparatus shown in figure 5.10 and described in section 5.8.2. The photo-elastic material was stressed by turning the centre screw under the beam and the stress of the beam was measured using conventional electrical strain gauges. The Camera was used without the focus and aperture lens and the CCD made to directly address the optical fibre output. RGB data was obtained for various beam strain settings at equal intervals. Calibration curves were derived by relating this chromatic output ith the output from the electric strain gauge.

2) 2-Dimensional Photoelastic Analysis

The 2-D Photoelastic strain analysis which involved addressing the whole surface area of a photoelastic element [e.g. barometer design (section 5.8.2)] utilised the apparatus shown in figures 5.11 and 5.12. Software was written to provide a reduced data set for sensor calibration. The circular diaphragm image was analysed by splitting the image into 10 circular concentric ring regions of equal area each with its centre corresponding to the centre of the diaphragm. The mean chromaticities x and y (Section 2.4.2.2) were obtained for each annular area and this data was utilised in performing further tests on system calibration using neural networks (section 4.2). Pressure was applied to the barometer between the ranges of 0 to 60 lb/in^2 and 20 images were obtained at equal pressure intervals. These images were then processed and analysed using the neural network system and placed for training.

6.3.3. Plasma Monitoring

The plasma monitoring experiments were performed using the CHELL Instruments plasma chamber described in section 5.8.4 and shown in figure 5.14. Two types of experiments were performed using the same apparatus. Firstly the spatial chromaticity of the plasma was monitored for different levels of r.f. power sustaining the plasma. All parameters that might affect the optical emission of the plasma (e.g. gas flow, plasma pressure etc.) were kept constant. The second type of experiment was the monitoring of a plasma etching process. In this experiment a sample wafer was placed inside the chamber and an etching run was made. At various points of the etching run, images of the plasma discharge were obtained. These image data were processed off-line to determine if the end point of the etching process could be identified using chromatic techniques.

6.3.4. Speckle Monitoring

Speckle patterns were monitored using the apparatus shown in figure 5.9 and described in section 5.8.3. For these experiments the Ar^+ laser was tuned to a frequency of 514.5 nm. The intensity of the two lasers was stepped down by the use of neutral density filters placed before the two beams such that the image obtained by the camera was not saturated. Images were obtained for four different angles of incidence of the Ar^+ with the HeNe Laser injecting light axially into the fibre. Images obtained from this arrangement were processed off-line using software techniques.

6.3.5. Chemical Indicators

A number of chemical indicators were tested during the course of this work. The sensors were the indicators themselves and the hardware arrangement was that shown in figures 5.5 and 5.6 (section 5.7) depending on the form of the indicators.

For experiments based on the Nephur medical strips, a set of calibration tests were performed to map the chromaticity values of the different areas of the comparison chart (section 2.6.3.1). The data was used in a look up table for checking with strips immersed in laberlled urine samples with known component concentrations.

Chemical indicator tests were also performed on a number of samples of electrically insulating oils from power transformers that were at different states of degradation. Chromatic tests on these samples were performed to identify if remote chromatic techniques could be employed to determine transformer oil quality.

CHAPTER 7
Experimental Results.

7.1. Hardware Tests

7.1.1. CCD Array Calibration.

The results for the calibration test (sections 5.8.1 and 6.2.1) on the CCD array are shown in figure 7.1. The graph shows that the CCD element may be considered as being composed of two distimulus detector elements as opposed to a single tristimulus element (red and blue responses do not overlap). The peak response of the red green and blue channels are at 620, 530 and 430 nm respectively. The spectral bandwidth of each detector is approximately 250nm.

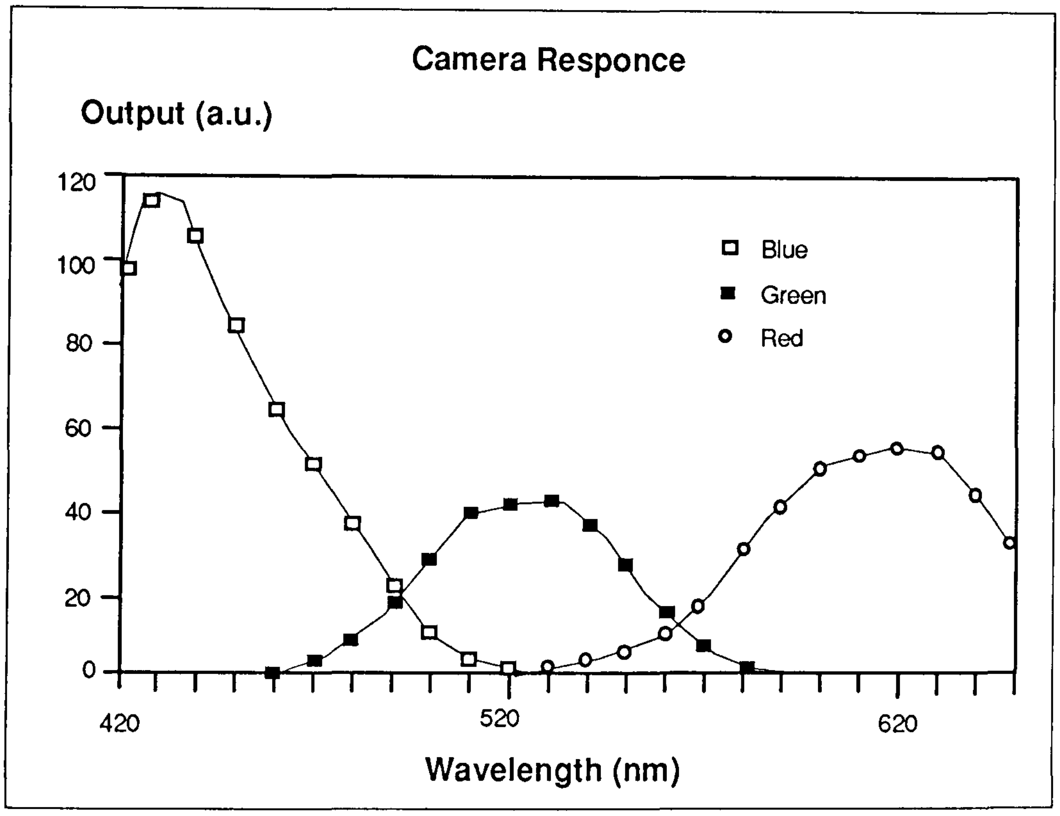


Figure 7.1 Graph showing CCD array wavelength response.

7.1.2. Gamma Compensation. Aperture Effects.

The dependence of the image on aperture size was tested to quantify the extent to which the aperture affected the chromaticity of the image. For this experiment the aperture values were read directly from the aperture dial. The figure 7.2 shows that the relationship between this value and the output signal is non-linear. It forms a sigmoid function with the effect of saturation observed for low aperture values (aperture open) and the non-linearity of aperture scaling for larger aperture values (aperture closed). The curves of figure 7.2 show also that the sigmoid curves for each of the 3 detectors are displaced with respect to each other along the aperture axis. This leads to an important effect at low light levels. A plot of the effect of the aperture on the x and y chromaticity values (figure 7.3) shows that as the aperture is reduced there is a shift in chromaticity towards the red. This effect is observable from the output of the FGC to the screen which shows a red tinge for a grey image when the intensity is low. In chromatic terms this shift is of the order of 0.08 in y and 0.15 in x which are significant changes compared with the resolution of the system.

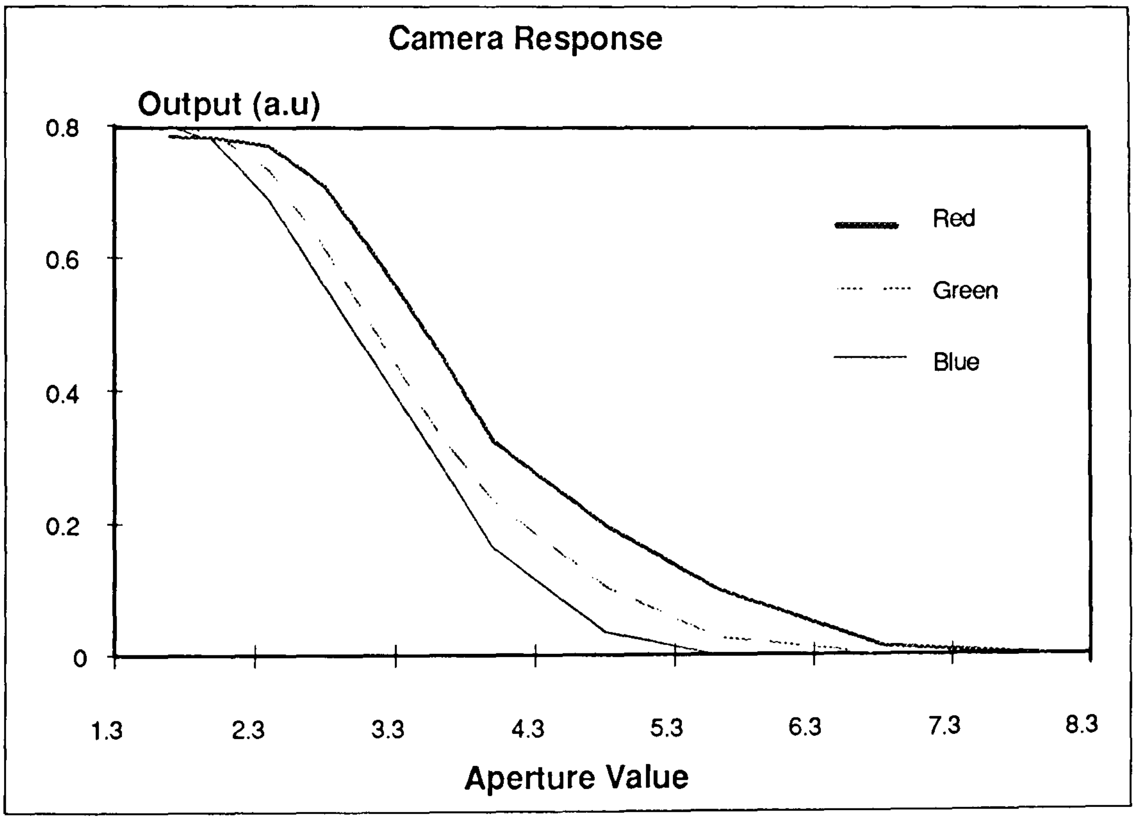


Figure 7.2 Graph showing variation of output with aperture value

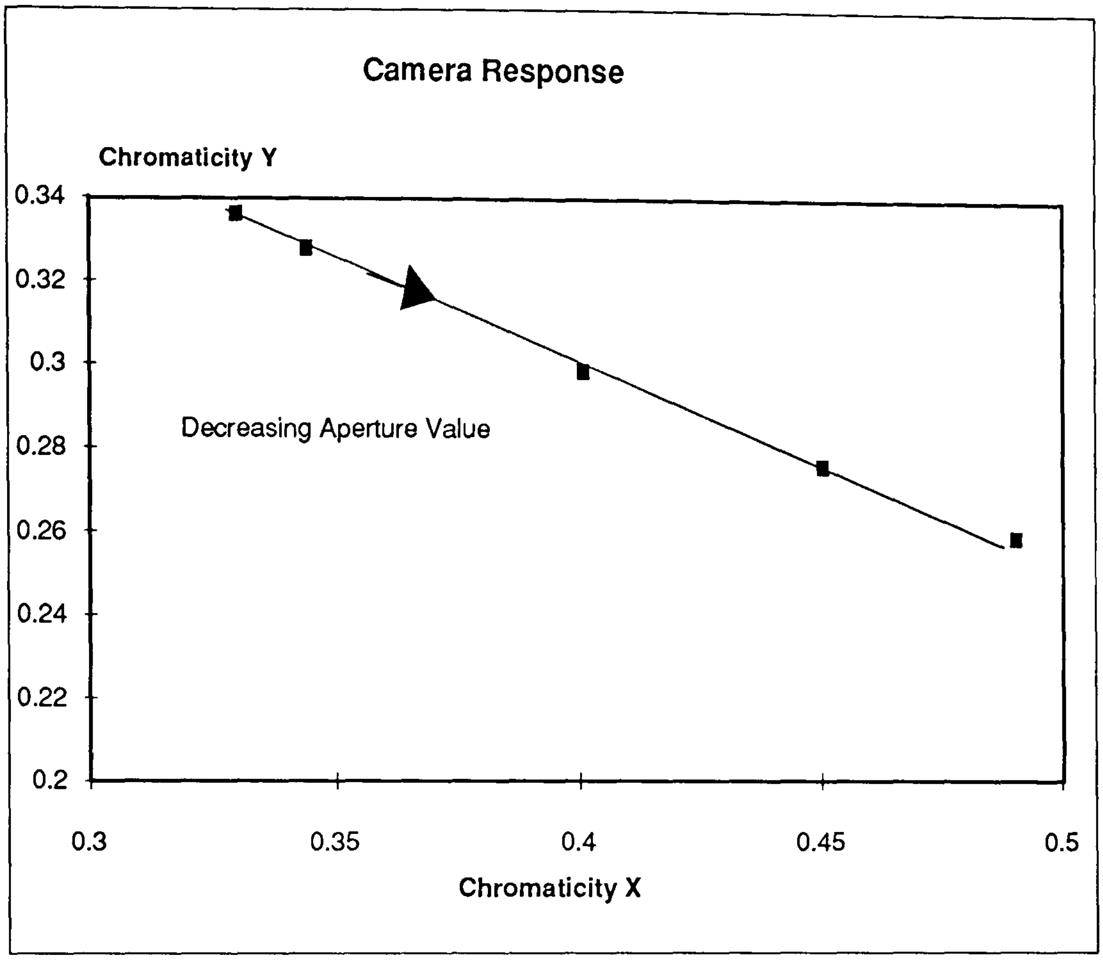


Figure 7.3 Graph showing Chromatic shift to the red due to intensity reduction.

The Gamma compensation switch on the camera (section 5.3.1) allows non-linear amplification to the CCD array. The effect of this gamma compensation is shown in Figure 7.4 below. The effect of switching gamma correction on for a single channel is shown in Figure 7.5. One point of interest is that with the gamma correction switched on, the red tinge apparent with low white light intensity (see figure 7.3), is greatly reduced. The effect of gamma compensation on the chromaticity is shown in Figure 7.6. Thus the chromatic changes due to the aperture effects are reduced from - 0.08 to - 0.03 for y and from +0.15 to + 0.09 for x. However the values remain within the resolution capability of the system.

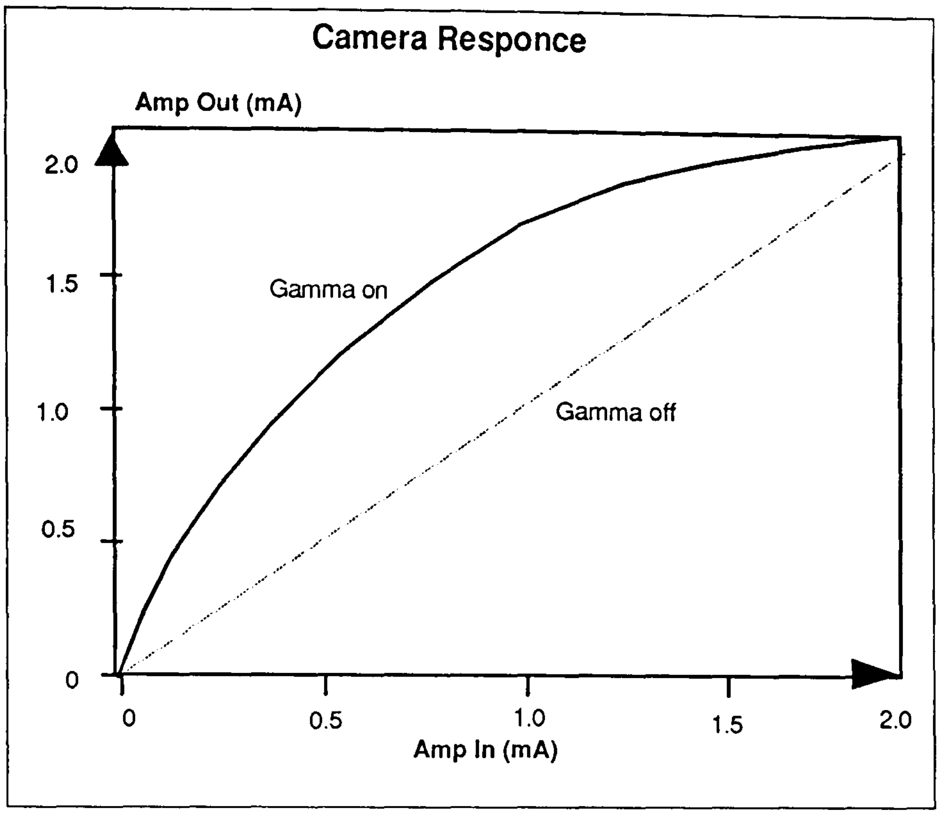


Figure 7.4 Figure showing the effect of non-linear gain with Gamma switching.

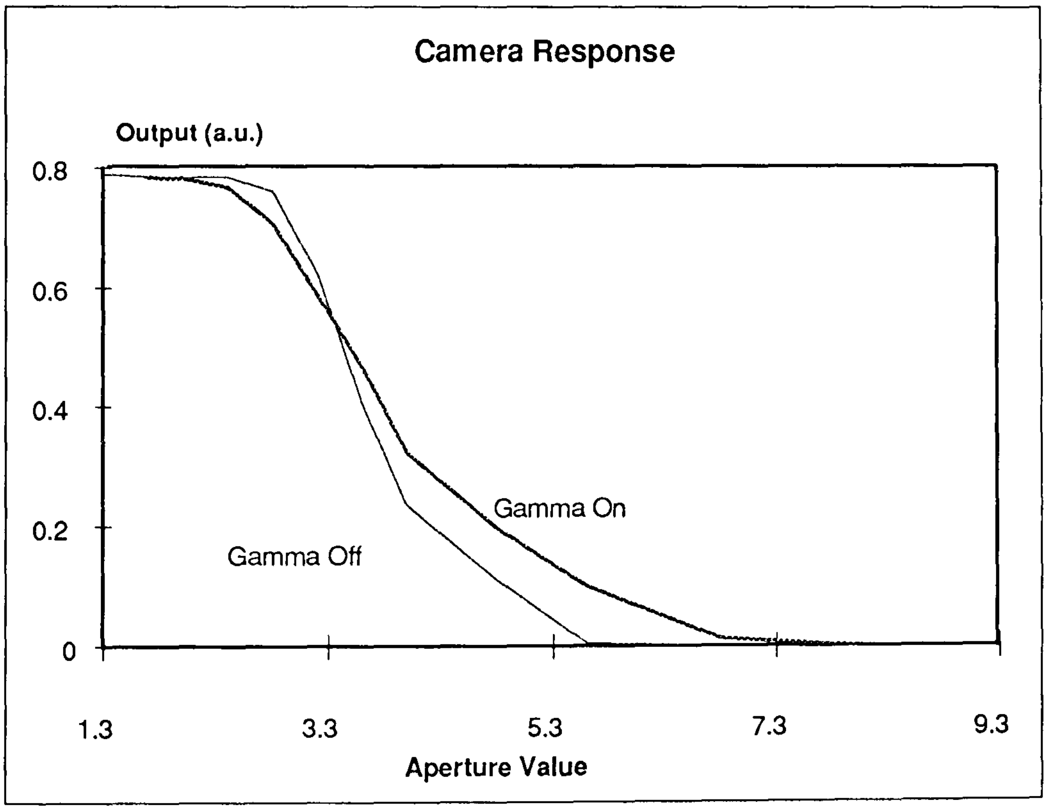


Figure 7.5. Graph showing effect of gamma compensation on output.

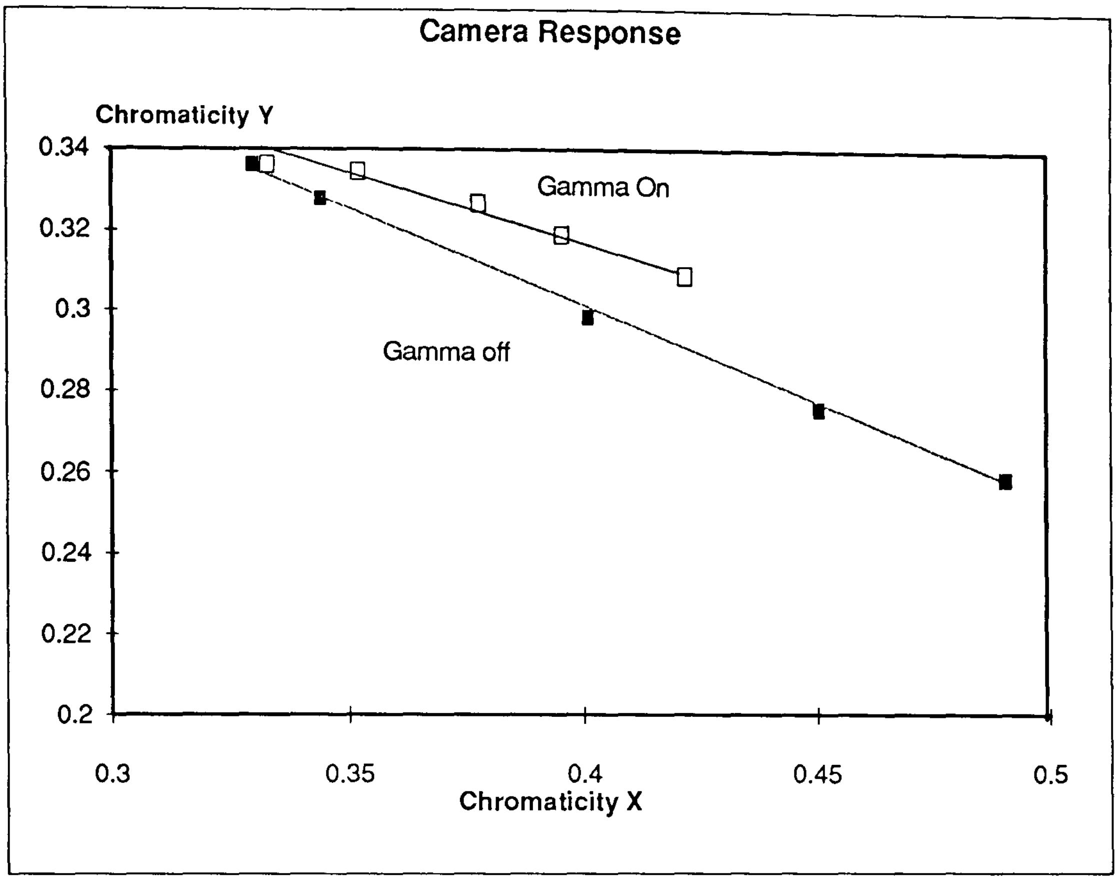


Figure 7.6 Graph showing chromatic change with aperture value for gamma on and off.

7.1.3. Dark Noise

Dark noise as explained in section 5.2.1, is evident when there is no input light incident on the camera. This noise level determines the minimum light intensity that can be monitored using the camera and so determines its chromaticity resolution. As this voltage level is below that which can be observed by the FGC the output of the camera was monitored directly with an oscilloscope. The experimental arrangement was described in section 6.2.1. The results show that the dark noise is in the order of 2 mV where the output range of the camera is 2V. This corresponds to a S/N ratio of 30dB thus suggesting that the camera has a maximum resolution of 10 bits. This shows that the camera signal output is better than the resolution of the FGC which is set to 8 bit resolution.

7.1.4. FGC ADC Tests.

The ADC tests considered the linearity and the effects of reference adjustments. The experimental methodology for these tests was described in section 6.2.2 and the results of linearity tests on the input ADC are shown in figure 7.7. This indicates that the input ADC exhibits good linear behaviour with input. The point of zero crossing is determined by the settings of the negative reference value and the maximum value is defined by the positive reference settings. The ADC produced corrupted output when the input voltage exceeded the value set by the positive threshold value.

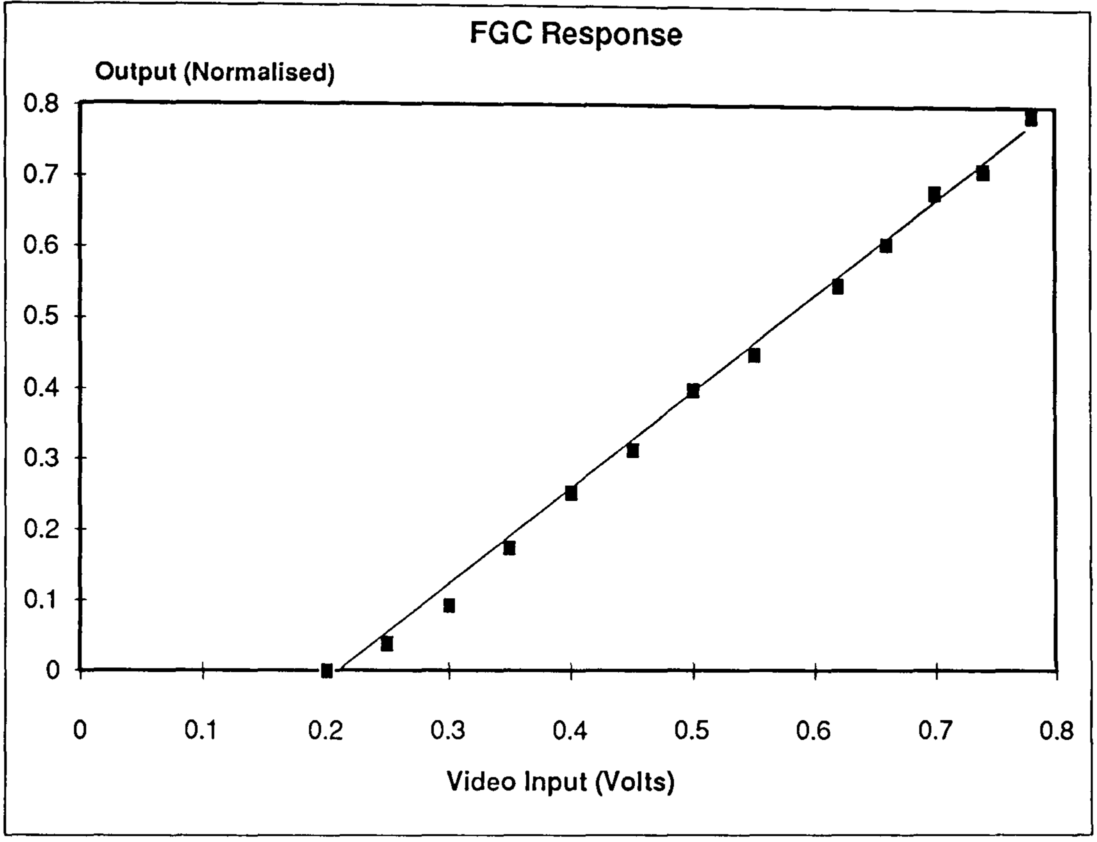


Figure 7.7 Graph showing output of FGC with input voltage (REF+=192, REF-=0)

Adjusting the negative and positive reference values set the input voltage range for the ADC. Figure 7.8 shows the normalised output as a function of input voltage with the reference value as parameter and varied in the range 0.341 to 1.95 V. The results show that as the reference value is reduced the input ADC gain is increased. The effect is noticed as a brightening of the output image from the FGC. However, it was also noted that using a low positive threshold value caused the image quality to be reduced. This is because noise becomes more significant as the difference between the positive and negative threshold values becomes sufficiently small.

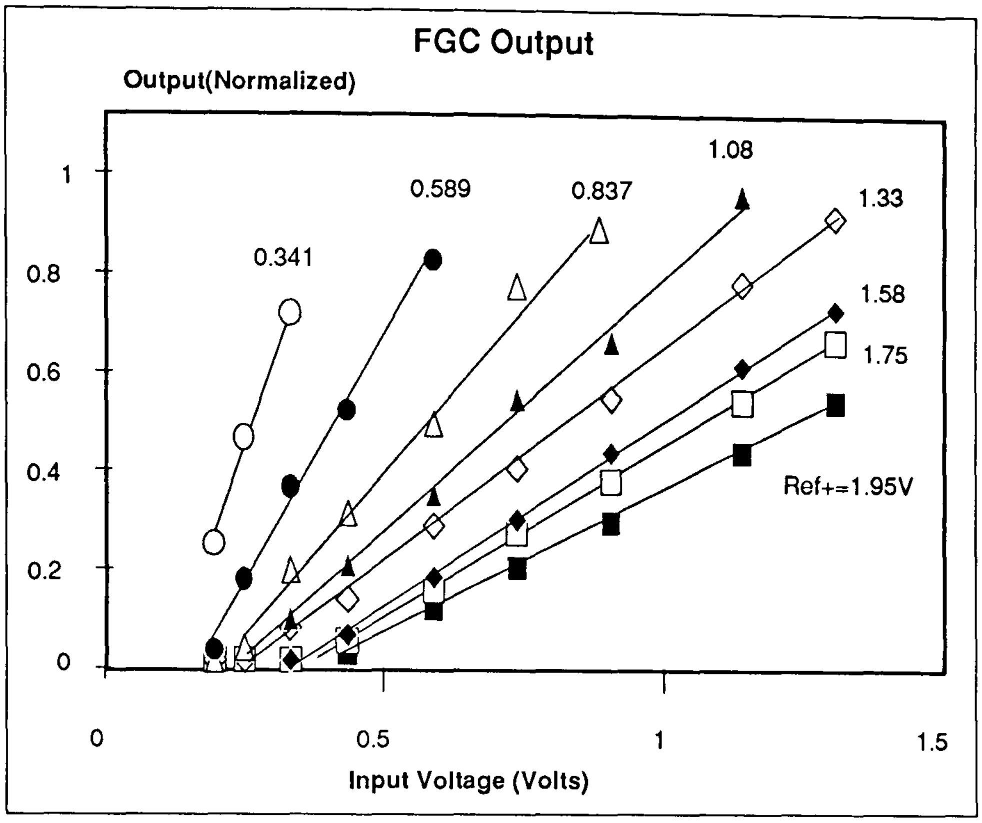


Figure 7.8 Graph showing ADC behaviour with changing Ref+

Figure 7.9 shows the normalised output as a function of input voltage with negative reference value as parameter in the range of 0.184 to 0.736. The results show that as the negative reference value is increased the position of the zero crossing is changed. Any voltage value below the negative threshold value appears as 0 at the output of the ADC. Increasing the negative reference value causes a darkening of the output image. Again it should be noted that the picture quality of the image is reduced when the difference between the positive and the negative threshold value is low. This is observed as "snow" on the output image.

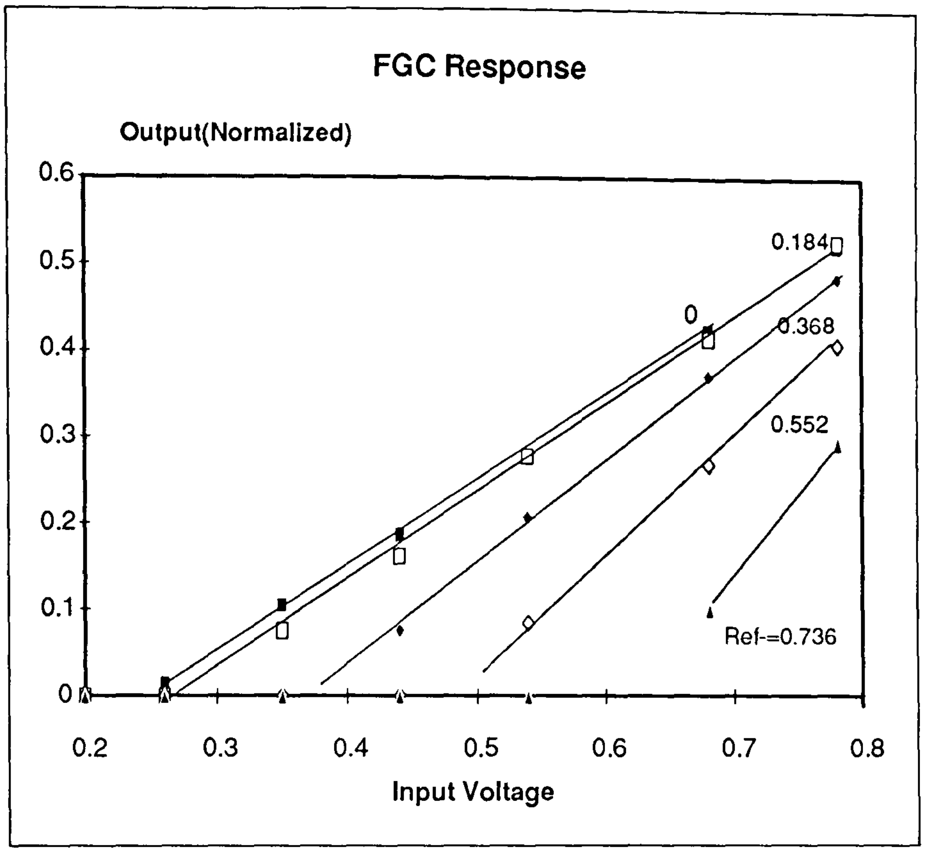


Figure 7.9 Graph showing ADC behaviour with ref- adjustment.

7.1.5 Long Term Drift Tests.

Long term drift tests were performed on the system as described in section 5.8. The result of a logging sequence over a 44 hour period is shown in figure 7.8. The results show that the system drift is small (~ 0.03 in 0.77, i.e. ~ 4%). There is a general decrease in intensity with time (this is possibly due to the light source degradation). but no daily periodic variations are apparent. A full quantification of drift on the system is difficult but for this experiment the effect of drift may be expressed as a S/N figure of around 14 dB.

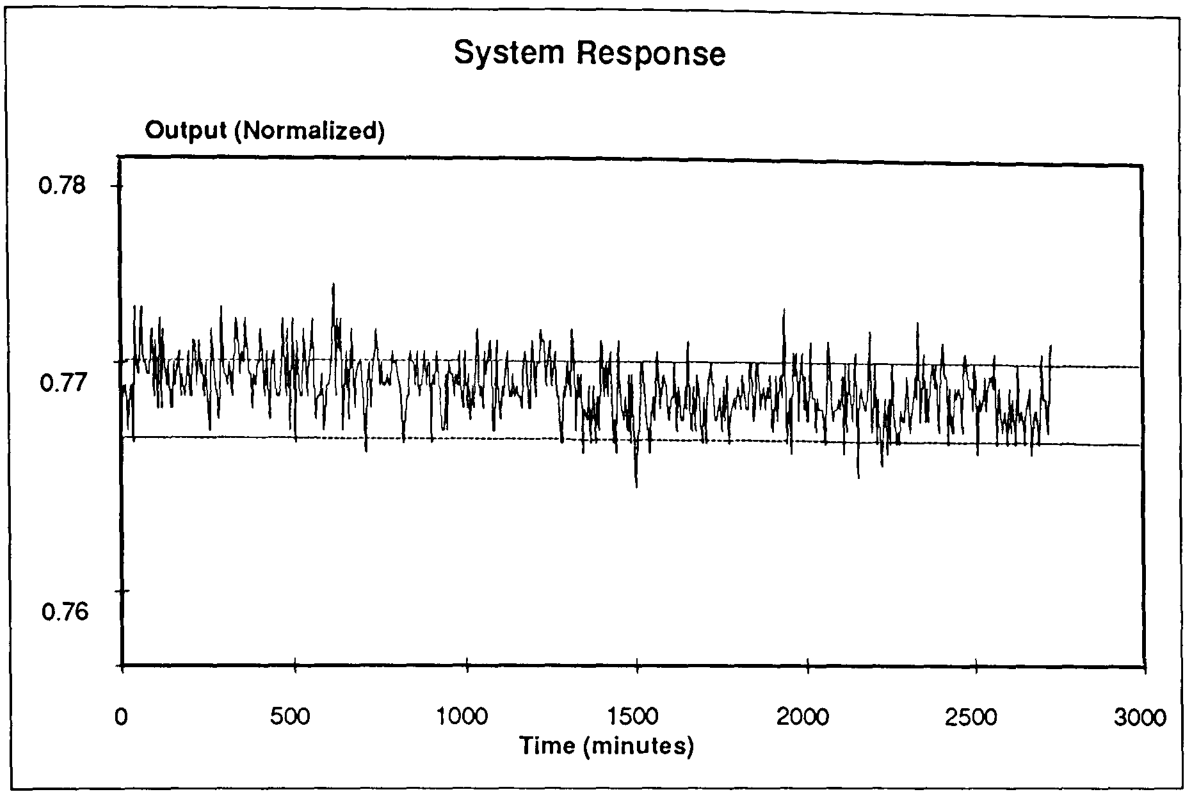


Figure 7.10 Graph showing long term drift over 44 hour data sampling period.

7.1.6 Glare Tests.

Glare tests for monitoring the effect of saturation of the image on the system (section 5.8) were performed using the apparatus shown in figure 5.8. The results of these experiments are shown in figure 7.9. The graph of each of the colour channels as intensity versus the angle of the test specimen with respect to the light source position, shows that when this angle is zero (i.e. the angle of incidence of the light on the sample is equal to the angle of observation of the image), the detectors are saturated giving maximum output values. The detectors remain saturated to an angle of 5^{o} away from the zero position decreasing gradually until the angle is increased to 15^{o} after which the detector outputs remain constant at a lower level.

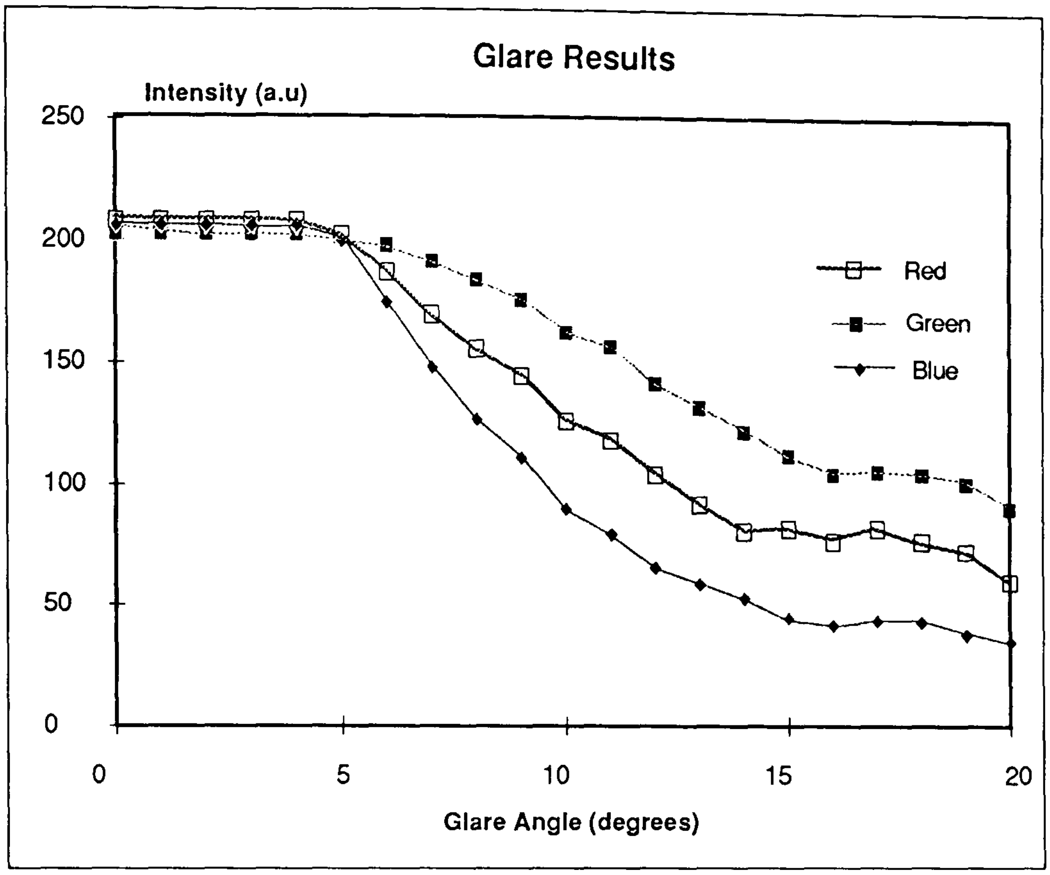


Figure 7.11 Graph showing variation of RGB intensities with glare angle.

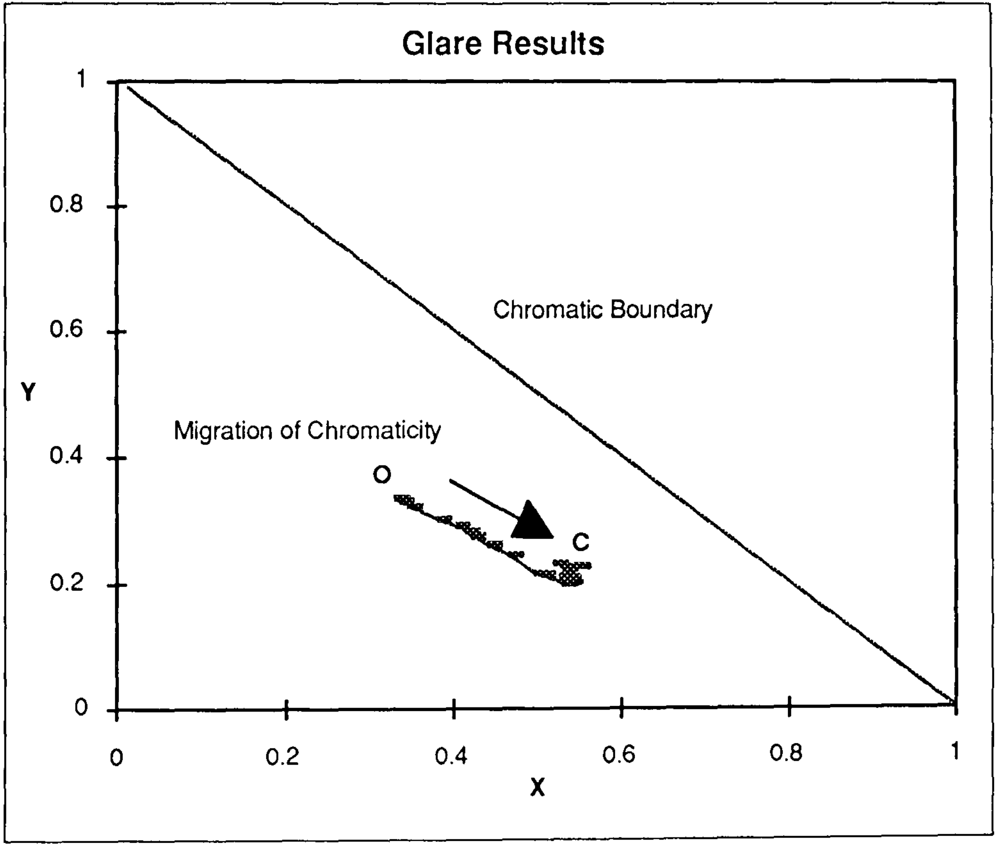


Figure 7.12. Graph showing variation of chromaticities X and Y with glare angle.

The effect of glare on a chromatic signature can be shown on a chromaticity diagram as illustrated on figure 7.10. Initially for angles between 0^o and 5^o the X and Y values lie close to the white light point, 0 (X= 0.33, Y = 0.33). As the glare angle is increased to 15^o the X and Y values migrate towards the point C (X=0.52, Y=0.22). Further increase in the angle causes negligible change in the chromaticities X and Y. C is therefore taken to be the true chromaticity of the image.

7.2 Thermochromic Calibration Experiments.

The thermochromic experiments were concerned with calibrating the thermochromic strip to act as a temperature sensor. Thus the calibration experiments were required to relate the chromaticity of the thermochromic material to temperature using the apparatus described in section 5.8.2 and the experimental methodology described in section 6.3.1. The calibration curves obtained from these experiments are shown in figure 7.11 in the form of the normalised output from each of the RGB pixels as a function of temperature. The results show that none of the detector outputs vary monotonically with temperature, so that a simple reliance upon the intensity variation of any channel as a temperature indicator would lead to ambiguous readings. The chromaticity of the sample varies with temperature in the order from blue to red, going back to blue. The peak red intensity is observed at a temperature of 30^{O}C and the green intensity peak is observed at 50^{O}C. This normalised input was then used as an input to a neural network for further processing.

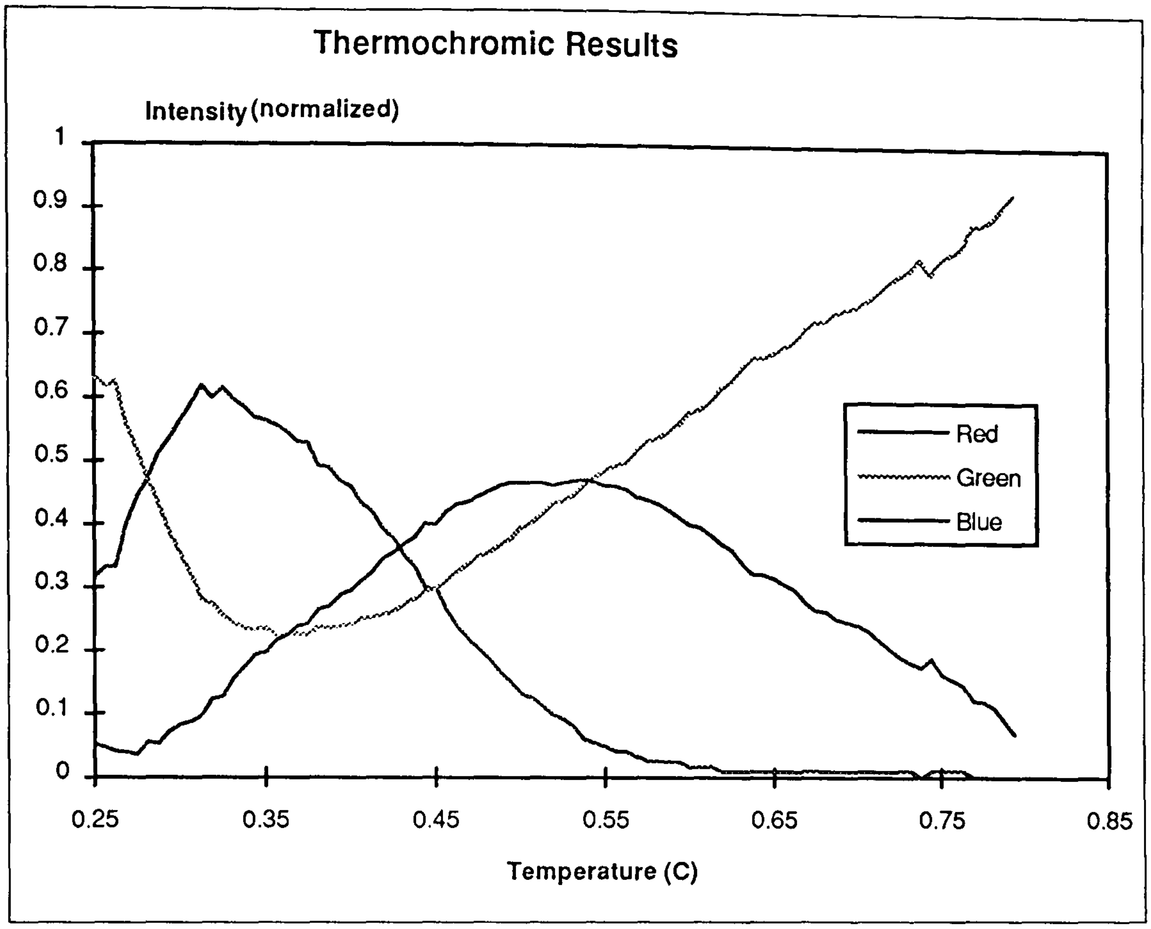


Figure 7.13. Graph showing variation in normalised RGB intensities with temperature for a thermochromic strip

7.2.1 Neural Network Results with Normalised Thermochromic Data.

The calibration results obtained from the thermochromic calibration tests (section 7.2) were used to test the performance of a number of neural networks to perform the mapping between normalised RGB inputs and temperature.

Neural networks were simulated and tested using a NeuralWorks™ package that is commercially available. The networks that were simulated were fully connected MLP using Standard Error Back Propagation Algorithms for learning (section 4.2.2).

Tests to observe mapping accuracy and generalisation ability of a network with normalised data was performed using a network of architecture 3:20:5:1. A data set containing 80 samples were used for training consisting of normalised RGB and the corresponding temperature values. A data set containing 20 data sets was retained to test the performance of the network with unseen data.

The results after a 12 hour training period (8 million iterations) of the neural network with the training data set are shown in figure 7.12.

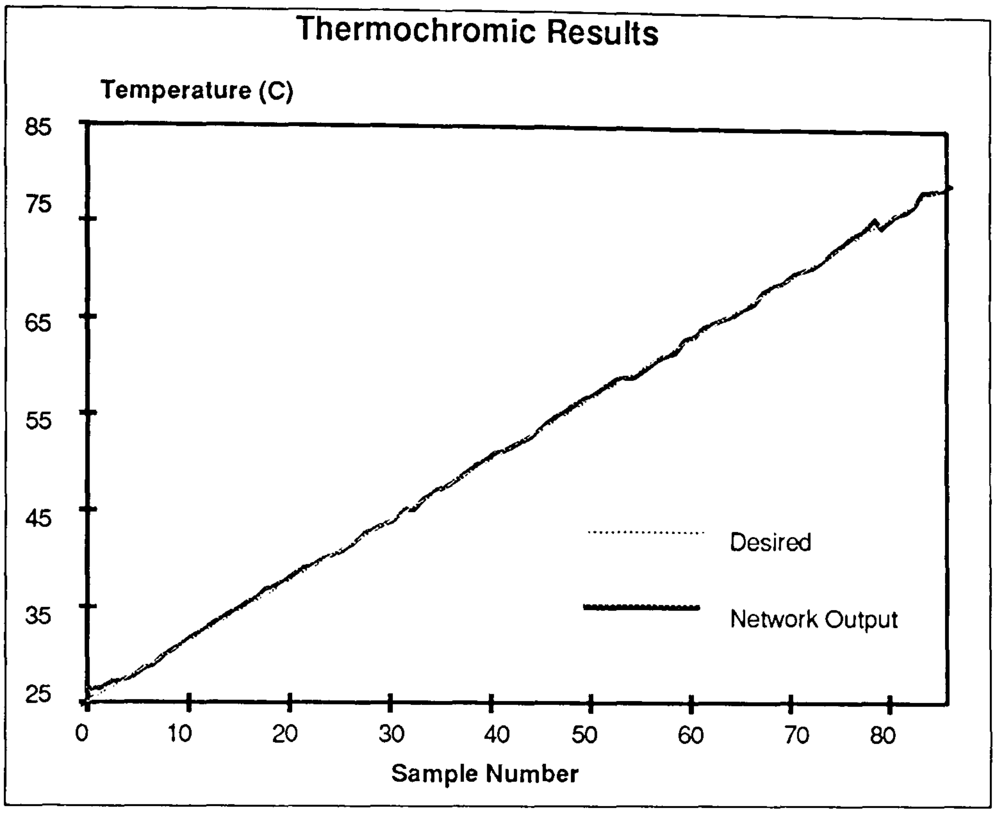


Figure 7.14. Graph showing 3,20,5,1 network output with normalised thermochromic data after 12 hours training.

The graph shows a good correlation between the desired output and the network output. The error between the desired and network output can be clearly seen in figure 7.13.

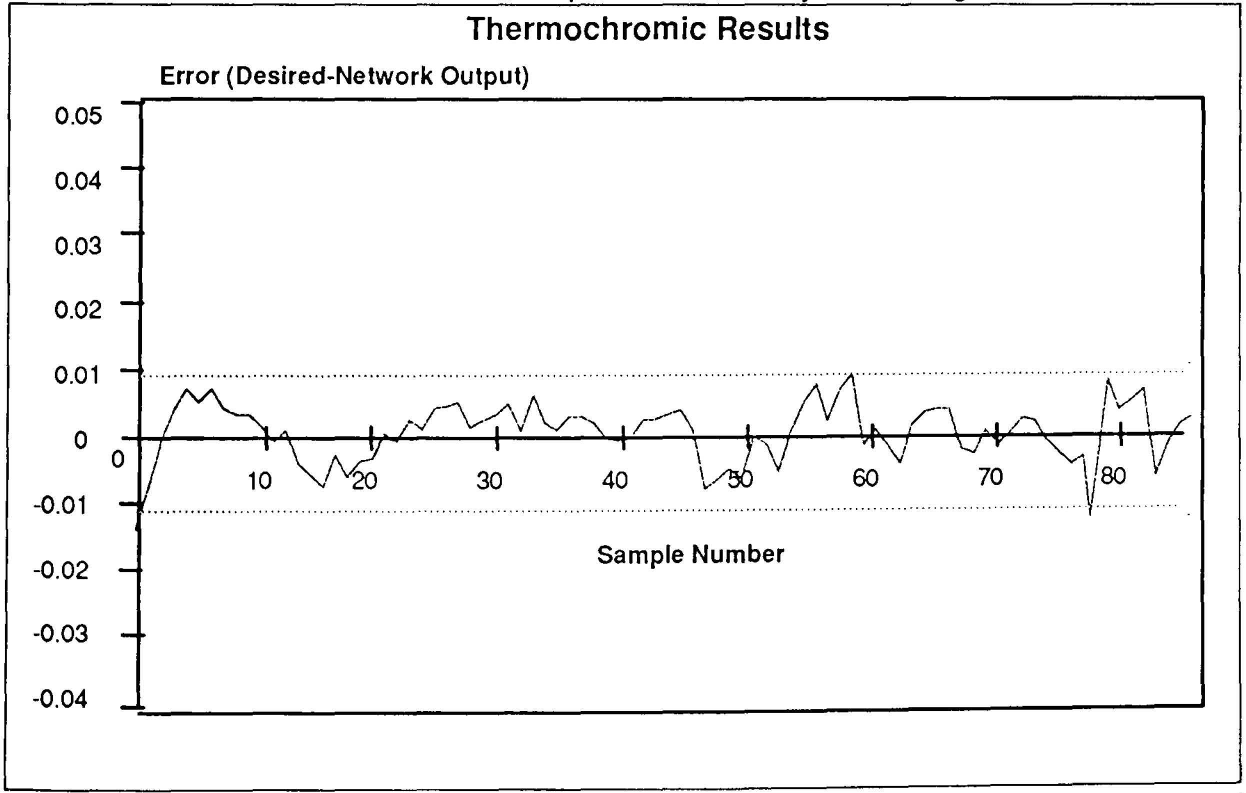


Figure 7.15. Graph showing 3,20,5,1 network error with normalised thermochromic data after 12 hours training.

The graph shows that there is an RMS error between desired and network output of 0.04% after 12 hours learning. With unseen data this network produced an RMS error less than 0.9%.

The effect of noise on the input data on the network performance was investigated by considering a smoothed data set. This smoothing was achieved by using a fifth order polynomial fit of the data. The polynomial fitting was performed by using Cricket Graph to generate the polynomial fit. The result of the fifth order polynomial fit is shown in figure 7.14. This graph compares well with the normalised thermochromic characteristics as shown in figure 7.13.

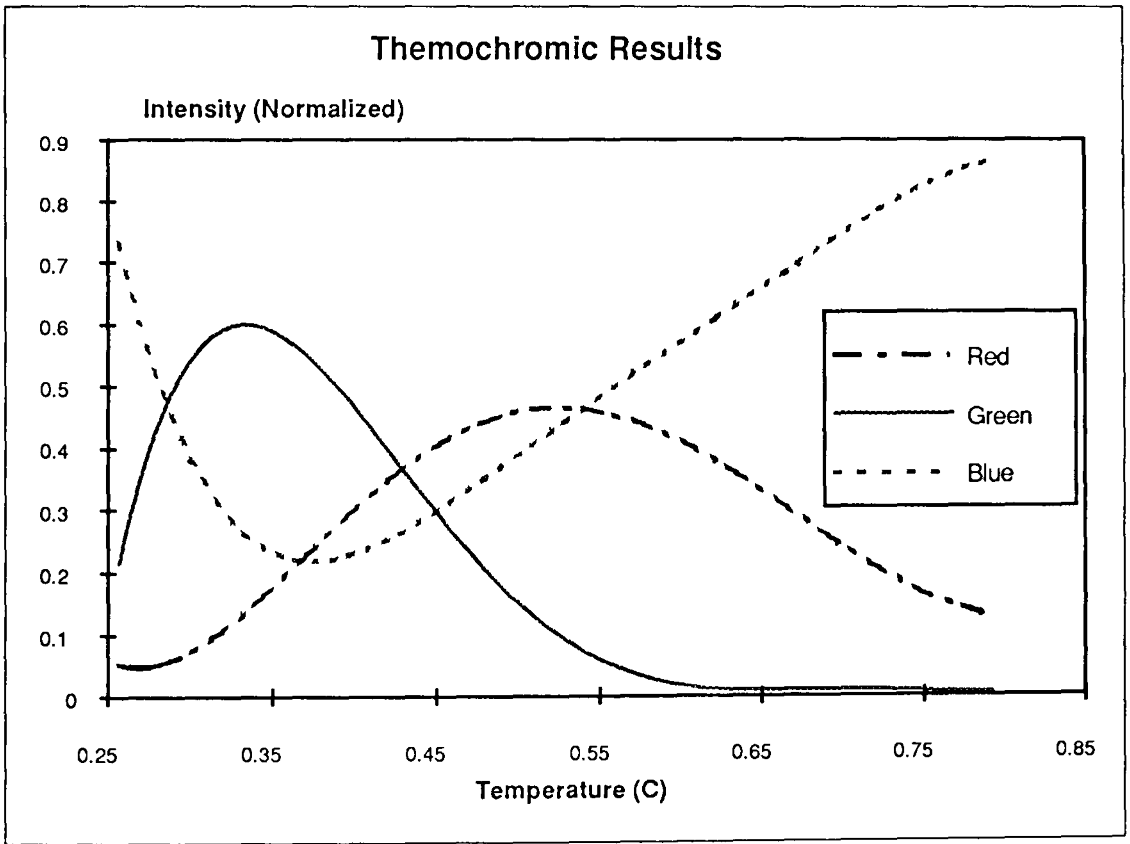


Figure 7.16 Graph showing effect of using fifth order polynomial on data set to reduce effect of noise.

The result of using an identical network architecture (3,20,5,1) and trained for the same period of time but using data generated using the polynomial smoothed data is shown in figure 7.15 and 7.16

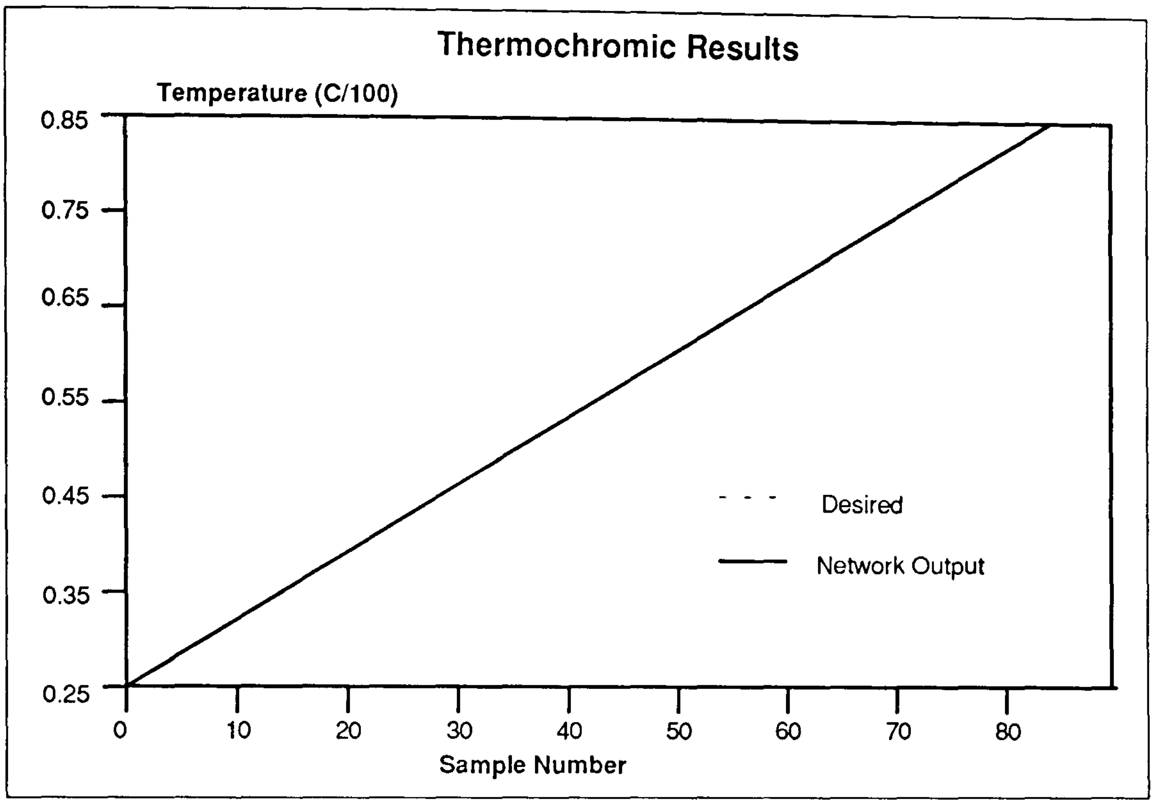


Figure 7.17 Result of using a 3,20,5,1 network with polynomial smoothed data after 12 hour iterations.

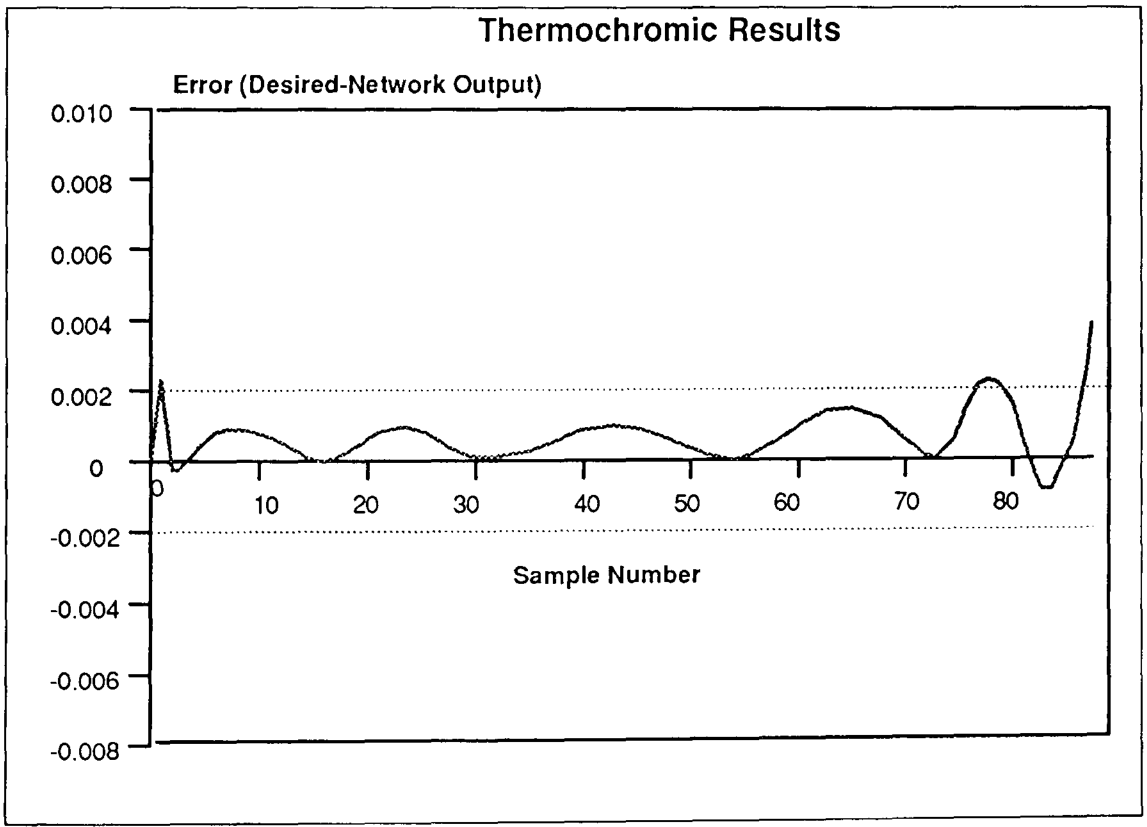


Figure 7.18. Graph showing error between desired and network output for a 3,20,5,1 network trained for 12 hours.

The graph shows that the error after using smoothed training data set is significantly smaller than those observed with normalised RGB data. Here the RMS error between the desired and network output was 0.01% and with unseen data the RMS error was 0.2%.

The effect of using the two different data sets (Normalised and smoothed polynomial fit) data on the learning rate can be seen in figure 7.17. This graph shows the RMS accuracy of the neural network at different stages of the learning process. After 1 million iterations the network trained with normalised data showed exhibited an RMS error with trained data of 0.058% while a similar network trained with polynomial data achieved an RMS error of 0.02%. This RMS error reduced logritmically with number of iterations going down to 0.04% and 0.01% after 8 million iterations for the normalised and polynomial data respectively. If this result was interpolated it suggests that a network trained for 80 million iterations using normalised data would achieve the same RMS accuracy as a network trained for 1 million iterations using polynomial data.

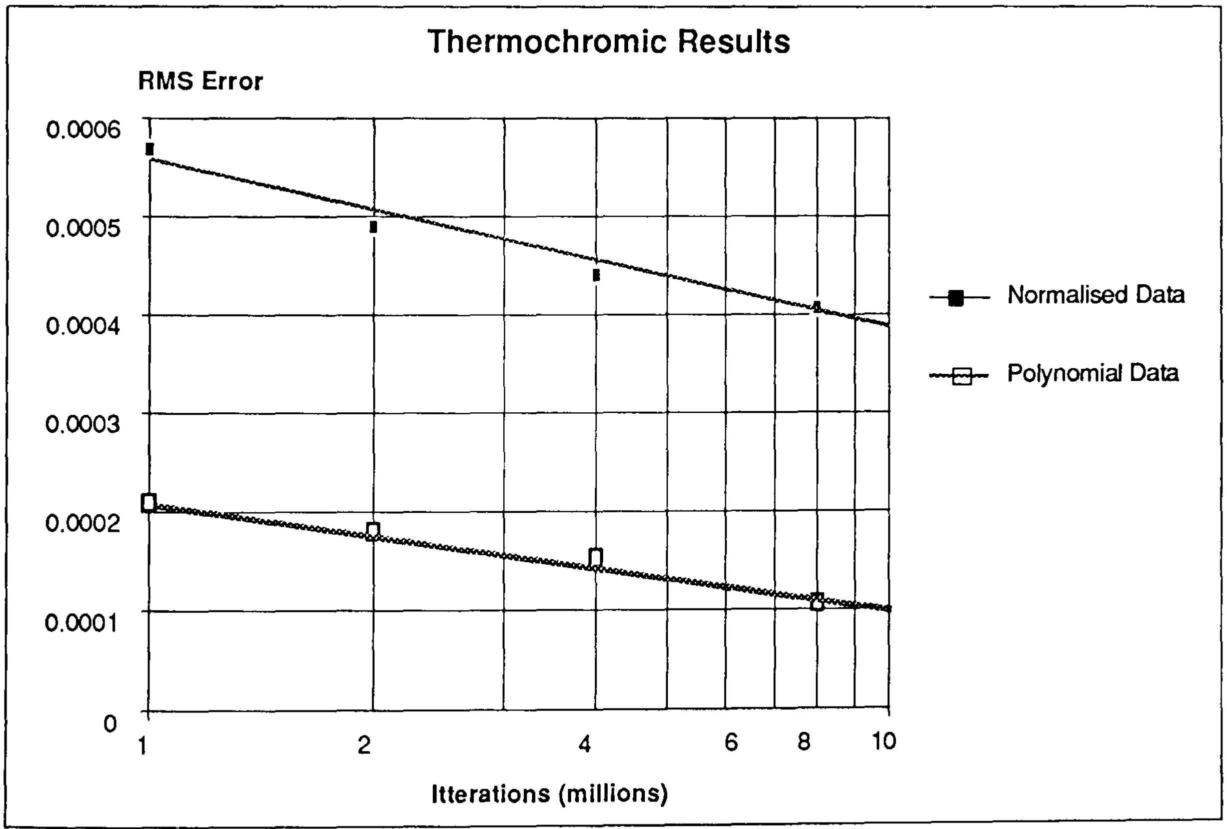


Figure 7.19 showing RMS error change with number of iterations for 3,20,5,1 networks trained with Normalised and polynomial data.

To check the generalisation ability of networks trained with polynomial data to map to real temperature data, networks that had been trained with polynomial data were tested with normalised data. The performance was considered at various stages of the learning process. The results of this experiment are shown in figure 7.18.

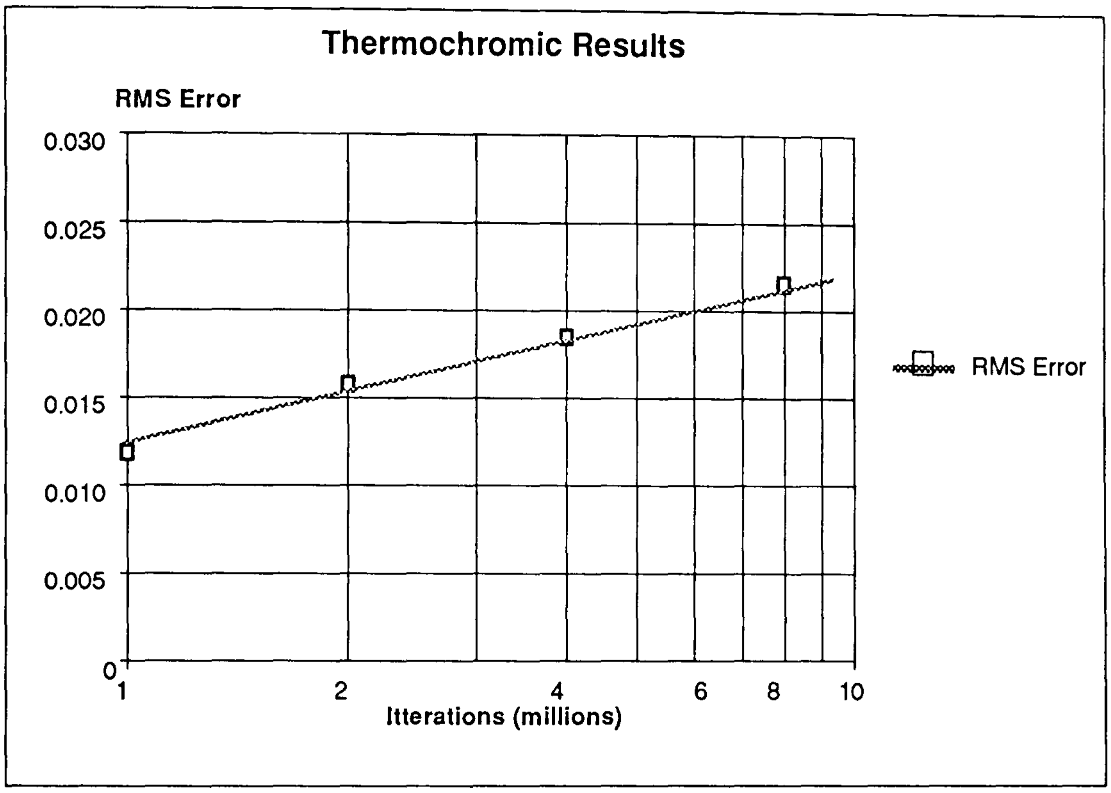


Figure 7.20 Graph showing RMS error of 3,20,5,1 network trained with polynomial data and tested using normalised thermochromic data.

The RMS error between the polynomial data set and the normalised data set is 2.9%. The result shown in figure 7.20 shows that as the number of iterations is increased from 1 to 8 million the RMS error of the network trained with polynomial data and presented with normalised data increases from 1.3% to 2.1%. It is expected that further training will make the RMS error tend towards the 2.9% value which is the RMS error between the polynomial and normalised data sets. It also shows that increased training reduces generalisation capability of the network.

7.3. Photo elastic Strain Sensing.

7.3.1. Single Point Analysis

The single point photoelastic strain sensor was calibrated with the system shown on figure 5.10 and the results are shown on figure 7.21 as the intensity variations of each of the three chromatic signals as a function of strain. The results show that as anticipated on the basis of theory (section) the variations in the output from each channel is cyclic in nature with the cycle period increasing as the dominant wavelength of the chromatic channel increases (i.e. in the order blue, green and red). The strain period of the red green and blue channels are 550, 450 and 350 microstrains respectively.

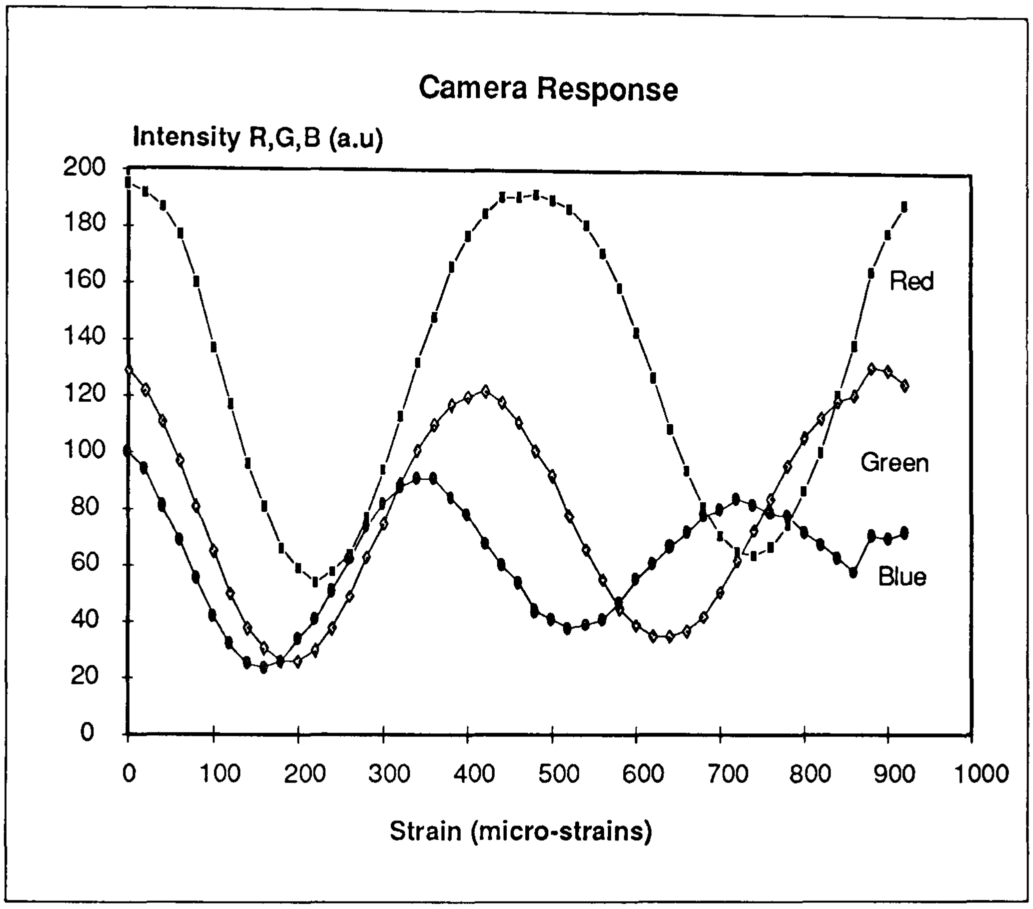


Figure 7.21 Graph showing variation of RGB channels with strain for single point analysis.

The resulting x chromaticity values, calculated from the individual RGB results, are shown in figure 7.22. The results show that the chromaticity x variation is also cyclic in nature and the period of the wave form is 760 nm. The amplitude of the chromaticity variation is observed to increase as the stress increases changing from approximately 0.15 to 0.24. A similar response is also observed for the chromaticity y plot. Figure 7.19 shows these changes on a 2 dimensional chromaticity diagram.

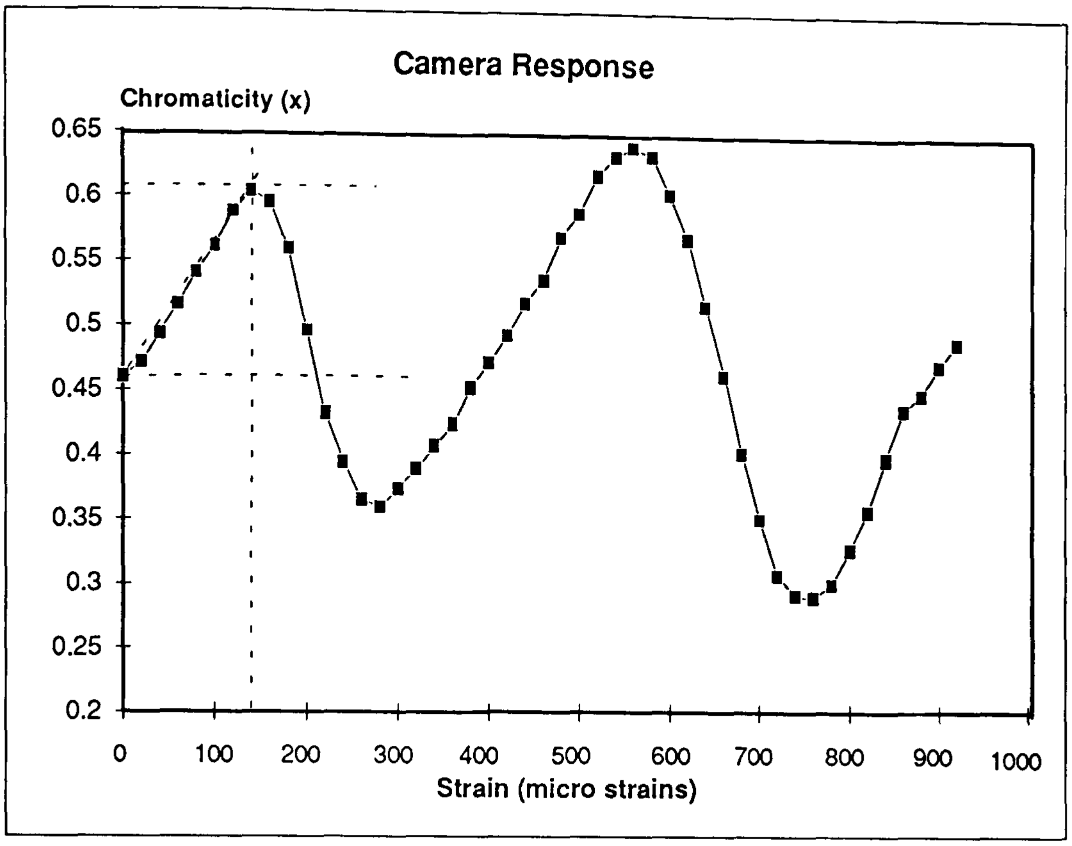


Figure 7.22 Graph showing Chromaticity X variation with Strain.

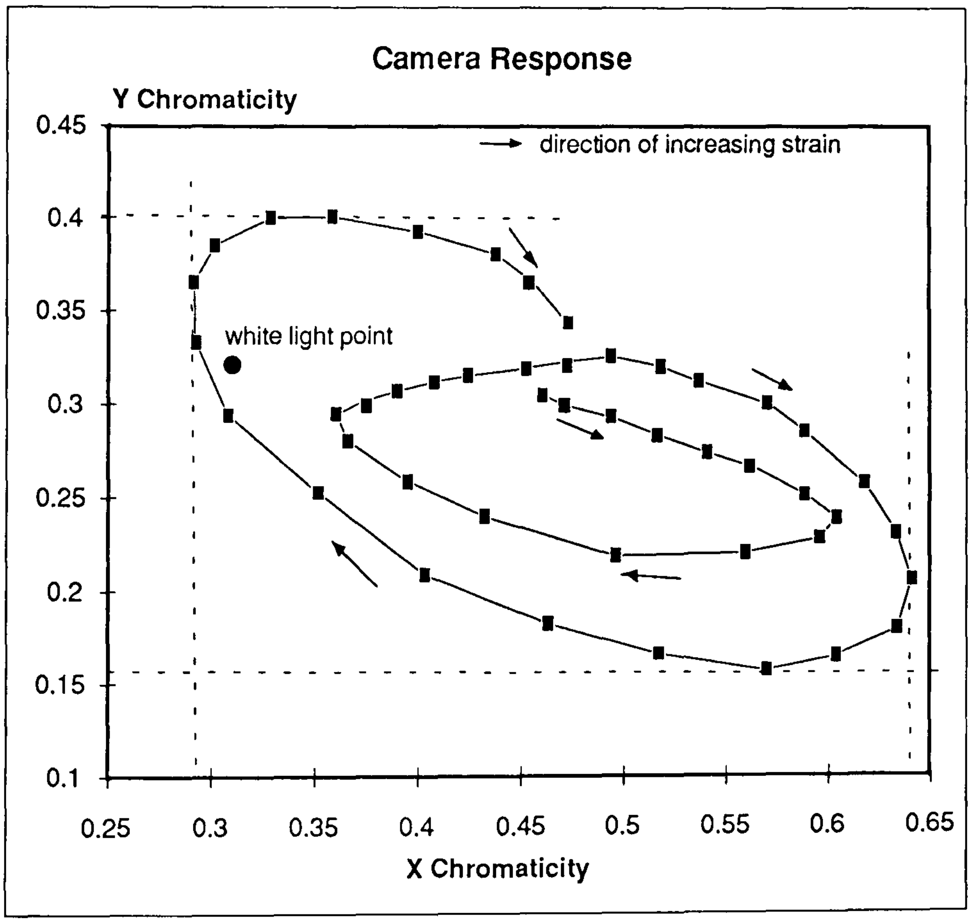


Figure 7.23 Graph showing chromaticity plot for incremental strain.

The results show that there is a modulation depth of .35 for each of the x and y chromatic coordinates. The range of strain measured was between 0 and 950 microstrains. Because the coordinates locus is of a spiral nature and for the range measured there is no intersection of the strain values the full strain range can be utilised by the use of look up tables. The resolution of such a system would be better than 20 microstrains i.e. approx. 2%.

7.3.2. 2 Dimensional Analysis

Two dimensional stress monitoring was tested on the barometric photoelastic diaphragm and the CCD Camera described in section 5.8.2. Images of the circular diaphragm at various pressures were collected for processing. The diaphragm image was then divided into concentric annular regions for chromatic calculations (section 6.3.2). A sequence of frames showing the diaphragm at different pressures is shown in figure 7.24. Also shown on this figure is the method used to obtain the annuli on the diaphragm for processing the information. The mean x and y chromaticity values for the resulting interference rings are shown in figures 7.20 and 7.21. These results show that the mean chromaticity values for the rings are complex in nature and that they have unequal periodicities. The modulation depth of the chromatic variation is 0.2 and 0.15 for the x and y chromaticities respectively. The difference in the periodicities suggests that there is significantly different chromatic activity in the mean values of the different rings of the sensor. Clearly, correlation of these points with stress cannot be performed using simple techniques such as Look Up Tables and Algorithmic methods. However, neural networks are capable of handling such complex waveforms and if suitably trained could provide a method of pattern extraction to obtain the desired calibration.

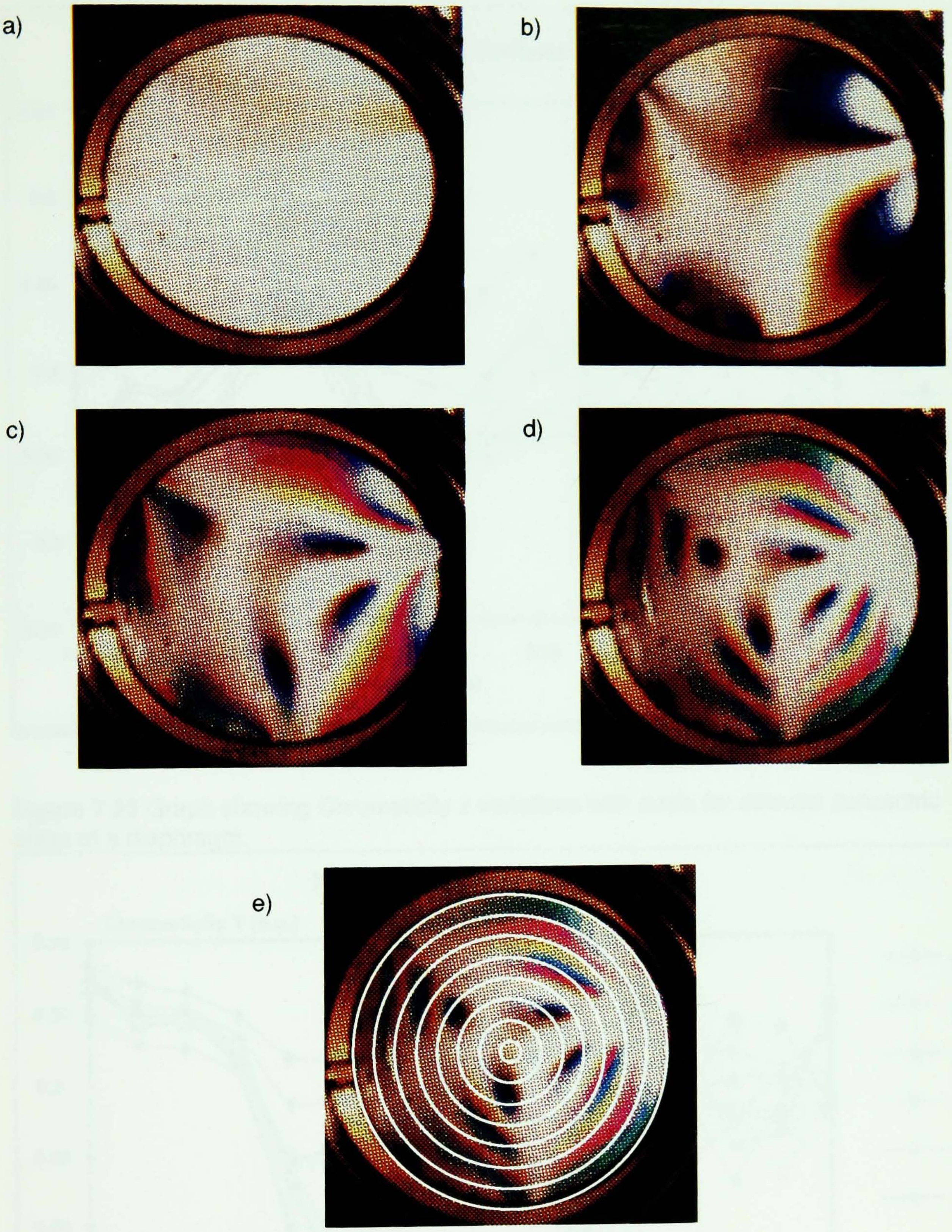


Figure 7.24. Images of Photoelastic Daiphragm at various pressures: a) 0 lb/in^2 b) 20 lb/in^2 c) 40 lb/in^2 d) 60 lb/in^2 e) Diagram showing how 2D analysis analysis is performed using ringed regions of the diaphragm.

(Note that in the monitoring system 10 rings of equal area were used)

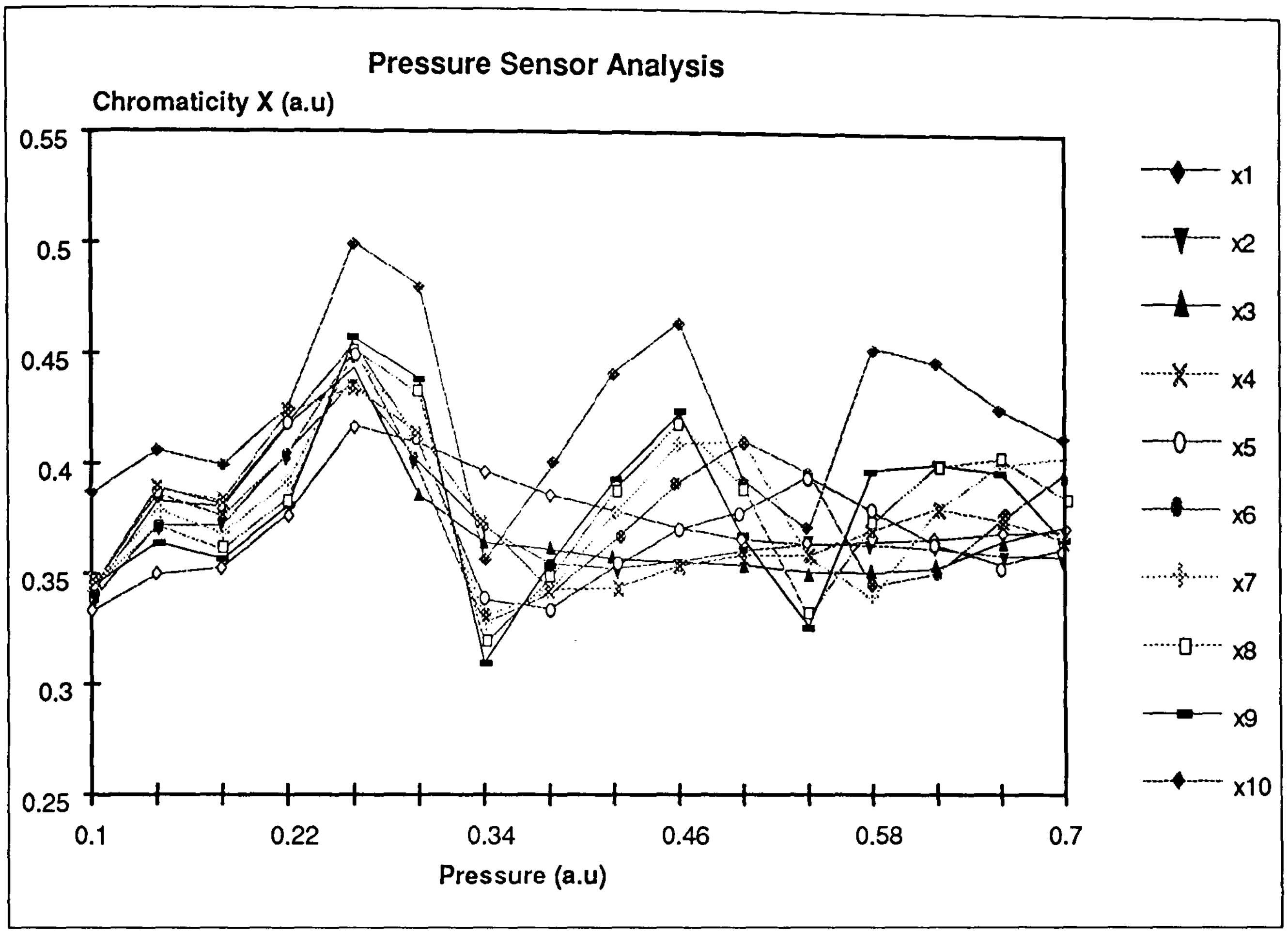


Figure 7.25 Graph showing Chromaticity x variations with strain for different concentric areas of a diaphragm.

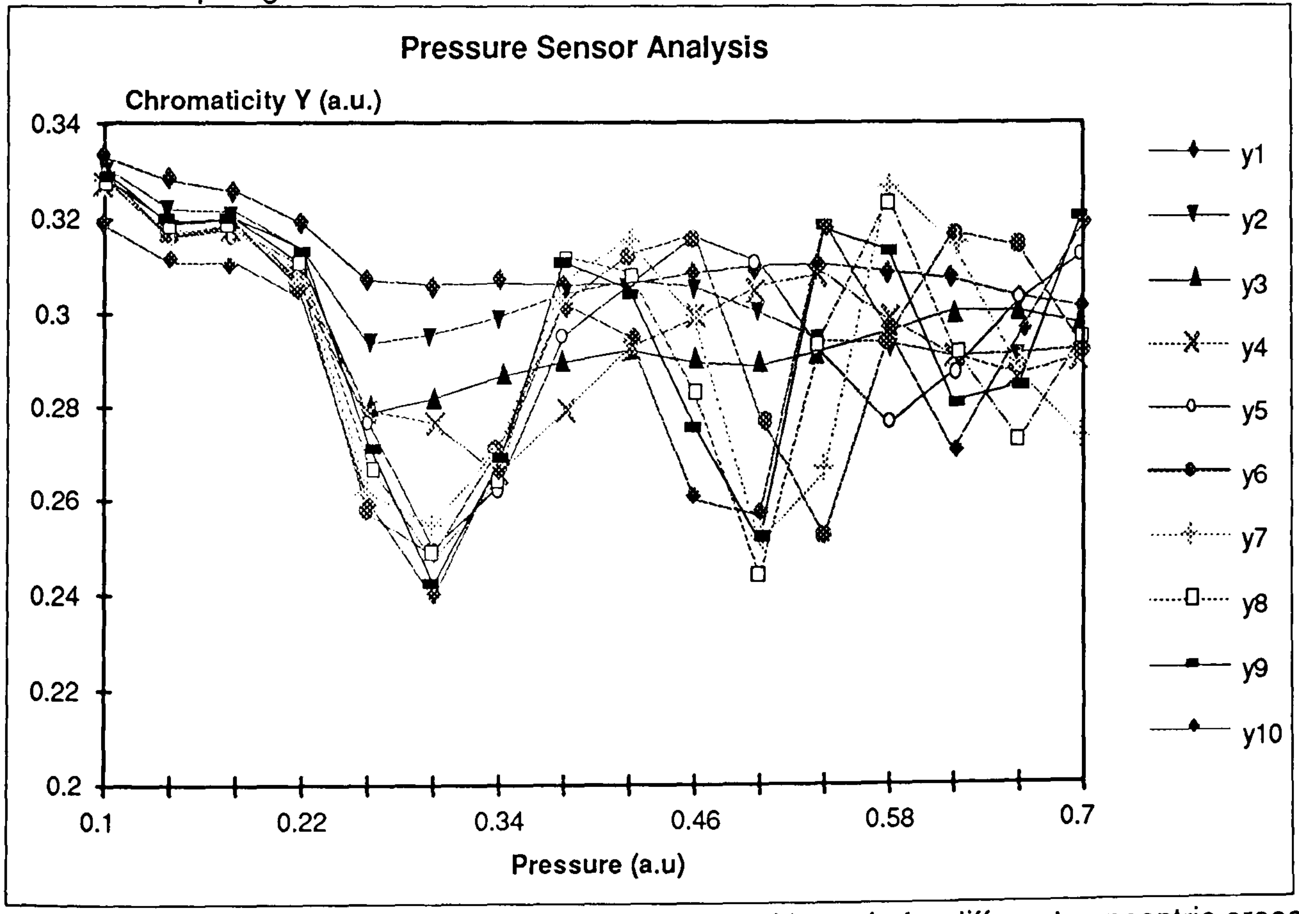


Figure 7.26 Graph showing Chromaticity y variation with strain for different concentric areas of a diaphragm.

7.3.3. Neural Network Training Results.

Neural Networks have been trained to map the mean chromatic values from the various rings obtained from the 2D diaphragm calibration experiments (section 7.3.2 above). A 20:20:10:1 architecture neural network was trained with chromatic data obtained from 15 barometer images at various pressures and taught to map the chromaticity x and y values with pressure. The network performances after 24 and 48 hours were obtained.

Figure 7.22 shows the network performance after 24 hours training in terms of a plot of pressure against sample number. This graph compares the network output with the desired target values. The error between the desired and network output is shown in figure 7.23. The results show that the network is capable of learning the data and that the accuracy of the system with the training data set was 0.17% (mean squared error). The error graph (figure 7.27) shows that if the first two data points are ignored then the error would be significantly reduced to 0.05%. This poor learning at initial values is due to the fact that the variation in the chromaticity in the initial stages of the stress sensing is small and thus the system is unable to extract useful information from the data.

The trained network was then tested with 5 data values that were not used in the training phase of the network to check the generalisation capability of the system.

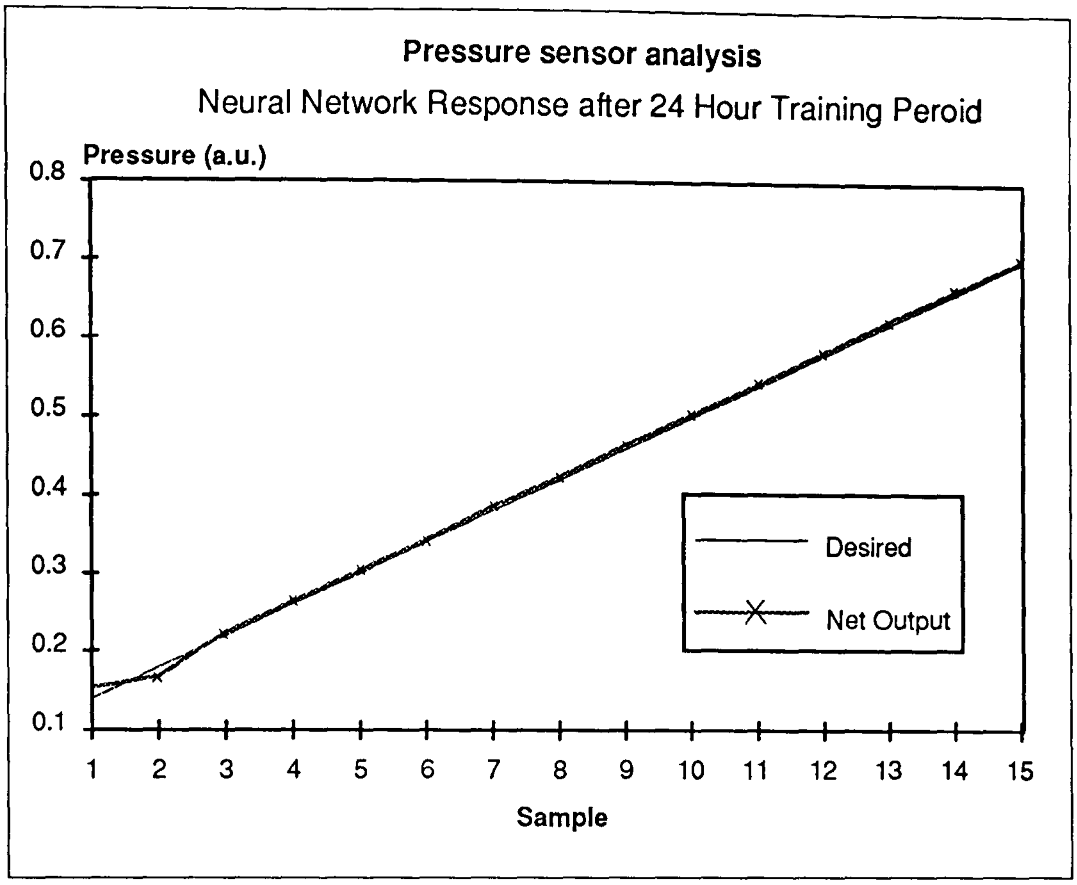


Figure 7.27. Graph Showing Network response to training data for pressure analysis.

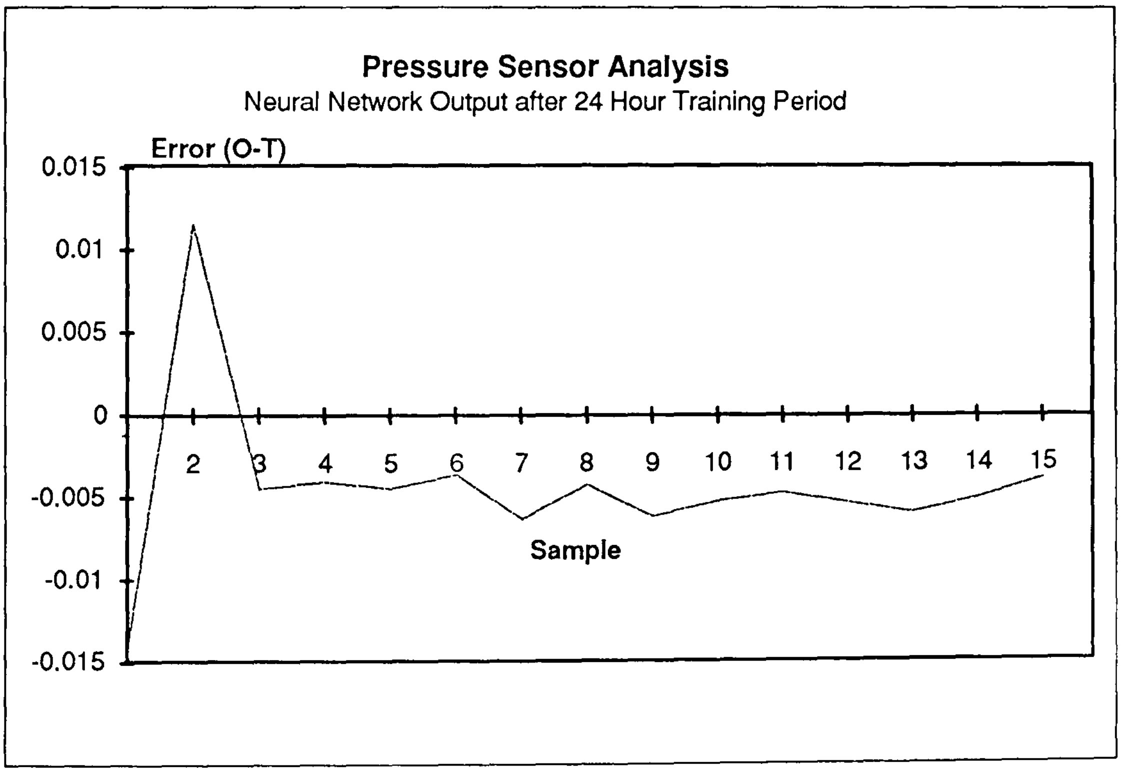


Figure 7.28 Error between desired and Network output after 24 hours training.

The results of the tests with unseen data are shown in figure 7.24. These results show that after a 24 hour training period the generalisation ability of the system is good. However the network is unlikely to achieve the same accuracy as that obtained with trained data. The mean squared error for these 5 samples shows an accuracy of 1.38%.

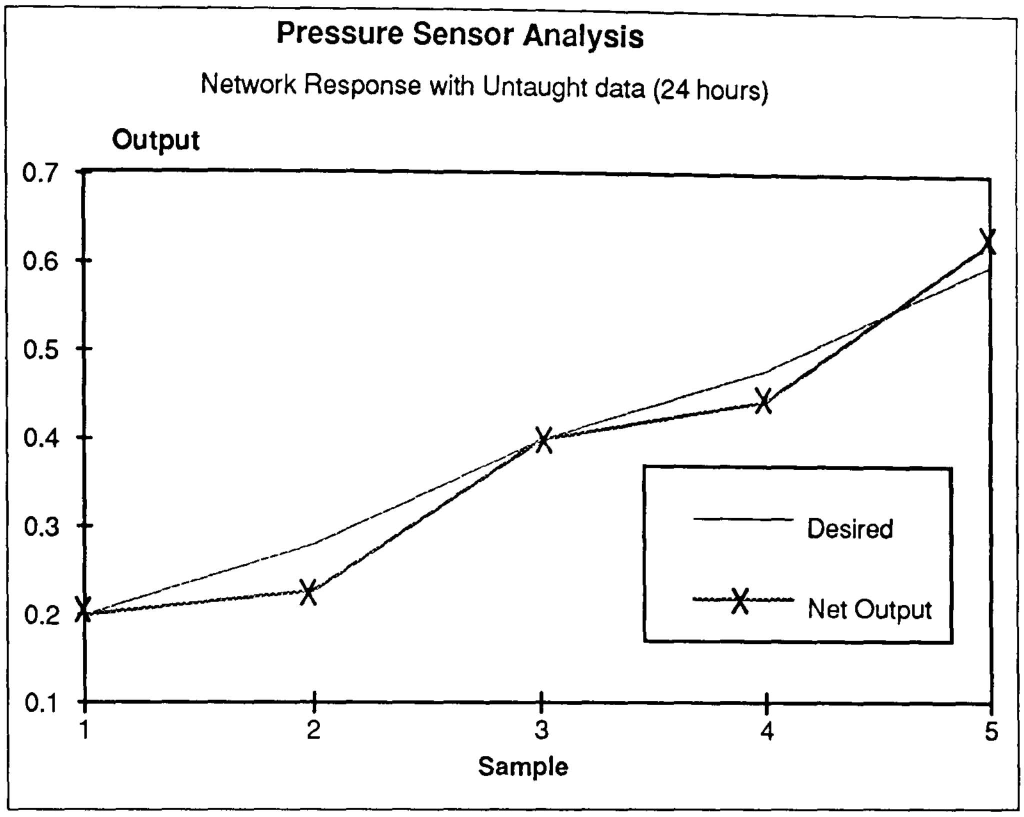


Figure 7.29. Graph showing desired and Network output with unseen data after the network had been taught for 24 hours.

The network was trained for a further 24 hours to examine the effect of further training on the performance with training data and unseen data. Figure 7.25 shows the network response after a 48 hour training period. The graph shows that the mapping with training data has improved with further learning time. The mean squared error here is observed to be 0.05% and again if the first two points are ignored then the mean squared error is reduced to less than 0.01%. The figure 7.26 illustrates the error between the desired and network output. The features that are observed in the figure 7.28 are also apparent in this graph suggesting that the errors generated are more a fundamental effect caused by poor data set than network performance.

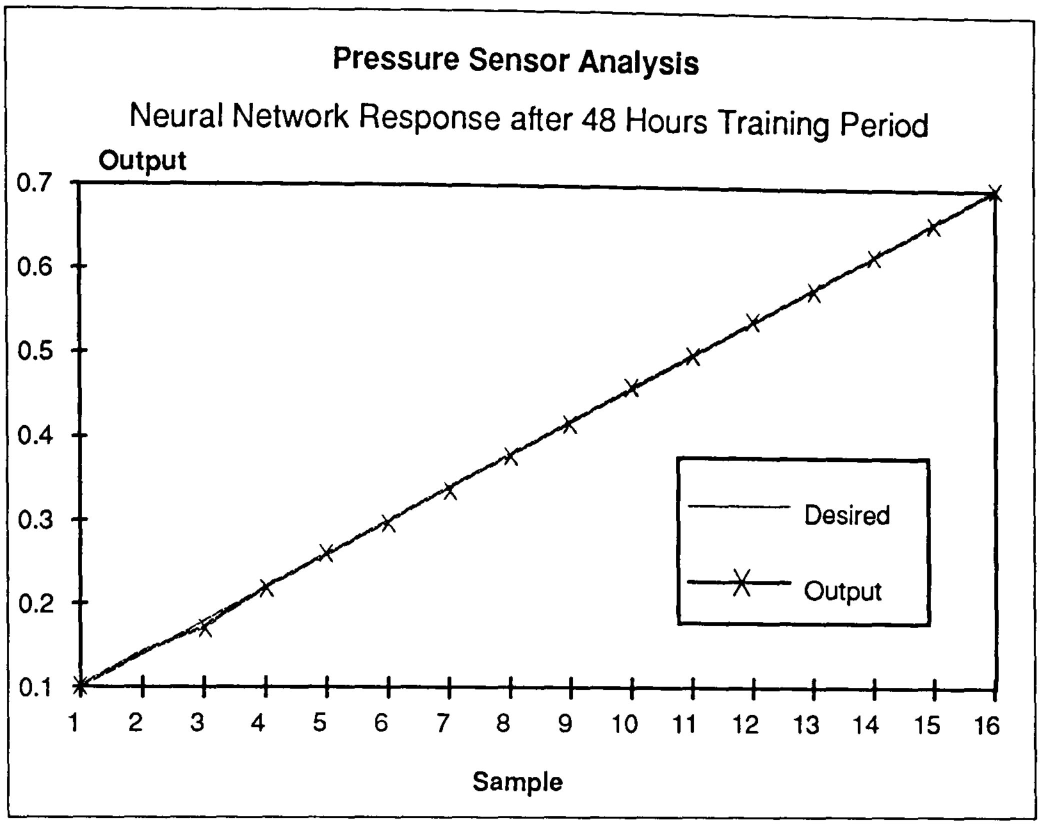


Figure 7.30. Network response to training data for pressure analysis after 48 hour training.

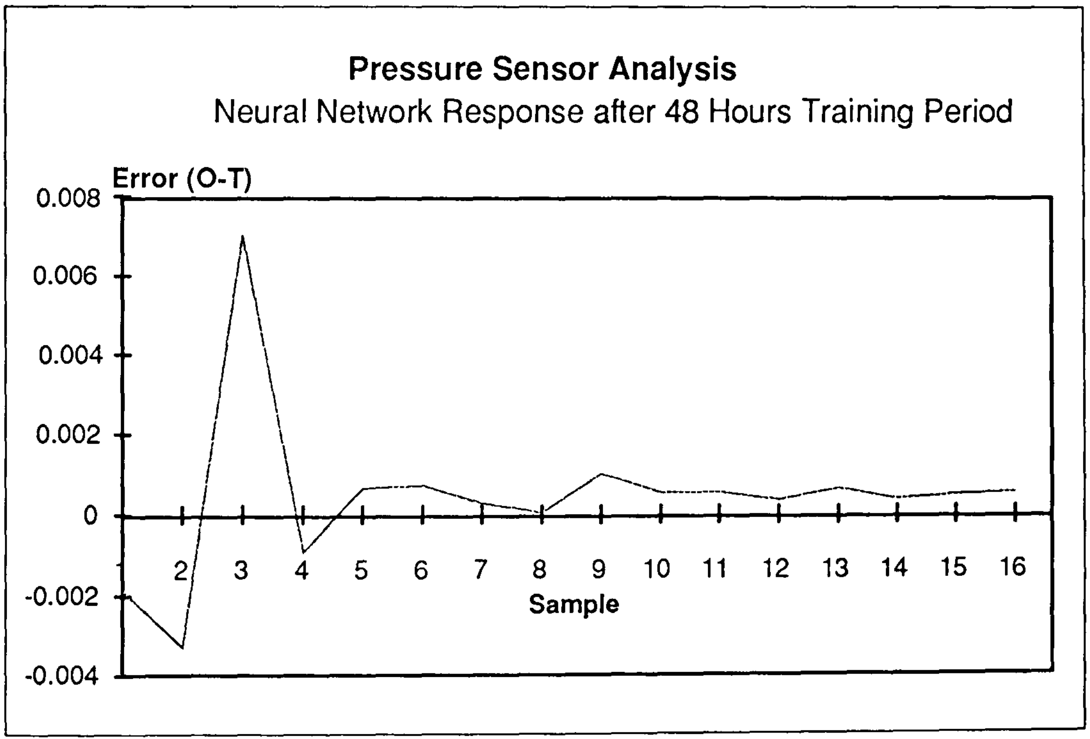


Figure 7.31. Error between desired and Network output after a 48 hour training period.

With unseen data the same network produces errors of 0.17% (Figure 7.27). The results show similar characteristics to those obtained after 24 hours training and apparently further training will not change the error characteristic curve appreciably. Thus the generalisation ability is not significantly affected by the increased learning period.

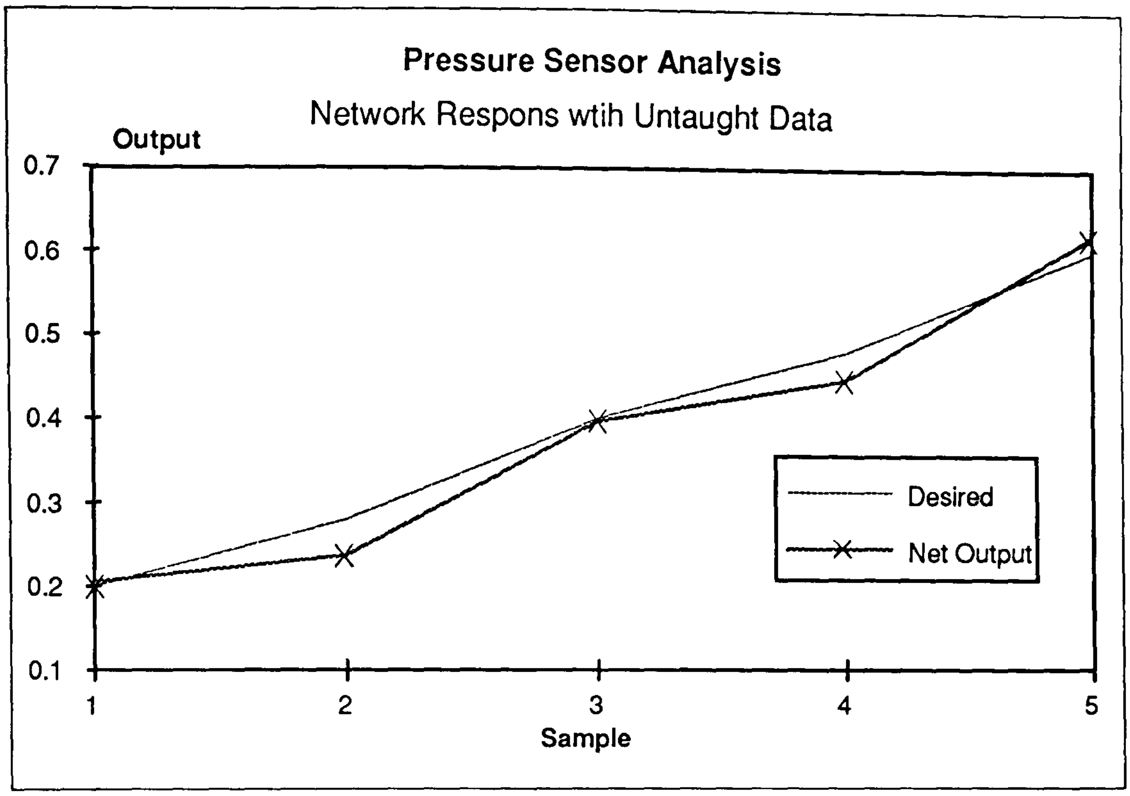


Figure 7.32. Graph showing desired and network output with unseen data after the network had been taught for 48 hours.

To determine the extent to which network size affects network learning and generalisation ability a second network was implemented using an architecture of 20:40:10:1. The network was trained for a 24 hour period. Note however that network size does affect the iteration time for a single learning pass (Thus after 24 hours of learning for a network of size 20:40:10:1 the number of iterations will be on average half that of a 20:20:10:1 network). The results obtained from this network are shown in figure 7.28. The results show a mean squared error of 0.2%. The pattern in the error (figure 7.29) is similar to those observed in the smaller network and it is expected that this network will be capable of mapping to the same accuracy as the smaller network (if not better) if trained for a longer period.

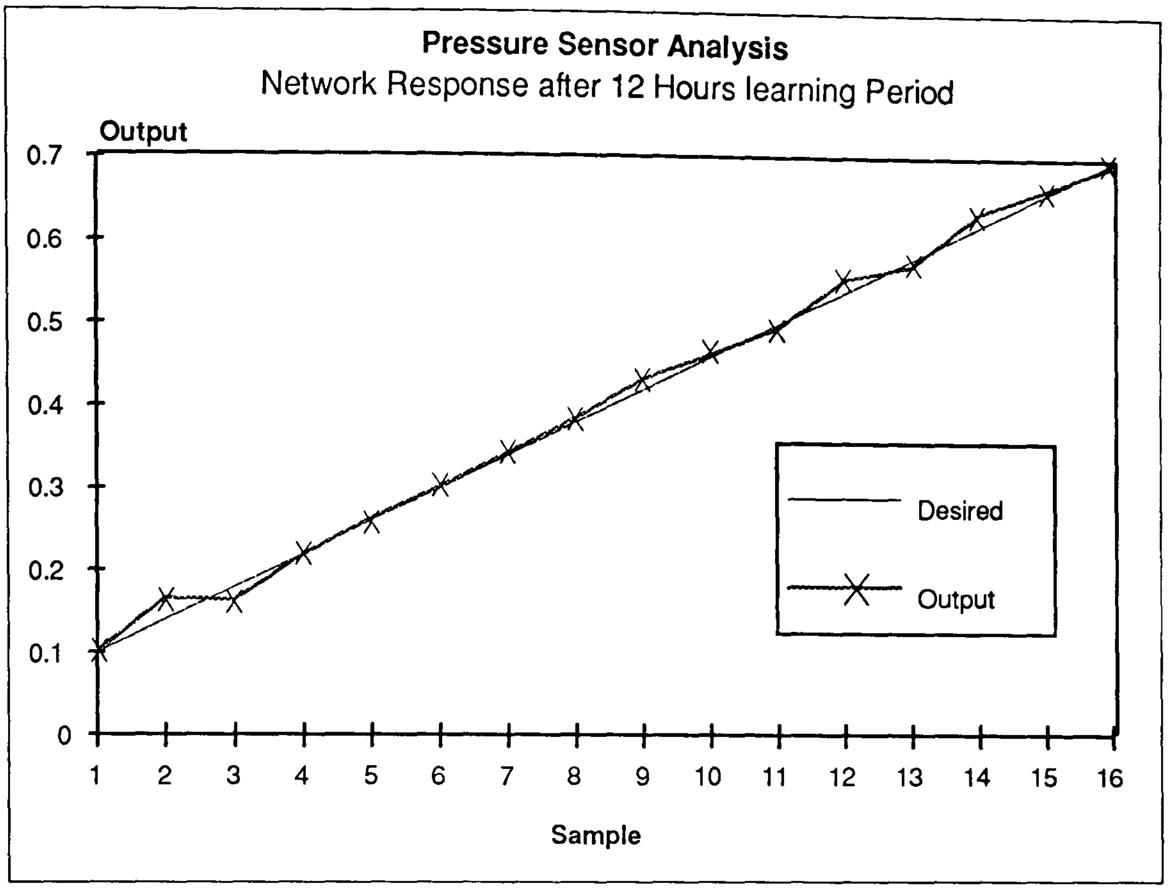


Figure 7.33. Network response to training data for pressure analysis after 24 hour training using a larger network.

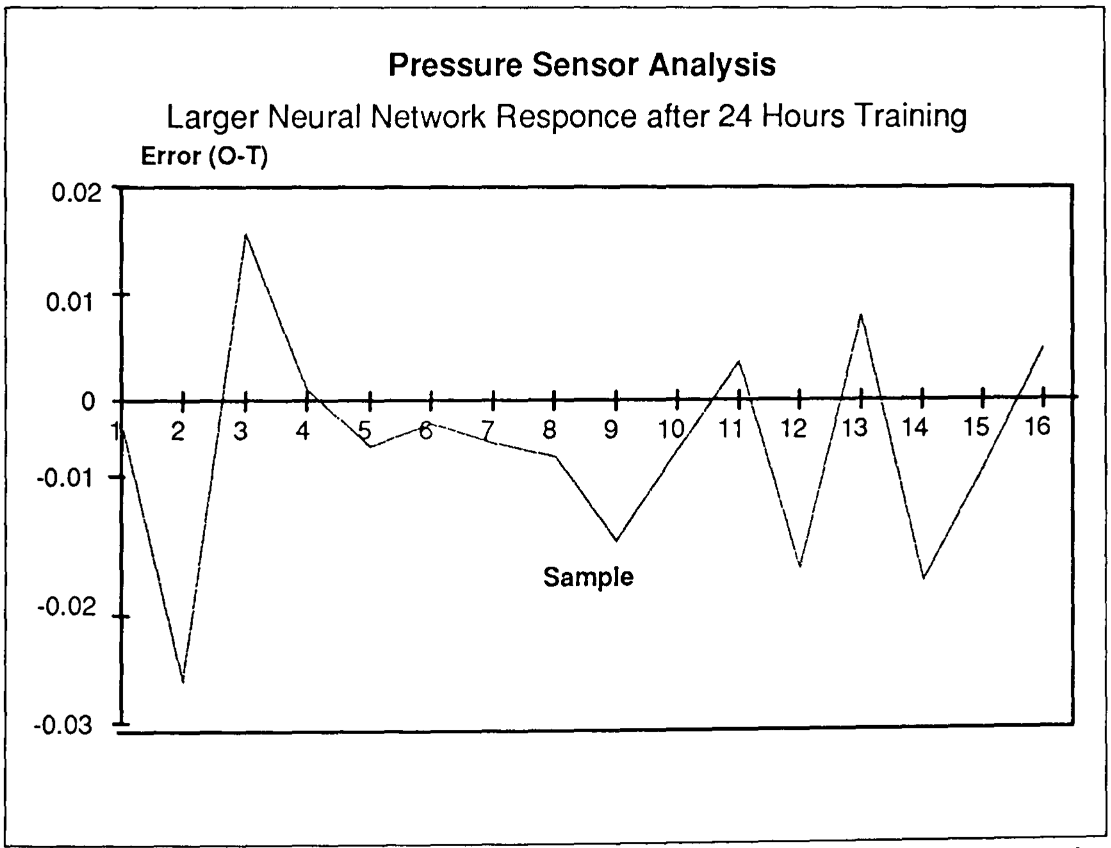


Figure 7.34. Error between desired and Network output after a 24 hour training period with a larger Network.

More interesting however is the behaviour of the large network with unseen data. This is shown in figure 7.30. The results indicate almost identical response with unseen data as the smaller networks. Accuracy for this network with unseen data is shown to be 1.62%. The effect of poor generalisation with unseen data therefore seems to be a more fundamental problem not directly associated with network size or learning period.

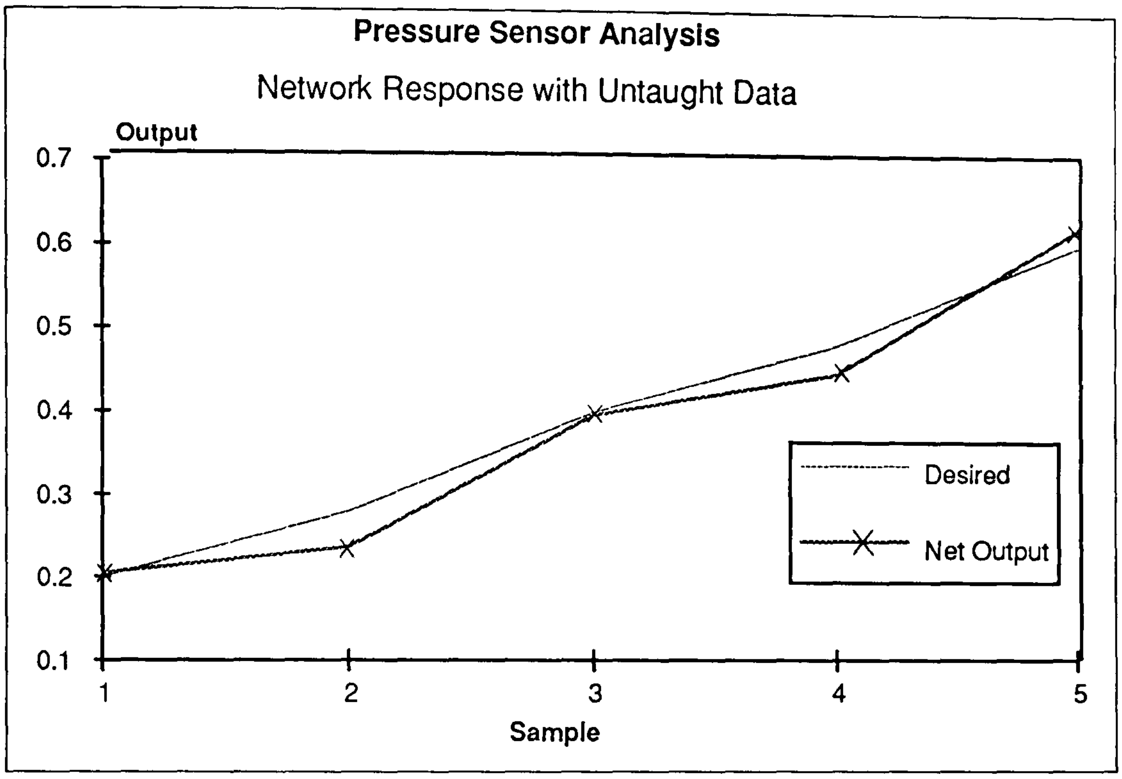


Figure 7.35. Desired and network output with unseen data after the network had been taught for 24 hours using a larger network.

7.4 Speckle Pattern Analysis Results

Speckle pattern analysis was performed using the apparatus described in section 5.8.3. and the experimental methodology of section 6.3.4. In these experiments chromatic speckle patterns are produced with HeNe and Argon Ion lasers that excite different propagating modes in an optical fibre can be achieved by launching light from one laser at different angles into the fibre (section 2.6.4). The HeNe laser excited only the Red photo detectors of the CCD camera (632.8 nm), the Ar^+ laser operating at 550nm primarily excited the green detectors. The angle of incidence of the HeNe laser was kept fixed (0^o from the principle axis) and that of the Ar^+ laser was varied from 3.84^o to 9.55^o. Figure 7.36 shows typical images of speckle patterns observed with the CCD camera for different angles of incidence of the Ar^+ laser light launched into a 400 micron multimode step index fibre, one meter long. The speckle pattern images were processed using statistical methods by considering concentric annular regions of the speckle image. Each annular region contained approximately 4000 pixels so statistical techniques are appropriate for analysis. 20 annuli of equal area were considered for the speckle images and mean chromaticities and their standard deviations were computed. The results of these calculations are shown in figures 7.36 to 7.43 for the range of different launch angles (3.84- 9.55^{o}) of the Ar^{+} laser radiation. These graphs show the variation in mean chromaticities x and y for the different regions of the speckle pattern. The error bars show +- 1 standard deviation of the chromaticities of each region. The y chromaticity graphs (figures 7.37,7.39,7.41,7.43) show the gradual extinction of lower order modes (lower annular values) as the angle of incidence of the Ar^{+} laser is increased. The analysis shows that even though the speckle produces a random distribution of light and dark patterns, chromatic statistical techniques can be readily employed to extract modal information from them. For this arrangement, at angles of incidence of and greater than 9.55^{o} the speckle patterns of the Ar+ and the HeNe lasers show only a marginal overlap. In this configuration optical fibre effects that differentially affect the modal propagation of light may be suitably investigated [Ahmed 1991][R. Smith 1994].

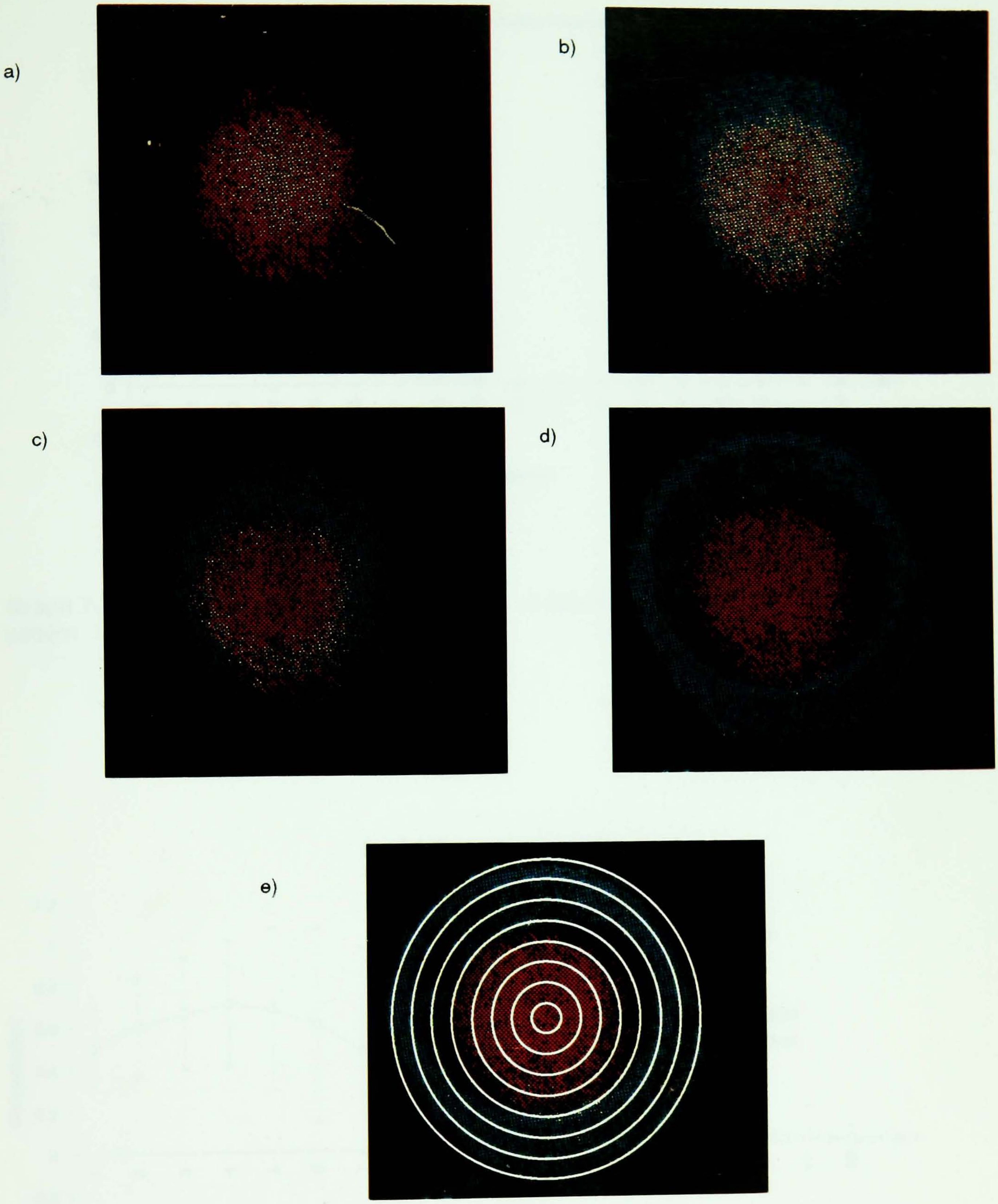
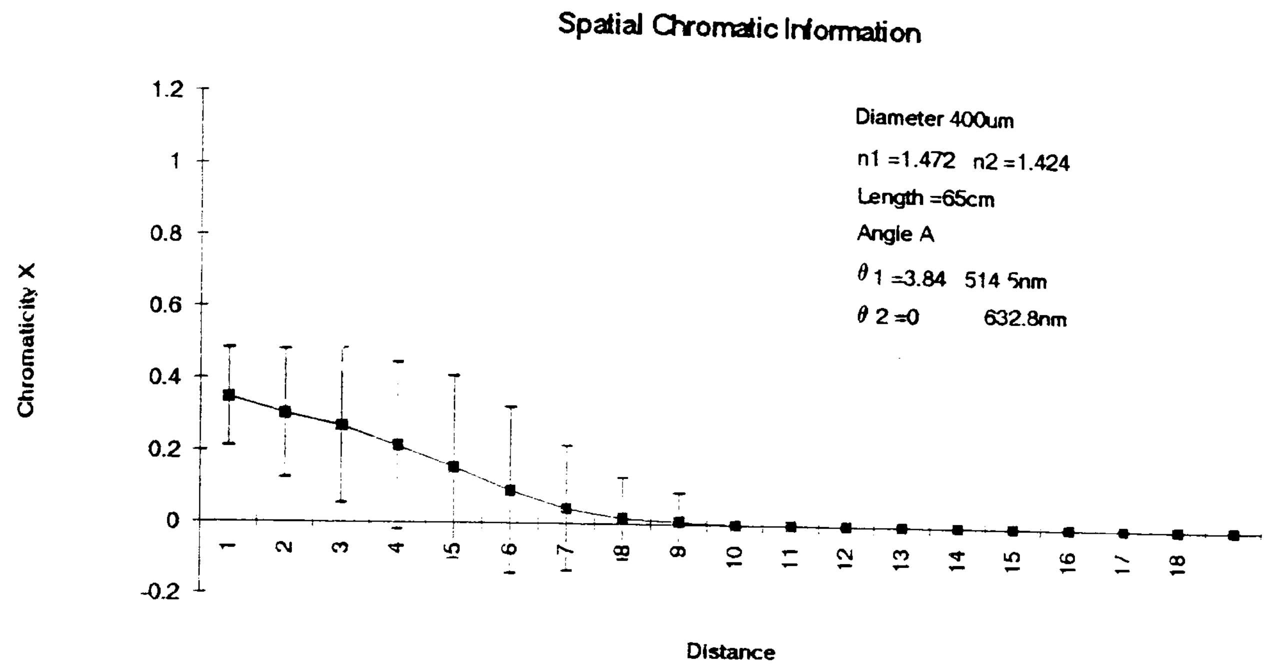
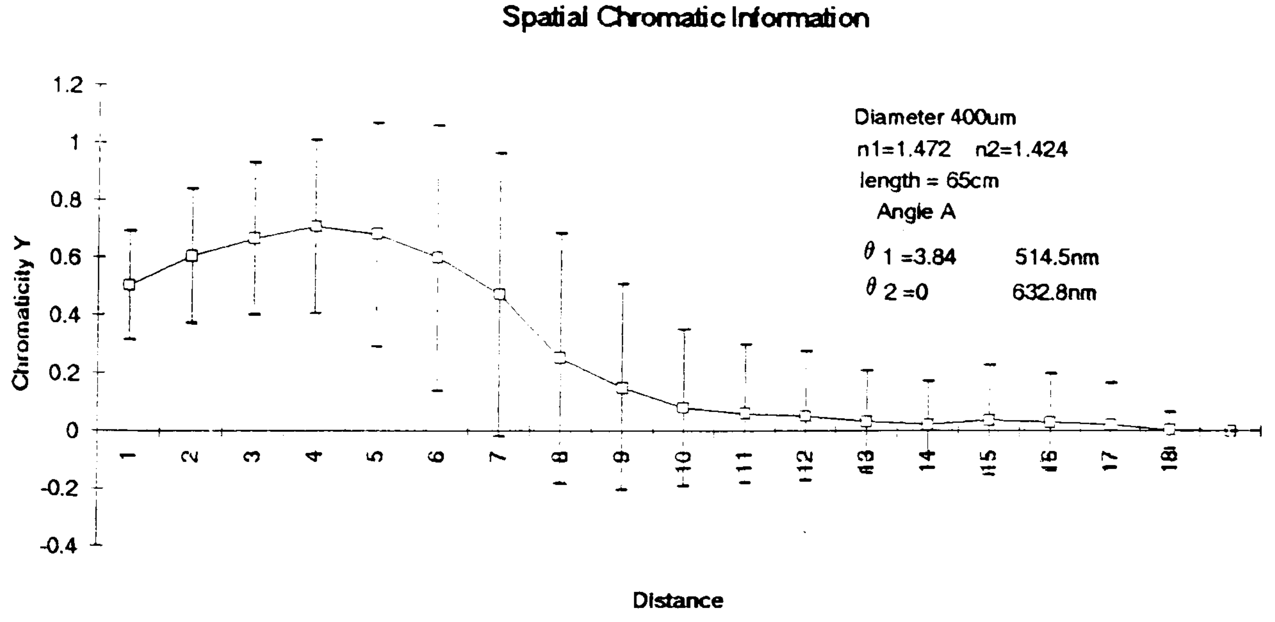


Figure 7.36. Images of speckle patterns obtained using HeNe and Ar$^+$ Lasers where the angle of incidence θ_1, of the Ar$^+$ laser was varied a) θ_1= 3.84° b) θ_1= 5.45° c) θ_1= 6.74° and d) θ_1= 9.55° e) Diagram showing how rings are used to analyse 2D speckle images.

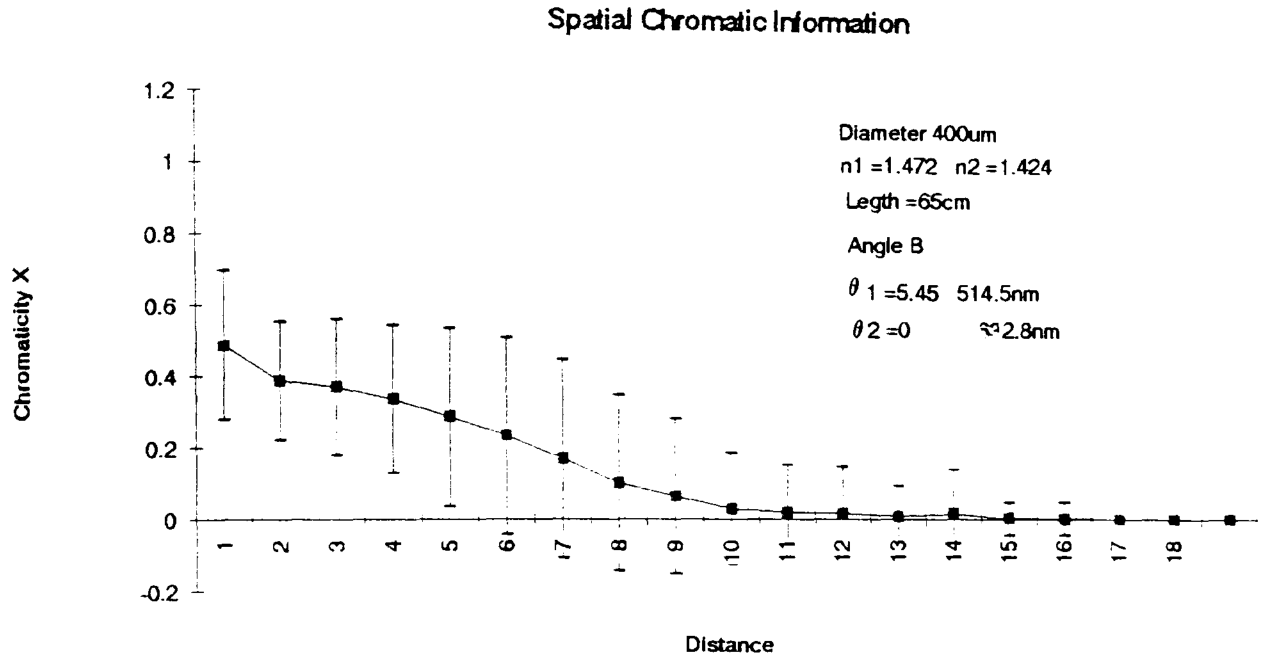
(Note that in the monitoring system 20 rings of equal area were used)



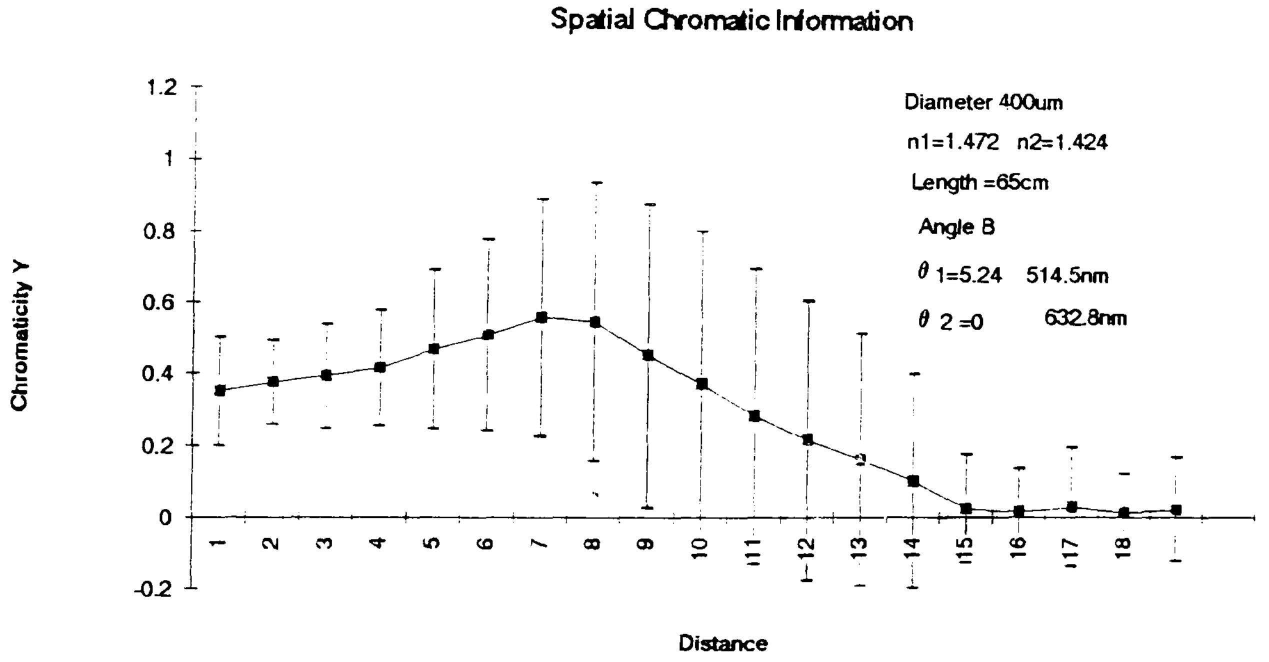
Graph 7.37. Graph showing mean chromaticity x of different ringed regions of the speckle pattern



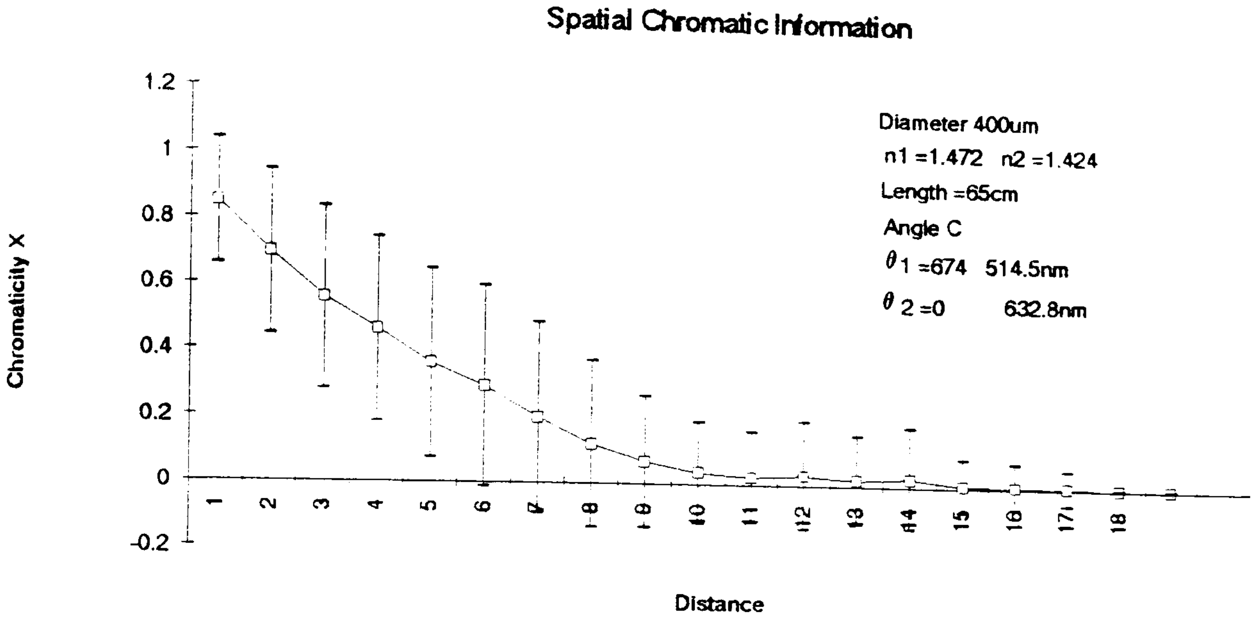
Graph 7.38. Graph showing mean chromaticity y of different ringed regions of the speckle pattern



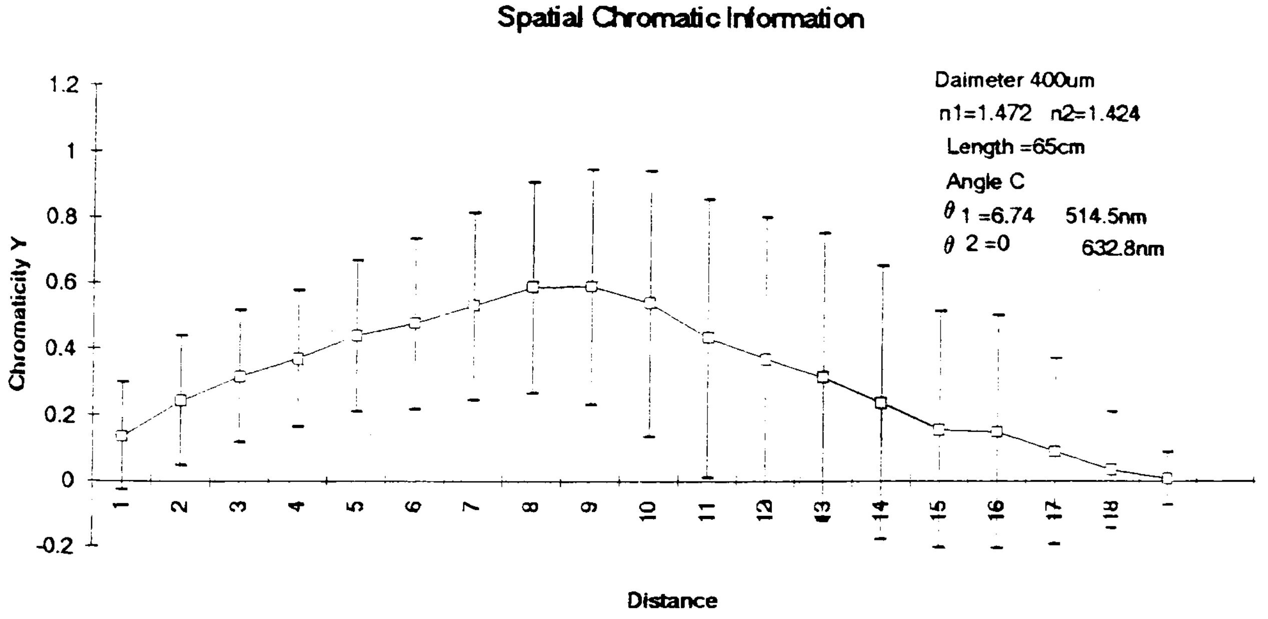
Graph 7.39. Graph showing mean chromaticity x of different ringed regions of the speckle pattern



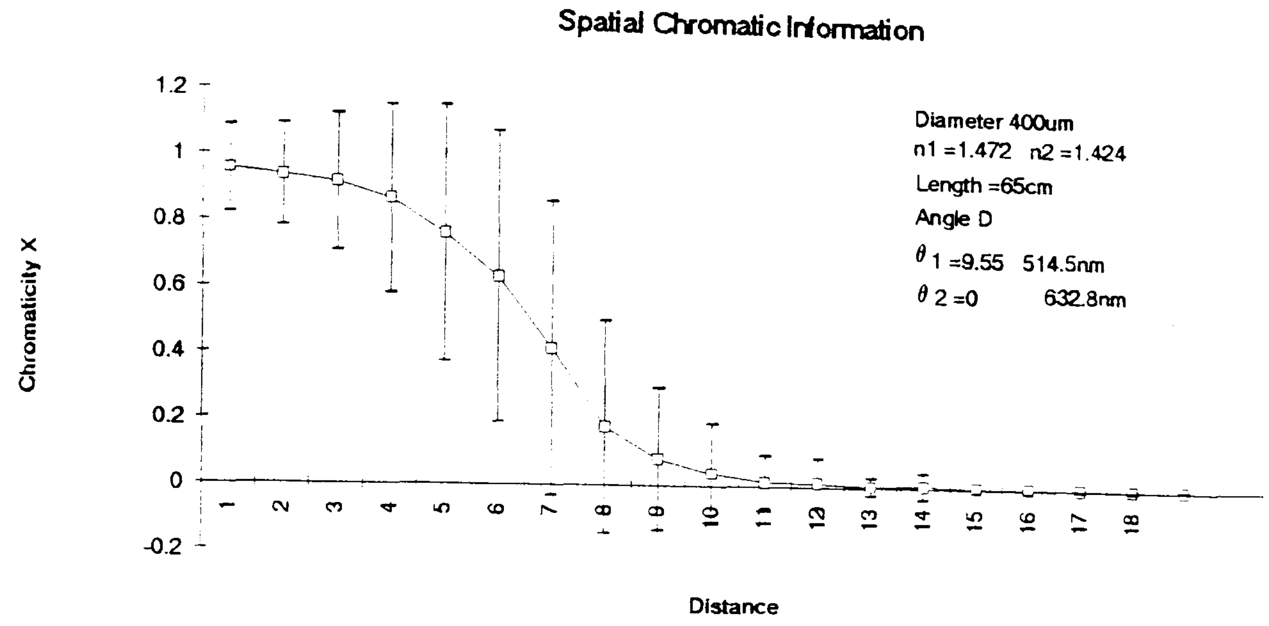
Graph 7.40. Graph showing mean chromaticity y of different ringed regions of the speckle pattern



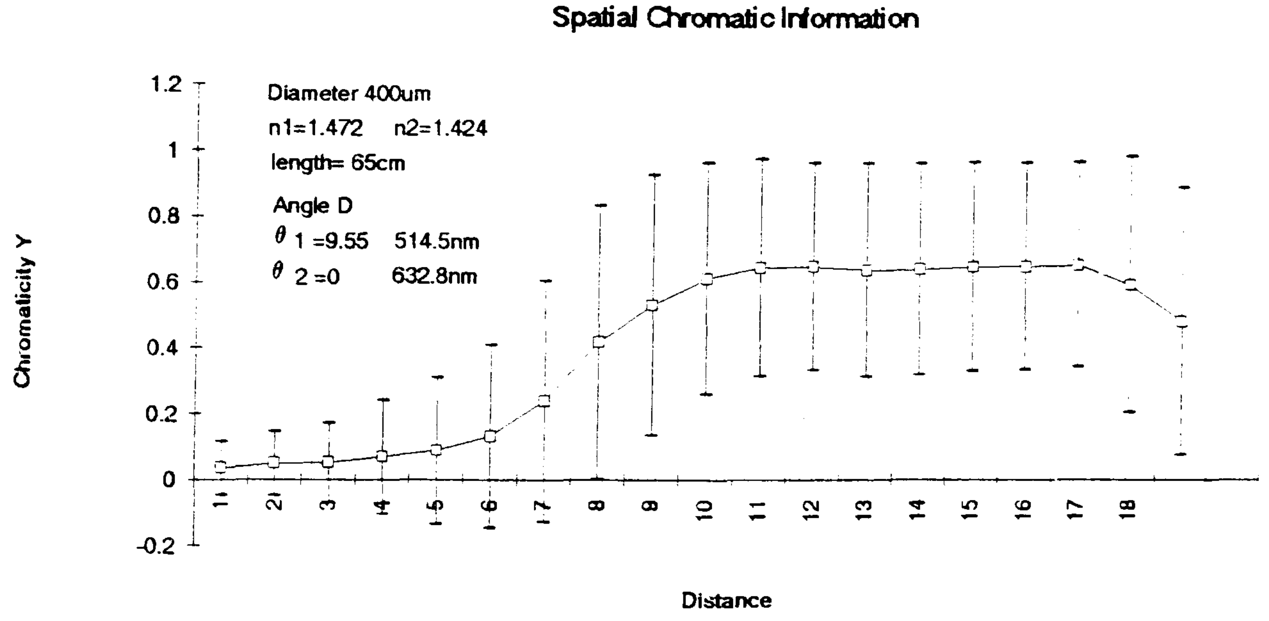
Graph 7.41. Graph showing mean chromaticity x of different ringed regions of the speckle pattern



Graph 7.42. Graph showing mean chromaticity y of different ringed regions of the speckle pattern



Graph 7.43. Graph showing mean chromaticity x of different ringed regions of the speckle pattern

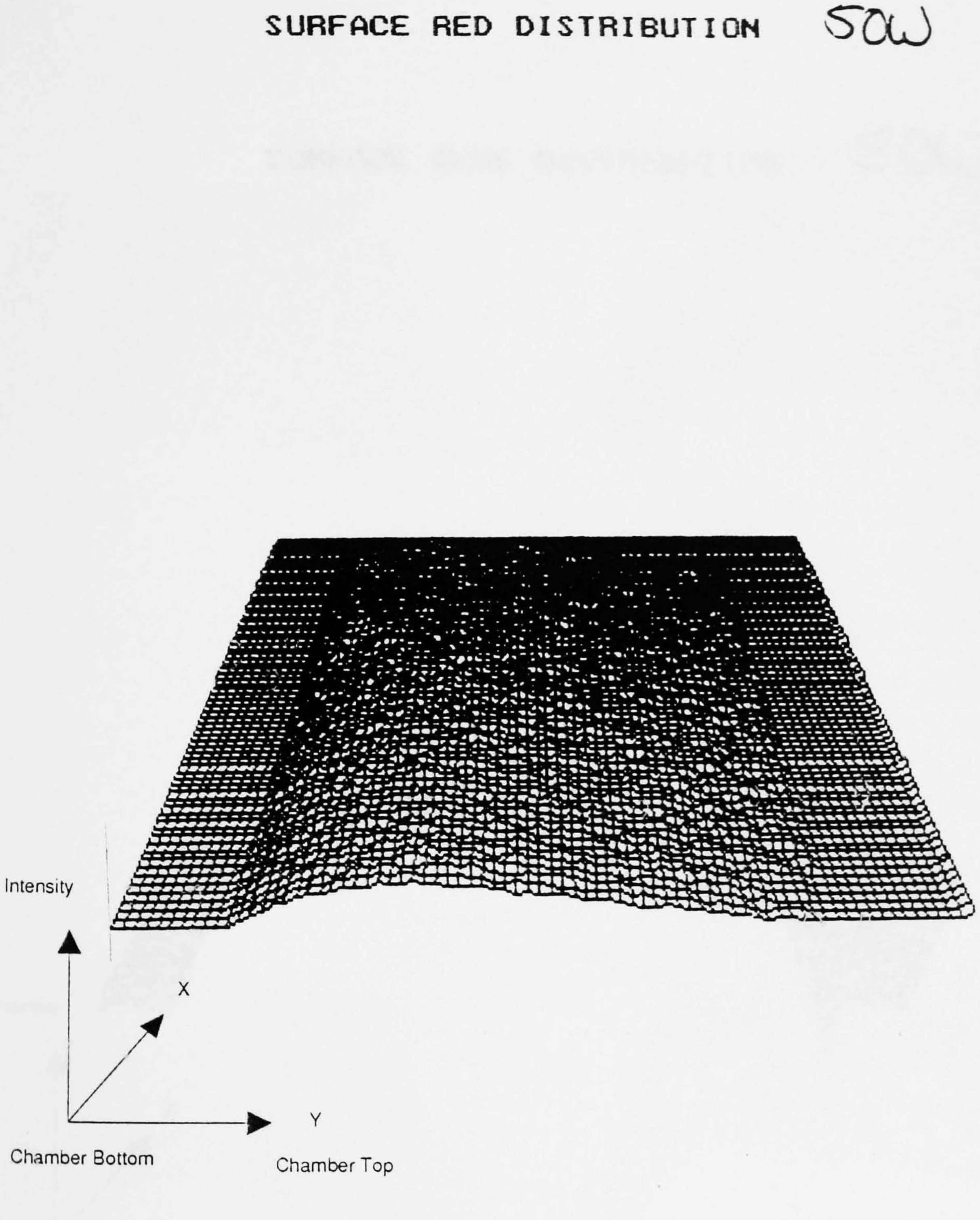


Graph 7.44. Graph showing mean chromaticity y of different ringed regions of the speckle pattern

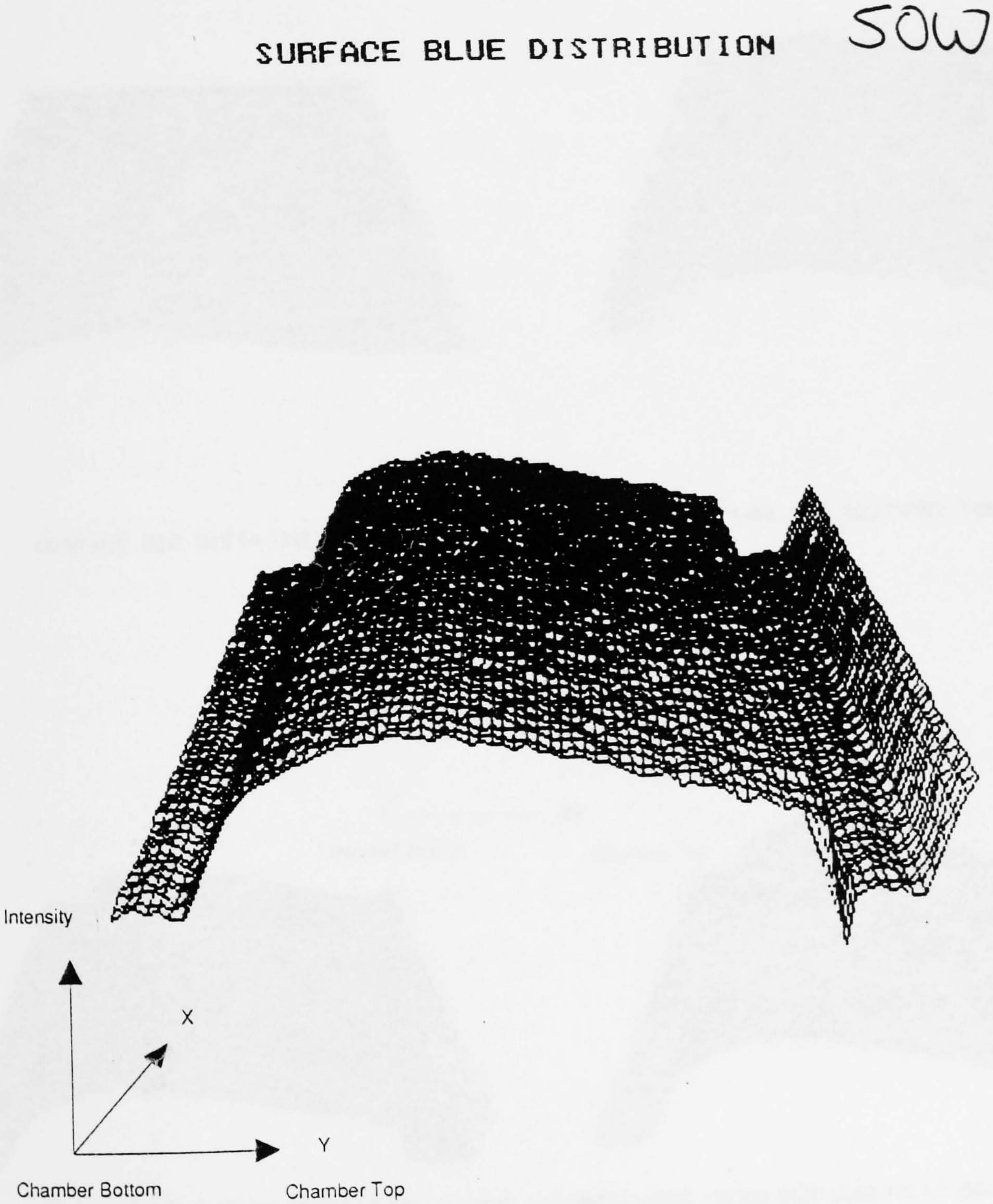
7.5 Plasma Emission Results

7.5.1. 2 D Plasma Monitoring

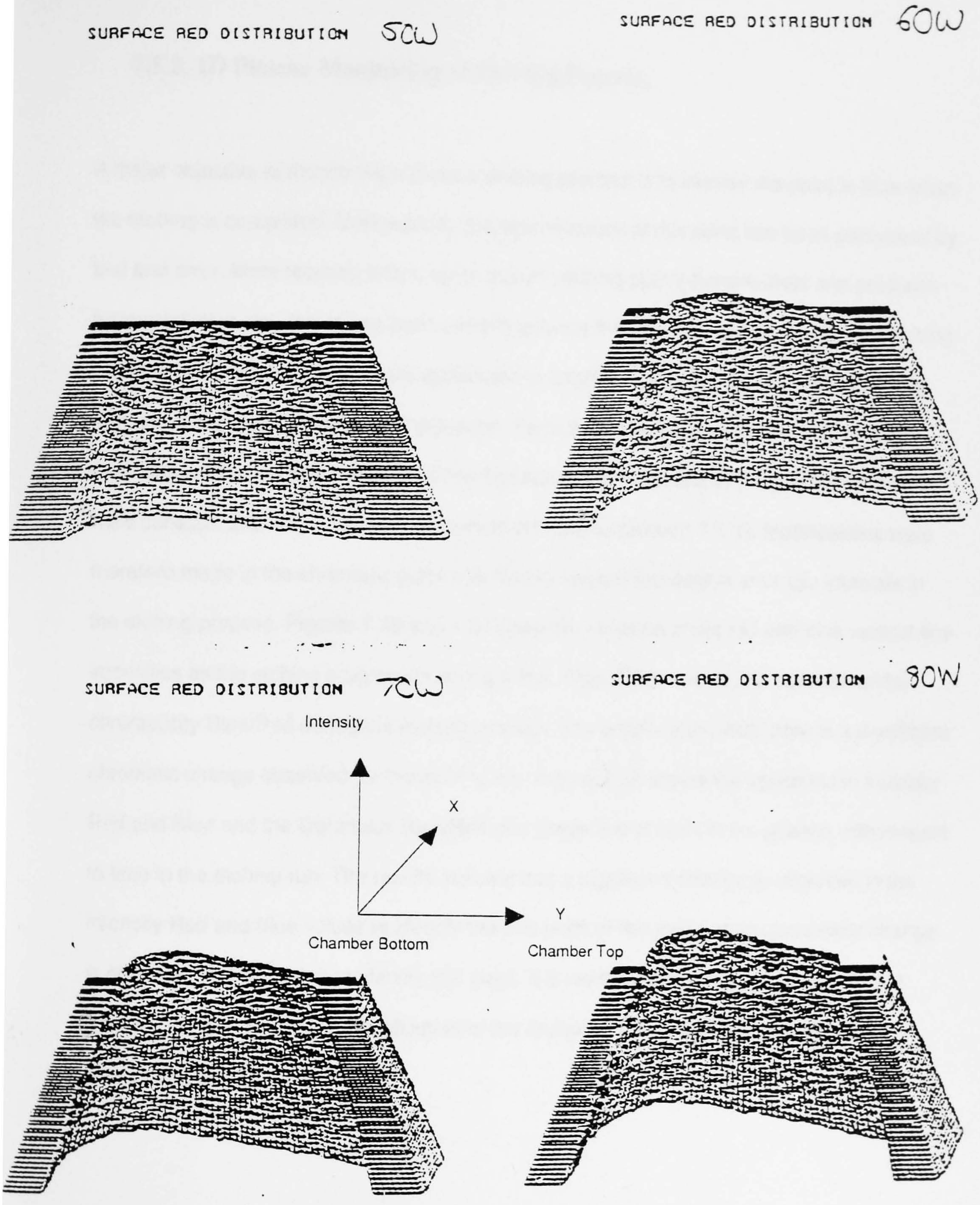
The tristimulus chromatic system developed in the University of Liverpool has shown good performance in monitoring plasma characteristics[Khandaker et al. 1994]. The monitoring system was arranged as shown in figure 5.14 and described in section 5.8. The camera was used to monitor the spatial blue and red components of the spectra as the green component of the spectra was very low. Images were obtained for various plasma powers to determine the effect of power on the plasma distribution. Figure 7.46 and 7.47 show the spatial blue and red distribution for the plasma energised to 50W. The graphs show that the intensity of the blue region of the spectra is significantly greater than the red part of the spectra and that they both have similar characteristics. The graph shows that there is little horizontal variation in the plasma but a significant variation if a vertical slice of the plasma is taken. The intensity is seen to increase gradually from the top of the chamber and gradually increase to a maximum value close to the bottom of the chamber it then falls of rapidly at the bottom of the chamber. To enable the monitoring of a wide range of plasma powers only the red component of the spectra was monitored. This was due to the physical limitation of the system which caused the blue channel to saturate for higher spectral powers. The 2D surface plots of the plasma intensity distribution for different powers is shown in figure 7.48. The results show that as the plasma power is increased gradually the form of the surface plot does not change significantly. However there is a significant increase in the spectral power distribution which is as expected for this system.



Graph 7.46. Graph showing spatial red distribution of a plasma energised to an r.f. power of 50W



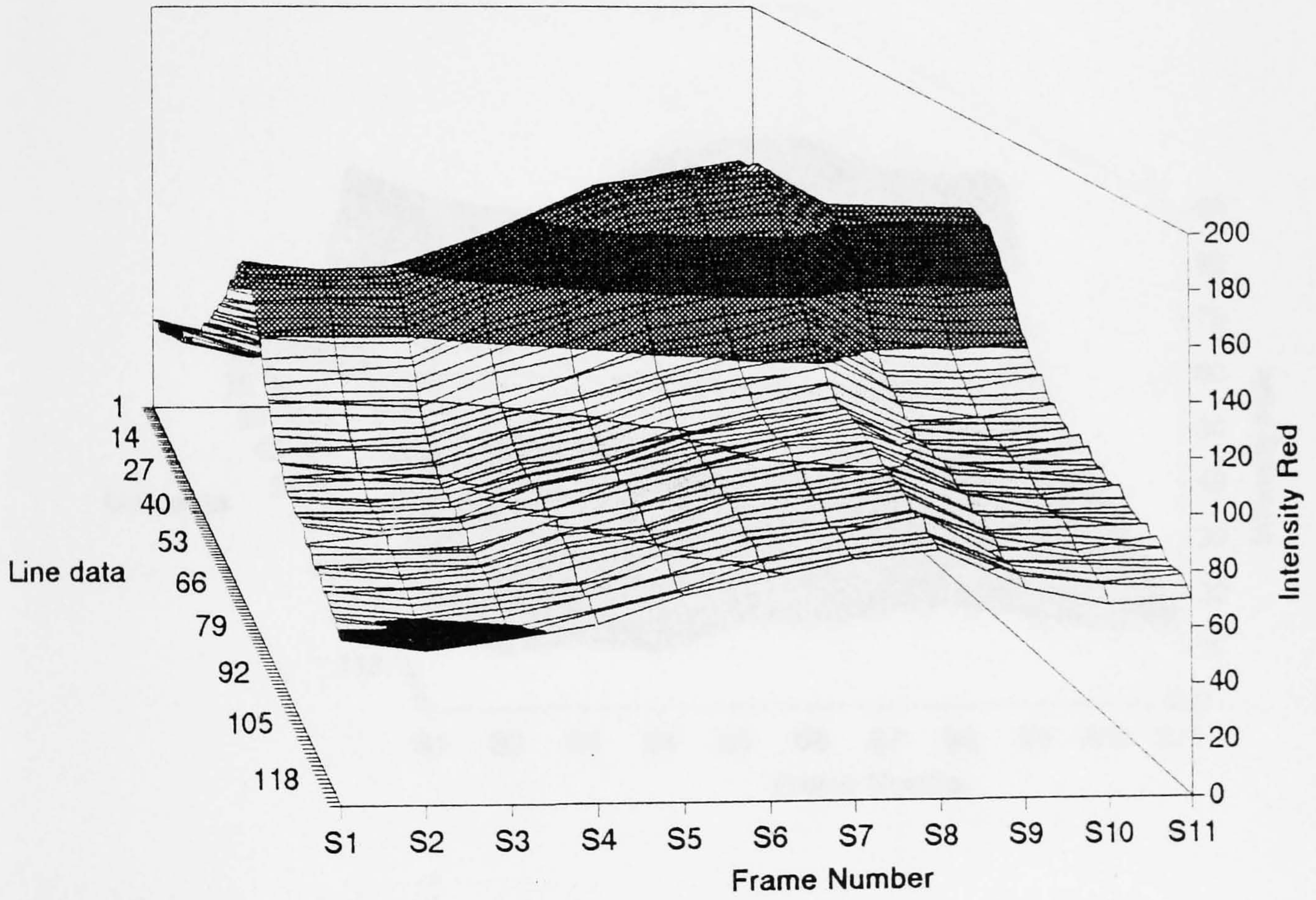
Graph 7.47. Graph showing spatial blue distribution of a plasma energised to an RF power of 50W



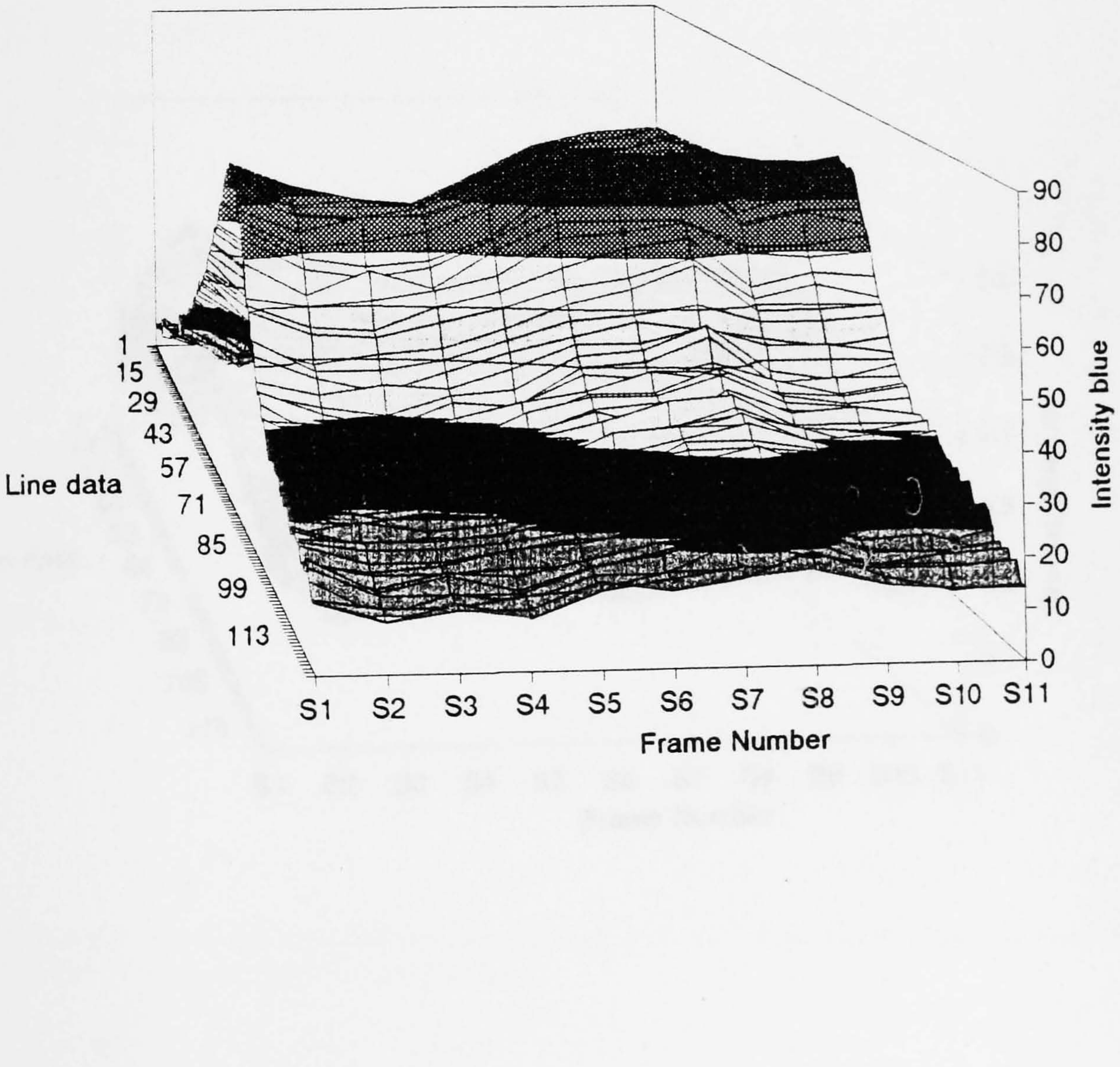
Graph 7.48. Graphs showing how spatial red distribution varies with plasma r.f. power a) 50W b) 60W c) 70W and d) 80W

7.5.2. 1D Plasma Monitoring of Etching Process.

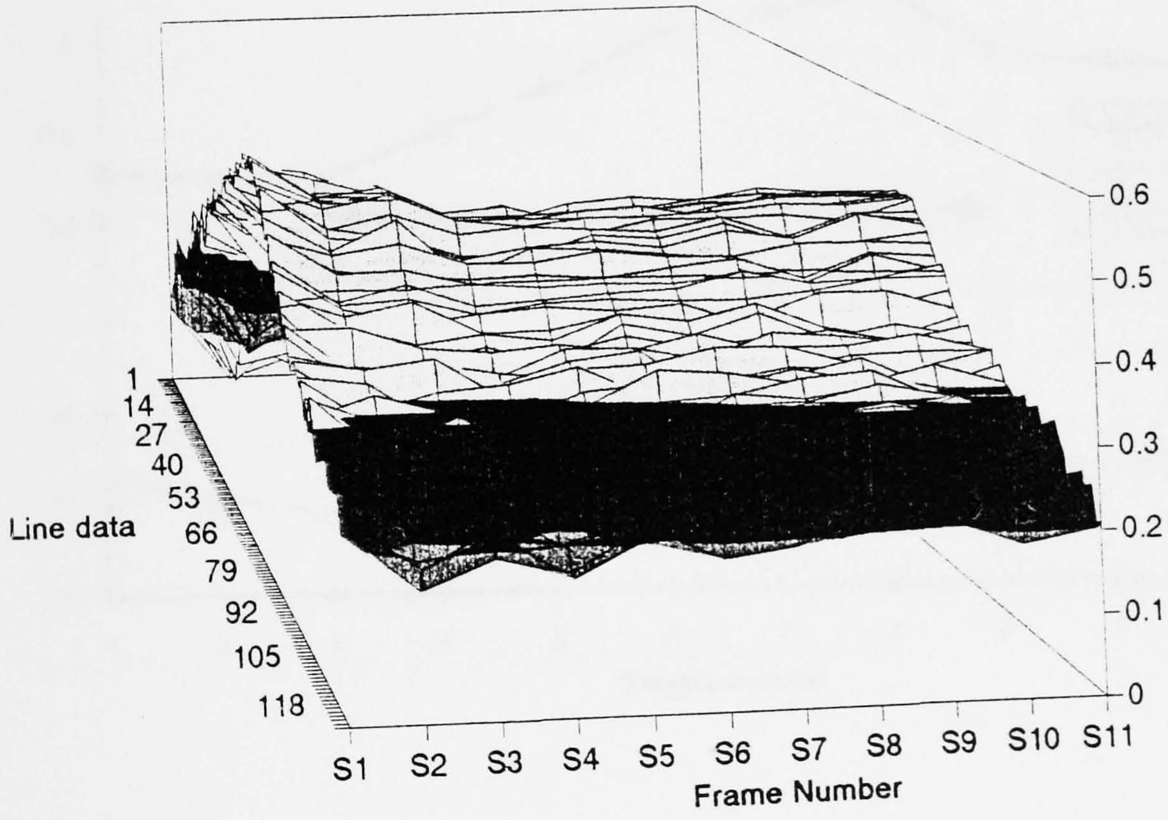
A major objective of monitoring a plasma etching process is to identify the point in time when the etching is completed. Until recently the determination of this point has been performed by trial and error. More recently a fibre optic system utilising optimised detectors and precision instrumentation electronics has been used to achieve this [Jones, Russell. 1993]. Tests using the remote chromatic system were performed to check its capability for monitoring etching rate and in particular for end point detection. From the experiments performed on the 2D monitoring of the plasma it was identified that horizontal variations in the plasma emissions were constant and only a vertical component was useful (section 7.5.1). Modifications were therefore made in the chromatic system to record vertical line data at strategic intervals in the etching process. Figures 7.49 and 7.50 show the variation in the red and blue vertical line intensities as the etching progresses during a test. Figure 7.51 shows the variation of the chromaticity Blue/Red during the etching process. The graphs show that there is a significant chromatic change observed for the etching run. Figure 7.52 shows the variations in intensity Red and Blue and the Distimulus Blue/Red of a single line of sight in the plasma with respect to time in the etching run. The results indicate that a significant change is observed in the intensity Red and Blue values to identify the end point of the etch but no discernible change is observed in the X value to identify this point. It is worth noting that the use of line data from the images allows the identification of the end point by the use of a simple peak detection technique.



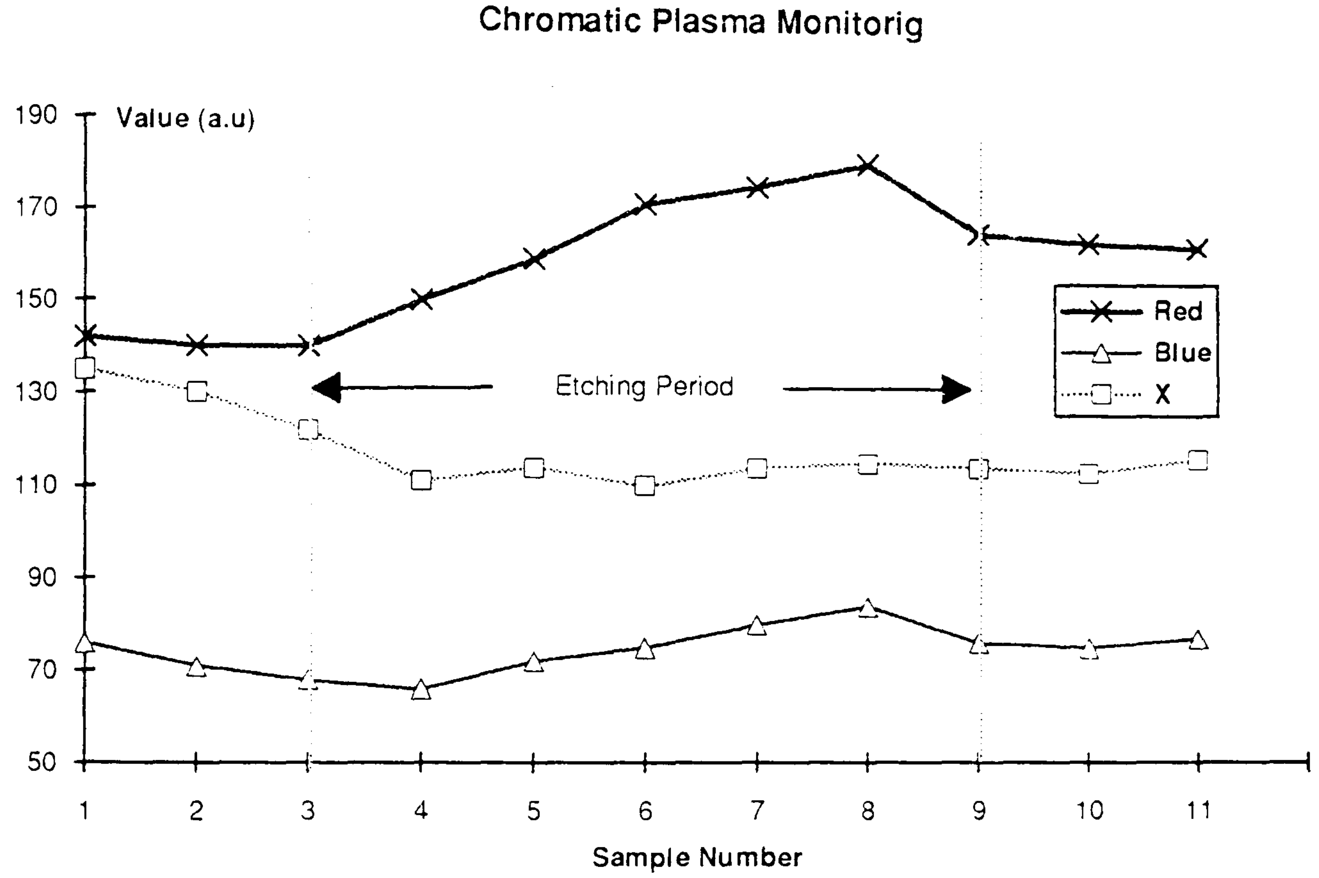
Graph 7.49 Graph showing plasma red intensity variation along a vertical line slice taken from the plasma image for different frame stages of the etching process.



Graph 7.50 Graph showing plasma blue intensity variation along a vertical line slice taken from the plasma image for different frame stages of the etching process.



Graph 7.51 Graph showing plasma Chromaticity X variation along a vertical line slice taken from the plasma image for different frame stages of the etching process.



Graph 7.52. Graph showing intensity Red and Blue and chromaticity Blue/Red variation of a single point during an etching process run.

7.6 Chemical Indicator Results

A number of chemical indicator tests have been performed using the remote chromatic processing system described in section 5.7. The results of these tests are presented below. Firstly tests on the Nephur medical test strips are presented. Finally results of monitoring various transformer oils are presented.

7.6.1. Medical Indicators

In these experiments each of elements in the Nephur medical strips section (2.6.3.1) colour chart was monitored using the remote monitoring system as described in section 5.7 and shown in figure 5.5. The RGB intensities of the various colour elements are shown in figure 7.53. The indicators were treated with a series of standard solutions to provide the full range of colour changes to be produced in real life clinical tests. The results of figure 5.73 show how the characteristics of each of the various pads vary with particular chemical to which it is sensitive. The graphs show that there is a significant change in the RGB and XY parameters with measureand for each pad. Simple look up table techniques can be utilised to relate these chromatic changes of the pads with measureand.

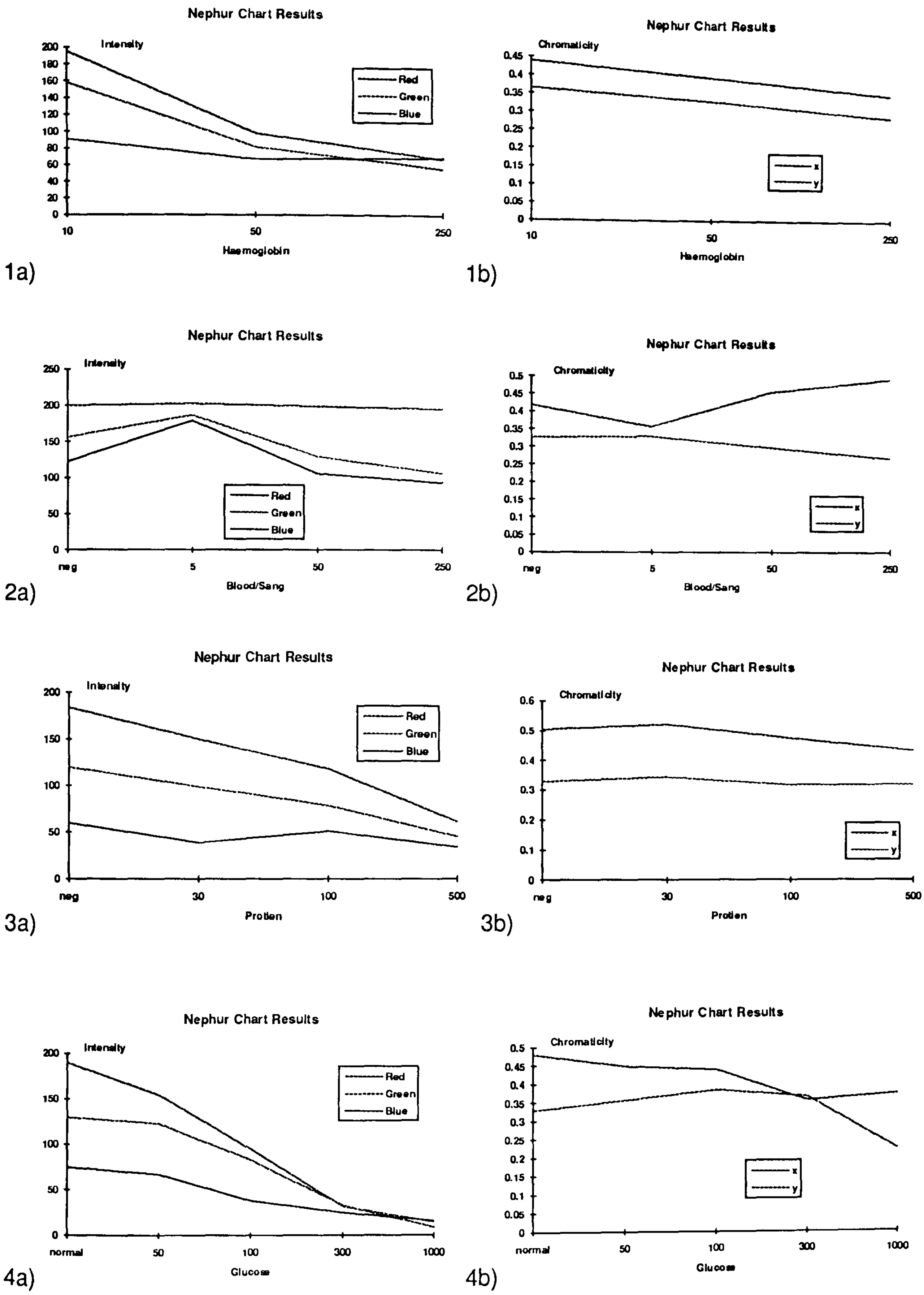


Figure 7.53 Graphs showing variation of RGB intensities (a) and XY chromaticities (b) of Combur chart with respect to measureand.

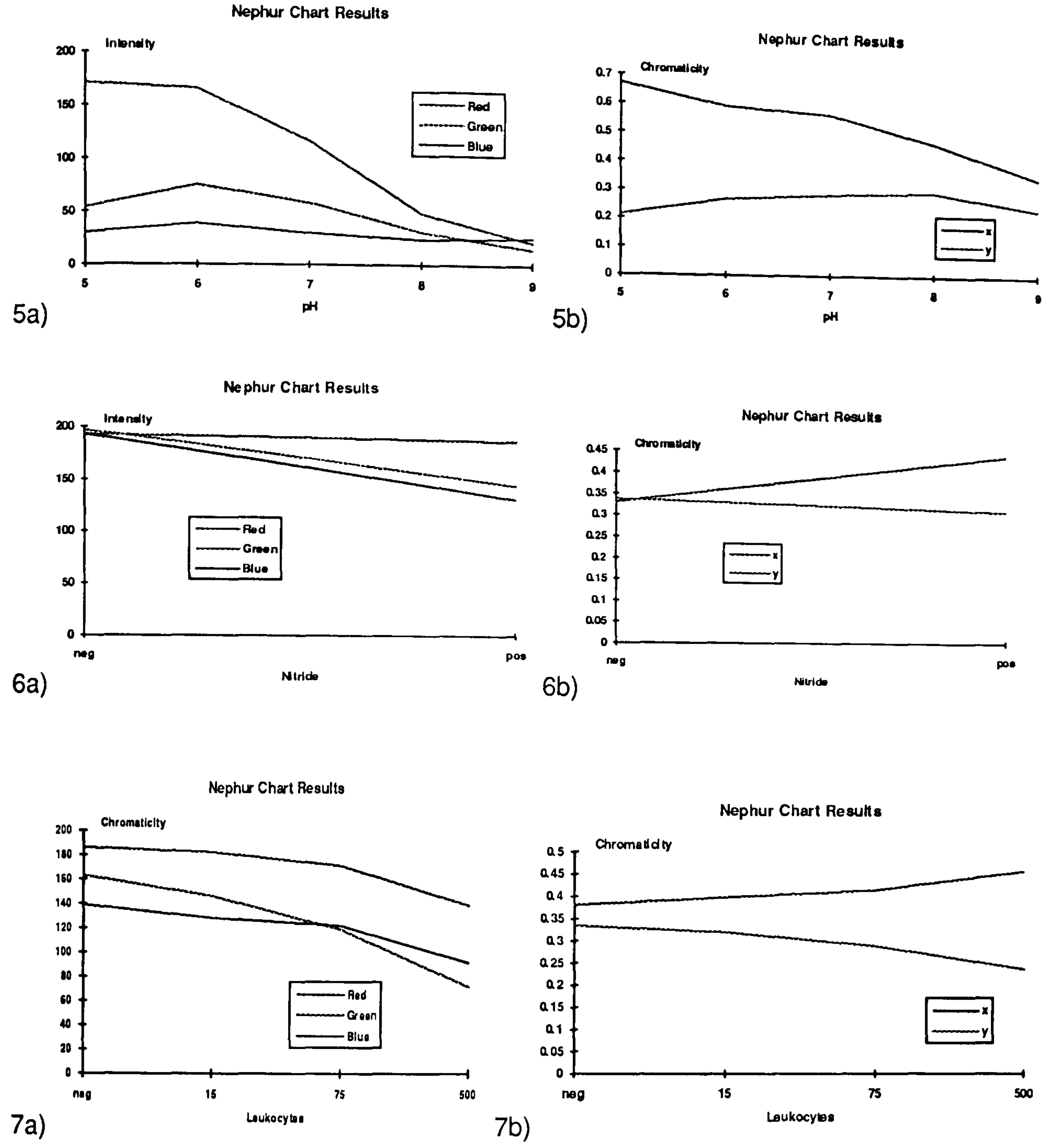


Figure 7.53 cont. Graphs showing variation of RGB intensities (a) and XY chromaticities (b) of Combur chart with respect to measureand.

7.6.3. Oil Monitoring

Samples of transformer oils at different stages of degradation from different electric power transformers were investigated using the remote chromatic monitoring system to monitor oil quality. The results of these tests are shown in figure 7.54. For each oil there are significant changes that occur with ageing. This suggests that chromatic techniques can be used in applications for quantitative analysis of oil quality.

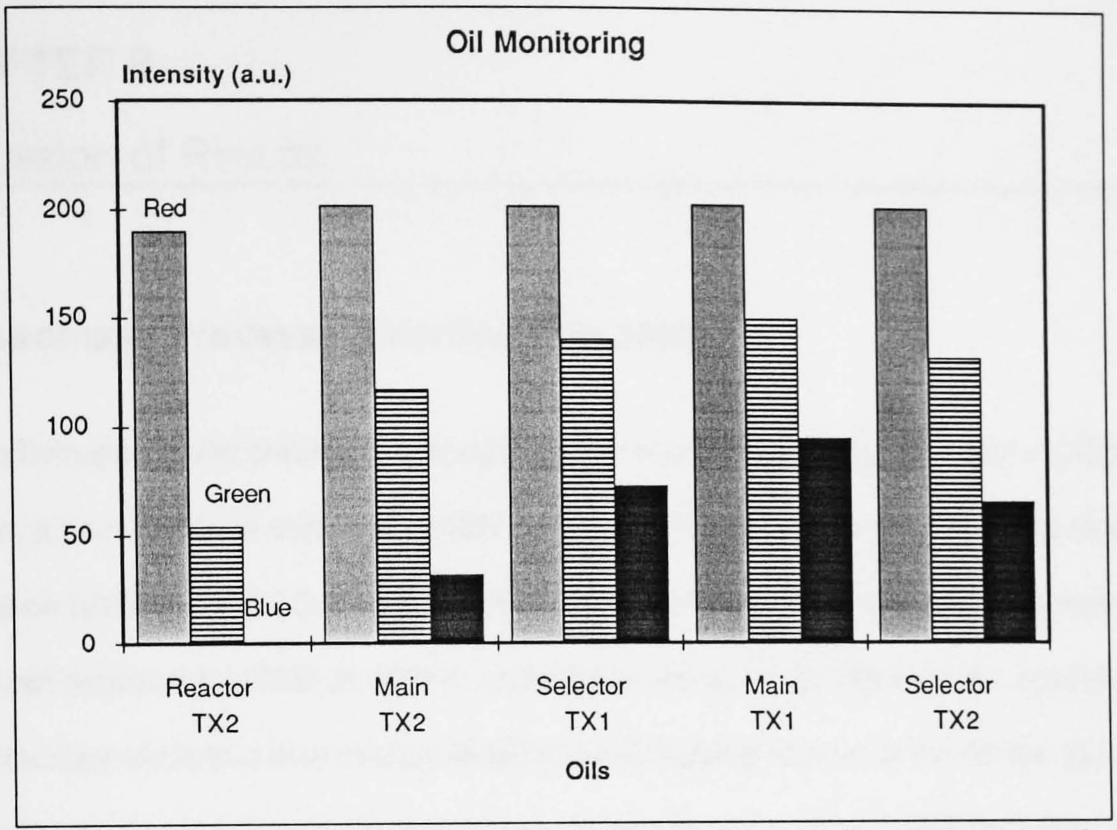


Figure 7.54. Graph showing variation of RGB values of different oil samples taken from a transformer.

CHAPTER 8

Discussion of Results.

8.1 Chromatic Processing Hardware System.

The performance of the chromatic processing system which consisted of a colour CCD camera, a frame grabber card and an IBM-PC compatible computer has been investigated. Calibration tests on the CCD array have shown that the three red, green and blue detectors have peek responses at 620nm, 530nm and 430nm respectively. The detector spectral characteristics show that they occupy wideband overlapping regions of the optical spectrum (Figure 7.1 section 7.1.1) and are suitable for chromatic processing. These detector characteristics differ from those found in the human eye as the spectral range of the blue detector does not overlap with the red detector. The significance of this is that the chromatic boundary of the chromaticity diagram is a triangle with co-ordinates (0,0), (1,0) and (0,1) in the XY space as opposed to the horse-shoe shaped boundary as shown in figure 2.7 (section 2.4.4.2) This is caused by an overlap between the red and blue detectors. The triangular chromaticity boundary indicates the region within which all detectable colours can be recorded. The relative sensitivities of the three detectors are observed to be different. This is because the RGB responses are calibrated to a CIE standard light source defined as a black body radiator at 3600^{o}K (section 5.3.3) and not to an ideal white light source. This relative sensitivity is similar to that of the human eye and therefore has the advantage of producing images that are visually correct.

The aperture of the camera is used to control the light incident onto the CCD array. Tests on the aperture of the camera were performed and showed that the output of the RGB channels varies with a sigmoid characteristic with aperture value (figure 7.2). More interestingly however is that aperture adjustments effect the chromaticity of the light being observed. There is a distinct shift toward the red as the aperture is changed from the open to the closed position (a change in the chromaticity X of 0.2). The red shift can be attributed to an effect of the CCD array with low optical power. As the optical power is reduced only those photons with higher energy (higher wavelengths) are able to cause excitation of the electrons in the CCD array surface.

The gamma correction feature provides a non-linear gain to the RGB signals obtained from the CCD array. The effect of Gamma compensation is to have increased gain for low light intensities and reduced gain for high intensities. This is analogous in behaviour to the pupil of the human eye. Gamma compensation also significantly reduces the chromatic changes observed with aperture adjustment and the chromaticity changes are halved as shown in figure 7.6 (section 7.1.1). Tests on the dark noise of the camera were also performed and this showed that the noise figure was considerably lower than that of the resolution of the FGC which was 8 bits.

The FGC is responsible for digitising the analogue RGB signals from the camera and storing the images obtained in the FGC memory. Tests on the input ADCs were performed and the results showed that the ADCs provide a linear output with input signal voltage. The ADCs used also have provision for software adjustment of the positive and negative reference voltage levels. This feature can be used for compensation and calibration purposes as they set the zero and full scale voltage range of the ADC (figures 7.8 and 7.9). However only a limited control is available as reducing the difference between the positive and negative refinance cause increased noise in the output analogue to digital conversion.

The effect of glare was investigated and the results showed that the chromaticity value migrates from the achromatic point to a chromatic value as the angle of reflection is increased (figures 7.11 and 7.12) and the glare reduced. This is because glare is the direct reflection of the source from the surface of the monitored object and what is observed is the additive mixing of the colour of the source and the object. As the angle is increased from the position where glare is observed, the effect of direct source reflection is reduced and the true chromaticity of the monitored object is observable. It was found that angle from glare of around 15 degrees was sufficient to reduce the effect of glare to an insignificantly low level. Increasing the angle from glare further causes negligible change in the object chromaticity.

8.2 Thermochromic Experiments

Thermochromic materials exhibit a chromatic change on their surface with varying temperature. Tests were performed with the chromatic monitoring system on thermochromic strip sensors with a temperature range from 25^{o} to 85^{o} C. Tests on the calibration of the thermochromic material showed that the sensor chromatically varied non-monotonically with temperature. The relationship between the chromaticity and temperature was complex (figure 7.13). Algorithms and look up tables were considered as possible solution to the mapping of chromaticity with temperature. However, because of the nature of remote chromatic monitoring, the input data is subject to a number of external effects (such as glare and lighting level changes), and the consideration of these effects in the system would make the system prohibitively complicated. Neural networks were identified as an alternative solution for the mapping of sensor chromaticity with temperature. Neural networks are simple models of the processing techniques employed by the human brain and exhibit characteristics like learning and generalisation. There are numerous neural network models that can be used for data processing tasks and the Multi Layered Perceptron (MLP) was identified as a suitable system for this project. The advantages of using such a neural network are:

a) they are simple to implement using software techniques;

b) they are good at processing analogue data and offer the scope for performing the complex mapping between input and output data;

b) once trained they are fast at processing the data for practical applications;

c) they can generalise to data which are corrupted or subject to external effects not considered;

d) They are able to be retrained to cope with other external effects at a later stage without increasing the complexity of the system.

However, the disadvantage of the system is that even though the accuracy of the neural network output with the training data can be identified, the accuracy of the generalisation property of the system is unknown. Experimental methods can be used to identify the performance of the system with unseen data and this can give an indication of the generalisation ability but it is difficult if not impossible to identify the system performance with all possible input data.

Tests were performed using the normalised RGB data obtained from the thermochromic strip calibration experiment on a MLP neural network with an architecture containing 3 input nodes, two hidden layers with 20 and 5 nodes respectively and 1 output node. The three normalised RGB data was applied to the three inputs and the system trained to map these three inputs with the temperature that was presented at the output node of the network. Two hidden layers were used to ensure that the network converges well and also to allow scope for learning larger data sets in future training by increasing the intrinsic memory of the network. RMS accuracy of 0.04% has been obtained with this system with the training data and 0.9% with unseen data. This indicates that the neural network used can reach sufficiently high accuracies for it to be practically useful for remote monitoring applications.

Tests on the effect of data quality in terms of noise have been performed on the same network to investigate the network learning and generalisation ability. The normalised thermochromic data was smoothed using fifth order polynomial fitting curves and a new data set created. This technique eliminated data ripple but still retained all the main features of the data characteristics. Tests on the network with polynomial data showed that the learning ability in terms of accuracy is significantly improved providing RMS errors with seen and unseen data of 0.01% and 0.2% respectively. Because the learning accuracy varies exponentially with iterations, the results indicated that use of polynomial data improves the network learning rate by a factor of 80 when compared with the normalised data.

Increased neural network learning makes the network more specific to the data set being trained and generalisation ability is reduced. Test on the performance of the network trained with polynomial data in terms of its generalisation ability was performed using the normalised data set as a sample. This demonstrates the ability of the network with real noisy data. The results showed that generalisation ability is indeed reduced by increased training and the RMS error for the system tended towards the RMS error between the polynomial and normalised data sets. From these tests it was identified that the best system in terms of learning and generalisation ability was the network trained using polynomial data and trained for 1 million iterations. This gives an RMS accuracy with seen data of 0.02% and 1.3% with unseen data. A major significance of the ability of the neural network to recognise signal under a high degree of corruption in that the effect of variable background light and glare can to a great extent be overcome.

8.3 Photoelastic Experiments

Photoelastic materials exhibit a chromatic change, when suitably observed through polarisers, with stress. The chromatic change appears as coloured fringes observed on the surface of the photoelastic material. Two systems have been developed for use with the remote chromatic processing system for addressing this photoelastic effect. The first using single point stress analysis utilised fibres to address the sensor. The monitoring system was used to interrogate the chromaticity of the light emergent from the system. The results of the single point calibration showed that the RGB colours from the system were cyclic in nature with stress. The chromaticity diagram is spiral in nature with no intersections in the chromaticity space with stress. This suggests that the full dynamic range of the system can be used to monitor stress (0 to 950 micro strains). The resolution of the system would be significantly better than 20 micro strains. Similar two detector (distimulus) systems have been developed [Murphy 1991] however they have smaller dynamic ranges as they utilise only a linear section of the chromaticity calibration curve.

Two dimensional stress analysis utilised the remote chromatic monitoring system to interrogate a circular photoelastic diaphragm configured as a barometer. Complex fringe patters have been observed on such systems. The fringe patterns are also complicated by imperfections in the interface between the photoelastic diaphragm and the container wall. To improve the resolution of the system the chromaticity of the full surface of the sensor was processed. To rationalise the data from the surface of the diaphragm in a meaningful manner a ring technique was used where the mean chromaticities of 10 concentric rings of the diaphragm surface were taken for processing. This ring approach was adopted to take into consideration the circular symmetry of the system. Results of calibration experiments on the photoelastic barometer showed that the ring chromaticities varied in a complex cyclic nature. Extraction of pressure from the 20 chromaticity data parameters would be prohibitively complex using conventional techniques and neural networks were identified as a possible solution to the mapping of ring chromaticities with pressure. A standard MLP with 20 input nodes (10 for the X parameters and 10 for the Y parameters), two hidden layers with 20 and 10 hidden nodes and 1 output giving the pressure value was employed for performing the data processing task. Training of such a network with the ring chromaticity values showed that pressure could be extracted from the barometer images using the ring technique. RMS errors of 0.05% and 0.17% have been obtained from such a network with seen and unseen data respectively. Tests on a larger network (architecture 20,40,10,1) did not provide significant improvement in the performance of the network in terms of learning accuracy and generalisation ability.

8.4 Speckle Pattern Analysis

Speckle patterns are observed at the output of an optical fibre when coherent light is incident into the fibre. This pattern is caused by the constructive and destructive interference of coherent light as it emerges from the fibre end. Chromatic speckle experiments have been performed with the use of two laser sources which preferentially excite different detectors in the monitoring system. The speckle images obtained have been processed using statistical techniques to obtain the mean and standard deviation of the speckle in equi-area concentric regions of the image. These regions can be related to the modal distribution of the speckle image where the effects of lower order modes are observed in the inner rings and the higher order modes are observed in the outer rings.

Selective modal excitation has been achieved by launching the laser light at an angle into the fibre. The results indicate firstly that chromatic techniques can be used to extract and separate the information contained by the speckle image in the optical fibre system. Also by selectively exciting different modes of the fibre information about the modal propagation of the fibre can be extracted. This technique has application in a wide range of intrinsic fibre sensors where the measurand is made to differentially affect the modal propagation of light through the system. These include optical fibre radiation and vibration monitoring systems [R. Smith 1994] [Cosgrave 1992].

8.5 Plasma Monitoring

Experiments using the remote monitoring system have been performed to monitor r.f. plasmas and also to monitor a silicon wafer plasma etching process. 2D monitoring of the plasma has shown that chromatic techniques can be used to extract information about the plasma state in terms of r.f. power. Using a 2D analysis also provides scope for extraction of more information like plasma pressure and gas concentrations since they also can affect the spatial distribution of the plasma chromaticity. The remote monitoring system was also used to monitor plasma etching process at different stages of the etching run and for this application a line slice of the plasma image was processed. These experiments illustrated that the system can be used in end point detection of the etching process as significant changes in the Red and blue outputs are observed.

8.6 Chemical indicators.

A number of chemical indicators have been investigated using the remote chromatic monitoring system. These include medical indicators for monitoring urine samples, Chemical dyes for monitoring concentration of different reagents in solution and transformer oils. The results of the medical indicator experiments showed that the remote chromatic monitoring system was capable of identifying the chromatic changes observed in the various pads. In some cases the chromatic changes were sufficiently pronounced to provide a more quantitative result than would be possible by manual use of the comparison chart. Automation of this process is possible and would not only eliminate the subjectivity of the quantification method but also increases the speed of analysis. Tests using the remote monitoring system on various transformer oils have shown that quantitative analysis of the oil quality is possible using such chromatic techniques.

8.7. Summary of Results

The aim of this project was to develop a remote chromatic monitoring system for general monitoring applications. The development of a monitoring system based on a colour camera, a frame grabber card and a computer workstation have been presented. The method of using colour information in image processing and data processing has been introduced with special attention to chromatic modulation as a means of handling colour data. As chromatic modulation usually produces complex and non-linear outputs, neural networks have been implemented as suitable means of interpreting the data. Tests have been performed on a number of systems based on the basic hardware arrangement mentioned for monitoring specific parameters.

Thermochromic materials, which show chromatic changes with temperature, have been tested and neural networks have been implemented to analyse the data. The results have shown that neural networks are able to process the data to a resolution of 0.01%. This indicates that the resolution of the system is dependent on the characteristics of the thermochromic sensor in terms of its modulation depth, hysteresis and ageing. Future work on the thermochromic tests includes tests to monitor the way the sensor behaves under ageing and extensive temperature cycling. These characteristics may in turn be programmed as added data for the neural network for calibration. The network may be trained to identify when the sensor has degraded significantly and requires changing.

The photoelastic barometer, based on the observation of fringes produced by a circular photoelastic diaphragm when stressed, have been analysed by the chromatic processing of two dimensional ringed regions on the surface of the diaphragm. The results show that there is a complex variation in the mean chromaticities of these rings with pressure. The ring chromaticity data have been applied to a neural network to calibrate the system. The results of the system have shown that the network is capable of processing the data meaningfully providing an accuracy with the training data of 0.01%. The network also shows good generalisation ability showing an accuracy of 0.17% with data that the system had not encountered. The limitation of the system, however, is that the system accuracy depends a great deal on the accuracy of the data that is used to train the network and the size of the data set presented to the network for training. Further work on this system will include the use of more accurate pressure monitoring systems for obtaining calibration of the photoelastic barometer.

Plasma monitoring, where the chromaticity of the plasma was acquired to monitor the process state, was performed and encouraging results obtained. Two dimensional analysis of the plasma to monitor plasma power showed that there is little variation in the plasma chromaticity horizontally across the plasma but significant variation vertically through the plasma. Also the power of the plasma can be monitored using this chromatic information. The etching end point of the reaction was also investigated using 1D analysis techniques. The results show that the end point of the etching can be suitably detected using the hardware using this technique. Further work on this system would include the monitoring of a wide range of parameters that affect the plasma, like pressure, relating them to the chromatic spatial information of the plasma. This will lead to the development of a simple technique for monitoring the plasma state for process control applications.

Speckle, the interference pattern observed when coherent light is emergent from a fibre, have been analysed using statistical techniques. The results show that a significant amount of data is available in the speckle information. This information can be readily processed using chromatic techniques if two lasers at different wavelengths are used. The chromatic and spatial information obtained from analysis of ringed regions of the speckle provide information on the propagation of light through the fibre in various modes. This information is valuable in situations where the fibre forms an intrinsic sensor, where the modal propagation of light through the fibre is dependent on the measurand. The excitation of different modes has been investigated by the changing the angle of incidence of the injected laser light. Further work on this system would include the development of specific intrinsic fibre sensors for monitoring specific parameters like pressure and radiation levels and relating the spatial chromatic variations of the speckle measurand.

A multitude of chemical indicators has been tested to observe their chromatic variations with measureand. The remote monitoring system has shown sufficient modulation depth with these indicators to be considered a practical means of quantification. These tests illustrate the flexibility of the system and the diversity of its potential applications. Further work could include development of a number of specific monitoring systems for monitoring a number of indicators and quantifying them. The scope of this work is as wide as the range of chemical indicators available commercially which produce chromatic change with measureand.

The success of this work has meant that this system can be used as a platform for preliminary tests on a wide range of systems to investigate their suitability as chromatic sensors and a number of different projects have evolved from this work.

Table 8.1 summarises some of the results obtained from the system.

Sensor	Measurement	Range	Resolution	Calibration	Applications
Thermochromic	Temperature	20-80° C	<0.01° C Theoretical	Neural Network	Remote temperature monitoring
Photoelastic	Stress	0-1000 µStrains	< 20µstrains	Look up Table	Point stress analysis
	Gas pressure	0-4137 mBAR	<70mBAR Theoretical	Neural Network	Non Contact Barometer
Plasma	Power	50-70W	1W	Look Up Table	Plasma Monitoring
	Etch End Point	-	-	Peek detection	Etch End Point Detection
Speckle	Various	-	Application dependent	Statistical	Intrinsic Optical Fibre Sensors
Medical Charts	Urine Sample	-	-	Look up Table	Automated Analysis of Urine
Transformer oil	Oil Quality	-	-	Look up Table	Oil Monitoring

Table 8.1. Table summarising results of various sensor experiments performed using the remote chromatic monitoring system.

CHAPTER 9

Conclusions and Further Work

9.1 Conclusions

The objective of this work was to develop a general purpose remote chromatic processing system for instrumentation purposes which operates in the visual region of the electromagnetic spectrum. This system was initially designed to automate the monitoring of sensors, or systems that show chromatic changes, which could be performed visually by a human observer. By using the existing technologies of cameras and television, which base their responses on the nature of the human eye, visual phenomena can be presented and analysed electronically. The main advantage of such a system is that it provides a non-contact means of addressing and interrogating the sensor. This allows a greater degree of freedom in the monitoring of environments that are hostile to conventional systems.

The use of wideband and spectrally overlapping detectors makes it possible for the extraction of the general characteristics of the optical spectra without the need for accurate spectrophotometric analysis. This method of selective filtering provides a meaningful reduction of spectral data into a form that is more usable. Electronic imaging systems that use the same chromatic response characteristics as the human eye mainly utilise it for visual representation of images and information. The availability of this technology in terms of its development of colour cameras and displays and its relatively low cost, makes the use of chromatic monitoring systems a practical and attractive method of achieving the goals of remote image processing. However, because chromatic processing relies on the manipulation of just three colour parameters, the precise spectral information is lost.

In the same way as chromatic processing reduces the number of data required to represent the spectral information using three parameters, image processing is another method used to reduce the amount spatial data necessary for analysis. This allows the system to identify the regions of the image that are of interest and a method of relating the spatial distribution of the chromatic data to the parameter being monitored. In real world applications it is sometimes necessary not only to locate the sensor but also to track it as it moves in the field of view. The problems associated with image processing are that they require a rigid criteria for correctly identifying sensor regions of interest but still require a degree of flexibility to be able to cope with real world images that can be corrupted. The selection of the modules to perform the image processing task and the complexity of the modules needs careful consideration to address the specific needs of image parameterisation. A number of different techniques used to parameterise the image into items that can positively identify the object is important to make decisions on the confidence of the system when the sensor is positively identified.

Because chromatic representation of spectral information of images are produced as a convolution of the different detector characteristics and the image spectra, they tend to show complex variations with measurand. The mapping between the chromatic information and the measurand is further complicated by the behaviour of the sensor used to convert the specific parameter to a spectral change, the chromatic characteristics of the detection system and other external optical effects like glare and lighting level changes. To take into consideration all or even a few of these effects in the mapping would become prohibitively complex using techniques such as look up tables and other algorithmic methods.

Neural networks have been utilised in the remote chromatic processing system to perform the required data processing mapping between measurand and chromatic information. Their ability to learn complex non-linear mappings and their ability to generalise with data which have not been trained makes them suitable in addressing the requirements of intelligent noise reduction which is essential in practical remote chromatic monitoring applications.

Another main advantage of using neural networks is that the underlying principles of the effects that cause the chromatic change does not need to be fully understood and, provided a suitable training set is presented for learning and the system has successfully learned, the system will be able to allow for these effects.

However, there are a number of disadvantages associated with the use of neural networks for data processing, the main one being that the behaviour of the network with all possible data combination is unknown and it is difficult to give a value for the system resolution. This unpredictability in the system has meant that neural networks have been unable to find main stream application in fields that require known system resolutions. However this work demonstrated firstly, that the theoretical resolution of a neural network system can be high (about 0.001%) provided the network architecture and training are carefully considered. Secondly the rate of learning is related to the nature of the data that is to be trained. Finally, the system showed that generalisation error based on presenting noisy data is related to the error between the trained data set and the noisy data set. This is because as the neural network is trained extensively its generalisation ability becomes poorer.

This work demonstrated that the three technologies, namely: chromatic processing, image processing and neural network processing can be merged together seamlessly, each of which addresses specific requirements of the remote chromatic processing system. Each of these technologies have their own specific advantages and limitations some of which have been presented here. However, the complete monitoring system includes the sensors, the light source and the surrounding environment which will also need to be considered. Some specific monitoring systems were investigated in this work to demonstrate the potentials and limitations of the system.

Lquid crystal thermochromic materials have been extensively used in the monitoring of temperature. These sensors produce a visible change in the chromaticity of the surface with temperature. However, these changes are non-linear and the data is susceptible to external effects like glare and lighting level fluctuations. Lighting level changes usually cause changes in the intensity of the received image and chromatic techniques are suitable in reducing the effects of intensity variations. Glare caused by the direct reflection of light from the surface of the sensor causes an additive ratiomatic mixing of the spectra of the light source and that of the sensor up to angle of approximately 15 degrees from the angle of reflectance of the source image. Thus, provided the monitoring angle is kept greater than 15 degrees the effect of glare can largely be ignored. This result is also applicable for other reflectance sensors that were investigated in this project (For example the medical indicators.)

Using neural networks to interrogate the chromatic information obtained by the system showed that once the system had been trained (taking many hours) the implementation of the network for performing the data processing was adequately fast to process the data in real time. The system showed that even with considerable changes in the source characteristics (induced by the introduction of a secondary light source) there is little change in the output temperature value. This kind of experiment indicates not only the benefit of using neural networks but also highlights one of the problems associated with this technique, that is, the quantification of the performance of the system in terms of external perturbances and data noise.

For the photoelastic barometer experiments image processing was performed by segmenting the image of a circular photoelastic disk into concentric annular regions and acquiring the mean chromaticities of these regions. The choice of using this method of segmentation, as opposed to quadrants, is based on the understanding of the nature of the pattern generated by the stressed photoelastic surface. Because the fringe patterns generated as the material is stressed is circularly symmetric, averaging over quadrant segments would yield a poorer data set than by processing annular regions. The patterns in mean chromaticity observed in these rings show complex cyclic variations with varying periodicities. Neural networks applied to this form of data are used for feature extraction. Because the features in the data that the network identifies as significant are not clearly apparent, it is hard to say that the system has

been trained correctly without the use of a more extensive data set showing many repeated cycles of the photoelastic barometer with more precise measurement of pressure. The size of the training data set then, plays an important role in the confidence that the network has leaned correctly the significant features of the data being trained.
Again, a figure for this confidence is hard to provide.

Plasma monitoring was only briefly touched upon to demonstrate the usefulness in monitoring environments that are unsuitable for conventional sensors. The elegance of the system is that it need neither light source nor sensor as both are an integral part of the plasma itself. Variations in the plasmas emission characteristics (both its spatial and spectral) depend on the condition of the plasma at that specific moment. There are a large numbers of parameters that affect the condition of the plasma like the ionic concentrations present, the pressure, the plasma power and electric field and changes in these parameters are observable in the emission characteristics. The work presented showed how plasma power and etch end point detection can be monitored using the system. However, further analysis of the spectral and spatial information may lead to more information about other parameters that affect the plasma. The problem associated with this technique is that as all the different parameters affect the plasma characteristics non linearly and also non orthogonally, it becomes harder to extract the desired parameter from the data without knowing at least some of the other parameters that affect the plasma. This is perhaps another field where neural networks and chromatic processing may find suitable application.

Chromatic speckle analysis was considered by investigating the propagation of light through the different propagating modes of a fibre. As different modes can be exited by launching laser light into the fibre at different angles and these modes can be colour coded by using lasers operating at different wavelengths. Both the spatial and spectral information obtained from the end of the speckle may be analysed statistically. The technique demonstrated that two sources may be spatially separated as it propagates through the fibre without overlap by controlling the angle of light injected into the fibre. This allows the scope for multiplexing the light spatially as well as with other conventional methods. Problems that may arise from using spatial multiplexing are firstly the design of the receiver and transmitter would become more complex. Furthermore, inter modal dispersion may cause interference between the two signals over long distances of fibre. More interesting however, is the use of this technique in the development of intrinsic optical fibre sensors. The light propagating in the outer modes are more susceptible to external phenomena like microbending, vibration and radiation. The spatial variations of the spectral distribution may provide information valid for quantifying the changes in the parameters around the fibre.

The diversity of the system has been demonstrated by monitoring chemical indicators, oils and other systems that exhibit a visible chromatic change. The system has shown that changes that are visibly observable can be quantified using the chromate monitoring system. The quantification of these changes not only removes the subjectivity of the observer but also provides scope for increased resolution and more precise automated measurements to be made. The system is useful for quantifying the chromatic changes observed for a multitude of other sensor systems. In this context the scope of this work is extensive and can lead to the development of dedicated monitoring systems once they have been proven with the general remote chromatic monitoring system.

9.2 Further Work

One aspect of work of this nature is that it brings to light a wide range of areas that require further study. Also they give indications of the possible directions that this work may take in the future. One key issue throughout the work undertaken is the quantification of the resolution and accuracy of the chromatic system. This is because the resolution of the chromaticity is related to the form of the spectra and the nature of the change. The use of alternate methods of representing colour, like HLS and La*b* may provide better solutions to the problems associated with resolution quantification. Indeed the cyclic nature of the Hue component of the HLS scheme is useful for representing the cyclic variations observed in fringe patterns. Also, this form of representation can provide a better visual concept of the form of the spectra than the chromatic representation.

Throughout the work on the complete chromatic monitoring system the chromatic effects of the monitoring system were largely ignored. The effect of aperture adjustment on the chromaticity of the observed light which showed a shift toward the red is a phenomena observed when the light levels being monitored becomes relatively small. However, as these low levels of illumination were never reached, these effects could be said to be negligible in the systems under investigation. A better understanding of the nature of these chromatic changes induced by the monitoring system, the effect of gamma compensation, the linearity of the detectors used in the camera in terms of intensity and how problems of this nature may be overcome are areas that require further input.

Continuation of the analysis of the thermochromic material is necessary. The effects of ageing, continuous temperature cycling and radiation degradation require further investigation. The dependence on a neural network ability to learn based on the number of training data sets used has already been mentioned. However, the overloading of data, its generalisation performance and the networks ability to extract information from different data values representing the same temperature needs to be investigated. The quantification of system insensitivity to data corruption by external phenomena may be possible using simulation studies and studies on how the network extracts the information from the data presented may provide directions for alternative methods of temperature quantification. Image processing modules should be rigorously tested in real world environments where there may be obstructions and other sensor like systems in the field of view. The utilisation of reference areas on the surface of the sensor and the compensation of external effects using these reference areas may be significant in improving resolution and accuracy of the system.

Analysis of the Photoelastic materials has demonstrated that the nature of the chromatic changes are complex in the annular regions of the material. The use of alternative shapes, for example square, of these diaphragms would also provide a suitable sensor for investigation using the chromatic system. The monitoring of different regions may also be considered for analysis as opposed to the annular regions used in this work. Further work would also include the monitoring of different photoelastic materials of different degrees of thickness and monitoring the effect of long term cycling and material fatigue. Temperature also plays an important role in the behaviour of the photoelastic material as temperature cycling is used to anneal the photoelastic material and this should be quantified if the system is to used in instrumentation.

Some indications of further work in plasma monitoring and speckle analysis has already been mentioned in the conclusions. The use of more than three detectors to monitor the spectral changes of the system may be useful in plasma monitoring. The identification of regions of the spectra which are modulated by the plasma as different parameter are changed in a controlled manner may lead to better instrumentation for plasma diagnosis. Also the use of optical data in conjunction with data obtained from other instrumentation used for monitoring the plasma may provide a system that will be able to account for the changes in the plasma.

Intrinsic sensors based on the analysis of modal effects induced by external phenomena on the fibre using the two dimensional techniques used in this work may be developed. The quantification of the variations in the modal propagation with different parameters will not only provide an insight into the nature of the light propagating through the fibre but also a method of relating the changes to measurand.

The nature of the general purpose image processing system lends itself well to a multitude of different chromatic applications. Monitoring of flames, power switches and radioactive environments are potential areas where the system may find future application.

References

References

[1] Ahmed S.. *Vehicle Detection And Weighing Using Optic Fibre Axle Sensor*, Msc. Thesis, University of Liverpool, October 1990

[2] Ahmed S., Russell P.C.. Lisboa P, Jones G.R., *Remote Sensing Using Neural Networks*, Workshop on Neural Networks: Techniques and application, Conference Presentation, (1993)

[3] Ahmed S., Jones G.R., Spindlow J., Stevens A., *Intrinsic Optical Fibre Sensing for Axle detection*, Traffic Engineering and Control, Nov. 1991, pp 527-530

[4] Alford M.W. A Requirements Engineering Methodology for Real Time Processsing Requirements. IEEE Trans. Soft. Eng. SE-3(1) Jan 1977

[5] AVC88 Conference *Proceedings of the fourth Alvey Vision Conference*, University of Manchester Sept 1988.

[6] Anderson D.Z.(Ed.), *Neural Information Processing Systems*, New York, American Institute of Physics, 1988

[7] Ballard D.H. and Brown C.M., *Computer Vision*, Prentice Hall, New Jersey 1982

[8] Ballard D.H., *Generalisation of the Hough transform to detect arbitary shapes*, Pattern Recogn, 13 No 2. (1981) pp 111-122

[9] Beaven C., *Colour Measurement in Optical Metrology*,PhD.Thesis, University of Liverpool, 1989.

[10] Benjamin R. *The Scope for Signal Processing in System Design.* International Specialist Seminar on the Impact of new Technoologies on Signal Processing. IEE Conf. Publ. 144 pp2-11 1976

[11] Born M. Wolf E., *Principles of Optics*, Pergamon Press, New York, 1970

[12] Boyle W.S. Smith G.E *Charge Coupled Semiconductor Devices* . Bell Syste. Tech. J. 49 pp 587 -593 1970

[13] Boyle R. D. , Thomas R. C., *Computer Vision A First Course*, Blackwell Scientific Publications, Oxford 1988

[14] Brown W.R. MacAdam. D.L. *Visual sensitivities to combined chromaticity and luminance differences* J. Opt. Soc. Am. 39 pp808-834 1949

[15] Chmurny G.N., Hilton B.D., et al. *NMR in Biomedicine* 1, 1988, 136-50

[16] CIE 1931. International Commission on Illumination, *Proceedings of the Eighth session*, Cambridge. England. Bureau Central de la Paris, 1931

[17] CIE 1978 International Commission on Illumination, *Recommendations for uniform colour spaces* Supplement No 2 (TC-1.3, 1978) to CIE Publication No 15 (E- 1.3.1., 1971), Bureau Central de Paris, 1978

[18] Cosgrave J., *Optical Fibre Acoustic Monitoring of Partial Discharge in Gas Insulated Substations*, MSc. Thesis University of Liverpool, **1992**

[19] Culshaw B., *Optical Fibre Sensing and Signal Processing*, P.Peregrinus Ltd. London, **1986**

[20] Dainty J.C., *The Statistics of Speckle Patterns*, Ed. E. Wolf. Progress in Optics, XIV, N. Holland, **1976** pp 1-44

[21] Davies E.R., *A Glance at image analysis- Chartered Mechanical Engineer,* (**1984**) pp 32-35

[22] Denker J. Schantz D. Wittner B. Sollia S. Hopfield J. Howard R. *Automatic learning, Rule Extraction and Generalisation*, Complesx Systems, 1, pp 877-922 **1987**

[23] Elser W., Ennulant R.D., *Advances in Liquid Crystals*, vol 2. , New York Academic Press, **1976** pp. 73-172

[24] Ersoy O.K. *Real Time Interpolation of Images obtained by image detector arrays.* Proc. Int. Conf. On Image Detection and Quality. Paris pp371-374 **1986**

[25] Ezeel B. *Borland C++ 3.0 Programming* 2nd Ed. Addison-Wesley Publications New York. **1992**

[26] Feldman J.A., Yakimovsky Y., *Decision Theory and Artificial Intelligence*, Artificial Intelligence 5, **1974** pp 349-371

[27] Feynman R.P., *The Feynman Lectures on Physics*, vol 2. Addison Wesley, **1969**

[28] Gloge. D. *Optical Powerflow in Multimode Fibres.* Bell. Syst. Tech. J. 51. 1767-1783 **1972**

[29] Grand Y.L., *Light Colour and Vision*, Chapman and Hall Ltd. ,2nd Edition, London **1968**.

[30] Goodman, J.W., Rawson, E.G., *Statistics of modal noise in optical fibres - a case of constrained speckle*, opt. lett. , July **1981**, 6 (7) p. 324

[31] Green W.B., *Introduction to Digital Image Processing,* Van Nostrand Reinhold Company, New York **1983**, pp 2-3

[32] Hect E., Zajac A., *Optics*, Addison Wiley, **1974**

[33] Helson, J. Opt. Soc. Amer. 33, p555, **1943**

[34] Henderson P., *Chromatic modulation systems for multiparameter measurement in physically demanding environments*, PhD, University of Liverpool (1989)

[35] Hendro D., Deby P., *Electromagnetic waves along long cylinders of dielectric,* Annal. Physik, 32(3), **1910**, pp465-476

[36] Howritz P., Hill W., *The Art of Electronics*, 2nd Ed. Cambridge University Press, New York, **1990**

[37] Hunt R.W., *Measuring Colour*, J. Wiley &Sons, New York, **1987**, pp. 60-70

[38] Imaging Technologies Inc. *CFG Hardware Reference Manual,* No 47-H30002-01, August **1991**

[39] iEC800 Caméra CCD rapide haute résolution, Manuel d'Utilisation Version V2.0 November **1989**

[40] Jessop H.T., Harris F.C., *Photoelasticity*, Dover Publications, **1960**

[41] Jones G.R., Beaven C., Henderson P., Lewis E., and Kwan S., *Optical fibre based sensing using chromatic modulation*, Opt. Laser Technol. 9. (**1987**) pp 297-303

[42] Jones G.R., Russell P. C., *Chromatic Modulation based Metrology*, Pure Appl. Opt. 2., **1993** pp.87-110

[43] Judd D. B. *Hue, Saturation and Lightness of Surface Colours with chromatic Illumination*, J. Opt. Soc. Amer., 23, **1940**, p2.

[44] Judd, D. B. & Wyszecki G. *Colour in business, science and industry* 2nd ed., Wiley New York **1975**

[45] Kaplan H., *Photonics at Work*, Photonics Spectra, September **1992**, pp94-95

[46] Kershaw D., *Chromatic Signal Processing for Optical Metrology*. PhD. Thesis, University of Liverpool, March **1991**.

[47] Kernighan B.W. Ritchie D.M. The C Programming Language, Prentice Hall New Jersey.

[48] Khandakar I., Glavas E., Jones G.R., *A fibre-optic oil condition monitor based on chromatic modulation*. Meas. Sci. Technol. 4. **1993** pp309-314

[49] Khandakar I., Glavas E., Morse K., Moruzzi S., Jones G. R., *Chromatic Modulation as an on-line plasma monitoring technique*, Vacuum/V45 1, **1994** pp109-113

[50] Kwan S. Beaven C., Jones G.R., *Displacement measurement using a Focusing Chromatic Modulator*, Meas. Sci.. Technol. 1 **1990** , 207-215

[51] Land, E. H. *Experiments in colour vision*, Scientific American, May **1959** Vol 200, No 5. pp 84-89

[52] Lisboa. P. (Ed.) *Neural Networks: Current Applications*, Chapman and Hall, **1992**

[53] Lowe, D. G., Binford D. O. *Segmentation and Aggregation: An approach to Figure Ground Phenomena*, Proc. DARPA Image Understanding Workshop, Palo Alto, **1982** pp 168-178

[54] Lunscher W.H. *Optimal edge detector design in Patramete Slection and Nise Effects*. IEEE Trans. Pattern Anal. Mach. Intell. 8(6) 679-698 **1986**

[55] MacAdam D.L., *Visual Sensitivities to colour differences in Daylight*, J. Opt. Soc. Am., 32(5) **1942**

[56] Moore A.J. Tyrer J.R. *An Electronic Speckle Pattern Interferometer for Complete In Plane Displacement Measurement*. Meas Sci. Technol 1. **1990** pp1024-1030

[57] Moghishi M., *Optical Voltage and current Sensing*, PhD, University of Liverpool, **1990**

[58] Misha A., Mahowald, Mead C. *The Silicon Retina*. Scientific American, pp40-46 May **1991**

[59] Minsky, Marvin Papert, Seymore, *Perceptrons*, Expanded Edition, MIT Press, 1988

[60] Murphy M. M., *Optical Fibre Structural Monitoring*, PhD Thesis, University of Liverpool, 1991

[61] Myllia R., Marsalec E, Kopola K., Wierzba H, *Measurements for meadical applications*, Proc. Int. Centre of Biocybernetics, Warsaw, Poland, Sept. 1991

[62] Niblack W., *An Introduction to Digital Image Processing*, Prentice Hall International, UK. 2nd Ed. 1986.

[63] Narendra K.S. Mukhopadyay S., *Intelligent Control using Neural Networks*, 1EEE Control Systems, April 1992 pp11-18

[64] Pauk. V. N, Makulov V.B. *Software for Image Processing on Personal Computers*, Opt. Eng. April 1992, V31, 4 pp782-788

[65] Poggio T. Girosi F. *Networks for Approximation & Learning.* Proc. IEEE 78. (9) 1481-1497 1990

[66] Ramsey M.M., Hockham G. A., *Propagation in Optical Fibre Waveguides*, in C.P Sandbank,(Ed.) Optical Fibre Communication Systems, J. Wiley, 1980, pp25-41

[67] Raynes E.P., *Electro-optic and thermo-optic effects in liquid crystals*, Phil. trans. R. Soc. Lond. A 309, 1983 pp 167-178

[68] Read H.H.. *Ruthleys Elements of Mineralogy*, 26th Edition, Murby 1981.

[69] Reber W.L. Lyman J. *An Artificial Neural Sytem for Rotating and Scale Invariant Pattern Recognition*, IEEE First Int.Conf. on Neural Networks, Vol 4, San Diego, Calif. pp277-283 June 1987

[70] Roberts L G, *Machine Perception of Three dimensional Solids, Optical and Electro-optical Information Processing.* Ed. J P Tipper et al., MIT Press, Cambridge, Massachusetts 1965.

[71] Rosenfield A., *Picture Processing by Computer*, Academic Press, New York, 1969

[72] Rumelhart D.E., Hinton G.E. R.J. Williams, *Learning Representations by Back-Propagating Errors*, Nature, 323,1986, pp 533-536

[73] Rumelhart D.E., McClelland J.L., *Parallel distributed processing: Exploration in the microstructure of cognition;* Vol 1. Foundations, MIT Press, 1986

[74] Russell P.C., Morse K., Khandakar I., Glavas E., Jones G.R., *Chromatic Modulation Technology for Plasma Processing*, 1990

[75] Schilling D., Belove C., *Electronic Circuits Discrete and Integrated*, Second Edition, Macgraw Hill Book Company, 1979

[76] Senior J.M., *Optical Fibre Communications, Principles and Practice*, Prentice Hall International, 1985

[77] Shanmogam K.S., Duickey F. M., Green J.A., *An Optimal frequency domain filter for edge detection in digital pictures.* IEEE Trans. Pattern Anal. Mach. Intell. 1. (1) pp-49 1979

[78] Shepherd R.N. *Neural Nets for Generalisation & Classification*, Psycological Review 97 pp579-580 1990.

[79] Smith R., Ahmed S., Voudras A,, Spencer J., Russell P.,. Jones G. R, *Chromatic Modulation for optical Fibre sensing Electromagnetic and Speckle noise analysis*, Journal Of Modern Optics Vol 39, 11, **1992** pp2301-2314

[80] Smith R., *Optical Sensors for Radioactive Environment*, PhD. Thesis, University of Liverpool, **1994**

[81] Smith R.V., *A Real Time Interactive IKBS for Plasma Processing,* MSc. Thesis, University of Liverpool, **1992**

[82] Snyder A.W. Young W.R. *Modes of Optical Waveguides* J. Opt. Soc. Am. 68 pp297-309 **1978**

[83] Soland. D.E. Narendra P.M. *Prototype Automatic Target Screener*, Proc. SPIE, Smart Sensors, 178, pp175-174 April **1979**

[84] Sproson W.N., *Colour Science in Television and Display Systems*, Hilger, **1983**

[85] Sutton R.S., Barto A.G., Williams R.J., *Reinforced Learning in Discrete Adaptive Optimal Control*, IEEE Control Systems, April **1992**,pp 19-21

[86] Travis M. *True Color Image Processing on the Desktop.* Photonics Spectra. pp 114-118 March **1991**

[87] Trickler R.L., *Optoelectronic Line Transmission: An Introduction to Fibre Optics,* Hienemann Professional Publication Ltd. **1989**, pp13-21

[88] Unger H.G., *Planar Optical Waveguides and fibres*, Clarendon Press, **1977**

[89] Watrose R.L., *Learning in Artificial for Connectionist Networks,: Applied Gradient Methods of Non-Linear Optimisation*, Proc. 1987 IEEE International Conference on Neural Networks, IEEE Press, New York, **1988** pp619-627

[90] Werbos P.J. *Back propagation Past and Future*, Proceedings. 1988 International Conference on Neural Networks, I(343-353) IEEE Press, New York, **1988**

[91] Weymouth T.E. *Using Object Description in a Schema Network for Machine Vision*, COINS Technical Report 86-124, Univ of Massachusetts **1986**

[92] Willis M. J., Massimo C.D., Montague G.A., Tham M., Morris A. J., *Artificial Neural Networks in Process Engineering*, IEE Proceeding D, Vol 138, 3 May **1991**, pp 256-266

[93] Wolf. H.F.(Ed.) *Handbook of Fibre Optics, Theory and applications*, Granada, **1981**, pp43-152

[94] Wyant J.C. Kolipoulo C. L. Bhushana B. Basila D. *Development of a three dimentional non--contact digital optical profiler.* Trans. ASME J. Tribol. 108 1-8, **1986**

[95] Zandman, *Photoelastic Coatings*, Iowa State University Press, Iowa, **1977**

[96] Zhang Y.X. Shen J.Y. Huang W.Q. *The Simulation and analysis of three WTA neural* network model Patt. Recog. Artif. Intell. 5(1). 1-7 **1992**

Appendix A1

Software listing

FRAME FUNCTIONS

FRAME FUNCTIONS

BCLASS. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure to implement a class called B_reg
which is used to define a direct input and output to the various FGC
ports by encapsulating the port address and data in an object Bclass
inherits its charecteristics from superclass F_reg and describes a byte
type class. The advantage of using this technique is that output for a
ports can be directly implemented as:
 Define a B_reg by : B_reg Control(0x01) where 0x01 is the port address;
 Read the port using : x = *Control where x is a Byte type
 Write to port using  : Control = x where x is a Byte type

 Includes: FCLASS.CPP defines __BCLASS_H

*********************************************************************/



#ifndef __FCLASS_H
#include "c:\final\frame\fclass.cpp"
#endif


#ifndef __BCLASS_H
#define __BCLASS_H

class B_reg:F_reg{

public:
      B_reg(unsigned int x){        //constructor
            init(x);
            }
      unsigned int operator*(void);   //operators overloaded for special
      void operator=(unsigned int x);           // functions
      void operator++(void);
      void operator--(void);

      void operator+=(unsigned int x);
      void operator-=(unsigned int x);
      void operator&=(unsigned int x);
      void operator|=(unsigned int x);
};


void B_reg::operator=(unsigned int x){
      outportb(location,x);
      }

void B_reg::operator++(void){
      outportb(location,inportb(location)+1);
      }
```

```
void B_reg::operator--(void){
      outportb(location,inportb(location)-1);
      }

void B_reg::operator+=(unsigned int x){
      outportb(location,inportb(location)+x);
      }

void B_reg::operator-=(unsigned int x){
      outportb(location,inportb(location)-x);
      }

void B_reg::operator&=(unsigned int x){
      outportb(location,inportb(location)&x);
      }

void B_reg::operator|=(unsigned int x){
      outportb(location,inportb(location)|x);
      }

unsigned int B_reg::operator*(void){
      return inportb(location);
      }
#endif
```

FBOX. CPP

```
/*******************************************************************

This is a Borland C ++ pprocedure to provide box drawing functionality
to FGC Programs.

includes FLINE.CPP,RGB.CPP,MATH.CPP
defines __FBOX.CPP, __minmax
procedures for:
put_fbox()   Put a image from menpry to screen bound by box
get_fbox()   Get an image are from FGC
sel_fbos()   select a boxed region on FGC memory
clear_boxdata() clears data stored in PC memory

*******************************************************************/

#ifndef __FBOX_H
#define __FBOX_H

#ifndef __FLINE_H
#include "c:\final\frame\fline.cpp"
#endif

#ifndef __RGB_H
#include "c:\final\col\RGB.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif


#ifndef __maxmin
#define __maxnin

  int max (int value1, int value2);

  int max(int value1, int value2)
  {
      return ( (value1 > value2) ? value1 : value2);
  }

  int min (int value1, int value2);

  int min(int value1, int value2)
  {
      return ( (value1 < value2) ? value1 : value2);
  }

#endif

/************ globals and prototypes *******************/

RGBREAL *BOX_DATA;
int BOX_STORED;

void put_fbox(int x1,int y1,int x2,int y2, int col);
void get_fbox(int x1,int y1,int x2,int y2,int *sizex,int *sizey);
void sel_fbox(int *x_1,int *x_2,int *y_1,int *y_2);
void clear_boxdata(void);


/*********** Function body starts here ***************** /

void get_fbox(int x1,int y1,int x2,int y2,int *sizex,int *sizey){

int x,y,r,g,b;

BOX_DATA=(RGBREAL *)farmalloc(abs(x1-x2)*abs(y1-y2)*sizeof(RGBREAL));
for(x=min(x1,x2);x<max(x1,x2);x++){
  for(y=min(y1,y2);y<max(y1,y2);y++){
  X=x;
  Y=y;
  get_fpix(&r,&g,&b);
  BOX_DATA[(x-min(x1,x2))*(y-min(y1,y2))]=RGBreal(r,g,b);
  }
}
  BOX_STORED=1;
  *sizex=(abs(x1-x2));
  *sizey=(abs(y1-y2));

}

void put_fbox(int x1,int y1,int x2,int y2, int col){
put_fline(x1,y1,x2,y1,col);
put_fline(x2,y1,x2,y2,col);
put_fline(x2,y2,x1,y2,col);
put_fline(x1,y2,x1,y1,col);
}

void sel_fbox(int *x_1,int *x_2,int *y_1,int *y_2){
int x1,x2,y1,y2;
m_status *m;
unsigned mouse_xval=8,mouse_yval=8;

sel_fpoint(CUR2,F_RED);
x1=*X;
y1=*Y;
X-=8;
Y-=8;
put_cursor(BLNK,F_RED);
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(CUR2,F_RED);
    m=m_pressed(1);

    while(m->button_status!=1)
    {
      m=m_pos();
      if(m->button_status==2){
      put_fbox(x1,y1,*X,*Y,0x00);
      x1=*X;
      y1=*Y;
      }
      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,F_RED);
        put_fbox(x1,y1,*X,*Y,0x00);
        mouse_xval=m->xaxis>>3;
        mouse_yval=m->yaxis>>3;

        X=mouse_xval*8;
        Y=mouse_yval*8;
        put_cursor(CUR2,F_RED);
        put_fbox(x1,y1,*X,*Y,F_RED);

       }
    }

    while(m->button_status) m=m_pos();
    *x_1=x1;
    *x_2=x2;
    *y_1=y1;
    *y_2=y2;
}

#endif
```

FCLASS. CPP

```
/*****************************************************************

Borland C++ class definition for FGC resiter control this is the
superclass from which B_regs and W_regs are defined and should be
included in those modules.
includes DOS.H
defines FBASE_ADDR, __FCLASS_H
superclass F_reg

*****************************************************************/
#ifndef __FCLASS_H
#define __FCLASS_H

#ifndef __DOS_H
#include <dos.h>
#endif

#define FBASE_ADDR       0x300

class F_reg{
protected:
      int location;
public:

      void init(int x){
            location=x+FBASE_ADDR;
            }
      ~F_reg(void){
            location=0;
            }
};


#endif
```

FFUNC. HPP

```
/*******************************************************************

This is a Borland C ++ header block for the Higher level FGC control
functions.

it defines the FGC colourss F_XXX; and the Cursors and defines all the
protoypes neccessary for the FFUNC_CPP.
includes COL.CPP also defines __FFUNC_HPP

********************************************************************/


#ifndef __FFUNC_HPP
#define __FFUNC_HPP

#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif


#define F_WHITE         0x00
#define F_BLUE          0x05
#define F_GREEN        0x0c
#define F_RED                0x03
#define F_BLACK         0x0e
#define F_LIGHTBLUE      0x0d
#define F_CYAN         0x0b
#define F_BROWN        0x0a
#define F_PURPLE       0X09
#define F_LIGHTRED           0X02
#define F_GREY          0x06


void init_refs(void);                 // init reference
void f_init(void);                    // frame initialisation
void clear(void);                     // clears the screen
void snap(void);                      // freezes the screen
void grab(void);                      // continuous acquisition
void freeze(void);                    // freezes FGC operations
void f_wait(void);                    // waits for a frame period.
void zoom(void);                      // zooms image
void unzoom(void);                    // unzooms image
void cls_fovl(void);                  // clears overlay image
void cls_fmem(void);                  // clears FGC Memory block
void get_fpix(int *r,int *g,int *b);  // gets a pixel data
void get_farea(int *r,int *g,int *b); // gets 8X8 data top left corner
void get_farea_max(int *r,int *g,int *b); // get max value of 8x8 area
void flush_buf(void);                     // flush buffer
void put_fpoint(int col);                 // put a point on overlay;
void put_fpix(int red,int green, int blue); // put a pixel on frame
memory.
void put_cursor(int *type, int col);        // draws a cursor
```

```
int BLNK[] =  { 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                 0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

int CUR1[] =  { 0x00,0xC0,0X20,0X10,0X08,0X04,0X04,0x7C,
                0x04,0x04,0x08,0x10,0x20,0xC0,0x00,0x00,
                0x00,0x07,0x09,0x11,0x21,0x41,0x40,0x7C,
                0x40,0x41,0x21,0x11,0x09,0x07,0x00,0x00 };

int CUR2[] = {  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7e,
                0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
                0x00,0x01,0x01,0x01,0x01,0x01,0x00,0xfc,
                0x00,0x01,0x01,0x01,0x01,0x01,0x00,0x00};

int CUR3[] =  {  0x00,0xff,0x01,0x01,0x01,0x01,0x01,0x01,
                0x01,0x01,0x01,0x01,0x01,0x01,0x01,0xff,
                0x00,0xff,0x80,0x80,0x80,0x80,0x80,0x80,
                0x80,0x80,0x80,0x80,0x80,0x80,0x80,0xff};

#endif
```

FFUNC. CPP

```
/**********************************************************************

This is a Borland C ++ program block for the Higher level FGC control
functions.

includes FGREGS.CPP, FFUNC.HPP


**********************************************************************/

#ifndef __FREGS_H
#include "c:\final\frame\fgreg.cpp"
#endif

#ifndef __FFUNC_H
#define __FFUNC_H
#include "c:\final\frame\ffunc.hpp"

void f_init(){
int x,y,z,ad;
int data[45]= {      0x00,0x00,0x00,             //this is the data set for
                        0x20,0x20,0xB0,             //setting the overlay
dac
                        0x20,0xA0,0x20,             //to produce the  right
                        0x30,0xC0,0xC0,             //colourd cursors and
                        0xB0,0x30,0x30,
                        0xB0,0x30,0xB0,
                        0x80,0x40,0x40,
                        0xC0,0xC0,0xCf,
                        0x50,0x50,0x50,
                        0x20,0x20,0xFF,
                        0x20,0xE0,0xE0,
                        0xE0,0x20,0x20,
                        0xE0,0x10,0xE0,
                        0xFF,0xFF,0x10,
                        0xFF,0xFF,0xFF,};

// all registers neccesary for frame initialisation are set here


  CON =0x44;                                         //control register
10001000
  OLUTA     =0X00;
  ACQ =0X00;
  PTRCON    =0X10;                                   //xincrenet by 1
  PANREG    =0X00;
  SCRREG    =0X00;
  VBCON     =0X12;                                   //vedeo control from
camera
  DMASK     =0XFF;
  DACCOM    =0x11;
  GOPBCON   =0X4040;
  GOHMASK   =0X0000;
  GOVMASK   =0X0000;
  RBPBCON   =0X4040;
  RBHMASK   =0X0000;
  RBVMASK   =0X0000;
  X         =0X0000;
  Y         =0X0000;
  init_refs();
                                                           //output luts set
here
  for(x=0;x<=3;x++)
  { y=x;
    y<<3;
    for(z=0;z<=3;z++)
    {
       OLUTCON=y|z;
       for(ad=0;ad<=0xfe;ad++)
       {
         OLUTA=ad;
         OLDATA=ad;
       }
    }
  }
                                                           //input adc luts
set here
  OLUTCON=0x00;
  RALUTA=0x00;
  for(x=0;x<0xff;x++) RALDATA=x;
  GALUTA=0x00;
  for(x=0;x<0xff;x++) GALDATA=x;
  BALUTA=0x00;
  for(x=0;x<0xff;x++) BALDATA=x;
  DLWADR=0x01;
  for(ad=0;ad<0xff;ad++)
    for(x=0;x<3;x++) DLDATA=ad;
  DOWADR=0x01;
  for(x=0;x<45;x++) DODATA=data[x];

  flush_buf();
  RBPBCON&=0x7f7f;                                  //clearing overlay
  GOPBCON&=0x7f7f;                                  //seting zmode and pbuf
  VBCON=0x12;                                       //enabled
  GOHMASK=0x0000;
  RBHMASK=0x0000;
  clear();                                          //full host access
  f_wait();                                         //clearing screen

  GOVMASK=0xff00;                                      //protecting overlay
video access
  RBPBCON&=0xefef;                                  //rgbo in zmode settings
  GOPBCON&=0xefef;

}

void init_refs()
{

  RALUTA  =0X00;                                    //setting up adc positive
  RADCCON =0X00;                                       //and negative
references
  RALUTA  =0X01;                                    //for red
  RNREF     =0X00;
  RALUTA  =0X02;
  RPREF     =0XC0;

  GALUTA  =0X00;                                    //for green
  GADCCON =0X00;
  GALUTA  =0X01;
  GNREF     =0X00;
  GALUTA  =0X02;
  GPREF     =0XC0;

  BALUTA  =0X00;                              //for blue
  BADCCON =0X00;
  BALUTA  =0X01;
  BNREF     =0X00;
  BALUTA  =0X02;
  BPREF     =0XC0;

}
void clear(void){                               //function clear
  int y;
  y=*ACQ;
  y&=0x7F;                                    //performing acq=01xx xxxx
  y|=0x40;                                    //for clear mode
  ACQ=y;
}

void snap(void){                              //function snap
  int y;
  y=*ACQ;
  y&=0xbf;                                    //performing acq=10xx xxxx
  y|=0x80;                                    //for clear mode
  ACQ=y;
}

void grab(void){                              //function grab
  int y;
  y=*ACQ;
  y|=0xd0;                                    //performing acq=11xx xxxx
  ACQ=y;
}

void freeze(void){                            //function freeze
  int y;
  y=*ACQ;
  y&=0x3f;                                    //performing acq=00xx xxxx
  ACQ=y;
}

void f_wait(void){                            //frame wait is a delay
      ACQ=0x40;
      while(*ACQ>=0x40);
}

void zoom(void){                              // zoom function
  CON=*CON|0x01;                              // performing con=xxxx xxx1
}

void unzoom(void){                            //unzoom function
  CON=*CON&0xfe;                              // performing con=xxxx xxx0
}


void get_fpix(int *r,int *g,int *b){
  *r=*RDATA;
  *g=*GDATA;
  *b=*BDATA;

}

void get_farea(int *r,int *g,int *b){
  int x,y,z;
  unsigned _r=0,_g=0,_b=0;
  flush_buf();
  RBPBCON=0x4040;
  GOPBCON=0x4040;
  for(y=1;y<=8;y++){
    *GODATA;
    *RBDATA;
    for(x=1;x<=8;x++){
       z=*RBPBUF;
       _r+=(z&0x00ff);
       _b+=(z&0xff00)>>8;
       z=*GOPBUF;
       _g+=(z&0x00ff);
    }
    ++Y;

  }
  Y-=8;
  *r=_r>>6;
  *g=_g>>6;
  *b=_b>>6;
}
void get_farea_max(int *r,int *g,int *b){
  int x,y,z;
  unsigned _r=0,_g=0,_b=0;
  flush_buf();
  RBPBCON=0x4040;
  GOPBCON=0x4040;
  for(y=1;y<=8;y++){
    *GODATA;
    *RBDATA;
    for(x=1;x<=8;x++){
       z=*RBPBUF;
       if(_r<(z&0x00ff))_r=z&0x00ff;
       if(_b<((z&0xff00)>>8))_b=(z&0xff00)>>8;
       z=*GOPBUF;
       if(_g<(z&0x00ff))_g=z&0x00ff;
    }
    ++Y;

  }
  y-=8;
  *r=_r;
  *g=_g;
  *b=_b;
}

void put_fpoint(int col){
  col=col<<8;
  col|=0xff;
  OLUTCON=0x00;
  GOPBCON=0x4040;
  GOHMASK=0x00ff;
  GODATA=col;
}

void put_fpix(int red,int green, int blue){
  GOHMASK=0xFF00;
  RBHMASK=0x0000;
  GODATA=green;
  RBDATA=(red|blue<<8);
}


void put_cursor(int *type, int col){
int x,y=0,z;
 X-=8;
 Y-=8;
 col=col<<8;
 col|=0xff;
 OLUTCON= 0x00;
 for(z=1;z<=2;z++)
 {
 flush_buf();
 GOHMASK=col;
 GOPBCON&=0xbfbf;
 GODATA=0x00;
 for(x=0;x<=15;x++)
 {
 y++;
 GOPBCON|=0x4040;
 OPBUF=type[y];
 GOPBCON&=0xbfbf;
 ++Y;
 GODATA=0x0000;
 }
 Y-=16;
 X+=8;
 }
 Y+=8;
 X-=8;

}

void cls_fovl()
{

   flush_buf();
   GOPBCON=0x1010;
   VBCON=0x12;
   GOHMASK=0x00FF;
   ACQ=0x40;
   while(*ACQ>=0x40);
   flush_buf();
}

void clear_fmem()
{
   flush_buf();
   GOPBCON=0x1010;
   RBPBCON=0x1010;
   VBCON=0x12;
   GOHMASK=0x0000;
   RBHMASK=0x0000;
   ACQ=0x40;
   while(*ACQ >= 0x40);
   flush_buf();
}
void flush_buf(void){
  RBHMASK=0xffff;                     //protecting the frame
memory
  GOHMASK=0xffff;
  RBPBCON|=0x5050;                    //setting xmode frame
access and
  GOPBCON|=0x5050;                    //enabling pbuf
xxpbcon=x1x1 xxxx x1x1 xxxx
  GODATA=0;                           //clearing buf with 0
  RBDATA=0;
}

#endif
```

FGH. CPP

```
/****************************************************************

This is a Borland C ++ program block for FGC control functions.

it includes all files for minimum FGC functionality of the FGC

******************************************************************/

#ifndef __FGH_H
#define __FGH_H

#include "ffunc.hpp"
#include "mouse.hpp"
#include "fproc.hpp"

#include "fclass.cpp"
#include "bclass.cpp"
#include "wclass.cpp"
#include "fgreg.cpp"
#include "mouse.cpp"
#include "ffunc.cpp"
#include "fproc.cpp"

#endif
```

fregs. CPP

```
/*****************************************************************

This is a Borland C ++ program block for FGC control functions.

It initilalises all register classes (W_regs and B_regs) to hold the
correct addresses of the ports.

******************************************************************/

#ifndef __FREGS_H
#define __FREGS_H

// here all the frame register are created
// for use by all frame handling function

#ifndef __BCLASS_H
#include "c:\final\frame\bclass.cpp"
#endif

#ifndef __WCLASS_H
#include "c:\final\frame\wclass.cpp"
#endif


B_reg CON     (0x00),
      OLUTCON (0x01),
      OLUTA   (0X02),
      PROTECT (0X03),
      ACQ      (0X04),
      PTRCON  (0X05),
      PANREG  (0X06),
      SCRREG  (0X07),
      OLDATA  (0X08),
      VBCON   (0X0A),
      RALUTA  (0X0C),
      RALDATA (0X0D),
      RADCCON (0X0E),
      RNREF   (0X0E),
      RPREF   (0X0E),
      GALUTA  (0X10),
      GALDATA (0X11),
      GADCCON (0X12),
      GNREF   (0X12),
      GPREF   (0X12),
      BALUTA  (0X14),
      BALDATA (0X15),
      BADCCON (0X16),
      BNREF   (0X16),
      BPREF   (0X16),
      DLWADR  (0X18),
      DLDATA  (0X19),
      DMASK   (0X1A),
      DLRADR  (0X1B),
      DOWADR  (0X1C),
      DODATA  (0X1D),
      DACCOM  (0X1E),
      DORADR  (0X1F),
        GPBUF    (0X26),
        OPBUF    (0X27),
        RPBUF    (0X2E),
        BPBUF    (0X2F),
        GDATA    (0X34),
        ODATA    (0X35),
        RDATA    (0X36),
        BDATA    (0X37);

W_reg GOPBCON (0X20),
        GOHMASK (0X22),
        GOVMASK (0X24),
        GOPBUF  (0X26),
        RBPBCON (0X28),
        RBHMASK (0X2A),
        RBVMASK (0X2C),
        RBPBUF  (0X2E),
        X       (0X30),
        Y       (0X32),
        GODATA  (0X34),
        RBDATA  (0X36);

#endif
```

FLINE. CPP

```
/*****************************************************************

This is a Borland C ++ pprocedure to provide line drawing functionality
to FGC Programs.

includes FPOINT.CPP,ALLOC.H
defines __FLINE.CPP,
procedures for:
put_fline()   Put a line on FGC screen
get_fline()   Get a line area from FGC
sel_fline()   select a line on FGC memory
clear_linedata() clears data stored in PC memory

*****************************************************************/

#ifndef __FLINE_H
#define __FLINE_H

#ifndef __FPOINT_H
#include "c:\final\frame\fpoint.cpp"
#endif

#ifndef __ALLOC_H
#include <alloc.h>
#endif

/************ globals and prototypes *******************/

RGBREAL     *LINE_DATA;
int LINE_STORED=0;

void put_fline(int x1, int y1, int x2,int y2, int col);
void get_fline(int x1, int y1, int x2,int y2,int *size);
void sel_fline(int *x_1,int *y_1,int *x_2,int *y_2);
void clear_linedata(void);

/*********** Function body starts here ***************** /

void put_fline(int x1, int y1, int x2,int y2, int col){
int x, y,temp1,temp2;
int z1,z2,z;
x1&=0x3ff;
x2&=0x3ff;
y1&=0x1ff;
y2&=0x1ff;
temp1=*X;
temp2=*Y;
if(x1==x2) z1=0;
if(y1==y2) z2=0;
if(x1>x2) z1=-1;
if(x2>x1) z1=1;
if(y1>y2) z2=-1;
if(y2>y1) z2=1;
if(z1==0&&z2==0)return;
if(((y2-y1)*z2)>((x2-x1)*z1)){
 for(y=0;y<=(y2-y1)*z2;y++){
  Y=y1+y*z2;
  X=x1+(((float)(x2-x1)/(float)(y2-y1))*y*z2);
  put_fpoint(col);
  }
}
else {
 for(x=0;x<=(x2-x1)*z1;x++){
  X=x1+x*z1;
  Y=y1+(((float)(y2-y1)/(float)(x2-x1))*x*z1);
  put_fpoint(col);
  }
}
X=temp1;
Y=temp2;
}
void get_fline(int x1, int y1, int x2,int y2,int *size){
int x, y,temp1,temp2;
int r,g,b;
int z1,z2,z;
if(LINE_STORED)free(LINE_DATA);
LINE_STORED=1;
x1&=0x3ff;
x2&=0x3ff;
y1&=0x1ff;
y2&=0x1ff;
temp1=*X;
temp2=*Y;
if(x1==x2) z1=0;
if(y1==y2) z2=0;
if(x1>x2) z1=-1;
if(x2>x1) z1=1;
if(y1>y2) z2=-1;
if(y2>y1) z2=1;
if(z1==0&&z2==0)return;
if(((y2-y1)*z2)>((x2-x1)*z1)){
 *size=(y2-y1)*z2;
 LINE_DATA=(RGBREAL *)farmalloc(*size*sizeof(RGBREAL));
 x=0;
 for(y=0;y<=(y2-y1)*z2;y++){
  Y=y1+y*z2;
  X=x1+(((float)(x2-x1)/(float)(y2-y1))*y*z2);
  get_fpix(&r,&g,&b);
  LINE_DATA[x].RED=r&0xff;
  LINE_DATA[x].GREEN=g&0xff;
  LINE_DATA[x].BLUE=b&0xff;
  x++;
  }
}
else {
 *size=(x2-x1)*z1;
 LINE_DATA=(RGBREAL *)farmalloc(*size*sizeof(RGBREAL));
 x=0;
 for(x=0;x<=(x2-x1)*z1;x++){
  X=x1+x*z1;
  Y=y1+(((float)(y2-y1)/(float)(x2-x1))*x*z1);
  get_fpix(&r,&g,&b);
  LINE_DATA[x].RED=r&0xff;
  LINE_DATA[x].GREEN=g&0xff;
  LINE_DATA[x].BLUE=b&0xff;
  x++;
  }
```

```
}
X=temp1;
Y=temp2;
}

void clear_linedata(void){
if(LINE_STORED)free(LINE_DATA);
LINE_STORED=0;
}


void sel_fline(int *x_1,int *y_1,int *x_2,int *y_2){
int x1,x2,y1,y2;
m_status *m;
unsigned mouse_xval=8,mouse_yval=8;

sel_fpoint(CUR2,F_RED);
x1=*X;
y1=*Y;
put_cursor(BLNK,F_RED);
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(CUR2,F_RED);
    m=m_pressed(1);
    while(m->button_status!=1)
    {
      m=m_pos();
      if(m->button_status==2){
      put_fline(x1,y1,*X,*Y,0x00);
      x1=*X;
      y1=*Y;
      }

      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,F_RED);
        put_fline(x1,y1,*X,*Y,0x00);
        mouse_xval=m->xaxis>>3;
        mouse_yval=m->yaxis>>3;

        X=mouse_xval*8;
        Y=mouse_yval*8;
        put_cursor(CUR2,F_RED);
        put_fline(x1,y1,*X,*Y,F_RED);

       }
    }
    while(m->button_status) m=m_pos();
    *x_1=x1;
    *x_2=x2;
    *y_1=y1;
    *y_2=y2;
}

#endif
```

FPOINT. CPP

```
/*******************************************************************

This is a Borland C ++ pprocedure to provide point manipulation
functionality to FGC Programs.

includes FPOINT.CPP,MOUSE.CPP,FFUNC.CPP
defines __FPOINT.CPP
procedures for:
init_cursor()  Initialises the cursor
sel_fpoint()   select a point on FGC memory

*******************************************************************/

#ifndef __FPOINT_H
#define __FPOINT_H

#ifndef __MOUSE_H
#include "c:\final\mouse\mouse.cpp"
#endif

#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif

/************ globals and prototypes *******************/

void init_cursor(void);
void sel_fopint(int *curtype,int col);

/*********** Function body starts here ****************** /

void init_cursor(void){
initialize_cursors();
m_ylimit(0,512);
m_xlimit(0,768);
}

void sel_fpoint(int *curtype,int col){
    m_status *m;
    unsigned mouse_xval=8,mouse_yval=8;
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(curtype,col);
    m=m_pressed(1);

    while(!m->button_status)
    {
      m=m_pos();

      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,col);
        mouse_xval=m->xaxis>>3;
        mouse_yval=m->yaxis>>3;
        X=mouse_xval*8;
        Y=mouse_yval*8;
        put_cursor(curtype,col);

      }
    }
    while(m->button_status) m=m_pos();
}
#endif
```

FPROC. CPP

```
/***************************************************************

This is a Borland C ++ procedure to provide Higer level FGC functions
for image selection and manipulation.

includes FPROC.CPP,MOUSE.CPP,FFUNC.CPP,FPROC.CPP

defines __FPROC_HPP, __FPROCS_H
procedures defines: (self explanatory )
 put_image();
 get_image();
 store_image();
 show_image();
 dge_detect();
 ref_adjust();
 sel_fpoint();
 init_cursor();
 sel_fline();
 put_fline();
 put_fbox();
 sel_fbox();

***************************************************************/

#ifndef  __FPROC_HPP
#define __FPROC_HPP

/************ globals and prototypes *******************/

void put_image(void);
void get_image(void);
void store_image(void);
void show_image(void);
void edge_detect(void);
void ref_adjust(int red,int green, int blue);
void sel_fpoint(int *curtype,int col);
void init_cursor(void);
void sel_fline(void);
void put_fline(int x1,int y1,int x2, int y2, int col);
void put_fbox(int x1,int y1,int x2, int y2, int col);
void sel_fbox(void);

#endif

#ifndef __FPROCS_H
#define __FPROCS_H

#ifndef __MOUSE_H
#include "c:\final\mouse\mouse.cpp"
#endif

#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif

#include "c:\final\frame\fproc.hpp"
```

```
void init_cursor(void){
initialize_cursors();
m_ylimit(0,512);
m_xlimit(0,768);
}

void sel_fpoint(int *curtype,int col){
    m_status *m;
    unsigned mouse_xval=8,mouse_yval=8;
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(curtype,col);
    m=m_pressed(1);

    while(!m->button_status)
    {
      m=m_pos();

      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,col);
        mouse_xval=m->xaxis>>3;
        mouse_yval=m->yaxis>>3;

        X=mouse_xval*8;
        Y=mouse_yval*8;
        put_cursor(curtype,col);

       }
    }
    while(m->button_status) m=m_pos();
}
void put_fbox(int x1,int y1,int x2,int y2, int col){
put_fline(x1,y1,x2,y1,col);
put_fline(x2,y1,x2,y2,col);
put_fline(x2,y2,x1,y2,col);
put_fline(x1,y2,x1,y1,col);
}

void put_fline(int x1, int y1, int x2,int y2, int col){
int x, y,temp1,temp2;
int z1,z2,z;
x1&=0x3ff;
x2&=0x3ff;
y1&=0x1ff;
y2&=0x1ff;
temp1=*X;
temp2=*Y;
if(x1==x2) z1=0;
if(y1==y2) z2=0;
if(x1>x2) z1=-1;
if(x2>x1) z1=1;
if(y1>y2) z2=-1;
if(y2>y1) z2=1;
if(z1==0&&z2==0)return;
if(((y2-y1)*z2)>((x2-x1)*z1)){
 for(y=0;y<=(y2-y1)*z2;y++){
  Y=y1+y*z2;
  X=x1+(((float)(x2-x1)/(float)(y2-y1))*y*z2);
  put_fpoint(col);
  }
}
else {
 for(x=0;x<=(x2-x1)*z1;x++){
  X=x1+x*z1;
  Y=y1+(((float)(y2-y1)/(float)(x2-x1))*x*z1);
  put_fpoint(col);
  }
}
X=temp1;
Y=temp2;
}
void get_fline(int x1, int y1, int x2,int y2){
int x, y,temp1,temp2;
int z1,z2,z;
x1&=0x3ff;
x2&=0x3ff;
y1&=0x1ff;
y2&=0x1ff;
temp1=*X;
temp2=*Y;
if(x1==x2) z1=0;
if(y1==y2) z2=0;
if(x1>x2) z1=-1;
if(x2>x1) z1=1;
if(y1>y2) z2=-1;
if(y2>y1) z2=1;
if(z1==0&&z2==0)return;
if(((y2-y1)*z2)>((x2-x1)*z1)){
 for(y=0;y<=(y2-y1)*z2;y++){
  Y=y1+y*z2;
  X=x1+(((float)(x2-x1)/(float)(y2-y1))*y*z2);
  }
}
else {
 for(x=0;x<=(x2-x1)*z1;x++){
  X=x1+x*z1;
  Y=y1+(((float)(y2-y1)/(float)(x2-x1))*x*z1);
  }
}
X=temp1;
Y=temp2;
}

void sel_fline(int *x_1,int *y_1,int *x_2,int *y_2){
int x1,x2,y1,y2;
m_status *m;
unsigned mouse_xval=8,mouse_yval=8;

sel_fpoint(CUR2,F_RED);
x1=*X;
y1=*Y;
put_cursor(BLNK,F_RED);
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(CUR2,F_RED);
    m=m_pressed(1);
    while(m->button_status!=1)
    {
      m=m_pos();
      if(m->button_status==2){
      put_fline(x1,y1,*X,*Y,0x00);
      x1=*X;
      y1=*Y;
      }

      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,F_RED);
        put_fline(x1,y1,*X,*Y,0x00);
        mouse_xval=m->xaxis>>3;
        mouse_yval=m->yaxis>>3;

        X=mouse_xval*8;
        Y=mouse_yval*8;
        put_cursor(CUR2,F_RED);
        put_fline(x1,y1,*X,*Y,F_RED);

       }
    }
    while(m->button_status) m=m_pos();
    *x_1=x1;
    *x_2=x2;
    *y_1=y1;
    *y_2=y2;
}
void sel_fbox(){
int x1,x2,y1,y2;
m_status *m;
unsigned mouse_xval=8,mouse_yval=8;

sel_fpoint(CUR2,F_RED);
x1=*X;
y1=*Y;
X-=8;
Y-=8;
put_cursor(BLNK,F_RED);
    m_hide();
    m_moveto(64,64);
    X=64;
    Y=64;
    put_cursor(CUR2,F_RED);
    m=m_pressed(1);

    while(m->button_status!=1)
    {
      m=m_pos();
      if(m->button_status==2){
      put_fbox(x1,y1,*X,*Y,0x00);
      x1=*X;
      y1=*Y;
      }
      if(((m->xaxis>>3)!=mouse_xval)||((m->yaxis>>3)!=mouse_yval))
       {
        put_cursor(BLNK,F_RED);
```

```
          put_fbox(x1,y1,*X,*Y,0x00);
          mouse_xval=m->xaxis>>3;
          mouse_yval=m->yaxis>>3;

          X=mouse_xval*8;
          Y=mouse_yval*8;
          put_cursor(CUR2,F_RED);
          put_fbox(x1,y1,*X,*Y,F_RED);

         }
     }

     while(m->button_status) m=m_pos();
}

#endif
```

WCLASS. CPP

```
/********************************************************************
 This is a Borland C ++ procedure to implement a class called W_reg
which is used to define a direct input and output to the various FGC
ports by encapsulating the port address and data in an object Wclass
inherits its charecteristics from superclass F_reg and describes a Word
type class. The advantage of using this technique is that output for a
ports can be directly implemented as:
 Define a W_reg by : W_reg Control(0x01) where 0x01 is the port address;
 Read the port using : x = *Control where x is a Word type
 Write to port using  : Control = x where x is a Word type

 Includes: FCLASS.CPP defines __WCLASS_H

********************************************************************/

#ifndef __WCLASS_H
#define __WCLASS_H

#ifndef __FCLASS_H
#include "c:\final\frame\fclass.cpp"
#endif

class W_reg:F_reg{

public:
      W_reg(unsigned int x){
            init(x);
            }
      unsigned int operator*(void);

      void operator=(unsigned int x);
      void operator++(void);
      void operator--(void);

      void operator+=(unsigned int x);
      void operator-=(unsigned int x);
      void operator&=(unsigned int x);
      void operator|=(unsigned int x);

};


void W_reg::operator=(unsigned int x){
      outport(location,x);
      }

void W_reg::operator++(void){
      outport(location,unsigned(inport(location))+1);
      }

void W_reg::operator--(void){
      outport(location,unsigned(inport(location))-1);
      }

void W_reg::operator+=(unsigned int x){
      outport(location,unsigned(inport(location))+x);
      }

void W_reg::operator-=(unsigned int x){
      outport(location,unsigned(inport(location))-x);
      }

void W_reg::operator&=(unsigned int x){
      outport(location,inport(location)&x);
      }

void W_reg::operator|=(unsigned int x){
      outport(location,inport(location)|x);
      }
unsigned int W_reg::operator*(void){
      return inport(location);
      }
#endif
```

IMAGE PROCESSING FUNCTIONS

AMALG. CPP

```
/*********************************************************************

This is a Borland C ++ implementation of the amalgamate all corner data
to determine if a set of data defines the bounds of a box.

includes IMPROC.CPP
defines __AMAL_H
Functios:

amalg_corner_data()

*********************************************************************/


#ifndef _AMALG_H
#define _AMALG_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif



int amalg_corner_data();

int amalg_corner_data(){
if(!CORNER_STORED) return 0;
int i,j,k,l;
int x,y;

for(y=0;y<PIC_Y-3;y++){
 for(x=0;x<PIC_X-3;x++){
  i=CORNER_DATA[x+y*PIC_X];
  j=CORNER_DATA[(x+1)+y*PIC_X];
  k=CORNER_DATA[x+(y+1)*PIC_X];
  l=CORNER_DATA[x+(y+1)*PIC_X];
  CORNER_DATA[x+y*PIC_X]=0;
  if(i||j||k||l){
   if((i==l)||(j==k))CORNER_DATA[x+y*PIC_X]=1;
   }
 }
}
return 1;
}

#endif
```

CONCOR. CPP

```
/*********************************************************************

This is a Borland C ++ implementation for sensor identification
purposes. it uses calc fit to determine how close a fit is with the
sensor parameters, Extracts features like Box in box (sensor design) And
sorts Corner data.

includes conio.h,stdio.h,improc.cpp,math.h
defines _CONCOR_H
Functios:

calc_fit();
find_boxinbox();
concor_detect();
sort_corner_data();
clear_concor();
concor_detect()


*********************************************************************/

#ifndef _CONCOR_H
#define _CONCOR_H

#ifndef __CONIO_H
#include <conio.h>
#endif

#ifndef __STDIO_H
#include <stdio.h>
#endif

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif


#define SENSOR_RATIO 6.2

float calc_fit(unsigned x,unsigned y);
int find_boxinbox(int far **corner_data,int abox,int *lastbox);
int concor_detect();
float sort_corner_data(int far **corner_data);
void clear_concor(void);

int concor_detect(){
int far *corner_data[4];
int w,x,y;
float conf;
int xa,xb,ya,yb;
if(!CONNECT_STORED) return 0;

corner_data[0]=(int far * )farmalloc(256*sizeof(int));
```

```
corner_data[1]=(int far * )farmalloc(256*sizeof(int));
corner_data[2]=(int far * )farmalloc(256*sizeof(int));
corner_data[3]=(int far * )farmalloc(256*sizeof(int));

for(x=0;x<256;x++){
 corner_data[0][x]=PIC_X;
 corner_data[1][x]=0;
 corner_data[2][x]=PIC_Y;
 corner_data[3][x]=0;
 }

for(x=0;x<PIC_X*PIC_Y;x++){
 y=CONNECT_DATA[x];
 xtoxy(x,&xa,&ya);
 if(xa<corner_data[0][y])corner_data[0][y]=xa;
 if(xa>corner_data[1][y])corner_data[1][y]=xa;
 if(ya<corner_data[2][y])corner_data[2][y]=ya;
 if(ya>corner_data[3][y])corner_data[3][y]=ya;
 }

if((conf=sort_corner_data(corner_data))==0)printf("\nno match found");
printf("\nprocess confidence is %f",conf);

farfree(corner_data[0]);
farfree(corner_data[1]);
farfree(corner_data[2]);
farfree(corner_data[3]);

return 1;

}
int find_boxinbox(int far **corner_data,int abox,int *y){
int x=1,x1,y1,x2,y2,xa,xb,ya,yb;
int flag=-1;

  xa=corner_data[0][abox];
  xb=corner_data[1][abox];
  ya=corner_data[2][abox];
  yb=corner_data[3][abox];

  while(corner_data[1][x]){
      x1=corner_data[0][x];
      x2=corner_data[1][x];
      y1=corner_data[2][x];
      y2=corner_data[3][x];

      if((ya<=y1)&&(yb>=y2)&&(xa<=x1)&&(xb>=x2)) {flag++;
                                                   *y=x;
                                                   }

      x++;
  }
  return flag;

}


float sort_corner_data(int far **corner_data){
int x=1,y;
int bestx,besty;
```

```
OBJECTS_FOUND=0;
float best_fit=0;
while(corner_data[1][x]){
  if(find_boxinbox(cornɘr_data,x,&y)==1){
       OBJECTS_FOUND++;

       if(best_fit<calc_fit(x,y)){
             best_fit=calc_fit(x,y);
             bestx=x;
             besty=y;
       }
      }
  x++;
}
      if(best_fit){
             OBJECTS_CORNERS[0][0]=corner_data[0][besty];
             OBJECTS_CORNERS[0][1]=corner_data[1][besty];
             OBJECTS_CORNERS[0][2]=corner_data[2][besty];
             OBJECTS_CORNERS[0][3]=corner_data[3][besty];

             OBJECTS_CORNERS[1][0]=corner_data[0][bestx];
             OBJECTS_CORNERS[1][1]=corner_data[1][bestx];
             OBJECTS_CORNERS[1][2]=corner_data[2][bestx];
             OBJECTS_CORNERS[1][3]=corner_data[3][bestx];

         }
      if(GROUP_SIZE!=NULL){farfree(GROUP_SIZE);
      GROUP_SIZE=NULL;
      }
      OBJ_CONF=best_fit;
      return best_fit;
}


float calc_fit(unsigned x,unsigned y){
  float temp;
  temp=GROUP_SIZE[x];
  temp/=GROUP_SIZE[y];
  temp/=SENSOR_RATIO;
  temp*=100;
  if(temp>100){
       temp=100-(temp-100);
       }
  return temp;
}

#endif
```

CORNER. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure for corner detection from Sobel
images. The corner detect program detects corners from points which show
changes in Sobel angles with their neighbours.

includes   improc.cpp,math.h
defines _CORNER_H

Function Corner_detect();
*********************************************************************/


#ifndef _CORNER_H
#define _CORNER_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif



int corner_detect();
unsigned char corn_sort(int i,int j,int k,int l);
void clear_corner(void);

int corner_detect(){
int i,j,k,l;
int x,y;
unsigned char *pntr;
farfree(CORNER_DATA);
if(!SOBEL_STORED) return 0;
CORNER_DATA=(unsigned char *)farmalloc((PIC_X)*(PIC_Y)*sizeof(unsigned
char));
if(CORNER_DATA==NULL) return 0;
pntr=CORNER_DATA;

for(y=0;y<PIC_Y-3;y++){
 for(x=0;x<PIC_X-3;x++){
  i=SOBEL_DATA[x+y*PIC_X];
  j=SOBEL_DATA[(x+1)+y*PIC_X];
  k=SOBEL_DATA[x+(y+1)*PIC_X];
  l=SOBEL_DATA[(x+1)+(y+1)*PIC_X];
  if(i||j||k||l){
      *pntr=corn_sort(i,j,k,l);

  }
  else *pntr=0;
  pntr++;
  }
  *pntr=0;
  pntr++;
  *pntr=0;
  pntr++;
  *pntr=0;
  pntr++;
 }
 for(x=0;x<PIC_X*3;x++){
      *pntr=0;
      pntr++;
      }
 CORNER_STORED=1;
 return 1;
}
void clear_corner(void){
farfree(CORNER_DATA);
CORNER_STORED=0;
}

unsigned char corn_sort(int i,int j,int k,int l){
 int x[4],flag=0;
 if(i==1||j==1||k==1||l==1){x[flag]=1;
                                 flag++;
                                };
 if(i==2||j==2||k==2||l==2){x[flag]=2;
                                 flag++;
                                };
 if(i==3||j==3||k==3||l==3){x[flag]=3;
                                 flag++;
                                };
 if(i==4||j==4||k==4||l==4){x[flag]=4;
                                 flag++;
                                };

 if(flag==2){
  if((x[0]==1)&&(x[1]==4))return 1;
  else return(x[flag-1]);
  }
 else return 0;

}
#endif
```

FBOUND. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure for Image tracking by centre
relocation

includes   math.h,ffunc.cpp,improc.cpp
defines __FBOUND_H

Function Corner_detect();
**********************************************************************/


#ifndef __FBOUND_H
#define __FBOUND_H


#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif

#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif

/************ globals and prototypes *******************/
int find_bounds(int *cx,int *cy);
int find_ref_bounds(int *cx,int *cy);

/*********** Function body starts here ****************** /

int find_ref_bounds(int *cx,int *cy){
int xstart,ystart;
int walk_range;
int r,g,b;
int rl,gl,bl;
int flag;
int track_status=1;
unsigned int top,left,bottom,right;
int THRESH=40,m_walk=10;
static int lx=0,ly=0;
xstart=*cx;
ystart=*cy;
X=xstart;
Y=ystart;
get_farea(&rl,&gl,&bl);

walk_range=0;
while(flag)
 {
 flag=1;
 walk_range++;
 X-=8;
 get_farea(&r,&g,&b);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
       left=*X&0x03ff;
       flag=0;
       }
 if(walk_range>m_walk){flag=0;
                               track_status=0;
                               }
 r=rl;
 g=gl;
 b=bl;
}
walk_range=0;
flag=1;
X=xstart;
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
 {
 X+=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    right=*X&0x03ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                            track_status=0;
                            }
 r=rl;
 g=gl;
 b=bl;
}

walk_range=0;
flag=1;
X=xstart;
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
 {
 Y+=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    bottom=*Y&0x01ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                            track_status=0;
                            }
 r=rl;
 g=gl;
 b=bl;
}
walk_range=0;
flag=1;
X=xstart;
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
```

```
 {
 Y-=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
        top=*Y&0x3ff;
        flag=0;
        }
 if(walk_range>m_walk){flag=0;
                             track_status=0;
                             }
 r=rl;
 g=gl;
 b=bl;
}



*cx=(right-left)/2+left;
*cx=*cx&0xfff8;
*cy=(bottom-top)/2+top;
*cy=*cy&0xfff8;

X=lx;
Y=ly;
put_cursor(BLNK,F_RED);
get_farea(&rl,&gl,&bl);
X=*cx;
Y=*cy;
put_cursor(CUR2,F_RED);
get_farea(&r,&g,&b);
if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH))
track_status=0;
lx=*cx;
ly=*cy;

return track_status;
}


int find_bounds(int *cx,int *cy){
int xstart,ystart;
int walk_range;
int r,g,b;
int rl,gl,bl;
int flag;
int track_status=1;
unsigned int top,left,bottom,right;
int THRESH=40,m_walk=20;
static int lx=0,ly=0;
xstart=*cx;
ystart=*cy;
X=xstart;
Y=ystart;
get_farea(&rl,&gl,&bl);

walk_range=0;
while(flag)
 {
 flag=1;
 walk_range++;
 X-=8;
 get_farea(&r,&g,&b);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    left=*X&0x03ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                            track_status=0;
                            }
 r=rl;
 g=gl;
 b=bl;
}
walk_range=0;
flag=1;
X=xstart;
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
 {
 X+=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    right=*X&0x03ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                            track_status=0;
                            }
 r=rl;
 g=gl;
 b=bl;
}

walk_range=0;
flag=1;
X=xstart;
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
 {
 Y+=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    bottom=*Y&0x01ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                            track_status=0;
                            }
 r=rl;
 g=gl;
 b=bl;
}
walk_range=0;
flag=1;
X=xstart;
```

```
Y=ystart;
get_farea(&r,&g,&b);
while(flag)
 {
 Y-=8;
 walk_range++;
 get_farea(&rl,&gl,&bl);
 if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH)){
    top=*Y&0x3ff;
    flag=0;
    }
 if(walk_range>m_walk){flag=0;
                           track_status=0;
                           }
 r=rl;
 g=gl;
 b=bl;
}


*cx=(right+left)/2;
*cx=*cx&0xfff8;
*cy=(bottom+top)/2;
*cy=*cy&0xfff8;

X=lx;
Y=ly;
put_cursor(BLNK,F_RED);
get_farea(&rl,&gl,&bl);
X=*cx;
Y=*cy;
put_cursor(CUR2,F_RED);
get_farea(&r,&g,&b);
if((abs(rl-r)>THRESH)||(abs(gl-g)>THRESH)||(abs(bl-b)>THRESH))
track_status=0;
lx=*cx;
ly=*cy;

return track_status;
}

#endif
```

FCONNECT. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure for implementing connected components
algorithm on GRAYSCALE Images.

includes  gray.cpp,math.h,improc.cpp
defines _FCONNECT_H

*********************************************************************/


#ifndef _FCONNECT_H
#define _FCONNECT_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __GSTORE_H
#include "c:\final\proc\gray.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif
#define  CON_THRESH     5
int inbounds(int x,int y);
int image_connect(void);
void group_this(int x,int far *con_stack);
void clear_connect(void);
int find_near(int start);
void reassign_group(int start,int obj_cnt);
int obj_cnt;

int image_connect(){
int x;
int far *con_stack;
if(!GRAY_STORED) gray_store();

if(CONNECT_DATA!=NULL){farfree(CONNECT_DATA);
                                CONNECT_DATA=NULL;
                                }
CONNECT_DATA=(unsigned char far*)farmalloc(PIC_X*PIC_Y*sizeof(unsigned
char));
if(CONNECT_DATA==NULL) return 0;
if(GROUP_SIZE!=NULL){farfree(GROUP_SIZE);
                            GROUP_SIZE=NULL;
                            }
GROUP_SIZE=(unsigned far *)farmalloc(256*sizeof(unsigned));
if(GROUP_SIZE==NULL) return 0;
obj_cnt=1;
con_stack=(int far *)farmalloc(PIC_X*PIC_Y*sizeof(int));
if(con_stack==NULL) return 0;
for(x=0;x<PIC_X*PIC_Y;x++)CONNECT_DATA[x]=0;
for(x=0;x<PIC_X*PIC_Y;x++){
  if(CONNECT_DATA[x]==0)group_this(x,con_stack);
  CONNECT_STORED=1;
```

```
}
farfree(con_stack);
return 1;
}

void clear_connect(void){
farfree(CONNECT_DATA);
CONNECT_DATA=NULL;
CONNECT_STORED=0;
}

void group_this(int start,int far *con_stack){
unsigned objsize=0,mstkl=0;
int x,z,xa,ya;
int stk_cnt=1;
con_stack[stk_cnt]=start;
CONNECT_DATA[start]=obj_cnt;

while(stk_cnt){
  x=con_stack[stk_cnt];
  stk_cnt--;
  objsize++;
  xtoxy(x,&xa,&ya);
  z=x-1;
  if(xa){
    if(CONNECT_DATA[z]==0&&inbounds(GRAY_DATA[x],GRAY_DATA[z])){
      stk_cnt++;
      if(mstkl<stk_cnt)mstkl=stk_cnt;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=obj_cnt;
      }
  }
  z=x+1;
  if(xa<PIC_X){
  if(CONNECT_DATA[z]==0&&inbounds(GRAY_DATA[x],GRAY_DATA[z])){
      stk_cnt++;
      if(mstkl<stk_cnt)mstkl=stk_cnt;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=obj_cnt;

      }
  }
  z=x+PIC_X;
  if(ya<PIC_Y){
  if(CONNECT_DATA[z]==0&&inbounds(GRAY_DATA[x],GRAY_DATA[z])){
      stk_cnt++;
      if(mstkl<stk_cnt)mstkl=stk_cnt;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=obj_cnt;

      }
  }
  z=x-PIC_X;
  if(ya){
  if(CONNECT_DATA[z]==0&&inbounds(GRAY_DATA[x],GRAY_DATA[z])){
      stk_cnt++;
      if(mstkl<stk_cnt)mstkl=stk_cnt;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=obj_cnt;
      }
  }

 }
 if(objsize>10&&mstkl>5)  {
        GROUP_SIZE[obj_cnt]=objsize;
        obj_cnt++;
       }

 else reassign_group(start,obj_cnt);

}

int find_near(int start){
int a,b,c;
int x,y,z;
z=0;
xtoxy(start,&x,&y);
if (x&&y)  z=CONNECT_DATA[xytox(x-1,y-1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (x)  z=CONNECT_DATA[xytox(x-1,y)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (y)  z=CONNECT_DATA[xytox(x,y-1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (y<PIC_Y)  z=CONNECT_DATA[xytox(x,y+1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (x<PIC_X)  z=CONNECT_DATA[xytox(x+1,y)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (x<PIC_X&&y<PIC_Y)  z=CONNECT_DATA[xytox(x+1,y+1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (x&&y<PIC_Y)  z=CONNECT_DATA[xytox(x-1,y+1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

if (x<PIC_X&&y)  z=CONNECT_DATA[xytox(x+1,y-1)];
if (z&&(z!=CONNECT_DATA[start]))return z;

return 0;
}

void reassign_group(int start,int obj_cnt){
int con_stack[10],stk_cnt=1;
int x,near_obj,z,xa,ya;
unsigned objsize=0;
con_stack[1]=start;

near_obj=find_near(start);
CONNECT_DATA[start]=near_obj;

while(stk_cnt){
  x=con_stack[stk_cnt];
  stk_cnt--;
  objsize++;
  xtoxy(x,&xa,&ya);
  z=x-1;
  if(xa>0){
    if(CONNECT_DATA[z]==obj_cnt){
      stk_cnt++;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=near_obj;
      }
  }
  z=x+1;
  if(xa<PIC_X){
  if(CONNECT_DATA[z]==obj_cnt){
      stk_cnt++;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=near_obj;
      }
  }
  z=x+PIC_X;
  if(ya<PIC_Y){
  if(CONNECT_DATA[z]==obj_cnt){
      stk_cnt++;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=near_obj;

      }
  }
  z=x-PIC_X;
  if(ya>0){
  if(CONNECT_DATA[z]==obj_cnt){
      stk_cnt++;
      con_stack[stk_cnt]=z;
      CONNECT_DATA[z]=near_obj;

      }
  }

 }

 GROUP_SIZE[near_obj]+=objsize;
}

int inbounds(int x,int y){
 if((abs(x-y))<CON_THRESH)return 1;
 else return 0;
}

#endif
```

```
EDGE_STORED=1;
return 1;

}

void clear_edge(void){
farfree(EDGE_DATA);
EDGE_STORED=0;
}

#endif
```

FNDBOX. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure for locating boxes using prrocedures
written in other modules.

includes  stdio.h,math.h,improc.cpp,screen.cpp,amalg.cpp
defines _BOXFIT_H


**********************************************************************/



#ifndef _BOXFIT_H
#define _BOXFIT_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __GRAYS_H
#include "c:\final\col\gray.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif

#ifndef __STDIO_H
#include <stdio.h>
#endif


#ifndef _CSCRH
#include "c:\final\scr\screen.cpp"
#endif

#ifndef __AMALG_H
#include "c:\final\proc\amalg.cpp"
#endif

#ifndef __CONIO_H
#include <conio.h>
#endif

typedef struct {int x, to;}box_id;
int box_detect(void);
void store_corner(int x,int col);
float fit_sobline(int x1,int x2,int col);
box_id far *corn_store[4];
void find_corner_pairs(int col1,int col2);
void find_full_box(void);
void show_corn_data(void);

void show_corn_data(void){
int x,y;
int xa,ya;
```

```
for(y=0;y<4;y++){
  x=0;
  while(corn_store[y][x].x){
    xtoxy(corn_store[y][x].x,&xa,&ya);
    putpixel(xa,ya,y+1);
    x++;
    }

  }
  getch();
}

int box_detect(void){
int x,y;
init_screen();
for(x=0;x<4;x++)
if((corn_store[x]=(box_id
far*)farmalloc(100*sizeof(box_id)))==NULL)return 0;
if(!CORNER_STORED) return 0;
if(!SOBEL_STORED) return 0;
if(!amalg_corner_data()) return 0;
for(x=0;x<(PIC_X)*(PIC_Y-4);x++)
 if(CORNER_DATA[x])store_corner(x,CORNER_DATA[x]);
 show_corn_data();
 find_corner_pairs(4,1);
 find_corner_pairs(1,2);
 find_corner_pairs(2,3);
 find_corner_pairs(3,4);
 find_full_box();
 for(x=1;x<4;x++)
 farfree(corn_store[x]);
 getch();
 closegraph();
 return 1;
}

void find_corner_pairs(int col1,int col2){
int x=0,y=0,flag;
col1--;
col2--;
int xc1,xc2,yc1,yc2;
int x1,x2;
int bestx;
float bestfit,afit;
while((x1=corn_store[col1][x].x)>0){
   bestx=0;
   bestfit=0.5;
   y=0;
   flag=0;
   while((x2=corn_store[col2][y].x)>0){
        afit=fit_sobline(x1,x2,col1);
        if(afit>bestfit){
          bestfit=afit;
          bestx=y;
          flag=1;
          }
      y++;
   }
  if(flag){
  xtoxy(x1,&xc1,&yc1);
  putpixel(xc1,yc1,col1+1);
  xtoxy(corn_store[col2][bestx].x,&xc2,&yc2);
  putpixel(xc2,yc2,col2+1);
  setcolor(6);
  line(xc1,yc1,xc2,yc2);
   corn_store[col1][x].to=bestx;
   }
  x++;
 }
}

void find_full_box(void){
int f1,f2,f3,f4;
int x1,y1;
int x=0;
while(corn_store[1][x].x){
  if((f1=corn_store[1][x].to)>0){
    if((f2=corn_store[2][f1].to)>0){
       if((f3=corn_store[3][f2].to)>0){
         if((f4=corn_store[0][f3].to)==x)  {


         xtoxy(corn_store[3][f2].x,&x1,&y1);
         printf("\nred coords %d %d",x1,y1);
         xtoxy(corn_store[0][f3].x,&x1,&y1);
         printf("\nblue coords %d %d",x1,y1);
         xtoxy(corn_store[1][f4].x,&x1,&y1);
         printf("\ngreen coords %d %d",x1,y1);
         xtoxy(corn_store[2][f1].x,&x1,&y1);
         printf("\n cyan coords %d %d",x1,y1);
         }
       }
    }
  }
 x++;
 }
}

void store_corner(int x,int col){
static int stk_cntr[4]={0,0,0,0};
col--;
corn_store[col][stk_cntr[col]].x=x;
corn_store[col][stk_cntr[col]+1].x=0;
corn_store[col][stk_cntr[col]].to=0;
stk_cntr[col]++;
}


float fit_sobline(int x1,int x2,int col){
int x,y ;
float dstore=0.0,dcount=0.0;
int xa,ya,xb,yb;
int xstep,ystep;
float m,c;
xtoxy(x1,&xa,&ya);
xtoxy(x2,&xb,&yb);
xstep=xb-xa;
ystep=yb-ya;
```

```
if(abs(xstep)>abs(ystep)){
x=xa;
m=float(float(ystep)/float(xstep));
c=ya-m*xa;
while(x!=xb){
 y=nint(m*float(x)+c);
 x+=xstep/abs(xstep);
 if(SOBEL_DATA[xytox(x,y)]==col+1)dstore++;
 dcount++;
 }


}
else{

y=ya;
m=float(float(xstep)/float(ystep));
c=xa-m*ya;
while(y!=yb){
 x=nint(m*float(y)+c);
 y+=ystep/abs(ystep);
 if(SOBEL_DATA[xytox(x,y)]==col+1)dstore++;
 dcount++;
 }
}
dstore/=dcount+1;
return(dstore);
}

#endif
```

GRAY. CPP

```
/*********************************************************************
 This is a Borland C ++ procedure for Converting FGC Images to Gray
Scale Images

includes  gray.cpp,pstore.cpp,improc.cpp
defines _GSTORE_H


**********************************************************************/

#ifndef _GSTORE_H
#define _GSTORE_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __GRAY_H
#include "c:\final\col\gray.cpp"
#endif

#ifndef __PSTORE_H
#include "c:\final\proc\pstore.cpp"
#endif

void clear_gray(void);
int gray_store(void);

int gray_store(void){

int x;
if(!PICT_STORED) store_screen();

if(GRAY_DATA!=NULL){
            farfree(GRAY_DATA);
            GRAY_DATA=NULL;
            }
GRAY_DATA=(unsigned char far *)farmalloc(PIC_X*PIC_Y*sizeof(unsigned
char));
if(GRAY_DATA==NULL) return 0;
for(x=0;x<(PIC_X*PIC_Y);x++){
 GRAY_DATA[x]=grayscale(PICT_DATA[x]);
 }
 GRAY_STORED=1;
 clear_pict();
 return 1;
}

void clear_gray(void){

farfree (GRAY_DATA);
GRAY_DATA=NULL;
GRAY_STORED=0;
}

#endif
```

HISTO. CPP

```
/**********************************************************************
 This is a Borland C ++ procedure for Generating Histogram data of the
FGC Imagess.

includes  rgb.cpp,improc.cpp
defines _HISTO_H


***********************************************************************/

#ifndef _HISTO_H
#define _HISTO_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __RGB_H
#include "c:\final\col\rgb.cpp"
#endif

void clear_histo(void);
int histogram(void);

int histogram(void){

int x;
RGBREAL c;
if(!PICT_STORED) return 0;

if(HISTO_DATA!=NULL)farfree(HISTO_DATA);
HISTO_DATA=(RGBREAL *)farmalloc(256*sizeof(RGBREAL));
if(HISTO_DATA==NULL) return 0;
for(x=0;x<(PIC_X*PIC_Y);x++){
  HISTO_DATA[x].RED=0;
  HISTO_DATA[x].GREEN=0;
  HISTO_DATA[x].BLUE=0;
  }

for(x=0;x<(PIC_X*PIC_Y);x++){
 c=RGBreal(PICT_DATA[x]);
 HISTO_DATA[c.RED].RED++;
 HISTO_DATA[c.GREEN].GREEN++;
 HISTO_DATA[c.BLUE].BLUE++;
 }
 HISTO_STORED=1;
 return 1;
}
void clear_histo(void){

farfree (HISTO_DATA);
HISTO_STORED=0;
}

#endif
```

IMPROC. CPP

```
/***********************************************************************
 This is a Borland C ++ procedure for general Image processing routines.
All finctions and global variables are defined here.
includes math.h,col.cpp
defines PIC_X,PIC_Y,__IMPROC_H
***********************************************************************/


#ifndef __IMPROC_H
#define __IMPROC_H

#ifndef __MATH_H
#include <math.h>
#endif
#ifndef __ALLOC_H
#include <alloc.h>
#endif

#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif
#define PIC_X      96
#define PIC_Y      64

void xtoxy(int xval,int *x,int *y);
int xytox(int x,int y);
int nint(float x);

RGBREAL far *PICT_DATA;
unsigned char    far *EDGE_DATA;
unsigned char    far *GRAY_DATA;
unsigned char    far *SOBLIN_DATA;
unsigned char    far *SOBEL_DATA;
unsigned char    far *CONNECT_DATA;
unsigned char    far *CORNER_DATA;
RGBREAL *HISTO_DATA;


unsigned char PICT_STORED=0;
unsigned char EDGE_STORED=0;
unsigned char HISTO_STORED=0;
unsigned char GRAY_STORED=0;
unsigned char SOBEL_STORED=0;
unsigned char SOBLIN_STORED=0;
unsigned char CONNECT_STORED=0;
unsigned char CORNER_STORED=0;

unsigned char OBJECTS_FOUND;
unsigned OBJECTS_CORNERS[2][4];
float OBJ_CONF;
unsigned far *GROUP_SIZE;

void clear_all_data(void);

void clear_all_data(void){
farfree(PICT_DATA);
farfree(EDGE_DATA);
farfree(HISTO_DATA);
farfree(GRAY_DATA);
farfree(SOBEL_DATA);
farfree(CORNER_DATA);
farfree(GROUP_SIZE);

farfree(CONNECT_DATA);

PICT_STORED=0;
EDGE_STORED=0;
HISTO_STORED=0;
GRAY_DATA=0;
CONNECT_STORED=0;
SOBEL_STORED=0;
CORNER_STORED=0;
}

void xtoxy(int xval,int *x,int *y){
*y=xval/PIC_X;
*x=xval-*y*PIC_X;
}

int xytox(int x,int y){
return(y*PIC_X+x);
}

int nint(float x){
double y;
if(modf(x,&y)<0.5) return (int)y;
else return (int)y+1;
}

#endif
```

NEWSOB. CPP

```
/***********************************************************************
 This is a Borland C ++ procedure for sOBEL oPERATIONS. All finctions
includes math.h,improc.cpp,gray.cpp,screen.cpp
defines _SOBEL_H,SOBEL_THRESH
************************************************************************/

#ifndef _SOBEL_H
#define _SOBEL_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __GRAYS_H
#include "c:\final\col\gray.cpp"
#endif

#ifndef _SCRH
#include "c:\final\scr\screen.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif
#define SOBEL_THRESH 500
int sobel_detect();
unsigned char calc_sobang(int z1,int z2);
void place_soblin(int x,int y, int z);
void clear_sobel(void);
void clear_soblin(void);
void show_lines(void);


int sobel_detect(){
if(!PICT_STORED) return 0;
int i,j,k,l,m,n,o,p,q;
int z1,z2;
unsigned char z;
int x,y;
unsigned char *pntr;
farfree(SOBEL_DATA);
if(!PICT_STORED) return 0;
SOBEL_DATA=(unsigned char far*)farmalloc((PIC_X)*(PIC_Y)*sizeof(unsigned
char));
SOBLIN_DATA=(unsigned char far*)farmalloc(150*115*sizeof(unsigned
char));

if(SOBEL_DATA==NULL) return 0;
if(SOBLIN_DATA==NULL)return 0;

for(x=0;x<150*115;x++)SOBLIN_DATA[x]=0;
pntr=SOBEL_DATA;

for(y=0;y<PIC_Y-2;y++){
 for(x=0;x<PIC_X-2;x++){
  i=grayscale(PICT_DATA[x+y*PIC_X])-16;
  j=grayscale(PICT_DATA[(x+1)+y*PIC_X])-16;
  k=grayscale(PICT_DATA[(x+2)+y*PIC_X])-16;

  l=grayscale(PICT_DATA[x+(y+1)*PIC_X])-16;
  m=grayscale(PICT_DATA[(x+1)+(y+1)*PIC_X])-16;
  n=grayscale(PICT_DATA[(x+2)+(y+1)*PIC_X])-16;

  o=grayscale(PICT_DATA[x+(y+2)*PIC_X])-16;
  p=grayscale(PICT_DATA[(x+1)+(y+2)*PIC_X])-16;
  q=grayscale(PICT_DATA[(x+2)+(y+3)*PIC_X])-16;

  z1=(-1*i)+(0*j)+(1*k)+(-2*l)+(0*m)+(2*n)
       +(-1*o)+(0*p)+(1*q);
  z2=(-1*i)+(-2*j)+(-1*k)+(0*l)+(0*m)+(0*n)
       +(1*o)+(2*p)+(1*q);
  z=calc_sobang(z1,z2);
  if(z)place_soblin(x,y,z);
  *pntr=z;
  pntr++;
  }
  *pntr=0;
  pntr++;
  *pntr=0;
  pntr++;

 }
 for(x=0;x<PIC_X*2;x++)   {*pntr=0;
                                 pntr++;
                                 }
 show_lines();
 SOBLIN_STORED=1;
 SOBEL_STORED=1;
 return 1;
}

void show_lines(){
int x,y;
float xa,ya,th,d;
for(x=0;x<100;x++){
 for(y=0;y<115;y++){
  if(SOBLIN_DATA[x+y*150]>5){
    th=x;
    th=th/200*3.141592654;
    d=y;
    xa=d*cos(th);
    ya=d*sin(th);
    if((ya<0)||(xa<0)){

      }
    }
   }
  }
}

void place_soblin(int x,int y, int z){
int xa,ya;
float d;

y=PIC_Y-y;
z-=16;
if(z>100)z-=100;
if(z-50){
  d=z+50;
  d=tan((d/100-0.5)*3.141592654)*x;
  if(int(d)<y)z+=100;
  }
xa=z;
d=z;
d/=100.0;
d-=0.5;
d*=3.141592654;
d=d-atan2(y,x);
d=cos(d)*sqrt(float(x*x+y*y));
ya=d;
SOBLIN_DATA[xa+ya*150]++;
}

void clear_sobel(void){
farfree(SOBEL_DATA);
SOBEL_STORED=0;
}

void clear_soblin(void){
farfree(SOBLIN_DATA);
SOBLIN_STORED=0;
}

unsigned char calc_sobang(int z1,int z2){
double z;
double za,zb;
unsigned char x;
za=z1;
za*=z1;
zb=z2;
zb*=z2;


z=sqrt(za+zb);
if(z<SOBEL_THRESH)return 0;

z=atan2(float(z1),float(z2));
z/=3.141592654;
z*=100;
z+=116;
x=z;
return (x);
}

#endif
```

PROC. CPP

```
#ifndef __PROC_H
#define __PROC_H

void calc_area_data(int *r,int *g,int *b);

void calc_area_data(int *r,int *g,int *b){
int x;

*r=0;
*g=0;
*b=0;
PIX *pntr;
pntr=PAREA;
for(x=0;x<64;x++){
      *r+=pntr->R;
      *g+=pntr->G;
      *b+=pntr->B;
      }
*r/=64;
*g/=64;
*b/=64;
}

void store_image();

RGBREAL pict[96][64];

void store_image(){

int x,y,r,g,b;

RGBCOL  p2;
init_screen();
for(x=0;x<768;x+=8){
   for(y=0;y<512;y+=8){
   set_x(x);
   set_y(y);
   get_pix_area();
   calc_area_data(&r,&g,&b);
   pict[x/8][y/8].RED=r;
   pict[x/8][y/8].GREEN=g;
   pict[x/8][y/8].BLUE=b;
   p2=conv2RGB(pict[x/8][y/8]);
   putpixel(x/8,y/8,conv2REAL(p2));
   }
 }
}
#endif
```

PSTORE. CPP

```
#ifndef __PSTORE_H
#define __PSTORE_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __RGB_H
#include "c:\final\col\rgb.cpp"
#endif

#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif

int  store_screen(void);
void clear_pict(void);

int store_screen(){
int x,y,r,g,b;
RGBREAL  p2;
RGBREAL far  *pntr;
if(PICT_DATA!=NULL){farfree(PICT_DATA);
                         PICT_DATA=NULL;
                         }
PICT_DATA=(RGBREAL far*)farmalloc(PIC_X*PIC_Y*sizeof(RGBREAL));
if(PICT_DATA==NULL) return 0;
pntr=PICT_DATA;

for(y=0;y<PIC_Y;y++){
   for(x=0;x<PIC_X;x++){
   X=x*8;
   Y=y*8;
   get_farea(&r,&g,&b);
   p2=RGBreal(r,g,b);
   *pntr=p2;
   pntr++;

   }

 }
 PICT_STORED=1;
 return 1;
}

void clear_pict(void){
farfree (PICT_DATA);
PICT_DATA=NULL;
PICT_STORED=0;
}
#endif
```

REFADJ. CPP

```
#ifndef __REFADJ_H
#define __REFADJ_H

#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif
void ref_adjust(int r,int g,int b);

void ref_adjust(int r,int g, int b){
int or,og,ob;
snap();
f_wait();
or=*RDATA;
og=*GDATA;
ob=*BDATA;
      if(or>r+2) {
                              RALUTA=0x02;
                              RPREF+=0x04;}
    else if(or<r-2) {
                              RALUTA=0x02;
                              RPREF-=0x04;}

      else {
                              RALUTA=0x02;
                              RPREF=*RPREF;}

      if(og>g+2) {
                              GALUTA=0x02;
                              GPREF+=0x04;}
    else if(og<g-2) {
                              GALUTA=0x02;
                              GPREF-=0x04;}
    else {
                              GALUTA=0x02;
                              GPREF=*GPREF;}

      if(ob>b+2){
                              BALUTA=0x02;
                              BPREF+=0x04;}

    else if(ob<b-2) {
                              BALUTA=0x02;
                              BPREF-=0x04;}
   else  {
                              BALUTA=0x02;
                              BPREF=*BPREF;}

}
#endif
```

SOBEL. CPP

```
#ifndef _SOBEL_H
#define _SOBEL_H

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __GRAYS_H
#include "c:\final\col\gray.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif


#define SOBEL_THRESH    70

int sobel_detect();
void clear_sobel(void);

int sobel_detect(){
int i,j,k,l,m,n,o,p,q;
int z1,z2,z3,z4,z,flag;
int x,y;
unsigned char *pntr;
farfree(SOBEL_DATA);
if(!GRAY_STORED) return 0;
SOBEL_DATA=(unsigned char far*)farmalloc((PIC_X)*(PIC_Y)*sizeof(unsigned
char));
if(SOBEL_DATA==NULL) return 0;
pntr=SOBEL_DATA;

for(y=0;y<PIC_Y-3;y++){
 for(x=0;x<PIC_X-2;x++){
  i=GRAY_DATA[x+y*PIC_X]-16;
  j=GRAY_DATA[(x+1)+y*PIC_X]-16;
  k=GRAY_DATA[(x+2)+y*PIC_X]-16;

  l=GRAY_DATA[x+(y+1)*PIC_X]-16;
  m=GRAY_DATA[(x+1)+(y+1)*PIC_X]-16;
  n=GRAY_DATA[(x+2)+(y+1)*PIC_X]-16;

  o=GRAY_DATA[x+(y+2)*PIC_X]-16;
  p=GRAY_DATA[(x+1)+(y+2)*PIC_X]-16;
  q=GRAY_DATA[(x+2)+(y+3)*PIC_X]-16;

  z1=(-1*i)+(0*j)+(1*k)+(-2*l)+(0*m)+(2*n)
       +(-1*o)+(0*p)+(1*q);
  z2=(-1*i)+(-2*j)+(-1*k)+(0*l)+(0*m)+(0*n)
       +(1*o)+(2*p)+(1*q);
  z3=(1*i)+(0*j)+(-1*k)+(2*l)+(0*m)+(-2*n)
       +(1*o)+(0*p)+(-1*q);
  z4=(1*i)+(2*j)+(1*k)+(0*l)+(0*m)+(0*n)
       +(-1*o)+(-2*p)+(-1*q);

  z=z1;
flag=1;
if(z2>z){z=z2;
              flag=2;
             }
if(z3>z){z=z3;
              flag=3;
            }
if(z4>z) flag=4;
if(z<SOBEL_THRESH)*pntr=0;
else *pntr=flag;
pntr++;
}
*pntr=0;
pntr++;
*pntr=0;
pntr++;

}
for(x=0;x<PIC_X*2;x++) {*pntr=0;
                              pntr++;
                              }
SOBEL_STORED=1;
return 1;
}
void clear_sobel(void){
farfree(SOBEL_DATA);
SOBEL_STORED=0;
}

#endif
```

DATA PROCESSING FUNCTIONS

CIE. CPP

```
#ifndef __CIE_H
#define __CIE_H


#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif
CIECOL CIE(float r,float g,float b);
CIECOL CIE(int r,int g,int b);
CIECOL CIE(RGBREAL x);
CIECOL CIE(RGBCOL x);

CIECOL CIE(float r,float g,float b){
CIECOL x;
x.X=r/(r+g+b);
x.Y=g/(r+g+b);
x.Z=b/(r+g+b);
return x;
};


CIECOL CIE(int r,int g,int b){
CIECOL x;
x.X=float(r)/float(r+g+b);
x.Y=float(g)/float(r+g+b);
x.Z=float(b)/float(r+g+b);
return x;
};

CIECOL CIE(RGBREAL x){
int r,g,b;
r=x.RED;
g=x.GREEN;
b=x.BLUE;
return CIE(r,g,b);
};

CIECOL CIE(RGBCOL x){
float r,g,b;
r=x.RED;
g=x.GREEN;
b=x.BLUE;
return CIE(r,g,b);
};

#endif
```

COL. CPP

```
#ifndef __COL_H
#define __COL_H

typedef struct HSVCOL {  float HUE;
                                 float SATURATION;
                                 float VALUE;
                           };

typedef struct RGBCOL {                      float RED;
                                                   float GREEN;
                                                   float BLUE;

                          };
typedef struct RGBREAL {                     unsigned RED;
                                                   unsigned GREEN;
                                                   unsigned BLUE;

                          };


 typedef struct CIECOL {  float X;
                                 float Y;
                                 float Z;
                                 };

 typedef struct HLSCOL {  float HUE;
                                 float SATURATION;
                                 float LIGHTNESS;
                            };
  float RGBmin(RGBCOL x);
  float RGBmax(RGBCOL x);


  float RGBmin(RGBCOL x){
        if(x.GREEN<x.RED)x.RED=x.GREEN;
        if(x.BLUE<x.RED)x.RED=x.BLUE;
        return x.RED;
  }

  float RGBmax(RGBCOL x){
        if(x.GREEN>x.RED)x.RED=x.GREEN;
        if(x.BLUE>x.RED)x.RED=x.BLUE;
        return x.RED;

  }
  #endif
```

COLCLASS. CPP

```
#ifndef __COLLIB_H
#include "c:\final\col\collib.cpp"
#endif

#ifndef __COLCLASS_H
#define __COLCLASS_H

class COL {
protected:
float RED;
float GREEN;
float BLUE;
public:
COL(void)
            {RED=0;
            GREEN=0;
            BLUE=0;
            };
COL(int r,int g,int b)
            {RED=float(r)/256.0;
            GREEN=float(g)/256.0;
            BLUE=float(b)/256.0;
            };
COL(float r,float g,float b)
            {RED=r;
            GREEN=g;
            BLUE=b;
            };

void set(int r,int g,int b)
            {RED=float(r)/256.0;
            GREEN=float(g)/256.0;
            BLUE=float(b)/256.0;
            };
void set(float r,float g,float b)
            {RED=r;
            GREEN=g;
            BLUE=b;
            };


~COL(void)
            {RED=0;
            GREEN=0;
            BLUE=0;
            };

RGBCOL _RGBcol(void);
RGBREAL _RGBreal(void);
int _grayscale(void);
CIECOL _CIE(void);
HLSCOL _HLS(void);
HSVCOL _HSV(void);
float R(void);
float G(void);
float B(void);
float H(void);
float S1(void);
float S2(void);
float L(void);
float V(void);
};

float COL::R(void){ return RED;
};

float COL::G(void){ return GREEN;
};

float COL::B(void){ return BLUE;
};

float COL::H(void){
HLSCOL x;
x=_HLS();
return x.HUE;
};

float COL::L(void){
HLSCOL x;
x=_HLS();
return x.LIGHTNESS;
};

float COL::S1(void){
HLSCOL x;
x=_HLS();
return x.SATURATION;
};

float COL::S2(void){
HSVCOL x;
x=_HSV();
return x.SATURATION;
};

float COL::V(void){
HSVCOL x;
x=_HSV();
return x.VALUE;
};


RGBCOL COL::_RGBcol(void){
      return (RGBcol(RED,GREEN,BLUE));
};

RGBREAL COL::_RGBreal(void){
      return (RGBreal(RED,GREEN,BLUE));
};

HLSCOL COL::_HLS(void){
      return (HLS(RED,GREEN,BLUE));
};

HSVCOL COL::_HSV(void){
      return (HSV(RED,GREEN,BLUE));
```

```
};

int COL::_grayscale(void){
      return (grayscale(RED,GREEN,BLUE));
};


#endif
```

COLLIB. CPP

```
#ifndef __COLLIB_H
#define __COLLIB_H

#include "c:\final\col\gray.cpp"
#include "c:\final\col\rgb.cpp"
#include "c:\final\col\cie.cpp"
#include "c:\final\col\hls.cpp"
#include "c:\final\col\hsv.cpp"

#endif
```

GRAY. CPP

```
#ifndef __GRAYS_H
#define __GRAYS_H

#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif

int grayscale(int r,int g,int b);
int grayscale(float r,float g,float b);
int grayscale(RGBREAL x);
int grayscale(RGBCOL x);


int grayscale(float r,float g,float b){
  return (int)((b+3*g+2*r)*40+16);
};

int grayscale(int r,int g,int b){
  return int(float(b+3*g+2*r)/6.4)+16;
};

int grayscale(RGBREAL x){
int r,g,b;
r=x.RED;
g=x.GREEN;
b=x.BLUE;
return grayscale(r,g,b);
};

int grayscale(RGBCOL x){
float r,g,b;
r=x.RED;
g=x.GREEN;
b=x.BLUE;
return grayscale(r,g,b);
};

#endif
```

HLS. CPP

```
#ifndef __HLS_H
#define __HLS_H

#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif

#ifndef __STDDEF_H
#include <stddef.h>
#endif

#ifndef __RGBSWAP_H
#include "c:\final\col\RGB.cpp"
#endif

HLSCOL HLS(RGBCOL x);
HLSCOL HLS(RGBREAL x);
HLSCOL HLS(int r,int g,int b);
HLSCOL HLS(float r,float g,float b);

HLSCOL HLS(int r,int g,int b){
return HLS(RGBcol(r,g,b));
};

HLSCOL HLS(float r,float g,float b){
return HLS(RGBcol(r,g,b));
};



HLSCOL HLS(RGBREAL x){
return HLS(RGBcol(x));
};




HLSCOL HLS(RGBCOL x){
      HLSCOL h;
      unsigned tmin,tmax;
      if((x.RED==x.GREEN)&&(x.RED==x.BLUE)) {
            h.HUE=NULL;
            h.SATURATION=0;
            h.LIGHTNESS=x.RED*100;
      }
      else{
            tmin=RGBmin(x);
            tmax=RGBmax(x);

            x.RED-=tmin;
            x.GREEN-=tmin;
            x.BLUE-=tmin;

            h.LIGHTNESS=100*((tmax+tmin)/2);

            if(!x.GREEN) h.HUE=120*((1+(x.RED/(x.RED+x.BLUE))));
            if(!x.BLUE) h.HUE=120*((1+(x.GREEN/(x.GREEN+x.RED))));
            if(!x.RED)  h.HUE=120*((2+(x.BLUE/(x.GREEN+x.BLUE))));

            if(h.LIGHTNESS<=50.0) h.SATURATION=100*((tmax-
tmin)/(tmax+tmin));
            else h.SATURATION=100*((tmax-tmin)/(200-tmax-tmin));

      }

      return h;
};

#endif
```

HSV. CPP

```
#ifndef __HSV_H
#define __HSV_H


#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif

#ifndef __MATH_H
#include <math.h>
#endif

#ifndef __STDDEF_H
#include <stddef.h>
#endif

#ifndef __RGBSWAP_H
#include "c:\final\col\RGB.cpp"
#endif
HSVCOL HSV(RGBCOL x);
HSVCOL HSV(RGBREAL x);
HSVCOL HSV(int r,int g,int b);
HSVCOL HSV(float r,float g,float b);

HSVCOL HSV(int r,int g,int b){
return HSV(RGBcol(r,g,b));
};

HSVCOL HSV(float r,float g,float b){
return HSV(RGBcol(r,g,b));
};



HSVCOL HSV(RGBREAL x){
return HSV(RGBcol(x));
};




HSVCOL HSV(RGBCOL x){
      HSVCOL h;
      float tmin,tmax;
      if((fabs(x.RED-x.GREEN)<0.01)&&(fabs(x.RED-x.BLUE)<0.01)) {
            h.HUE=NULL;
            h.SATURATION=0;
            h.VALUE=x.RED;
      }
      else{
            tmin=RGBmin(x);
            tmax=RGBmax(x);

            x.RED-=tmin;
            x.GREEN-=tmin;
            x.BLUE-=tmin;
            h.VALUE=tmax;

            if(!x.GREEN) h.HUE=120*((0+(x.RED/(x.RED+x.BLUE))));
            if(!x.BLUE) h.HUE=120*((1+(x.GREEN/(x.GREEN+x.RED))));
            if(!x.RED) h.HUE=120*((2+(x.BLUE/(x.GREEN+x.BLUE))));

            h.SATURATION=100*((tmax-tmin)/tmax);

      }

      return h;
};

#endif
```

RGB. CPP

```
#ifndef __RGBSWAP_H
#define __RGBSWAP_H

#ifndef __COL_H
#include "c:\final\col\col.cpp"
#endif
RGBCOL RGBcol(float r,float g,float b);
RGBCOL RGBcol(int r,int g,int b);
RGBCOL RGBcol(RGBREAL r);
RGBCOL RGBcol(RGBCOL r);

RGBREAL RGBreal(float r,float g,float b);
RGBREAL RGBreal(int r,int g,int b);
RGBREAL RGBreal(RGBREAL r);
RGBREAL RGBreal(RGBCOL r);


RGBCOL RGBcol(float r,float g,float b){
RGBCOL x;
x.RED=r;
x.GREEN=g;
x.BLUE=b;
return x;
};

RGBCOL RGBcol(int r,int g,int b){
RGBCOL x;
x.RED=float(r)/256.0;
x.GREEN=float(g)/256.0;
x.BLUE=float(b)/256.0;
return x;
};

RGBCOL RGBcol(RGBREAL r){
RGBCOL x;
x.RED=float(r.RED)/256.0;
x.GREEN=float(r.GREEN)/256.0;
x.BLUE=float(r.BLUE)/256.0;
return x;
};

RGBREAL RGBreal(int r,int g,int b){
RGBREAL x;
x.RED=r;
x.GREEN=g;
x.BLUE=b;
return x;
};

RGBREAL RGBreal(float r,float g,float b){
RGBREAL x;
x.RED=int(r*256.0);
x.GREEN=int(g*256.0);
x.BLUE=int(b*256.0);
return x;
};
```

```
RGBREAL RGBreal(RGBCOL r){
RGBREAL x;
x.RED=int(r.RED*256.0);
x.GREEN=int(r.GREEN*256.0);
x.BLUE=int(r.BLUE*256.0);
return x;
};

#endif
```

NEURAL. CPP

```
#ifndef  __NEURAL_HPP
#define  __NEURAL_HPP

#include <math.h>
#include <conio.h>


/* defines for neural.i */

#define No_in 3
#define No_hid1 20
#define No_hid2 5
#define No_out 1
#define No_train 87


/* global variables for neural.i */
float bias = 1, bias1[No_hid1], bias2[No_hid2], bias3[1], in[No_in];
float wtsin1[No_hid1][No_in], wts12[No_hid2][No_hid1],
wts2out[No_out][No_hid2];
unsigned char wgtselected=0;

/* prototypes for meural.i */
unsigned calc_sum(int imax, int jmax, float * sum, float * sigout, float
* bias, float * wts, float * indat);
unsigned readwts(imagestkptr ifs,msclickptr ms);

unsigned getwts(int imax, int jmax, float * biasin, float * wtsin);
unsigned calc_thick(float * thick);
unsigned calc_sum(int imax, int jmax, float * sum, float * sigout, float
* bias, float * wts, float * indat);
unsigned convstr(char * str, int stlen, float * data);

#endif


#ifndef  __NEURAL_H
#define  __NEURAL_H


unsigned optneural(imagestkptr ifs,msclickptr ms)
{
      int k;
      float req_thick, output;
      FILE *trainfile;
      unsigned x=100,y=100,w=200,h=100;
      imagestkptr fs;

      if(wgtselected==0) readwts(ifs,ms);

        get_pix();
        in[0]=(float)PIXEL.p.R/256;
        in[1]=PIXEL.p.G/256;
        in[2]=PIXEL.p.B/256;
        calc_thick(&output);
        quickframe(&fs,&x,&y,&w,&h);
        fmttegltextxy(120,130,"%f",output);
        definebuttonclick(fs,150,160,imageOK,close_frame);
      return(1);
}

FILE *nwts;

unsigned readwts(imagestkptr ifs,msclickptr ms)
{
      optopenfilesel();


      if ((nwts = fopen(fileselected, "rt")) == NULL)
      {
         text_string1="weight file could not be opened";
         messageframe(ifs,ms);
      }

      getwts(No_in, No_hid1, &bias1[0], &wtsin1[0][0]);
      getwts(No_hid1, No_hid2, &bias2[0], &wts12[0][0]);
      getwts(No_hid2, No_out, &bias3[0], &wts2out[0][0]);
      wgtselected=1;
      fclose(nwts);
      return(1);
}


unsigned getwts(int imax, int jmax, float * biasin, float * wtsin)
{
      int i,j;
      char inline[81];
      float inno[3];

      for (j=0; j<jmax; j++)
      {
        fgets(inline, 80, nwts);

        convstr(inline, 80, &inno[0]);

        *biasin++ = inno[1];

        for (i=0; i<imax; i++)
        {
          fgets(inline, 80, nwts);
          convstr(inline, 80,&inno[0]);

         *wtsin++ = inno[1];
        }
      }
      return(1);
}

unsigned convstr(char * str, int stlen, float * data)
{
   int i,j,k,cnt, flag, endflag;
   float tempval;
   char tempstr[10], ch;

   i=0;
   k=0;
   j=0;
   cnt=0;
   flag =0;
   endflag =0;

   while ((i < stlen) && (endflag == 0))
   {
      ch = str[i];
      if (ch == '\t') ch = ' ';

      if ((ch == ' ') && (flag ==1))
      {
         flag = 0;
         if ((cnt!=0) && (cnt != 3))
         {
           tempstr[k] = '\0';
           tempval = (float) atof(tempstr);
           data[j] = tempval;
           j++;
         }
         cnt++;
         k=0;
      }

      if (ch != ' ')
      {
        tempstr[k++] = ch;
        flag =1;
      }

      i++;

      if (cnt==4) endflag =1;
   }
   return(1);
}


unsigned calc_thick(float * thick)
{
      float sum1[No_hid1], sum2[No_hid2], sum3[No_out];
      float val1[No_hid1], val2[No_hid2], val3[No_out];

      calc_sum(No_in, No_hid1, &sum1[0], &val1[0], &bias1[0],
&wtsin1[0][0], &in[0]);
      calc_sum(No_hid1, No_hid2, &sum2[0], &val2[0], &bias2[0],
&wts12[0][0], &val1[0]);
      calc_sum(No_hid2, No_out, &sum3[0], &val3[0], &bias3[0],
&wts2out[0][0], &val2[0]);

      *thick = val3[0];
      return(1);
}

unsigned calc_sum(int imax, int jmax, float * sum, float * sigout, float
* bias, float * wts, float * indat)
{
   float biasval=1.0, tempwt, tempsum, *tmpptr, *ptrsum;
   int i,j;
   ptrsum = sum;

   for(j=0; j<jmax; j++)
   {
      tmpptr = indat;
      *ptrsum = biasval * bias[j];

      for (i=0; i<imax; i++)
      {
      tempwt = *wts++;
      tempsum = *tmpptr++;
      *ptrsum += (tempwt*tempsum);
      }
      ptrsum++;
   }

   for(j=0; j<jmax; j++)
      *sigout++ = 1/(1+exp(- *sum++));
   return(1);
}

#endif
```

DISPLAY FUNCTIONS

PSHOW. CPP

```
#ifndef __SHOW_H
#define __SHOW_H

#ifndef __SCREEN_H
#include "c:\final\scr\screen.cpp"
#endif

#ifndef __IMPROC_H
#include "c:\final\proc\improc.cpp"
#endif

#ifndef __CONIO_H
#include <conio.h>
#endif

void show_pict(unsigned char far *data,int xtop,int ytop);

void show_pict(unsigned char far *data,int xtop,int ytop){
int x,y;
for(y=0;y<PIC_Y;y++){
 for(x=0;x<PIC_X;x++){
      putpixel(x+xtop,y+ytop,data[x+y*PIC_X]);
      }
 }
}

#endif
```

3-D PLOT

```
#ifndef __3DPLOT_H
#define __3DPLOT_H


/* defines for plot.i */

#define POINT     1
#define FULL      2
#define REF       3
#define nxsteps 75
#define nysteps 70


/* global variables for plot.i */

int points[8],xscrn[4],yscrn[4];
float height[1+nxsteps][1+nysteps];
int yoffset=400;
int xoffset=300;
float sc = 500.0;
float di=4.0;
float ht= -2.0;
unsigned Rlst=0,Glst=0,Blst=0;


/* prototypes for plot.i */
void Surfplot(void);
void surfplot(void);
void calcheights(void);
void normalise(void);
void holdit(void);
void tile(void);
void do_calcs(void);
void conv3d2d(float x3d,float y3d,float z3d,int k);

float get_data(int i,int k);
void findtilecoords(int i,int k);
float fnx(int x);
float fny(int y);

#endif

#include <graphics.h>


void Surfplot(void)
{
int gdriver=DETECT,gmode,errorcode;
int i,k,flag=1;
char c;
initgraph(&gdriver,&gmode,"c:\\app\\borlandc\\bgi\\");
calcheights();
grab();
//normalise();
surfplot();
holdit();
//closegraph();
}


void surfplot()
{
int k,i;

for (k=nysteps-1;k>=0;k--)

{
      for (i=0;i<=nxsteps-1;i++)
      {
      findtilecoords(i,k);
      tile();
      };
};

};


void normalise()
{
/* ensure all heights are in the range 0..1 */
/* then add height above surface ht */
int i,k;
float max,min;
max= -999999.999;
min=999999.999;

/* find min and max */
for (i=0;i<=nxsteps;i++) {
for (k=0;k<=nysteps;k++) {
if (height[i][k]>max) {
      max=height[i][k];
      };
if (height[i][k]<min) {
      min=height[i][k];
      };
};
};
/* now normalise to range 0..1 */
for (i=0;i<=nxsteps;i++) {
for (k=0;k<=nysteps;k++) {
height[i][k]=ht+(height[i][k]-min)/(max-min);
};
};

}


void calcheights()
 {
  int i,k;
  for (i=0;i<=nxsteps;i++) {
      for (k=0;k<=nysteps;k++) {
      height[i][k]=get_data(i,k);
      }
   }
}
```

```
float get_data(int i,int k)
{ float red,green,blue,intensity,val ;
  xpos(8*i);
  ypos(8*k);
  get_pix_area(&red,&green,&blue);
  intensity = red+blue+green;
    if(intensity) val = red/intensity;
    if(intensity==0) val = 0;
    return (val);



}

void conv3d2d(float x3d,float y3d,float z3d,int k)

{
/* convert float coords x3d,y3d,z3d to integer
   screen coords in xscrn[k], yscrn[k] using perspective
   transformation
*/
xscrn[k] = (int) sc*x3d/(y3d+di)+ xoffset;
yscrn[k] = (int) (400-(sc*z3d/(y3d+di)))-yoffset;
}


/* draw a tile using coordinates stored in arrays
   xscrn[] and yscrn[]
*/

void tile()
{
int npoints=4;
/* draw filled square in red then
  draw a yellow outile around that */
setfillstyle(SOLID_FILL,BLACK);
points[0]=xscrn[0];
points[1]=yscrn[0];
points[2]=xscrn[1];
points[3]=yscrn[1];
points[4]=xscrn[2];
points[5]=yscrn[2];
points[6]=xscrn[3];
points[7]=yscrn[3];
fillpoly(npoints,points);
setcolor(WHITE);
drawpoly(npoints,points);
}


void findtilecoords(int i,int k)
{
/* find tile coordinates for point
      x index=i;
      y index=k;
*/
float x0,y0,z0,x1,y1,z1;
float x2,y2,z2,x3,y3,z3;
//float fnx(),fny();
x0=fnx(i);
y0=fny(k);
z0=height[i][k]-2.0;
conv3d2d(x0,y0,z0,0);
x1=fnx(i+1);
y1=fny(k);
z1=height[i+1][k]-2.0;
conv3d2d(x1,y1,z1,1);
x2=fnx(i+1);
y2=fny(k+1);
z2=height[i+1][k+1]-2.0;
conv3d2d(x2,y2,z2,2);
x3=fnx(i);
y3=fny(k+1);
z3=height[i][k+1]-2.0;
conv3d2d(x3,y3,z3,3);
}



void holdit()
{
char c;
c=getch();
}


/* functions relating x or y point index
   to the range -1..1 allows normalised
   plot with any number of points in x or y
   directions.
*/

float fnx(int i)
{
float result;
result=1.0+2.0*(i-nxsteps)/((float) nxsteps);
return(result);
}


/*this function is identical to the one above
   but is included for logical clarity !!*/
float fny(int i)
{
float result=1.0+2.0*(i-nysteps)/((float) nysteps);
return(result);
}
```

GUI FUNCTIONS

EVENT. CPP

```
#ifndef __EVENT_H
#define __EVENT_H

#ifndef __STDLIB_H
#include <stdlib.h>
#endif

#ifndef __STDIO_H
#include <stdio.h>
#endif
class event_box;

class event_box{
public:
 int id;
 int xt;
 int yt;
 int width;
 int height;
 void (*repaint_func)(event_box *temp);
 event_box *next_box;

event_box(int id_num,int _xt,int _yt,int _width,int _height,void
(*paint_func)(event_box *) =NULL){
  id=id_num;
  xt=_xt;
  yt=_yt;
  width=_width;
  height=_height;
  repaint_func=paint_func;
  next_box=NULL;
}

~event_box(){
  id=0;
  xt=0;
  yt=0;
  width=0;
  height=0;
  repaint_func=NULL;
  next_box=NULL;
}

void link(event_box *next){
 next_box=next;
}

int retval(void){
 return id;
}

};

event_box *start;
```

```
class event_handle{
public:
 event_box *start;

event_handle(void){
start=new event_box(0,0,0,0,0);
}

~event_handle(void);

void add_item(event_box *item);
void add_item(int id_num,int _xt,int _yt,int _width,int _height);
void remove_item(int id_num);
void move_item(int id_num,int x,int y);
event_box *search_inbounds(int x,int y);
void to_end(int id_num);
void repaint_all(void);
};

void event_handle::move_item(int id_num,int x,int y){
event_box *temp;
temp=start;
while (temp->id!=id_num||temp!=NULL)temp=temp->next_box;
if(temp!=NULL){
  temp->xt=x;
  temp->yt=y;
  }
}

event_handle::~event_handle(void){
event_box *temp,*temp2;
temp2=start;
temp=start->next_box;
while(temp!=NULL){
  delete temp2;
  temp2=temp;
  temp=temp->next_box;
  }
delete temp2;
}


void event_handle::add_item(event_box *item){
event_box *temp;
temp=start;
while(temp->next_box!=NULL)temp=temp->next_box;
temp->next_box=item;
}

void event_handle::add_item(int id_num,int _xt,int _yt,int _width,int
_height){
event_box *temp,*item;
item=new event_box(id_num,_xt,_yt,_width,_height);
temp=start;
while(temp->next_box!=NULL)temp=temp->next_box;
temp->next_box=item;
```

```
}


event_box *event_handle::search_inbounds(int x,int y){
event_box *temp;
event_box *retval;
retval=start;
temp=start;
while(temp!=NULL){
 if((x>=temp->xt)
       &&(y>=temp->yt)
        &&(y<=(temp->yt+temp->height))
            &&(x<=(temp->xt+temp->width))) {
                   retval=temp;
            }
       temp=temp->next_box;
  }
return retval;
}

void event_handle::remove_item(int id_num){
event_box *temp,*temp2;
temp=start;
temp2=NULL;
while(temp->id!=id_num){
      temp2=temp;
      temp=temp->next_box;
      if(temp==NULL)break;
      }
 if(temp2!=NULL)  {
        temp2->next_box=temp->next_box;
        delete temp;
        }
}

void event_handle::to_end(int id_num){
event_box *temp,*temp2;
temp=start;
temp2=NULL;
while(temp->id!=id_num){
      temp2=temp;
      temp=temp->next_box;
      if(temp==NULL)break;
      }

if(temp2!=NULL)  {temp2->next_box=temp->next_box;
       temp2=start;
       while(temp2->next_box!=NULL)temp2=temp2->next_box;
       temp2->next_box=temp;
       temp->next_box=NULL;
       }
}

void event_handle::repaint_all(void){
event_box *temp;
temp=start;
while(temp->next_box!=NULL){temp=temp->next_box;
                                                                       (temp-
>repaint_func)(temp);
                                                                 };
```

```
}
event_handle menu;
event_handle gwindow;
event_handle proc_chain;

#endif
```

GUI. CPP

```
#ifndef __GUI_H
#define __GUI_H


#ifndef _CSCRH
#include "c:\final\scr\screen.cpp"
#endif

#ifndef __MOUSE_H
#include "c:\final\mouse\mouse.cpp"
#endif

#ifndef __CONIO_H
#include <conio.h>
#endif

#define GMAX_X 319
#define GMAX_Y 199


void initguienv();
void draw_bin();

void initguienv(){
int x,y;
char s[10];
init_screen();
gray256();
initialize_cursors();
//gray256();
cleardevice();
Set_Graphic_Cursor(ARROW);
m_xlimit(0,GMAX_X*2);
m_ylimit(0,GMAX_Y);
m_moveto(10,10);
m_hide();
setfillstyle(SOLID_FILL,90);
bar(0,0,GMAX_X,GMAX_Y);
setcolor(4);
rectangle(0,0,GMAX_X,GMAX_Y);
outtextxy(50,50,"hello");
draw_bin();
m_show();
}


void draw_bin(){

FILE *binfile;
int x,y,c;
x=295;
y=170;


binfile=fopen("c:\\final\\gui\\bin.ico","rt");
while(!feof(binfile)){
  c=fgetc(binfile);
  switch (c){
        case '.': x++;
                              break;
        case '#': putpixel(x,y,BLACK);
                              x++;
                              break;
        case '+': putpixel(x,y,LIGHTGRAY);
                              x++;
                              break;
        case '-': putpixel(x,y,WHITE);
                              x++;
                              break;
        case '\n':y++;
                              x=295;
                              break;
        default:  x++;
                              break;
        }
}
fclose(binfile);
}



#endif
```

MAIN. CPP

```
#include "c:\final\gui\gui.cpp"

#ifndef __MENU_H
#include "c:\final\menu\menu.cpp"
#endif



void main(){
int c;
initguienv();
barmenu(barmenu1);
while(event_controller()!=1);
closegraph();
}
```

MENU. CPP

```
#ifndef _MENU_PRO
#define _MENU_PRO

void barmenu(MENUTYPE *menuitem);
void far *drop_pullmenu(MENUTYPE *pullmenu,int offx,int offy);
void remove_pullmenu(MENUTYPE *pullmenu,int offx,int offy,void far
*menuback);
void handle_barmenu(event_box *menuitem);
int handle_dropmenu(int id);
int handle_pullmenu(int id,int newstart);

void handle_window(event_box *winditem);

void proc_dropmenu(int id);
int event_controller(void);

#endif

#ifndef __ALLOC_H
#include <alloc.h>
#endif

#ifndef __GRAPHICS_H
#include <graphics.h>
#endif

#ifndef __STRING_H
#include <string.h>
#endif

#ifndef __MEN1_H
#include "c:\final\menu\men1.cpp"
#endif

#ifndef __EVENT_H
#include "c:\final\gui\event.cpp"
#endif
```

```
#ifndef __MOUSE_H
#include "c:\final\mouse\mouse.cpp"
#endif

#ifndef __MPROCS_H
#include "c:\final\gui\mprocs.cpp"
#endif

#ifndef __GUI_H
#include "c:\final\gui\gui.cpp"
#endif

#ifndef __MPROCS_H
#include "c:\final\gui\mprocs.cpp"
#endif


#ifndef __MENU_H
#define __MENU_H


void barmenu(MENUTYPE far *menuitem){

int x=0,count=0;
m_hide();
setcolor(17);
setfillstyle(SOLID_FILL,180);
bar(0,0,GMAX_X,16);
rectangle(0,0,GMAX_X,16);
while(strcmp(menuitem[x].name,"")){

if(menuitem[x].stat==MNORMAL)menu.add_item(menuitem[x].id,count*8+1,1,(s
trlen(menuitem[x].name)+1)*8,14);
  setcolor(BLACK);
  if(menuitem[x].stat==MGRAYED)setcolor(150);
  outtextxy((count*8)+4,4,menuitem[x].name);
  count+=strlen(menuitem[x].name)+1;
  x++;
  }

m_show();

}


void remove_pullmenu(MENUTYPE far *pullmenu,int offx,int offy,void far
*menuback){
int x=0;
while(strcmp(pullmenu[x].name,"")){
      if(pullmenu[x].id&&(pullmenu[x].stat==MNORMAL))menu.remove_item(pu
llmenu[x].id);
      x++;
      }
m_hide();
putimage(offx,offy,menuback,COPY_PUT);
m_show();
farfree(menuback);
}
```

```
void far *drop_pullmenu(MENUTYPE far *pullmenu,int offx,int offy){
void far *menuback;
MENUTYPE *menuitem;
int width,height;
char menuname[20];
int count=0,x,y;
int maxstrlen=0;
menuitem=pullmenu;
m_hide();
while(strcmp(menuitem->name,"")){
      count++;
      strcpy(menuname,menuitem->name);
      if(maxstrlen<strlen(menuname))maxstrlen=strlen(menuname);
      menuitem++;
      }
width=(maxstrlen+1)*8;
height=(count+1)*8;
setfillstyle(SOLID_FILL,180);
menuback=farmalloc(imagesize(offx,offy,offx+width,offy+height));
getimage(offx,offy,offx+width,offy+height,menuback);
bar(offx,offy,offx+width,offy+height);
setcolor(17);
rectangle(offx,offy,offx+width,offy+height);
for(x=0;x<count;x++){
 if(!strcmp(pullmenu[x].name,"--")){
      for(y=0;y<=maxstrlen;y++)
             outtextxy(offx+(y*8),offy+4+(x*8),"Ä");
      }
 else {
      setcolor(BLACK);
       if(pullmenu[x].stat==MGRAYED)setcolor(150);
       outtextxy(offx+4,offy+4+(x*8),pullmenu[x].name);

if(pullmenu[x].stat==MNORMAL)menu.add_item(pullmenu[x].id,offx+1,offy+(x
*8)+4,(maxstrlen+1)*8-2,8);
       }
}
m_show();
return menuback;
}

void handle_barmenu(event_box *_item){
int c=0;
void far *itemimg;
m_hide();
itemimg=farmalloc(imagesize(_item->xt,_item->yt,_item->xt+_item-
>width,_item->yt+_item->height));
getimage(_item->xt,_item->yt,_item->xt+_item->width,_item->yt+_item-
>height,itemimg);
putimage(_item->xt,_item->yt,itemimg,NOT_PUT);
handle_dropmenu(_item->id);
m_hide();
putimage(_item->xt,_item->yt,itemimg,COPY_PUT);
m_show();
farfree(itemimg);

gwindow.repaint_all();
m_show();
}
```

```
int handle_dropmenu(int id){
int c=0;
void far *back1;
switch(id){
 case 1: back1=drop_pullmenu(starmenu,1,17);
                 c=handle_pullmenu(1,100);
                 remove_pullmenu(starmenu,1,17,back1);
                 break;

 case 2: back1=drop_pullmenu(filemenu,16,17);
                 c=handle_pullmenu(2,200);
                 remove_pullmenu(filemenu,16,17,back1);
                 break;

 case 3: back1=drop_pullmenu(framemenu,50,17);
                 c=handle_pullmenu(3,300);
                 remove_pullmenu(framemenu,50,17,back1);
                 break;

 case 4: back1=drop_pullmenu(procmenu,120,17);
                 c=handle_pullmenu(4,401);
                 remove_pullmenu(procmenu,120,17,back1);
                 break;

 case 5: back1=drop_pullmenu(graphmenu,180,17);
                 c=handle_pullmenu(5,601);
                 remove_pullmenu(graphmenu,180,17,back1);
                 break;
 }
                 if(c)proc_dropmenu(c);
                 return c;
}

int handle_pullmenu(int from,int newstart){
 m_status *m;
 event_box *_item;
 int curid,l_xt,l_yt,c=0;
 void far *itemimg=NULL,*back2;
 int last_id=0;
 m=m_pos();
 while(m->button_status){
      m=m_pos();
      _item=menu.search_inbounds(m->xaxis/2,m->yaxis);
      curid=_item->id;
      if((curid<newstart)&&(curid)&&(curid!=from))break;
      if(curid&&(curid>=newstart)&&(curid!=last_id)){
            m_hide();
            if(itemimg!=NULL) {putimage(l_xt,l_yt,itemimg,COPY_PUT);
                                   farfree(itemimg);
                                  }
            itemimg=farmalloc(imagesize(_item->xt,_item->yt,_item-
>xt+_item->width,_item->yt+_item->height));
            getimage(_item->xt,_item->yt,_item->xt+_item->width,_item-
>yt+_item->height,itemimg);
            putimage(_item->xt,_item->yt,itemimg,NOT_PUT);
            l_xt=_item->xt;
            l_yt=_item->yt;
            last_id=curid;
            m_show();
            switch(curid){
                  case 403: back2=drop_pullmenu(imagemenu,180,36);
                                      c=handle_pullmenu(403,500);

remove_pullmenu(imagemenu,180,36,back2);
                                      break;

                  case 601: back2=drop_pullmenu(gallmenu,230,19);
                                      c=handle_pullmenu(601,700);

remove_pullmenu(gallmenu,230,19,back2);
                                      break;

                  case 602: back2=drop_pullmenu(gallmenu,230,27);
                                      c=handle_pullmenu(602,700);

remove_pullmenu(gallmenu,230,27,back2);
                                      break;

                  case 603: back2=drop_pullmenu(gallmenu,230,35);
                                      c=handle_pullmenu(603,700);

remove_pullmenu(gallmenu,230,35,back2);
                                      break;

              case 604: back2=drop_pullmenu(gallmenu,230,43);
                                      c=handle_pullmenu(604,700);

remove_pullmenu(gallmenu,230,43,back2);
                                      break;

              case 605: back2=drop_pullmenu(gallmenu,230,51);
                                      c=handle_pullmenu(605,700);

remove_pullmenu(gallmenu,230,51,back2);
                                      break;

              }
      }


 }
if(itemimg!=NULL)farfree(itemimg);
if(c)proc_dropmenu(c);
return curid;

}

#endif
```

MENU1. CPP

```
#ifndef __MEN1_H
#define __MEN1_H

#define MGRAYED 1
#define MNORMAL 2
typedef struct far MENUTYPE {char *name;int id;int stat;};


MENUTYPE barmenu1[]={{"* ",1,MNORMAL},
                                              {"File ",2,MNORMAL},
                                              {"Frame ",3,MNORMAL},
                                              {"Process ",4,MNORMAL},
                                              {"Graph ",5,MGRAYED},
                                              {"",0,MNORMAL},
                                              };

MENUTYPE starmenu[]={{"About..>", 101,MNORMAL},
                                              {"--",0,MNORMAL},
                                              {"Clock", 102,MNORMAL},
                                              {"Help", 103,MGRAYED},
                                              {"Info..>", 104,MNORMAL},
                                              {"",0,MNORMAL},
                                              };

MENUTYPE filemenu[]={{"New", 201,MNORMAL},
                                              {"Open", 202,MNORMAL},
                                              {"Save", 203,MNORMAL},
                                              {"Save as", 204,MNORMAL},
                                              {"Close", 205,MNORMAL},
                                              {"--",0,MNORMAL},
                                              {"Change Dir", 206,MNORMAL},
                                              {"OS shell", 207,MNORMAL},
                                              {"Quit", 208,MNORMAL},
                                              {"",0,MNORMAL},
                                              };

MENUTYPE framemenu[]={{"Snap", 301,MNORMAL},
                                              {"Grab", 302,MNORMAL},
                                              {"Zoom", 303,MNORMAL},
                                              {"Unzoom", 304,MGRAYED},
                                              {"Initialize", 305,MNORMAL},
                                              {"",0,MNORMAL},
                                              };

MENUTYPE procmenu[]={{"Sel Point", 401,MNORMAL},
                                              {"Sel Line", 402,MNORMAL},
                                              {"Image..   ¯", 403,MNORMAL},
                                              {"Auto_detect", 404,MNORMAL},
                                              {"",0,MNORMAL},
                                              };

MENUTYPE imagemenu[]={{"Gray", 502,MNORMAL},
                                              {"Edge", 503,MNORMAL},
                                              {"Connect", 504,MNORMAL},
                                              {"Sobel", 505,MNORMAL},
                                              {"",0,MNORMAL},
                                                    };

MENUTYPE graphmenu[]={{"GRAY..¯", 601,MNORMAL},
                                                    {"RGB.. ¯", 602,MNORMAL},
                                                    {"CIE.. ¯", 603,MNORMAL},
                                                    {"HLS.. ¯", 603,MNORMAL},
                                                    {"HSV.. ¯", 604,MNORMAL},
                                                    {"Net.. ¯", 605,MNORMAL},
                                                    {"",0,MNORMAL},
                                                    };

MENUTYPE gallmenu[]={{"Bar",701,MNORMAL},
                                                    {"Trace",702,MNORMAL},
                                                    {"Meter",703,MNORMAL},
                                                    {"",0,MNORMAL},
                                                    };


#endif
```

MOUSE. CPP

```
#define call_mouse int86(0x33, &inreg, &outreg)
#define EVENTMASK 0x7F
#define SOFTWARE 0
#define HARDWARE 1
#define OFF 0
#define ON 1

#ifndef __DOS_H
#include <dos.h>
#endif

#ifndef __STDIO_H
#include <stdio.h>
#endif


int m_cursor= ON, m_view = OFF;
union REGS inreg, outreg;
typedef struct {int present,
                  buttons;
                  } m_result;
typedef struct {int button_status,
                        button_count,
                        xaxis, yaxis;
                  } m_status;
typedef struct {int x_count,
                        y_count;
                  } m_movement;
typedef struct {unsigned flag,
                              button,
                              xaxis, yaxis;
                  } mouse_event;
typedef struct {unsigned *image,
                              xkey, ykey;
                  } g_cursor;


void m_show(void);
void m_hide(void);
m_result *m_reset(void);
m_status *m_pos(void);
void m_moveto(int xaxis, int yaxis);
m_status *m_pressed(int button);
m_status *m_released(int button);
void m_xlimit(int min_x, int max_x);
void m_ylimit(int min_y, int max_y);
void m_graphic_cursor(int xaxis,int yaxis,
                              unsigned mask_Seg,
                              unsigned mask_Ofs);
void m_text_cursor(int type, unsigned start, unsigned stop);
m_movement *m_motion();
void m_inst_task(unsigned mask,
                  unsigned task_seg,
                  unsigned task_ofs);
void m_ligthpen(int set);
void m_move_ratio(int xsize, int ysize);
void m_conceal(int left, int top, int right, int bottom);
void m_speed(int speed);
void far m_handler();

void initialize_cursors();
void Set_Graphic_Cursor(g_cursor ThisCursor);

m_result *m_reset()
{
      static m_result m;
      inreg.x.ax=0;
      call_mouse;
      m.present= outreg.x.ax;
      m.buttons= outreg.x.bx;
      return(&m);
}

void m_show()
{
      if(m_cursor && !m_view)
      {
            inreg.x.ax=1;
            call_mouse;
            m_view = ON;
      }
}

void m_hide()
{
      if(m_cursor && m_view)
      {
            inreg.x.ax=2;
            call_mouse;
            m_view =OFF;
      }
}

m_status *m_pos()
{
      static m_status m;
      inreg.x.ax=3;
      call_mouse;
      m.button_status=outreg.x.bx;
      m.xaxis=outreg.x.cx;
      m.yaxis=outreg.x.dx;
      return(&m);
}

void m_moveto(int xaxis, int yaxis)
{
      inreg.x.ax=4;
      inreg.x.cx=xaxis;
      inreg.x.dx=yaxis;
      call_mouse;
}

m_status *m_pressed(int button)
{
      static m_status m;
      inreg.x.ax=5;
      inreg.x.bx=button;
      call_mouse;
      m.button_status=outreg.x.ax;
      m.button_count = outreg.x.bx;
      m.xaxis=outreg.x.cx;
      m.yaxis=outreg.x.dx;
      return(&m);
}

m_status *m_released(int button)
{
      static m_status m;
      inreg.x.ax=6;
      inreg.x.bx=button;
      call_mouse;
      m.button_status=outreg.x.ax;
      m.button_count =outreg.x.bx;
      m.xaxis=outreg.x.cx;
      m.yaxis=outreg.x.dx;
      return(&m);
}

void m_xlimit(int min_x, int max_x)
{
      inreg.x.ax=7;
      inreg.x.cx= min_x;
      inreg.x.dx= max_x;
      call_mouse;
}

void m_ylimit(int min_y, int max_y)
{
      inreg.x.ax=8;
      inreg.x.cx=min_y;
      inreg.x.dx= max_y;
      call_mouse;
}

void m_graphic_cursor(int xaxis,
                              int yaxis,
                              unsigned mask_Seg,
                              unsigned mask_Ofs)
{
      struct SREGS seg;
      inreg.x.ax=9;
      inreg.x.bx=xaxis;
      inreg.x.cx=yaxis;
      inreg.x.dx=mask_Ofs;
      seg.es=mask_Seg;
      int86x(0x33, &inreg, &outreg, &seg);
}

void m_text_cursor(int type, unsigned start, unsigned stop)
{
      inreg.x.ax=10;
      inreg.x.bx=type;
      inreg.x.cx=start;
      inreg.x.dx=stop;
      call_mouse;
}
```

```
m_movement *m_motion()
{
      static m_movement m;
      inreg.x.ax=11;
      call_mouse;
      m.x_count = _CX;
      m.y_count = _DX;
      return(&m);
}

void m_inst_task(unsigned mask,
                    unsigned task_seg,
                    unsigned task_ofs)
{
      struct SREGS seg;
      inreg.x.ax=12;
      inreg.x.cx=mask;
      inreg.x.dx=task_ofs;
      seg.es =task_seg;
      int86x(0x33, &inreg, &outreg, &seg);
}

void m_ligthpen(int set)
{
      if(set) inreg.x.ax=13;
             else inreg.x.ax=14;
      call_mouse;
}

void m_move_ratio(int xsize, int ysize)
{
      inreg.x.ax=15;
      inreg.x.cx=xsize;
      inreg.x.dx=ysize;
      call_mouse;
}

void m_conceal(int left, int top, int right, int bottom)
{
      inreg.x.ax=16;
      inreg.x.cx=left;
      inreg.x.dx=top;
      inreg.x.si=right;
      inreg.x.di=bottom;
      call_mouse;
}

void m_speed(int speed)
{     inreg.x.ax=19;
      inreg.x.dx=speed;
      call_mouse;
}
static unsigned arrow_image[32]=
             {      0x3FFF,  0x1FFF,  0x0FFF,  0x07FF,
                    0x03FF,  0x01FF,  0x00FF,  0x007F,
                    0x003F,  0x001F,  0x01FF,  0x10FF,
                    0x30FF,  0xF87F,  0xF87F,  0xFC3F,
                    0x0000,  0x4000,  0x6000,  0x7000,
                    0x7800,  0x7C00,  0x7E00,  0x7F00,
                    0x7F80,  0x7FC0,  0x7C00,  0x4600,
                  0x0600, 0x0300, 0x0300, 0x0180};
static g_cursor ARROW = {NULL,1,1};

static unsigned check_image[32] =
            {     0xF00F, 0xF00F, 0xF00F, 0xE007,
                  0xC003, 0x8001, 0x8001, 0x8001,
                  0x8000, 0x8001, 0x8001, 0xC003,
                  0xE007, 0xF00F, 0xF00F, 0xF00F,
                  0x07E0, 0x07E0, 0x07E0, 0x0810,
                  0x1108, 0x2108, 0x2108, 0x2108,
                  0x21C6, 0x2008, 0x2008, 0x1008,
                  0x0810, 0x07E0, 0x07E0, 0x07E0};
static g_cursor CHECK = {NULL,5,10};

static unsigned cross_image[32]=
            {     0Xf83f, 0xE01F, 0xE00F, 0xC557,
                  0x8C63, 0x1C71, 0x0001, 0x0101,
                  0x0001, 0x1C71, 0x8C63, 0xC447,
                  0xE00F, 0xF01F, 0xF83F, 0xFFFF,
                  0x0000, 0x07C0, 0x0920, 0x1110,
                  0x2108, 0x4140, 0x4004, 0x7C76,
                  0x4004, 0x4104, 0x2108, 0x1110,
                  0x0920, 0x07C0, 0x0000, 0x0000};
static g_cursor CROSS ={ NULL,7,7};

static unsigned glove_image[32]=
            {     0xF9FF, 0xE1FF, 0xE9FF, 0xE9FF,
                  0xE9FF, 0xE849, 0xE800, 0x8924,
                  0x0924, 0x0986, 0x0DFC, 0x2FFC,
                  0x3FFC, 0xEFFC, 0x0000, 0x8001,
                  0x0C00, 0x1200, 0x1200, 0x1200,
                  0x1200, 0x13B6, 0x1249, 0x7249,
                  0x9249, 0x9001, 0x9001, 0x8001,
                  0x8001, 0x8001, 0x8001, 0x7FFE};
static g_cursor GLOVE = {NULL, 4,0};

static unsigned ibeam_image[32]=
            {     0xE187, 0xE007, 0xF81F, 0xFC3F,
                  0xFC3F, 0xFC3F, 0xFC3F, 0xFC3F,
                  0xFC3F, 0xFC3F, 0xFC3F, 0xFC3F,
                  0xFC3F, 0xF81F, 0xE007, 0xE187,
                  0x0C30, 0x0240, 0x0180, 0x0180,
                  0x0180, 0x0180, 0x0180, 0x0180,
                  0x0180, 0x0180, 0x0180, 0x0180,
                  0x0180, 0x0180, 0x0240, 0x0C30};

static g_cursor IBEAM = {NULL, 7, 7};

static mouse_event far *m_events;

void far m_handler()
{
      mouse_event far *save;
      unsigned a, b, c, d;
      a= _AX,
      b= _BX,
      c= _CX,
      d= _DX;
      save= (mouse_event *)MK_FP(_CS-0x10, 0x00C0);
      save -> flag=a;
```

```
     save -> button=b;
     save -> xaxis=c;
     save -> yaxis=d;
}
void initialize_cursors()
{
ARROW.image= arrow_image;
      CHECK.image=check_image;
      CROSS.image=cross_image;
      GLOVE.image=glove_image;
      IBEAM.image=ibeam_image;}
void Set_Graphic_Cursor(g_cursor ThisCursor)
{
      m_graphic_cursor(ThisCursor.xkey,
                      ThisCursor.ykey,
                      _DS,
                      (unsigned)ThisCursor.image);}
#endif
```

MPROCS. CPP

```
#ifndef __MPROCS_H
#define __MPROCS_H

#ifndef __MOUSE_H
#include "c:\final\mouse\mouse.cpp"
#endif

#ifndef __EVENT_H
#include "c:\final\gui\event.cpp"
#endif

#ifndef __CONIO_H
#include <conio.h>
#endif

#ifndef __WPROCS_H
#include "c:\final\wind\wprocs.cpp"
#endif


#ifndef __FFUNC_H
#include "c:\final\frame\ffunc.cpp"
#endif

#ifndef _MENU_PRO
#include "c:\final\menu\menu.hpp"
#endif

#ifndef __PAINT_H
#include "c:\final\menu\paint.cpp"
#endif
#ifndef __PROCTST_H
#include "c:\final\find\proctst.cpp"
#endif
int ev_hit(void);

int ev_hit(void){
m_status *m;
int flag=0;
if(kbhit()) flag=1;
m=m_pos();
if(m->button_status) flag=2;
return flag;

}

int event_controller(void){
int flag=0;
m_status *m;
event_box *event;
int c;

while(!flag){
  flag=ev_hit();
}

switch (flag){
```

```
            case 2:    m=m_pos();
                                 event=menu.search_inbounds(m->xaxis/2,m-
>yaxis);
                                 if(event->id)handle_barmenu(event);
                                 event=gwindow.search_inbounds(m-
>xaxis/2,m->yaxis);
                                 if(event->id)handle_window(event);
                                 break;
            case 1:    break;

            };

return flag;
}


void proc_dropmenu(int id){
switch(id){


 case 208: exit(1);
                         break;

 case 301: snap();
                         break;
 case 302: grab();
                         break;
 case 303: zoom();
                         framemenu[2].stat=MGRAYED;
                         framemenu[3].stat=MNORMAL;
                         break;
 case 304: unzoom();
                         framemenu[3].stat=MGRAYED;
                         framemenu[2].stat=MNORMAL;
                         break;
 case 305: f_init();
                         break;

 case 404: auto_find_obj();
                     break;
 case 502: new gwind(2,20,20,96,64,paint_gray);
                     imagemenu[0].stat=MGRAYED;
                         break;
 case 503: new gwind(4,30,30,96,64,paint_box);
                         break;
 case 504: new gwind(6,40,40,96,64,paint_connect);
 }


}

void handle_window(event_box *witem){
m_status *m;
int x,y;
int xa,ya,heighta,widtha;

void far *rect[4];
rect[0]=NULL;
Set_Graphic_Cursor(GLOVE);
m=m_pos();
```

```
x=m->xaxis;
y=m->yaxis;

xa=witem->xt;
ya=witem->yt;
heighta=witem->height;
widtha=witem->width;
gwindow.to_end(witem->id);
setcolor(4);

while(m->button_status)
  {
  m=m_pos();
  if((x!=m->xaxis)||(y!=m->yaxis)){
      m_hide();
      if(rect[0]!=NULL)gput_ubox(xa,ya,widtha,heighta,rect);

      xa+=(m->xaxis-x)/2;
      if(xa<1||xa+widtha>GMAX_X-1)xa-=(m->xaxis-x)/2;
      ya+=m->yaxis-y;
      if(ya<18||ya+heighta>GMAX_Y-1)ya-=m->yaxis-y;
      x=m->xaxis;
      y=m->yaxis;
      gget_ubox(xa,ya,widtha,heighta,rect);
      rectangle(xa,ya,xa+widtha,ya+heighta);
      m_show();
      }
}
  m_hide();
  if(rect[0]!=NULL)gput_ubox(xa,ya,widtha,heighta,rect);
  if(rect[0]!=NULL){for(x=0;x<4;x++)farfree(rect[x]);}
  setfillstyle(SOLID_FILL,90);
  bar(witem->xt,witem->yt,witem->xt+widtha,witem->yt+heighta);
  witem->xt=xa;
  witem->yt=ya;
  if((m->xaxis/2>295)&&(m->yaxis>170)
       &&(m->xaxis/2<315)&&(m->yaxis<195))gwindow.remove_item(witem-
>id);
  gwindow.repaint_all();
  m_show();
  Set_Graphic_Cursor(ARROW);
  draw_bin();
}

#endif
```

SCREEN. CPP

```
#ifndef _DAC256_
#define _DAC256_
typedef unsigned char DacPalette256[256][3];
#endif

extern int far _Cdecl Svga256_fdriver[];

/* These are the currently supported modes */
#ifndef   SVGA320x200x256
#define     SVGA320x200x256         0     /* 320x200x256 Standard VGA */
#endif

#ifndef XNOR_PUT
#define     XNOR_PUT    5
#define NOR_PUT         6
#define NAND_PUT  7
#define TRANS_COPY_PUT  8     /* Doesn't work with 16-color drivers */
#endif

#define RGB(r,g,b) ((r & 31)<<10) | ((g & 31)<<5) | (b & 31)

int RealDrawColor(int color)
{
  if (getmaxcolor() > 256)
    setrgbpalette(1024,(color>>10)&31,(color>>5)&31,color&31);
  return(color);
}

int RealFillColor(int color)
{
  if (getmaxcolor() > 256)
    setrgbpalette(1025,(color>>10)&31,(color>>5)&31,color&31);
  return(color);
}

int RealColor(int color)
{
  if (getmaxcolor() > 256)
    setrgbpalette(1026,(color>>10)&31,(color>>5)&31,color&31);
  return(color);
}
/* vga extended mode initilaisation and gray level palette formation */

#ifndef __GRAPHICS_H
#include <graphics.h>
#endif

int WhitePixel(void);
int huge DetectVGA256(void);
void init_screen(void);
void gray256(void);


int WhitePixel()
{
  if (getmaxcolor() > 256)
    return(32767);
  return(255);
}

int huge DetectVGA256()
{
  return 0;
}


void init_screen()
{
  int Gd = DETECT, Gm;
  installuserdriver("Svga256",DetectVGA256);
  initgraph(&Gd,&Gm,"c:\\final\\scr\\");
}
void gray256(){
int x;
  for(x=0;x<240;x+=3){
      setrgbpalette(x+16,x>>2,x>>2,x>>2);
      setrgbpalette(x+17,x>>2,x>>2,(x+3)>>2);
      setrgbpalette(x+18,x>>2,(x+3)>>2,(x+3)>>2);
      }

}

#endif
```

WIND. CPP

```
#ifndef __WIND_H
#define __WIND_H

#ifndef __EVENT_H
#include "c:\final\gui\event.cpp"
#endif
class gwind{
public:
 event_box *wind_evnt;
 int id_val;

gwind(int id,int x,int y,int width, int height,void
(*paint_func)(event_box *));
~gwind(void);
void move_to(int x,int y);
void pop_up(void);
};

gwind::gwind(int id,int x,int y,int width, int height, void
(*paint_func)(event_box *)){
 id_val=id;

 wind_evnt=new event_box(id,x,y,width,height,paint_func);
 gwindow.add_item(wind_evnt);
}

gwind::~gwind(void){
 gwindow.remove_item(wind_evnt->id);
 gwindow.repaint_all();

}

void gwind::pop_up(void){
      gwindow.to_end(wind_evnt->id);
      gwindow.repaint_all();
}

void gwind::move_to(int x,int y){
      gwindow.move_item(id_val,x,y);
      gwindow.repaint_all();
      }
#endif
```

WPROCS. CPP

```
#ifndef __GRAPHICS_H
#include <graphics.h>
#endif

#ifndef __ALLOC_H
#include <alloc.h>
#endif
#ifndef __WPROCS_H
#define __WPROCS_H
void gget_ubox(int x,int y,int width, int height,void far *u_box[4]);
void gput_ubox(int x,int y,int width, int height, void far *u_box[4]);

void gget_ubox(int x,int y,int width, int height,void far *u_box[4]){
if(u_box[0]!=NULL){ farfree(u_box[0]);
                                  farfree(u_box[1]);
                                  farfree(u_box[2]);
                                  farfree(u_box[3]);
                                 }

u_box[0]=farmalloc(imagesize(x,y,x+width,1));
u_box[1]=farmalloc(imagesize(x,y,x+width,1));
u_box[2]=farmalloc(imagesize(x,y,1,y+height));
u_box[3]=farmalloc(imagesize(x,y,1,y+height));
getimage(x,y,x+width,y+1,u_box[0]);
getimage(x,y+height,x+width,y+height+1,u_box[1]);
getimage(x,y,x+1,y+height,u_box[2]);
getimage(x+width,y,x+1+width,y+height,u_box[3]);

}

void gput_ubox(int x,int y,int width, int height, void far *u_box[4]){
putimage(x,y,u_box[0],COPY_PUT);
putimage(x,y+height,u_box[1],COPY_PUT);
putimage(x,y,u_box[2],COPY_PUT);
putimage(x+width,y,u_box[3],COPY_PUT);

}

#endif
```

VARIOUS PROGRAMS

DISPDAT.C

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <alloc.h>
#include "fg2.i"
#include "gprint.i"
#include "mouse.i"

typedef struct{int xl,yt,xr,yb;}win;

void *bitim1;
int graphdriver = DETECT,graphmode;
char *drivepath = "\\TC";
struct time now;
int MaxX,MaxY,Xlst=0,Ylst=0;
m_status m;
int Xlst,Ylst;
int flag=0,sample_cnt=0;
unsigned char Rlst,Glst,Blst;
attr *q;
int Cx,Cy,Cxl=0,Cyl=0,Tnew,Tlst=0;
float Tac;

main()
{  int x,c;

   init_cfg();
   init_graph();
   init_mouse();
   init_screen();
   init_options();
   while(!flag) sel_option();
   closegraph();
   exit(1);

}
init_cfg()
{
  freeze();
  init_regs();
/*  set_cfgframe();  */
  empty_buffer;
  olut1();
  ilut();
  olut2();
   snap();


}

sel_option()
{
```

```
int selx,sely,opt;

   m.button_status=0;
   m_xlimit(276,424);
   m_ylimit(55,205);
   m_pos(286,56);
   m_show();

   while(!m.button_status) m=*m_pos();
   selx=(int)((m.xaxis-275)/50)+1;
   sely=(int)((m.yaxis-55)/74)+1;
   opt=selx+(sely*3)-3;
   switch(opt)
   {
     case 1: break;

     case 2: {grab();
           break;}

     case 3: {snap();
           break;}

     case 4: {proc_data();
           break;}

     case 5: {sel_pos();
           break;}

     case 6: {flag=1;
           break;}
   }
}

proc_data()
{
   cls_ovl();
   m.button_status=0;
   while(!m.button_status)
   {
   outb(f_DACcommand,0x1c);
   m=*m_pos();
   acq_data();
   do_calcs();
   disp_data();
   }
}




do_calcs()
{ int x;
  double temp=0;
  Cx=0;
  Cy=0;
  sample_cnt++;
  gettime(&now);
  if((x=(Rlst+Glst+Blst))!=0)
```

```
    {
    Cx=(int)(Rlst*100/x);
    Cy=(int)(Blst*100/x);
    }
    if(Rlst!=0)
    {
    temp= atan2((double)Blst,(double)Rlst);
    Tnew=(int)(50*temp)+10;
    }
    else Tnew=89;
    if(Rlst==0 && Blst==0) Tnew=10;
    Tac=((float)(Tnew-10)/5.2) +22.0;
 }
 init_graph()
 {
   initgraph(&graphdriver,&graphmode,drivepath);
   MaxX = getmaxx();
   MaxY = getmaxy();
   setbkcolor (8);
   setcolor (4);
   rectangle(0,0,MaxX,MaxY);
   rectangle(4,4,MaxX-4,MaxY-4);

 }

 init_mouse()
 {
   m_result*rodent;
   rodent=m_reset();
   if(rodent->present)
   {
   initialize_cursors();

   }
   else
   {
   outtextxy(10,10,"no mouse present");
   exit(1);
   }

 }

init_screen()


{ unsigned clip_size;
  settextstyle(1,0,2);
  settextjustify(1,0);
  setcolor(14);
  outtextxy((int)MaxX/2,30,"TEMPERATURE MONITOR");
  settextstyle(2,0,4);
  outtextxy(90,45,"MONITORING POSITION");
  outtextxy(100,197,"PRIMARY COLOUR INTENSITIES ");
  outtextxy(530,70,"OBSERVATION COORDINATES");
  outtextxy(530,100,"PRIMARY COLOUR VALUES");
  outtextxy(530,130,"CIE PLOT VALUES");
  outtextxy(530,45,"SYSTEM INFORMATION");
  outtextxy(490,180,"SAMPLE COUNT");
  outtextxy(490,200,"TIME        ");
  outtextxy(490,160,"TEMPERATURE ");
  outtextxy(300,245,"CIE PLOT");
  outtextxy(530,295,"CIE HISTORY");
  outtextxy(230,45,"TEMPERATURE");
  setcolor(15);
  outtextxy(90,465," BLUE      GREEN      RED");
  setfillstyle(1,1);
  bar(20,50,175,155);
  setfillstyle(1,7);
  bar(186,251,419,469);
  line(18,456,172,456);
  line(195,457,400,457);
  line(195,457,195,276);
  rectangle(18,48,175,155);
  rectangle(18,200,172,470);
  rectangle(185,250,420,470);
  rectangle(440,50,620,250);
  rectangle(440,300,620,470);
  rectangle(200,50,260,200);

  settextjustify(0,0);
  outtextxy(450,80,"XAXIS        YAXIS     ");
  outtextxy(450,110,"RED     GREEN      BLUE");
  outtextxy(450,140,"X VALUE        Y VALUE  ");
  outtextxy(290,465,"x");
  outtextxy(189,365,"y");
  outtextxy(207,150,"24 C");
  outtextxy(207,90,"39 C");
  outtextxy(600,160,"C");
  putpixel(266,389,4);
  putpixel(266,390,4);
  putpixel(267,389,4);
  putpixel(267,390,4);
  setcolor(4);
  setfillstyle(1,4);
  pieslice(240,170,0,360,8);
  rectangle(235,55,245,170);
  clip_size=imagesize(440,300,620,470);
  bitim1=malloc(clip_size);
}

init_options()
{ setcolor(63);
  setfillstyle(1,20);
  bar(276,56,424,204);
  rectangle(275,55,425,205);
  line(325,55,325,205);
  line(375,55,375,205);
  line(275,105,425,105);
  line(275,155,425,155);
  outtextxy(275,80,"  STOP     GRAB     SNAP ");
  outtextxy(275,130,"  PROC   SEL POS    REF ");


}
sel_pos()
{
   m_xlimit(0,768);
   m_ylimit(0,512);
   m_hide();
   outb(f_DACcommand,0x14);

   cls_ovl();
   Stroll();
   xpos(Xlst);
   ypos(Ylst);
   outb(f_otlutcntrl,0x00);

}

Stroll()
{
   m.button_status=0;
   m_hide();
   while(!m.button_status)
    {
      settextjustify(2,0);
      setcolor(4);
      m=*m_pos();

      if(m.xaxis!=Xlst)
      {
        putpixel(20+Xlst/5,50+Ylst/5,1);
        blank(Xlst,Ylst);
        Xlst=m.xaxis;
        cross(Xlst,Ylst);
        putpixel(20+Xlst/5,50+Ylst/5,15);
        gprintxy(500,80,"    ");
        gprintxy(500,80,"%d",Xlst);
      }
      if(m.yaxis!=Ylst)
      {
          putpixel(20+Xlst/5,50+Ylst/5,1);
        blank(Xlst,Ylst);
        Ylst=m.yaxis;
        cross(Xlst,Ylst);
        putpixel(20+Xlst/5,50+Ylst/5,15);
        gprintxy(580,80,"    ");
        gprintxy(580,80,"%d",Ylst);
      }
   }

}

disp_data()
{

  settextjustify(2,0);
  getimage(442,301,619,469,bitim1);
  if(Rlst!=q->RED)
   {    setcolor(8);
      rectangle(140,455-Rlst,170,455);
      Rlst=(int)q->RED;
      setcolor(4);
      gprintxy(490,110,"    ");
      gprintxy(490,110,"%d",Rlst);
      rectangle(140,455-Rlst,170,455);

   }
   if(Glst!=q->GREEN)
   {    setcolor(8);
      rectangle(80,455-Glst,110,455);
      Glst=(int)q->GREEN;
      setcolor(4);
      gprintxy(550,110,"   ");
      gprintxy(550,110,"%d",Glst);
      setcolor(2);
      rectangle(80,455-Glst,110,455);
   }

   if(Blst!=q->BLUE)
   {
      setcolor(8);
      rectangle(20,455-Blst,50,455);
      Blst=(int)q->BLUE;
      setcolor(4);
      gprintxy(610,110,"   ");
      gprintxy(610,110,"%d",Blst);
      setcolor(1);
      rectangle(20,455-Blst,50,455);
   }
   setcolor(4);
   gprintxy(520,140,"   ");
   gprintxy(520,140,"%d",Cx);
   gprintxy(610,140,"   ");
   gprintxy(610,140,"%d",Cy);
   gprintxy(580,160,"     ");
   gprintxy(600,180,"        ");
   gprintxy(600,200,"        ");
   gprintxy(580,160,"%5.2f",Tac);
   gprintxy(600,200,"
%02d:%02d:%02d",now.ti_hour,now.ti_min,now.ti_sec);
   gprintxy(600,180,"%d",sample_cnt);
   putpixel(200+(Cxl*2),455-(Cyl*2),7);
   putpixel(200+(Cxl*2),456-(Cyl*2),7);
   putpixel(201+(Cxl*2),455-(Cyl*2),7);
   putpixel(201+(Cxl*2),456-(Cyl*2),7);
   Cxl=Cx;
   Cyl=Cy;
   putpixel(200+(Cx*2),455-(Cy*2),1);
   putpixel(200+(Cx*2),456-(Cy*2),1);
   putpixel(201+(Cx*2),455-(Cy*2),1);
   putpixel(201+(Cx*2),456-(Cy*2),1);
   setfillstyle(1,8);
   bar(236,170-(Tlst),244,170);
   bar(619,301,619,469);
   setfillstyle(1,4);
   bar(236,170-(Tnew),244,170);
   Tlst=Tnew;

   putpixel(619,469-(Cx*1.5),4);
   putpixel(619,469-(Cy*1.5),1);
   putimage(441,301,bitim1,0);

}

acq_data()
{

     int x,y,totr=0,totg=0,totb=0;
     snap();
     for(x=0;x<=7;x++)
     {
     for(y=0;y<=7;y++)

       {
      xpos(X1st+x);
      ypos(Y1st+y);
        q=get_pix();
      totr=totr+q->RED;
      totg=totg+q->GREEN;
      totb=totb+q->BLUE;
        }
      }
      q->RED=totr/64;
      q->GREEN=totg/64;
      q->BLUE=totb/64;
      grab();

}
```

Publications

Intrinsic optical fibre sensors for axle detection

by S. Ahmed and Professor G. R. Jones, *University of Liverpool*
and by J. R. Spindlow and A. Stevens, *Transport and Road Research Laboratory*

Introduction. The detection of vehicles may be achieved using distributed electronic or electromagnetic sensors embedded across a carriageway. Such sensors respond to the passage of a vehicle either by the weight of the vehicle compressing a pneumatic, capacitive, piezoelectric or peizoresistive element to produce an electronic signal or through the vehicle itself modifying the electromagnetic coupling of an inductive loop.

Whereas such sensors are providing good service, improvements would nonetheless be welcome for minimising service requirements, reducing installation costs and effort, improving reliability and life, and extending the function for detecting stationary (as opposed to moving) vehicles and possibly for vehicle type discrimination.

It may be argued that the time is right for investigating whether a generically new form of sensing technology would be capable of meeting these demands better than the existing technology. Such a technology, which has the potential for meeting these requirements and which is currently rapidly emerging, utilises optical fibres for sensing purposes (as distinct from their use for data transmission in telecommunications).

Optical fibres for the detection of distributed stress are already available. There is industrial experience with such systems for intruder detection and for security purposes. Systems based upon these fibres have recently been evaluated for vehicle detection through a joint project involving the Transport and Road Research Laboratory at Crowthorne and the Department of Electrical Engineering and Electronics of the University of Liverpool. Such fibres have been suitably encapsulated and subjected to rigorous tests both in the laboratory and on a carriageway site. The tests indicate that such optical fibre sensing technology has the potential for meeting the improvements sought in axle sensing.

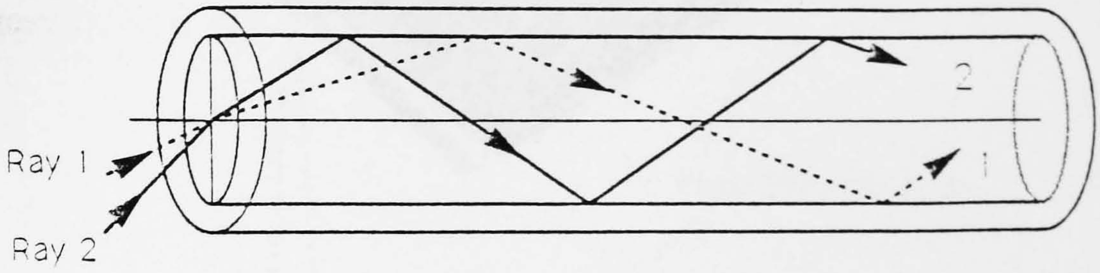


Fig 1. Ray model for light propagation along an optical fibre.

Optical fibre propagation

Optical fibres are extended lengths of small diameter (5 µm to 1 mm) transparent materials (glass or polymer) suitably encapsulated for the transmission of light signals. Such fibres have been utilised in medical endoscopes for many years and more recently for high data-rate telecommunication links. The present trend is for their use in industrial process monitoring as optical sensing elements.

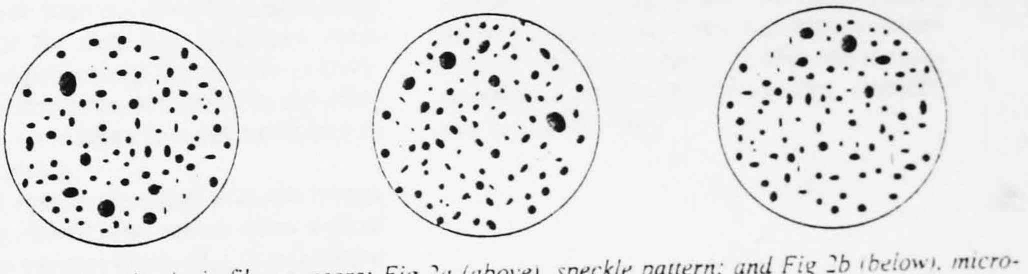
Provided that the fibre diameter is much greater than the wavelength of light, optical propagation through such fibres may be described in terms of a large number of rays which are continuously reflected from the cylindrical wall of the inner glass fibre (Fig 1). This propagation may be disturbed by mechanically compressing the fibre at various locations along its length.

If light from a highly coherent source such as a laser is transmitted through the fibre then interference between monochromatic light rays forms a randomly-distributed pattern of bright and dark spots at the fibre end known as a 'speckle pattern' (Fig 2a). As the fibre condition is perturbed this interference pattern flickers and the occurrence of such change may be detected with a highly-tuned optoelectronic detection system.

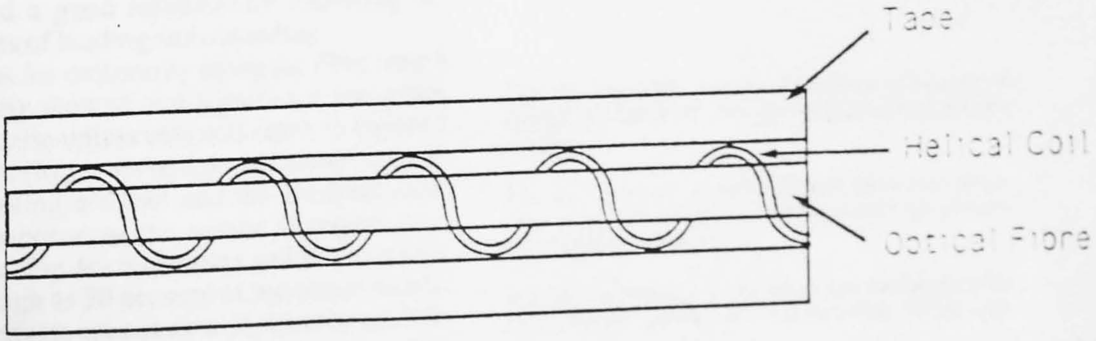
Propagation through a fibre is also affected by regular microscopic perturbations of a precisely-determined periodicity along the length of the fibre. These may be produced either by a helically-wound fibre around the light-propagating fibre (Fig 2b) or by regular corrugations in a supporting substrate. Compression of the microbend structure increases the leakage of light rays from the fibre core leading to the attenuation of the propagated light. The proportion of light leaked in this manner is also wavelength dependent so that the spectrum of the transmitted light is modified by the microbending.

These effects provide the basis for the optical detection of fibre compression via the monitoring of the optical signal propagated. The speckle pattern approach forms the basis of an intruder detection system commercially produced by Pilkington whilst the microbend approach is utilised in a safety system produced by Herga Electric Ltd. The latter system has the potential of being upgraded for possible stress measurement using chromatic monitoring methods developed at the University of Liverpool for discriminating the wavelength dependent changes produced by various degrees of microbending.

These systems have been examined for possible adaptation for road sensing applications with regard to their capability of being suitably encapsulated for carriageway use, having sufficient robustness and for adequate sensitivity for detecting stationary and types of vehicles.



Stress-responsive intrinsic fibre sensors: Fig 2a (above), speckle pattern; and Fig 2b (below), microbending helix.



encapsulation

as the optical fibres used in the present
igation have already withstood opera-
der robust industrial conditions which
their normal market targets the encap-
on necessary for carriageway operation
ery much an unknown factor. Thus not
does the encapsulation need to meet the
lished demands for carriageway instal-
(robustness, reliability, ease of manu-
e and installation, and carriageway ad-
) but it also needs to be compatible
the sensing fibre element. It needs to
ve adequate sensor sensitivity whilst
ting against excessive overloads, the
sulation materials need to be compati-
th the optical fibre materials and there
s to be reasonably uniform sensitivity
the length of the encapsulated sensor.

right). Optical fibre encapsulation.

suitable encapsulation procedure was
oped by the Transport and Road
rch Laboratory. This was achieved by
the fibre sensing elements in a modi-
T-section mould (Fig 3), split horizontal-
wo levels so that thin threads could be
across to form suspensions for an out-
and a return fibre. At the remote end the
was formed into a loop with a minimum
adius of 25 mm (to minimise unneces-
bend-induced attenuation) and held
light tension with further threads. The
cable was sheathed with heat-shrink
g at its entry and exit points from the
and coated with primer. The mould
lled with Devcon Flexane 80 liquid and
ed to cure for one week.

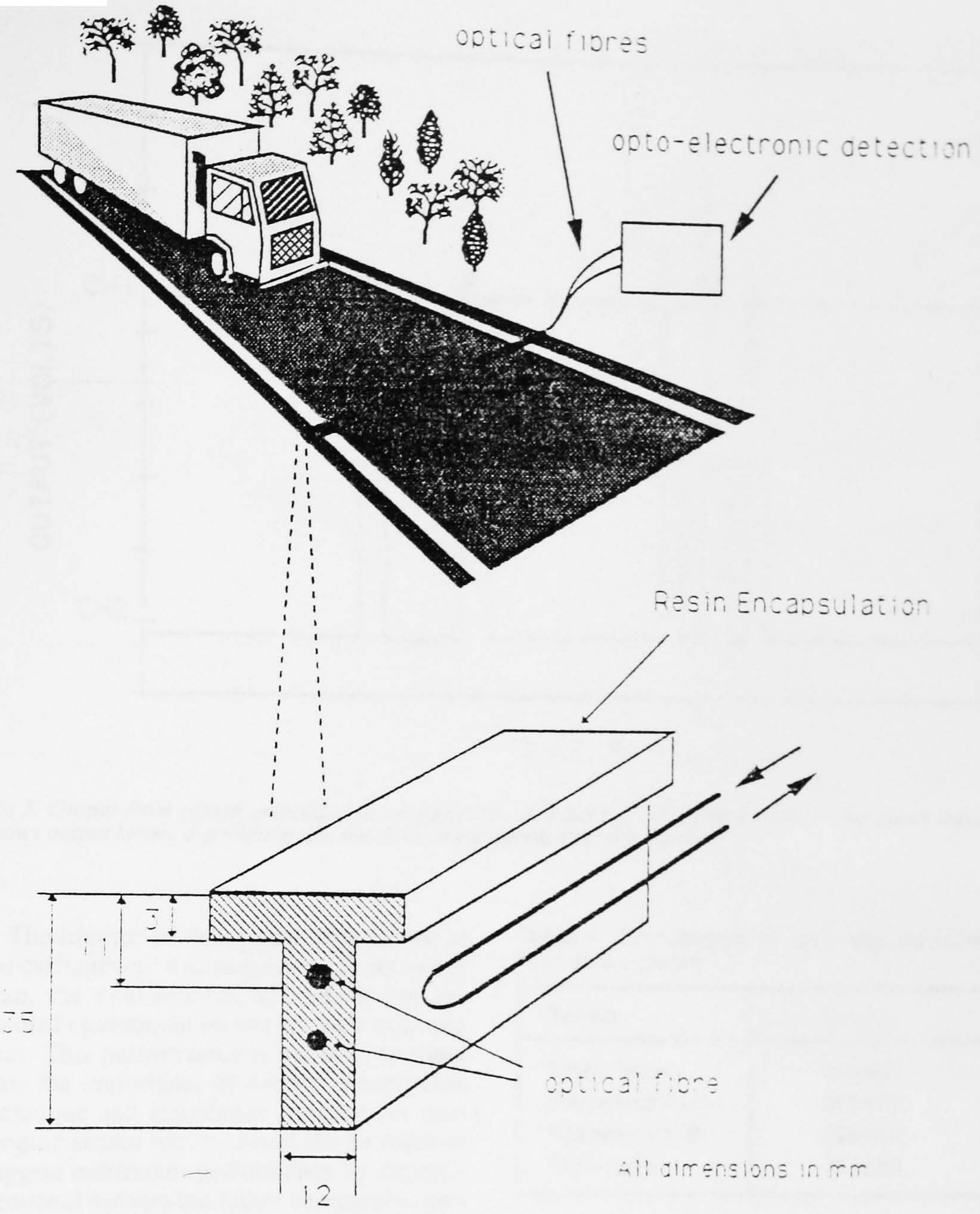
oring instrumentation

tput of the optical fibre sensors was
red both with the units of the commer-
nufacturers and also with experimen-
omatic units developed at the Univer-
Liverpool.

Pilkington unit detected fluctuations
speckle pattern via an AC-sensitive
tector unit which responded only to
ts rather than steady optical signals.
tem has been optimised to respond to
ansients and so provides extremely
e responses free from the effects of
erm light intensity drifts. However.
uires the laser diode source to be
abilised.

Herga unit operates in a pulse ampli-
dulation mode. When the pulse am-
rops below an adjustable threshold.
o increased microbending, a relay is
d to implement a safety interlock.
em therefore operates in a fail-safe
the sense that regular optical pulses
e through the system in its dormant
establishing its integrity.

omatic detection system utilises at
photodetectors with overlapping
esponses to monitor changes in the
signature of the transmitted light.
m is intensity independent over al-
orders of intensity and is therefore
to the deficiencies of intensity-
tems (e.g. source intensity fluctua-
ct of fibre connectors, etc.).



Laboratory tests

Laboratory tests were performed on the various fibre sensing systems to establish sensitivities, uniformity of response, temperature stability and long-term stability.

The Pilkington speckle pattern system is a highly-sensitive system whose performance was not impaired by encapsulation. Its mode of operation makes it insensitive to temperature fluctuations and long-term drift. However, the penalties for such a performance are that the system does not have the potential for weighing nor for detecting stationary vehicles. More significantly the system is oversensitive to vibrations remote from the sensing location, so leading to a susceptibility to spurious counting.

Some test results obtained with the Herga system and a ribbed fibre sensor show a good sensitivity to various loads (Fig 4a) despite a two orders of magnitude reduction in intensity following encapsulation. The system showed a good repeatability following sequences of loading and unloading.

Tests for uniformity along the fibre length (Fig 4b) showed that significant variations could arise unless care was taken to ensure a fairly uniform, not so tight buffering between the sensing element and the encapsulation. The response of the sensor to temperature change (Fig 4c) is complex and at worse may be as high as 50 per cent of the output signal. Such effects are not over-serious for axle detection purposes since they result in slow zero point drift well within the dynamic measurement range of the system so that the transient signals due to a passing vehicle may be discriminated. The possible advantage of the Herga system for vehicle signature discrimination should not be overlooked and should not be too disadvantaged by the above spurious effects if relying on amplitude or waveform change rather than absolute values.

Test results obtained for a helically microbent fibre with chromatic detection gave similar results to those with the Herga instrumentation, but with somewhat improved resolution and less tendency to drift and to temperature dependence. Long-term stability was also improved.

Fig 4a (opposite, upper). Variation of output of Herga system with load for encapsulated Herga sensor.

Fig 4b (opposite, centre). Graph showing variation of sensitivity to load with position for encapsulated Herga sensor.

Fig 4c (opposite, lower). Intensity variation with temperature cycling of encapsulated Herga sensor.

riageway tests

Laboratory tests provided sufficient evi- to make site tests under real carriage- conditions worthwhile. To date these been restricted to axle detection tests have mainly been conducted with the cle pattern system, although some tests also been implemented with the micro- sensor system. The primary aim has to establish count accuracy and lifetime real operating conditions. The first set ts were performed at a TRRL test site on M4 motorway near Theale in Berkshire.

The T-section encapsulated transducer 3) was mounted in a carefully-cut slot in carriageway half filled with Stag Traffic adhesive. For comparison purposes con- onal electronic sensors were also in- ed one metre apart, next to the optical sensor. These were triboelectric sup- by Traffic Technology Ltd, piezo- tric made by Pioneer Weston and lied by Golden River Traffic Ltd, piezo- tric – high sensitivity from the Gates er Co. and piezoresistive, also from the s Rubber Co. The processed output puls- m each sensor were counted electroni- over the lift of the sensor element.

gure 5 shows a typical response of the al fibre sensor to a vehicle travelling at ximately 90 km/h. The upper and lower s show the output signals before and digitisation.

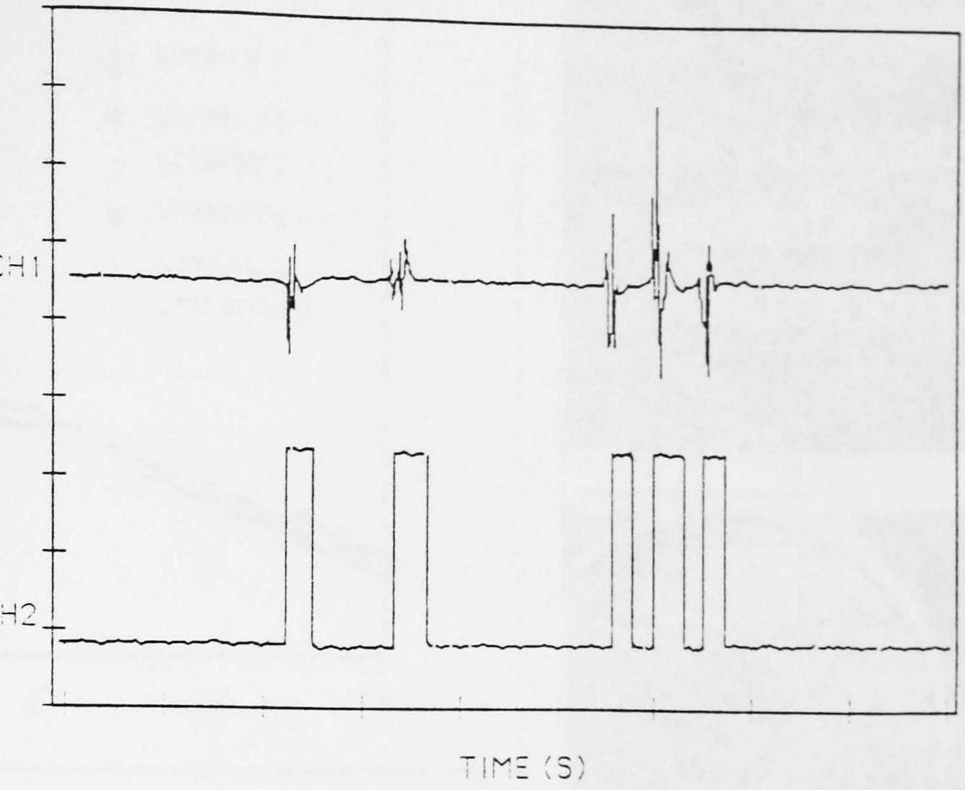


Fig 5. Output from sensor embedded in carriageway, detecting an articulated lorry — the upper trace shows output before digitisation and the lower trace output after digitisation.

The lifetime of the optical fibre sensor in the carriageway was as good as, if not better than, the conventional sensors. It has remained operational on site for well over one year. This performance is believed to illustrate the importance of correct construction technique and installation practice for prolonging sensor life. It should not be taken to suggest indifferent performance by the conventional sensors but rather comparable performance by the optical fibre sensor.

The behaviour of the optical fibre sensor with regard to correct axle counting has also been compared with the conventional electronic sensor (Table I). The results indicate the optical sensor to give a count accuracy of better than 1 per cent. Since the optical fibre sensor gave the lowest reading there is the possibility that an occasional axle was missed due to slow recovery or inappropriate signal discrimination during the passage of two closely-spaced axles.

Preliminary tests performed with a Herga type system at a second carriageway site have indicated that not only is axle-counting possible with this type of sensor, but that there is also evidence of a capability for vehicle type discrimination.

Table I. Comparison of axle counts from four axle sensors

Sensor	Count
Triboelectric	264 687
Piezoelectric (A)	263 982
Piezoelectric (B)	265 898
Fibre-optic	263 089

Conclusions

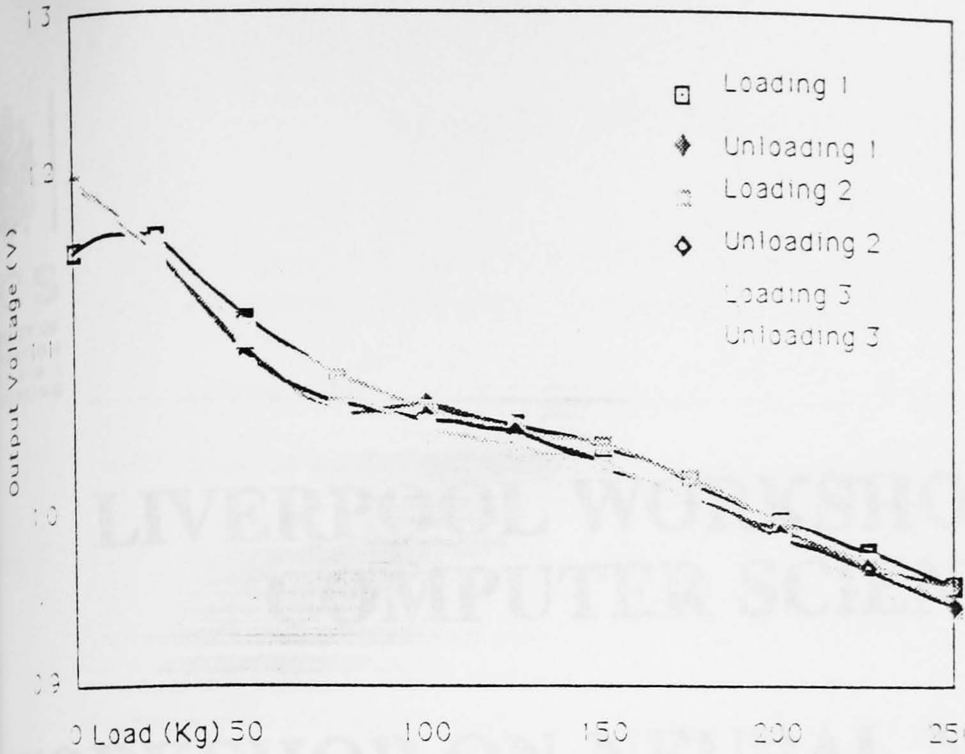
Sufficient knowledge has been gained to indicate that optical fibres may be suitably encapsulated to withstand the rigours of carriageway operation to have a useful life expectancy. Their axle-count accuracy is satisfactory and certain forms have the potential for discriminating between different vehicle types. Some types also have the capability for detecting stationary vehicles and to give fail-safe operation.

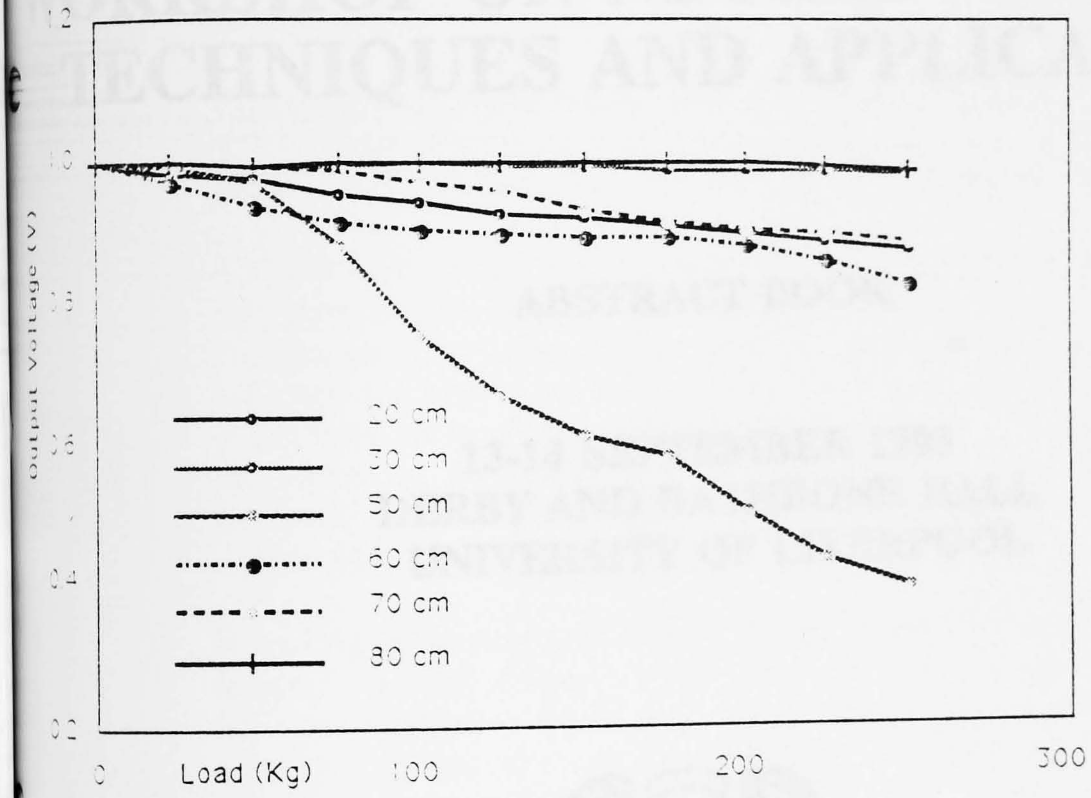
Sufficient laboratory tests have been performed to provide a knowledge base from which future developments can evolve. Nonetheless, more extensive evaluation is required in order to determine the most appropriate forms for specific duties and to address the question of the cost-effectiveness of the technology.

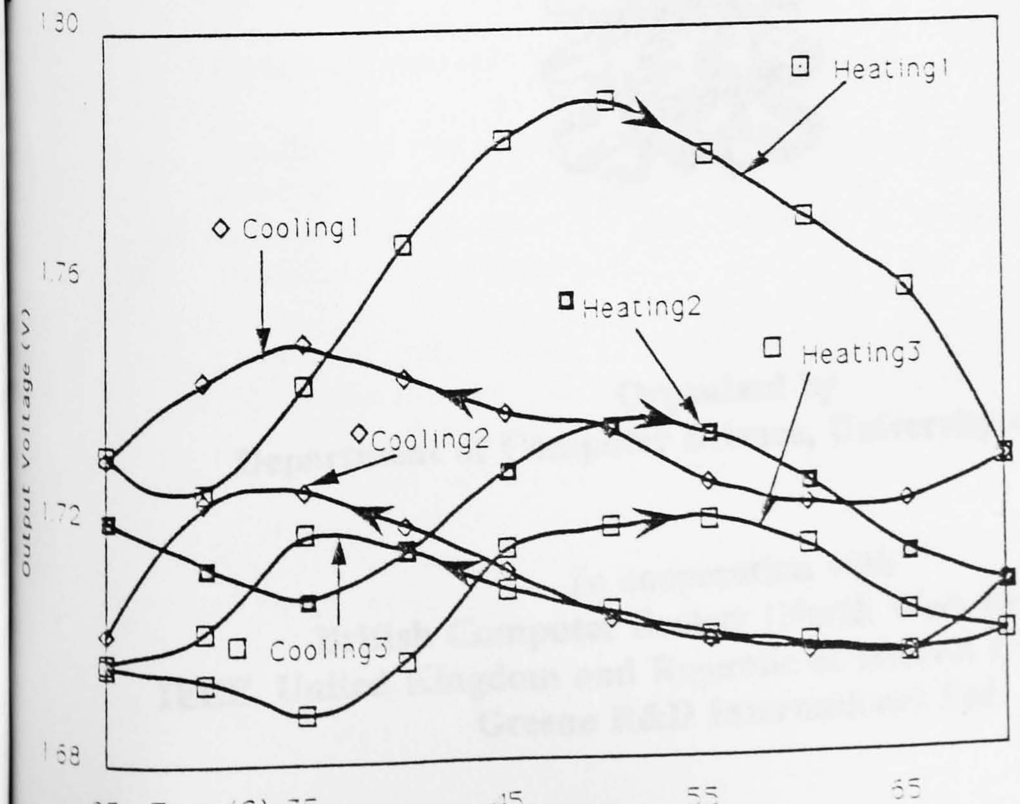
However, there is a need for the industrial sector to realise that to achieve a successful product through the use of a generically new technology such as optical fibre sensing there is a need for strong collaboration between end-users with expertise in the vehicle counting sector and organisations with the necessary fundamental knowledge of optical fibre sensing.

The work described in this paper forms part of the programme of the Transport and Road Research Laboratory and the paper is published by permission of the Director.

The address of Professor Jones: Department of Electrical Engineering and Electronics, University of Liverpool, Brownlow Hill, P.O. Box 147, Liverpool L69 3BX; and of Messrs Spindlow and Stevens: Traffic Operations Division, Transport and Road Research Laboratory, Crowthorne, Berkshire RG11 6AU.


1.3
1.2
1.1
1.0
0.9
Output Voltage (V)
Loading 1
Unloading 1
Loading 2
Unloading 2
Loading 3
Unloading 3
0 Load (Kg) 50
100
150
200
250


1.2
1.0
0.8
0.6
0.4
0.2
Output Voltage (V)
20 cm
30 cm
50 cm
60 cm
70 cm
80 cm
0
Load (Kg)
100
200
300


1.80
1.76
1.72
1.68
Output Voltage (V)
Heating1
Cooling1
Heating2
Heating3
Cooling2
Cooling3
25 Temp(C) 35
45
55
65

CS
OCIETY OF
MATION
TEMS
NEERING

Institute of Electrical &
Electronics Engineers

LIVERPOOL WORKSHOPS ON COMPUTER SCIENCE

WORKSHOP ON NEURAL NETWORKS: TECHNIQUES AND APPLICATIONS

ABSTRACT BOOK

13-14 SEPTEMBER 1993
DERBY AND RATHBONE HALL
UNIVERSITY OF LIVERPOOL

Organised by
Department of Computer Science, University of Liverpool

In cooperation with
British Computer Society (North West Branches)
IEEE, United Kingdom and Republic of Ireland Computer Chapter
Greene R&D International Ltd

REMOTE SENSING USING NEURAL NETWORKS

S. Ahmed, P.C. Russell, P. Lisboa and G.R. Jones
Department of Electrical Engineering and Electronics, University of Liverpool

Introduction

Remote sensing is a technique whereby measurements of specific parameters are made using non contact techniques, usually at a distance from the object under observation. The most common form of remote sensing is achieved using optical techniques where the object is observed at a distance using a camera. The development of CCD cameras and computer acquisition hardware have made this technology a suitable solution for remote sensing in applications where standard contact sensing is hazardous or impractical. This paper describes a new way to use colour information to measure physical parameters remotely, using a neural network to linearise the complex path traversed in colour space as a function of the value of the measureand.

The tasks of real world remote sensing is complex and subject to a great number of external effects. These effects include light source variations, glare, shadowing and sensor degradation. These effects, whether alone or in combination, will degrade the quality of the image data.

A colour CCD camera can be used to acquire the image providing information in the form of red, green and blue intensities. Chromatic processing [1,2,3] is a technique whereby colour information in the image can be quantified independently of the individual intensities of the red green and blue values but rather as a fixed combination of all three. Thus, sensors used in this work are carefully selected so that they produce a reproducible chromatic change over the surface of the sensor which depends on the measureand. However, output from a chromatic sensor obtained in this way usually exhibits a very non-linear relationship with the measureand.

Neural networks were identified as an ideal method for analysing the data from the sensor images. A number of neural network topologies exist, offering analogue and digital processing. For these applications the Multi Layered Perceptron (MLP) was used, trained using standard error back propagation, since this neural network architecture is capable of handling complex non-linear analogue functions.

Chromatic Sensors

Results from two specific sensor systems are presented in this report. Firstly, temperature monitoring systems using thermochromic LCD materials. These materials rely upon brag diffraction in the cholesteric liquid crystals to provide a change in textural colour which is dependent on temperature [4].

The second system tested employed a circular photoelastic diaphragm used as a barometer. Photoelastic materials [5], when interrogated using polarised light, show surface stress patterns in the form of chromatic regions. These iso-chromic regions represent areas of equal stress. The chromatic patterns are complicated for a circular diaphragm and are made more complex by imperfections in the boundary of the diaphragm and its mounting. Monitoring of a single point on the sensor diaphragm is unsuitable because the colour pattern cycles over a number of fringe orders which limits the modulation depth of the system.

Neural Network Architecture

Data from the chromatic sensors were used to train a variety of MLPs with different architectures. A MLP with two hidden layers gave good mapping with the data sets. Pre- and post processing of the data was used to aid in the training of the system. Pre-processing included filtering data noise, implementing chromatic processing and data normalisation. For the 2D diaphragm analysis the image was broken up into concentric rings and the average chromaticity of each of the rings presented to the network. Post-processing included scaling the net output and error checking. This processing helps to reduce the load on the network in a simple and meaningful manner and so allows a greater scope for learning, generalisation and feature extraction.

Results

The results of the remote sensing system have been encouraging and have indicated that neural networks provide a practical solution to the associated problems. Figure 1 shows a schematic diagram of the remote sensing system developed.

Graph 1 shows the variation in the red green and blue intensities for the thermochomic strip with temperature. Using this raw data to train the neural network yielded an accuracy of at best 1%, for a 3, 20, 5, 1 (3 input nodes, 2 hidden layers with 20 and 5 nodes and 1 output node) net architecture, after extensive learning. Graph 2 shows the network performance after a 12 hours learning period. A fifth order polynomial curve fit was then used to reduce experimental noise, the data was also normalised. This improved the network accuracy by an order of magnitude (an accuracy of better than 0.1%). Graph 3 shows the error obtained in mapping the neural network output with data from the training set. The system was tested with other sets of data and was observed to have good generalisation capability. The neural network also coped well with light source variations.

Analysis of data from the photoelastic barometer was much more difficult. Pre-processing consisted of calculating the mean chromaticity in concentric rings over the area of the diaphragm surface (figure 2). The Data obtained from this analysis for various pressures is shown in graphs 4 and 5. A total of 10 rings, providing 2 chromatic parameters (x and y), were used as input to a 20, 20, 10, 1 network. A data set consisting of 20 pressure values were obtained and the system trained using 15 sets of this data . The results obtained for the training sets are shown in graph 6. The system was then tested with the remaining 5 data sets to check the generalisation and interpolation capabilities of the network. Graphs 7 and 8 show that, after 48 hours learning, the system gave an accuracy of 0.05%, using the training set, and an accuracy of 1.17% with unseen data. Improvements in the generalisation capability were not immediately visible for networks of greater size.

Conclusions

These results demonstrate that neural network technology can be successfully employed in processing optical information for sensing purposes, even when the data is subject to a wide range of external effects. The network learning performance is greatly dependent on the data set presented to it and, if neural networks are to be used in data processing, care and consideration must be given to preprocessing of the data.

REFERENCES.

[1] Jones, G.R., Kwan, S., Henderson, P., and Lewis, E., *Opt. Laser Technol.*, 19, 1987, 297 - 303.

[2] Kwan, S., Beavan, C.M., and Jones, G.R., *Meas. Sci. Technol.*,1, 1990, 207-215.

[3] Jones,G.R., Russell, P.C., *Pure Appl. Opt.*, 2, 1993, 87-110

[4] Raynes, E.P., *Phil. Trans. R.Soc London a 309*, 1983,167-178

[5] Jessop,H.T., Harris, F.C., *Photoelasticity*, 1970, Dover Publications

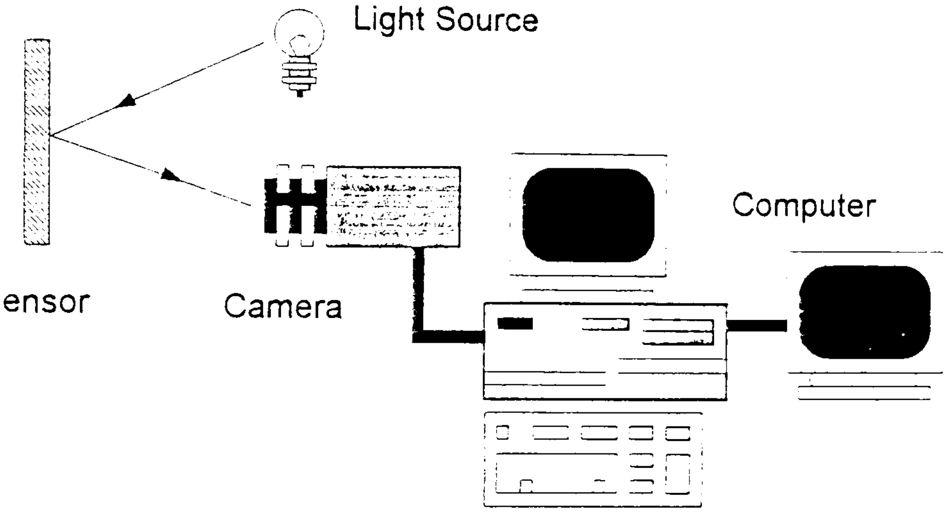


FIGURE 1. Diagram showing Remote Sensing System

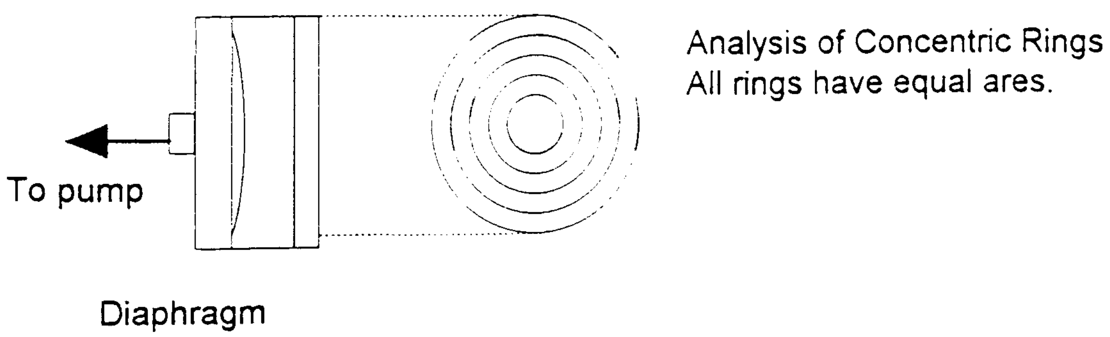
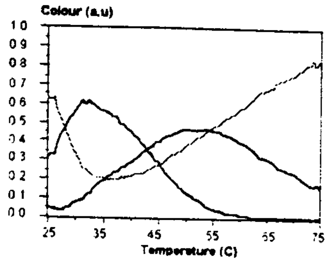
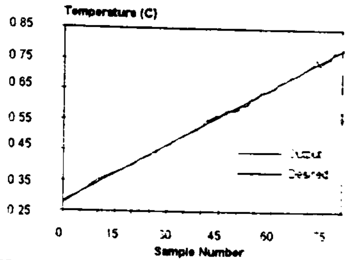


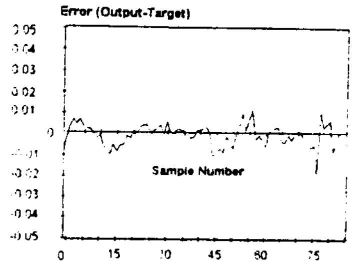
FIGURE 2. Diagram showing Baramoter and 2D analysis approach



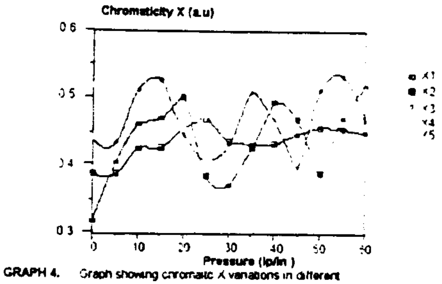
GRAPH 1 Graph showing Red, Green and Blue Variation for Thermochromic Strip with temperature



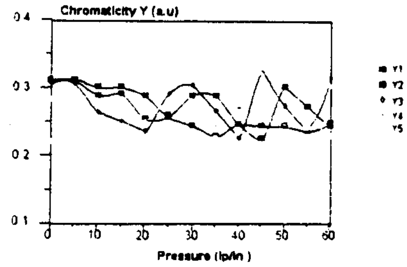
GRAPH 2. Graph showing output of a 3,20 10 1 architecture network trained for 12 hours to learn temperature characteristics



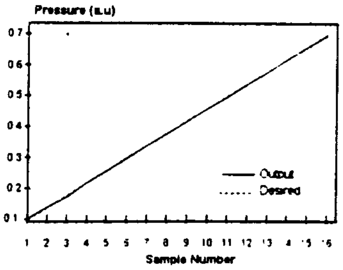
GRAPH 3. Graph showing error between output and target for Network trained for 12 hours to learn temperature charectenstics



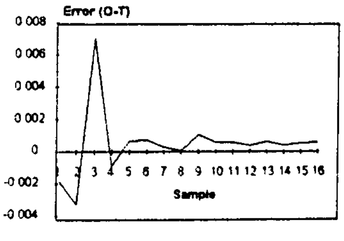
GRAPH 4. Graph showing chromaitc X variations in different parts of the photoelastic diaphragm with pressure



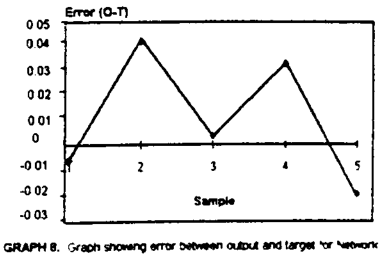
GRAPH 5. Graph showing chromaitc Y variations in different parts of the photoelastic diaphragm with pressure



GRAPH 6. Graph showing output of a 20,20,10,1 architecture network trained for 24 hours to learn barometer charecteristics



GRAPH 7. Graph showing error between output and target for Network trained for 24 hours to learn Barometer charectenstics



GRAPH 8. Graph showing error between output and target for Network with unseen data after training for 24 hours

JOURNAL OF MODERN OPTICS, 1992, VOL. 39, NO. 11, 2301–2314

Chromatic modulation for optical fibre sensing
Electromagnetic and speckle noise analysis

R. SMITH, S. AHMED, A. VOURDAS,
J. SPENCER, P. RUSSELL and G. R. JONES

Department of Electrical Engineering and Electronics,
The University of Liverpool, Brownlow Hill, P.O. Box 147,
Liverpool L69 3BX, UK

(*Received 7 April 1992 and accepted 16 July 1992*)

Abstract. Changes in the chromaticity of the transmitted electromagnetic field in an optical fibre are considered. An electromagnetic analysis that predicts the radial dependence of the chromaticity at the exit is presented and coupled to a speckle noise analysis which provides an estimate of the error. Theoretical predictions are compared with experimental results and good agreement between the two is demonstrated. The results can be useful in the design of optical fibre sensors which use the chromatic modulation method.

1. Introduction

The use of optical fibres for sensing and monitoring physical and chemical parameters has been hitherto limited by a number of factors including a deeper understanding of the fibre transmission process itself. The properties of light which might be used for sensing, amplitude, frequency, polarization, are all susceptible to extraneous influences more so than if the chromaticity (spectral distribution) of the light is used.

The chromaticity of an electromagnetic field has been rigorously defined in the literature [1, 2] as a quantitative measure of colour. The original definition together with some generalizations proposed here are discussed in section 2. The chromatic modulation method has been used for optical sensing in a variety of ways [3, 4] and it has been found to be a sensitive whilst at the same time a cost effective method.

Of fundamental importance for the chromatic technique, is a better understanding of how multimode fibres transmit over a wide range of wavelengths and how, as a result, the chromaticity of the light at the fibre output is affected. Little information exists in the literature about such propagation. The purpose of this contribution is an initial study of polychromatic propagation. For simplicity we consider a polychromatic signal composed of only two wavelengths and we study the two-dimensional distribution of the chromaticity of the light emerging from a fibre, following transmission via a multiplicity of propagation modes. Both experimental observations of the chromaticity of the signal and a rigorous theoretical analysis of multiwavelength propagation in the fibre are presented.

Due to speckle noise these predictions cannot be directly compared with the experimental measurements so a 'local statistical analysis' is introduced which divides the total exit region into many sub-regions and calculates the average value, the variance and the other higher moments of the chromaticity within each sub-region. The average value of the chromaticity together with the error bars are then presented as a function of position; and this is compared with theoretical predictions. The existing analysis for speckle noise [7] is extended into the variables of chromaticity and probability distributions for these variables are derived.

2. Chromaticity of an electromagnetic field

Let $P(A; \lambda)$ be the electromagnetic energy flow as a function of the wavelength λ. The $P(A; \lambda)$ can be expressed in terms of the electric and magnetic field with the use of the Poynting vector. Consider N photodiodes with responsivity curves $R_\mu(\lambda)$ $(\mu=1, \ldots, N)$ and define the quantities

$$
\left.\begin{aligned}
x_\mu(A) &= \frac{\displaystyle\int P(A; \lambda) R_\mu(\lambda)\,\mathrm{d}\lambda}{\displaystyle\sum_{\mu=1}^{N} \int P(A; \lambda) R_\mu(\lambda)\,\mathrm{d}\lambda}, \\
\sum_{\mu=1}^{N} x_\mu(A) &= 1, \qquad 0 \leqslant x_\mu \leqslant 1.
\end{aligned}\right\} \qquad (1)
$$

The 'generalized chromaticity' of the electromagnetic field at the point A is then defined as $(x_1, \ldots, x_N)$. This formalism is a generalization of the CIE formalism [1, 2] which quantifies the concept of colour in human vision. If $N=3$ and the specific photodiode responses correspond to those of the human vision system, (x_1, x_2, x_3) provide a quantitative characterization of the usual concept of colour, in the sense explained in [1, 2]. If however different types of photodiodes or a different number of photodiodes are used, then the $(x_1, \ldots, x_N)$ system becomes a generalized chromaticity which characterizes the distribution of power into various wavelength regions. Of course this has no direct relationship to the usual concept of colour. For example, by using the photodiodes whose response curves are shown in figure 1, if the value of x_1 is close to 1 and the values of x_2, x_3 close to zero, then the power is mainly distributed between $\lambda_1=400\,\text{nm}$, $\lambda_2=500\,\text{nm}$, etc. A spectrum analyser can of course, provide the exact distribution $P(A; \lambda)$ as a function of λ, but for many sensing applications such superfluous information is unnecessary, and costly and cumbersome to acquire. It is only the integral of the power over a region of wavelengths, that is needed. In these cases it is less expensive and easier, to use this method of chromaticity. Moreover the method can be conveniently used with a CCD camera (section 3 below), and it can provide the chromaticity with good sensitivity and spatial resolution.

The chromaticity of a signal can be represented by a point in a $(N-1)$-dimensional space which we call chromaticity space (the Nth coordinate is not an independent variable, equation (1)). The method of chromatic modulation associates changes in the chromaticity with changes in the value of the quantity which we want to measure. As the value of this quantity varies, the chromaticity follows a path in the chromaticity space.

3. Experimental investigation

Two monochromatic sources are used to excite modes within a step index multimode fibre. The Argon ($\lambda=496{\cdot}5\,\text{nm}$) and Helium-Neon ($\lambda=632{\cdot}8\,\text{nm}$) sources are combined into a single coaxial beam before entering the input plane of the fibre. A microscopic objective focused this coaxial beam to a point beyond the plane of the fibre so that the core of the fibre is completely covered by the beam (figure 2).

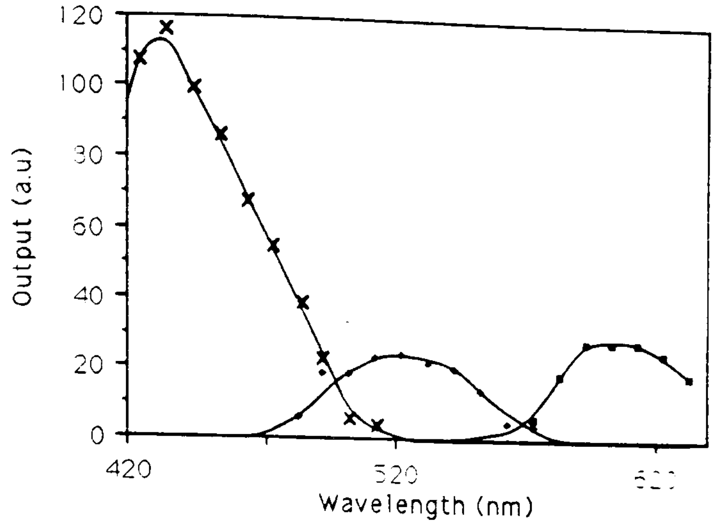


Figure 1. Responsivity curves of the CCD camera. R_1 ■ Red, R_2 ◆ Green, R_3 × Blue.

The optical fibre is a step index multimode fibre, with a core diameter of 50 μm and a cladding diameter of 125 μm. The relative refractive indices of the core and cladding are 1·472 and 1·462 respectively. The length of fibre used in the test is 500 mm; this was firmly held between two supports. The fibre did not sag nor was it excessively strained but was firmly clamped.

To view the exit of the fibre a colour CCD camera, without a lens system was placed between 2 and 4 mm from the exit plane. The CCD array consists of 512 by 512 pixels, each pixel being formed by three elements with optical responses covering different areas of the visible spectrum. The responsivities of these colour sensitive elements are shown in figure 1. The CCD array was connected to a frame-grabber card in a DOS workstation, allowing the image of the fibre to be grabbed and stored. The signal from each of the colour sensitive elements was converted to an 8-bit binary number and stored within the frame grabber card for future storage and processing. Usually the image of the fibre occupied 300 by 300 pixels.

Light intensities may be varied by interposing neutral density filters in the path of each light source. Correct illumination of the CCD array is assured so that saturation is avoided but maintaining maximum sensitivity.

The chromaticity at each pixel is given by

$$x_i = \frac{I_i}{I_1 + I_2 + I_3}, \qquad i = 1, 2, 3, \qquad (2)$$

where I_i is the output of the i detector at this pixel.

Where the light intensity had fallen below the sensitivity of the camera system, therefore appeared black, these values where not considered.

Figure 3 shows the spatial distribution of the chromaticity variables x_1 and x_2 at the output of the fibre as determined from the measurements.

The results show not only the random nature of the speckle pattern but also a radial dependence of the signal.

4. Electromagnetic analysis

A step index optical fibre is considered with core radius a, core refractive index n_1 and cladding refractive index n_2. The propagation of electromagnetic waves via a

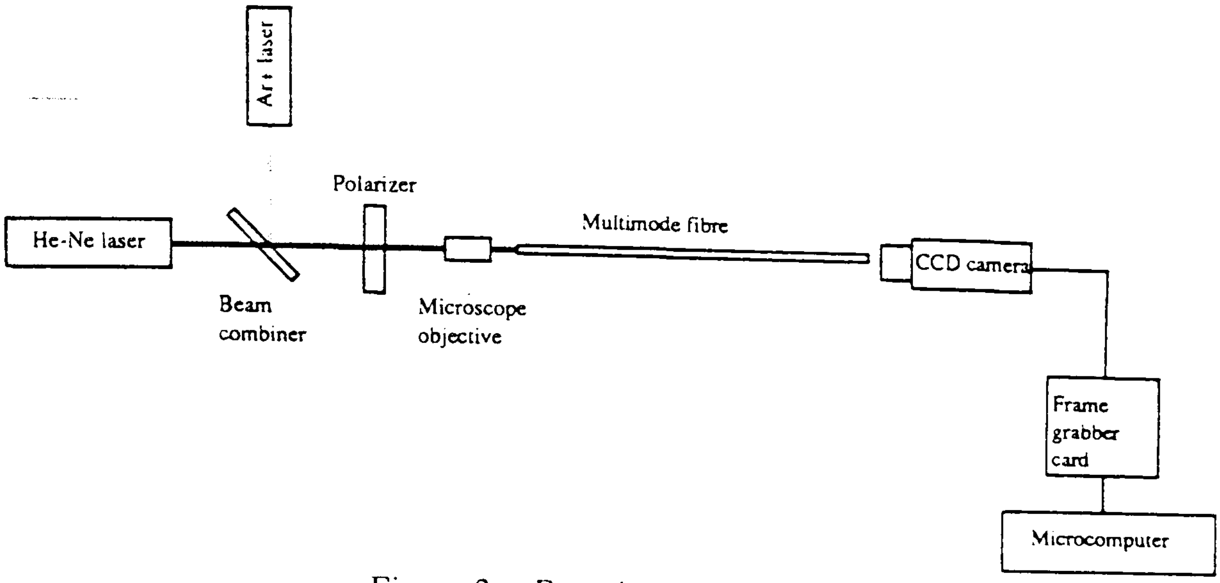
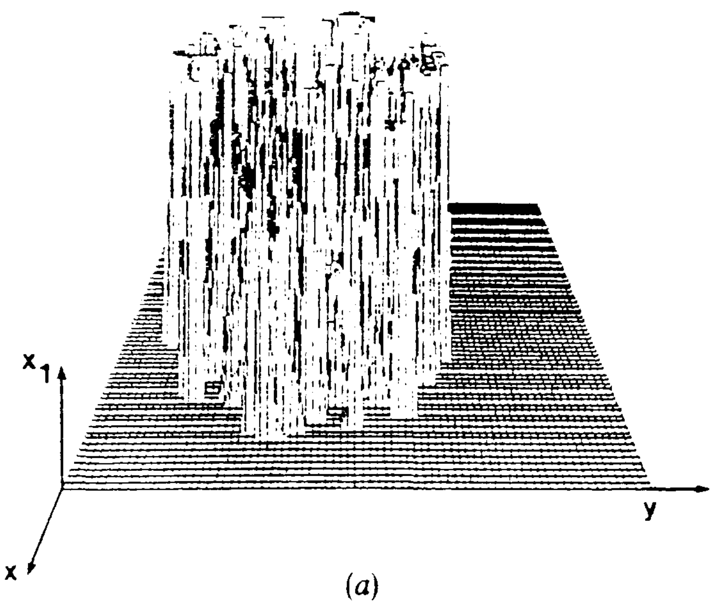
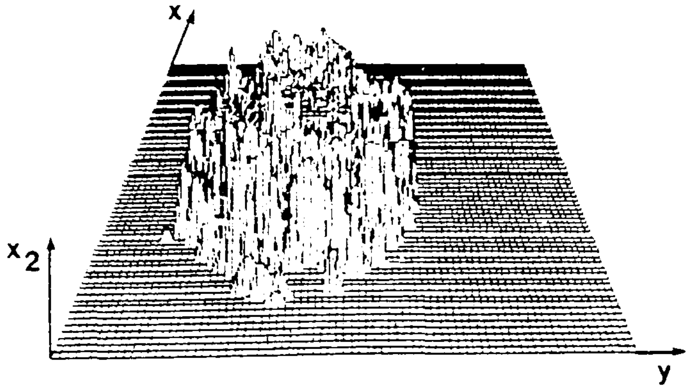


Figure 2. Experimental apparatus.



(*a*)



(*b*)

Figure 3. The chromaticity variables (*a*) x_1, and (*b*) x_2 as measured by the CCD camera at the various pixels in the plane $x-y$.

number of propagation modes may then be described in terms of the following parameters [5, 6]

$$k=\omega(\varepsilon_0\mu_0)^{1/2}, \tag{3}$$

$$u^2=a^2(k^2n_1^2-\beta^2), \tag{4}$$

$$w^2=a^2(\beta^2-k^2n_2^2), \tag{5}$$

$$u^2+w^2=a^2k^2(n_1^2-n_2^2), \tag{6}$$

where k is the propagation constant in free space and β is the propagation constant in the z-direction corresponding to the fibre axis. The electromagnetic modes are characterized by three propagation constants (ν, u, β). ν is the azimuthal propagation constant; and due to the periodicity it assumes only integer values ($\nu=0, 1, 2, \ldots$). u is the radial propagation constant in the core and due to the boundary conditions between core and cladding it takes discrete values which are labelled u_m with $m=1, 2, \ldots$. w is the radial propagation constant in the cladding and is dependent on u in the sense that it obeys equation (6). w is also labelled with the index m as w_m. Let $A_{m\nu}(\lambda)$ be the amplitude of the (m, ν) mode.

Varying both ν and m yields all the propagation modes corresponding to a certain value of k. Since the optical fibres considered are made from linear optical materials (with refractive indices which are constant, i.e. independent of the transmitted optical power) any linear combination of the above modes can in principle propagate in the fibre. It is the nature of the light source and the way that this source is coupled to the fibre, that determines which combination of modes will propagate. In experimental systems similar to the one being investigated, it is usually assumed that all modes carry the same amount of power (i.e. $A_{m\nu}(\lambda)$ is independent of m, ν). This problem has been discussed extensively in the literature [8–11] but the more general problem of how to launch a specific combination of modes into the fibre and whether this can be advantageous for certain applications remains to be resolved. Experimentally the laser beam profile was adjusted such that the intensity across the beam launched into the fibre was constant and the beam was converging. This was achieved using a microscope objective. This beam convergence should excite a large number of modes in the fibre. However the input conditions are critical in determining which mode groups are excited.

Using the Poynting vector, the power $P(r, \theta)$ has been calculated and by integration over θ, the power as a function of r has been determined. In order to get a feeling about the power distribution, the case of one laser (Argon, $\lambda=496{\cdot}5$ nm) has been considered and the power as a function of the radius has been plotted in figure 4. Two different assumptions about the power distribution into different modes, have been made:

(i) That the power is equally distributed into all modes (i.e. $A_{m\nu}$ is independent of m, ν).

(ii) That

$$A_{m\nu}=\text{const}\left(1-\frac{\nu}{\nu_{\max}}\right), \tag{7}$$

where $\nu_{\max}$ is the maximum value of ν which is determined by the radius of the fibre. In this case it is clear that the lower angular modes are preferentially excited.

It is seen clearly that these two different assumptions lead into different power distributions. Subsequently equation (1) has been used to calculate the chromaticity (x_1, x_2) as a function of r for the case of the two lasers which have been used in our

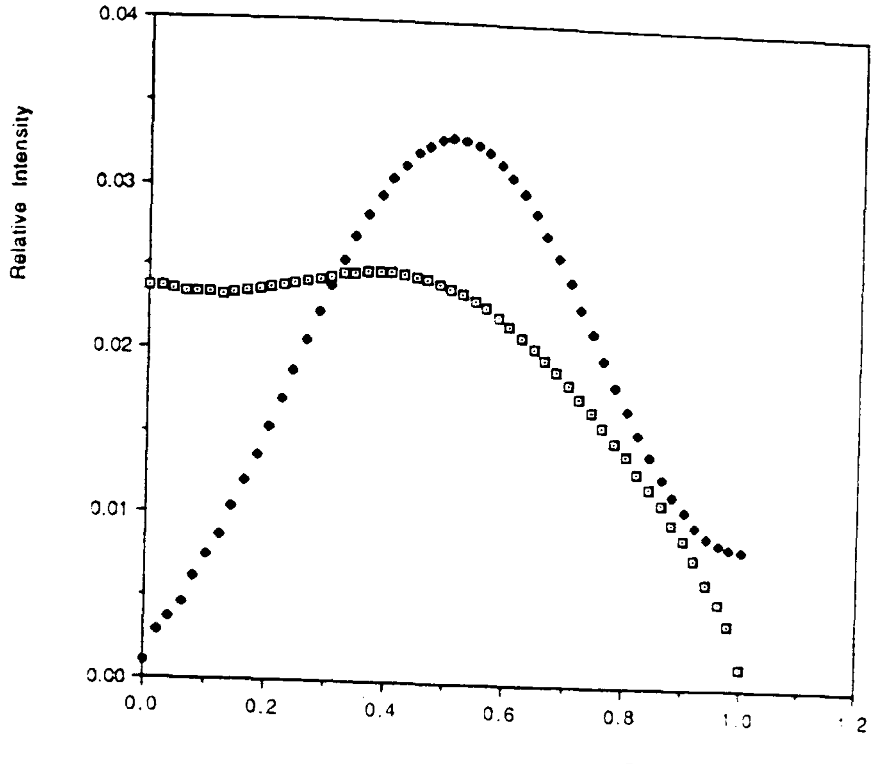


Figure 4. Theoretical calculation of the transmitted electromagnetic power as a function of the radius, for monochromatic light ($\lambda = 496{\cdot}5$ nm) propagating in the fibre described in the text. ▢ Low modes preferentially excited; ◆ all modes equally excited.

experiment and for the responsivity curves given in figure 1. In order to demonstrate the dependence of our theoretical predictions on the distribution of the power into various modes we have repeated the calculations under four different assumptions:

(1) It is assumed that for both lasers the lower angular modes are preferentially excited (as described by equation (7)).

(2) It is assumed that for the Arg laser ($\lambda = 496{\cdot}5$ nm) all modes are equally excited; and for the He–Ne laser ($\lambda = 632{\cdot}7$ nm) the lower angular modes are preferentially excited (as described by equation (7)).

(3) It is assumed that for both lasers all modes are equally excited.

(4) It is assumed that for the He–Ne laser ($\lambda = 632{\cdot}8$ nm) all modes are equally excited; and for the Ar laser ($\lambda = 496{\cdot}5$ nm) the lower modes are preferentially excited (as described by equation (7)).

The results are given in figures 5 and 6. It is seen that assumptions 1, 3 lead to very similar results, while assumptions 2, 4 lead to results which are significantly different. As we explained above, for the experiment considered here, assumption 3 is a reasonable one. It is clear from the theoretical analysis presented here that assumption 1 also leads to very similar theoretical predictions.

5. Statistical analysis of experimental results

The predictions of electromagnetic theory for x_1 and x_2 as a function of the position r cannot be directly compared with the experimental measurements of figure 3 because of the existence of speckle noise. In order to make such a comparison, a 'local statistical analysis' of the experimental results needs to be performed to give the mean values and the other higher moments of x_1, x_2 as a function of the position r. The speckle pattern region is divided into many sub-regions and within each sub-region the mean values and the other higher moments of

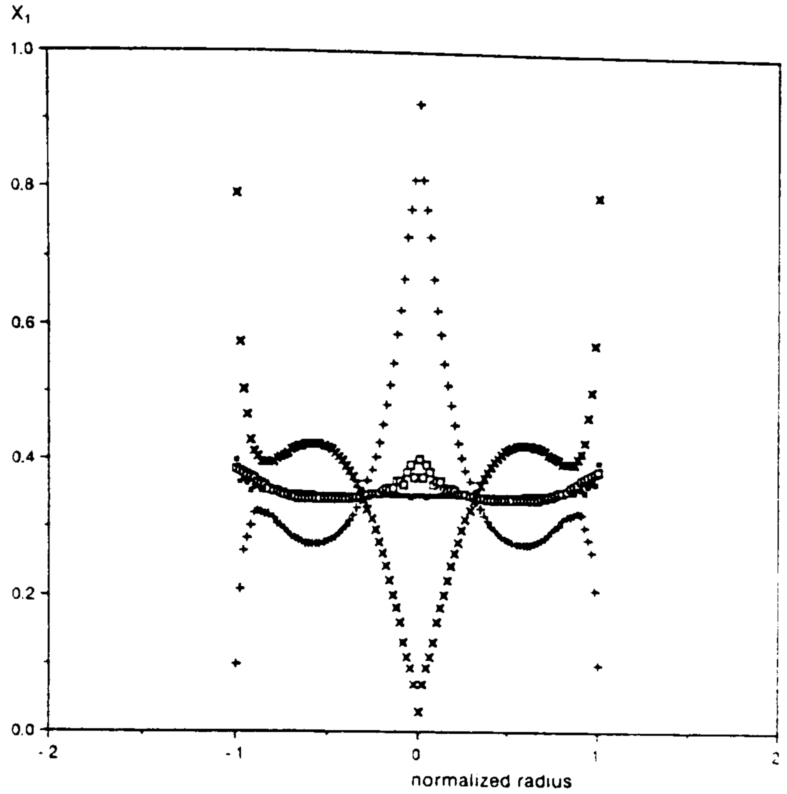


Figure 5. Theoretical calculation of the chromaticity variable x_1 as a function of the (normalized) radius under four different assumptions about the power distribution into the various modes, explained in text. ■ X1 theoretical (using assumption 1); + X1 theoretical (using assumption 2); □ X1 theoretical (using assumption 3); × X1 theoretical (using assumption 4).

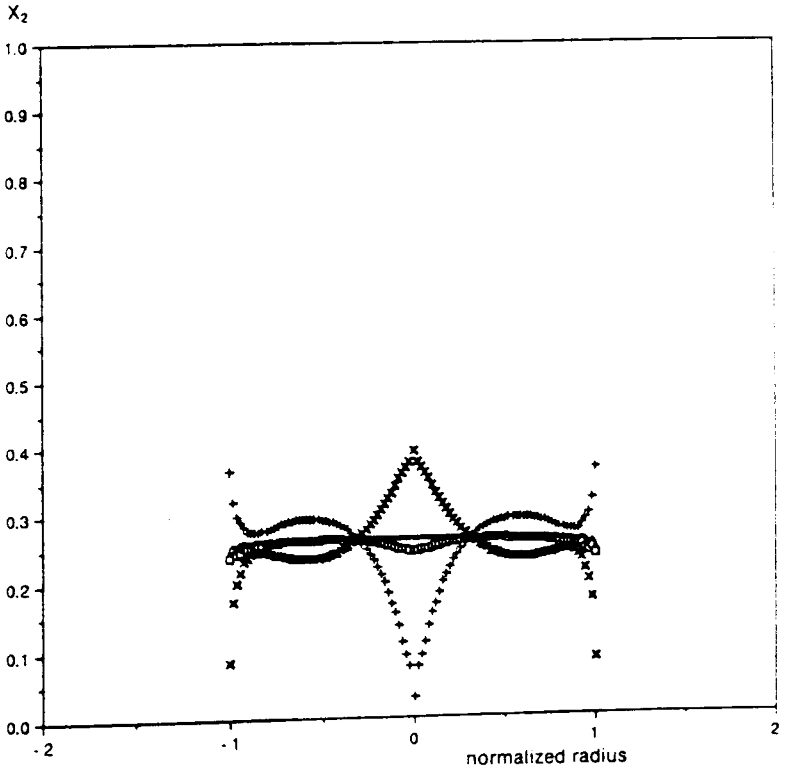


Figure 6. Theoretical calculation of the chromaticity variable x_2 as a function of the (normalized) radius under four different assumptions about the power distribution into the various modes, explained in the text. ● X2 theoretical (using assumption 1); + X2 theoretical (using assumption 2); □ theoretical (using assumption 3); × X theoretical (using assumption 4).

x_1, x_2 are calculated. Each sub-region should be large enough to contain several measurements for good statistical analysis and at the same time it should be small enough to give the variation of the statistical quantities as a function of the position (r, θ). Since we are interested in the radial dependence only, the speckle pattern region is divided into five radial sub-regions (figure 7), each of which has approximately the same area and contains approximately the same number of pixels ($N = 20\,000$ pixels). There is no reason to consider more radial subregions because the theoretically predicted curve is almost flat. The large number of pixels in each subregion ensures that the statistical results are very good. If $x_i(M)$ $(i = 1, 2)$ is the chromaticity at the M^{th} pixel $(M = 1, \ldots, N)$, the quantities which have been evaluated are

$$\begin{aligned} \langle x_i \rangle &= N^{-1} \sum_{M=1}^{N} x_i(M), \\ \sigma_\mu(x_i) &= N^{-1} \sum_{M=1}^{N} [x_i(M) - \langle x_i \rangle]^\mu \qquad (\mu = 2, 3, 4). \end{aligned} \tag{8}$$

The results are presented in the table. They are the experimentally measured moments of the probability distributions $p(x_1)$, $p(x_2)$ for the chromaticity in each subregion.

In figures 8 and 9 the mean values are plotted in the middle of the appropriate regions. The standard deviation $(\sigma_2^{1/2})$ is the half-width of the plotted error bars. The theoretical predictions of the previous section under the assumptions 3 and 1, are also shown. Across the image the theoretical predictions fall within the error bars of the experimental results. The fit between theory and experiment is good within the centre of the image, while towards the periphery, theoretical and experimental results show slightly larger discrepancies, although the trends in both are the same. The error bars of the experimental results are due to the speckle noise in contrast with the theoretical results which are based on purely electromagnetic calculations. A speckle noise analysis is performed in the next section.

6. Speckle noise analysis

The theoretical work for speckle noise analysis is reviewed in [7] and here it is extended into the context of chromaticity. As usual, the electromagnetic field is for simplicity described by a scalar field with amplitude

$$A_i = I_i^{1/2} \exp(i\theta_i), \tag{9}$$

where the index i indicates the various wavelengths (in the present case $i = 1, 2$), I_i is the intensity, and θ_i is the phase. Let $P(A_1, A_2)$ be the probability distribution for the amplitudes A_1, A_2 in a certain sub-region. Using statistical arguments related to the central limit theorem, it is known from the literature [7] that:

$$P(A_1, A_2) = [4\pi^2(\tau_1\tau_2 - |\tau|^2)]^{-1} \exp[-\tau_1|A_1|^2 - \tau_2|A_2|^2 + \tau A_1 A_2^* + \tau^* A_1^* A_2], \tag{10}$$

$$\tau_1 = [\langle I_1 \rangle (1 - |\mu|^2)]^{-1}, \tag{11}$$

$$\tau_2 = [\langle I_2 \rangle (1 - |\mu|^2)]^{-1}, \tag{12}$$

$$\tau = \mu(1 - |\mu|^2)^{-1} [\langle I_1 \rangle \langle I_2 \rangle]^{-1/2}. \tag{13}$$

The physical meaning of the parameters $\langle I_1 \rangle$, $\langle I_2 \rangle$, μ is seen from the fact that:

$$\langle I_1 \rangle = \int |A_1|^2 P(A_1, A_2)\, \mathrm{d}^2 A_1\, \mathrm{d}^2 A_2, \tag{14}$$

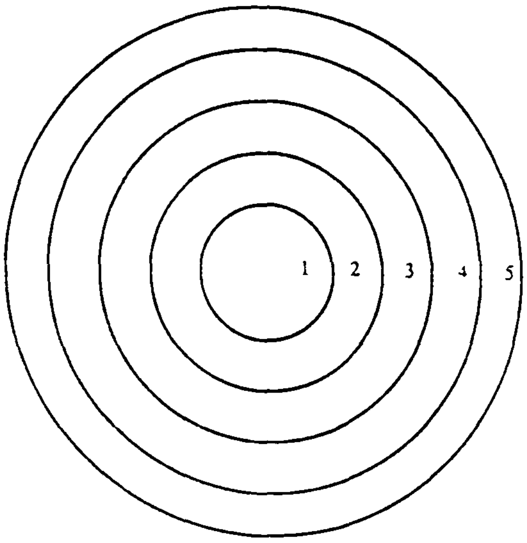


Figure 7. Division of the speckle pattern into various regions.

Experimental results for the first four moments of the chromaticity probability distributions $p(x_1)$, $p(x_2)$ in the various regions. The results in parenthesis are theoretical predictions (figure 10) for $|\mu| = 0{\cdot}88$

Region	1	2	3	4	5
$\langle x_1 \rangle$	0·3478 (0·3551)	0·3768	0·3675	0·3341	0·2540
$\sigma_2(x_1)$	0·0192 (0·0391)	0·0584	0·1038	0·1351	0·1339
$\sigma_3(x_1)$	0·0057 (0·0078)	0·0382	0·0966	0·1491	0·1456
$\sigma_4(x_1)$	0·0036 (0·0056)	0·0242	0·0706	0·1235	0·1297
$\langle x_2 \rangle$	0·2987 (0·2339)	0·2344	0·1492	0·0900	0·0529
$\sigma_2(x_2)$	0·0074 (0·0068)	0·0161	0·0193	0·0141	0·0095
$\sigma_3(x_2)$	−0·0007 (0·00002)	−0·0008	0·0013	0·0017	0·0015
$\sigma_4(x_2)$	0·0002 (0·0001)	0·0005	0·0007	0·0005	0·0004

$$\langle I_2 \rangle = \int |A_2|^2 P(A_1, A_2)\, \mathrm{d}^2 A_1\, \mathrm{d}^2 A_2, \tag{15}$$

$$\mu = (\langle I_1 \rangle \langle I_2 \rangle)^{-1/2} \int A_1 A_2^* P(A_1, A_2)\, \mathrm{d}^2 A_1\, \mathrm{d}^2 A_2. \tag{16}$$

It is clear that $\langle I_1 \rangle$, $\langle I_2 \rangle$ are the mean intensities of the electromagnetic field in the two wavelengths and μ is the correlation between the two fields which may be expressed as:

$$\mu = |\mu| \exp(\mathrm{i}\psi). \tag{17}$$

The value of $|\mu|$ depends upon the roughness at the exit of the fibre amongst other things. It has to be derived experimentally in the manner described later.

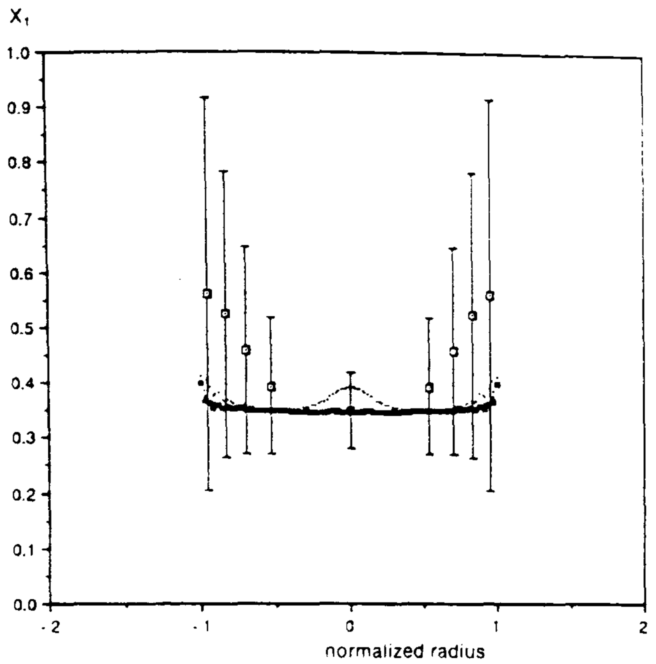


Figure 8. Experimental results (points with error bars) and theoretical predictions (curves) of the chromaticity variable x_1 as a function of the (normalized) radius. ⊡ X1 experimental; · X1 theoretical (using assumption 3); ■ X1 theoretical (using assumption 1).

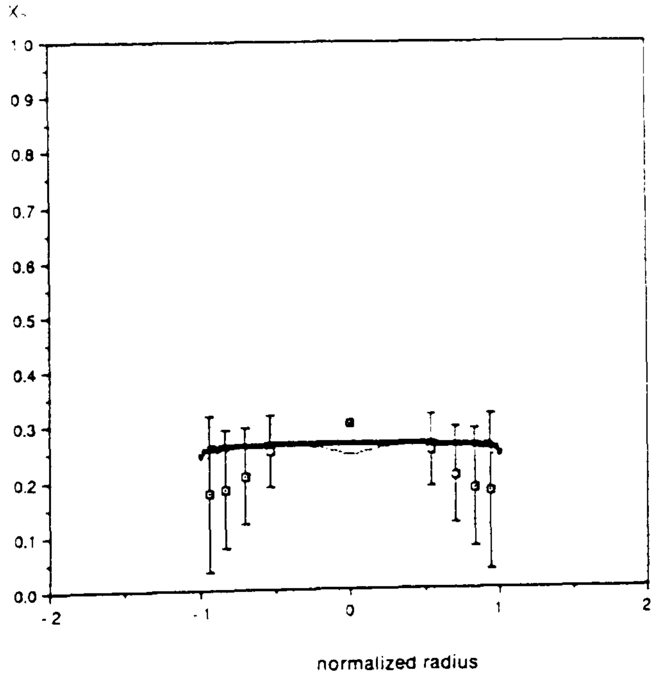


Figure 9. Experimental results (points with error bars) and theoretical predictions (curves) of the chromaticity variable x_2 as a function of the (normalized) radius). ⊡ X2 experimental; · X2 theoretical (using assumption 3); ■ X2 theoretical (using assumption 1).

Substituting equations (9), (17) into equation (10) and re-expressing the probability distribution as

$$P(I_1,\theta_1;I_2,\theta_2)=[4\pi^2(\tau_1\tau_2-|\tau|^2)]^{-1} \times\exp[-\tau_1 I_1-\tau_2 I_2+2|\tau|(I_1 I_2)^{1/2}\cos(\theta_1-\theta_2+\psi)], \tag{18}$$

and integration over the angular variables gives

$$P(I_1,I_2)=[\tau_1\tau_2-|\tau|^2]^{-1}\exp[-\tau_1 I_1-\tau_2 I_2]I_0[2|\tau|(I_1 I_2)^{1/2}], \tag{19}$$

where I_0 is the zero order modified Bessel function of the first kind given by [12]

$$I_0(z)=\sum_{k=0}^{\infty}(\tfrac{1}{4}z^2)^k[k!]^{-2}. \tag{20}$$

— In order to calculate the probability distribution for the chromaticity variables x_1, x_2, equation (1) is used with

$$P(\lambda)=I_1\delta(\lambda-\lambda_1)+I_2\delta(\lambda-\lambda_2), \tag{21}$$

to give the result

$$x_1=\frac{Au+B}{Cu+D}, \tag{22}$$

$$x_2=\frac{Eu+F}{Cu+D}, \tag{23}$$

where

$$u=\frac{I_1}{I_2}, \tag{24}$$

$$A=R_1(\lambda_1), \tag{25}$$

$$B=R_1(\lambda_2), \tag{26}$$

$$C=\sum_{\mu=1}^{3}R_k(\lambda_1), \tag{27}$$

$$D=\sum_{\mu=1}^{3}R_\mu(\lambda_2), \tag{28}$$

$$E=R_2(\lambda_1), \tag{29}$$

$$F=R_2(\lambda_2). \tag{30}$$

Knowing the probability distribution $P(I_1, I_2)$ the probability distribution for the variable u may be calculated, using the well known formula [13]

$$P(u)=\int_0^{\infty}I_2P(uI_2, I_2)\,\mathrm{d}I_2. \tag{31}$$

From equations (19), (31), (11), (12), (13) after integration

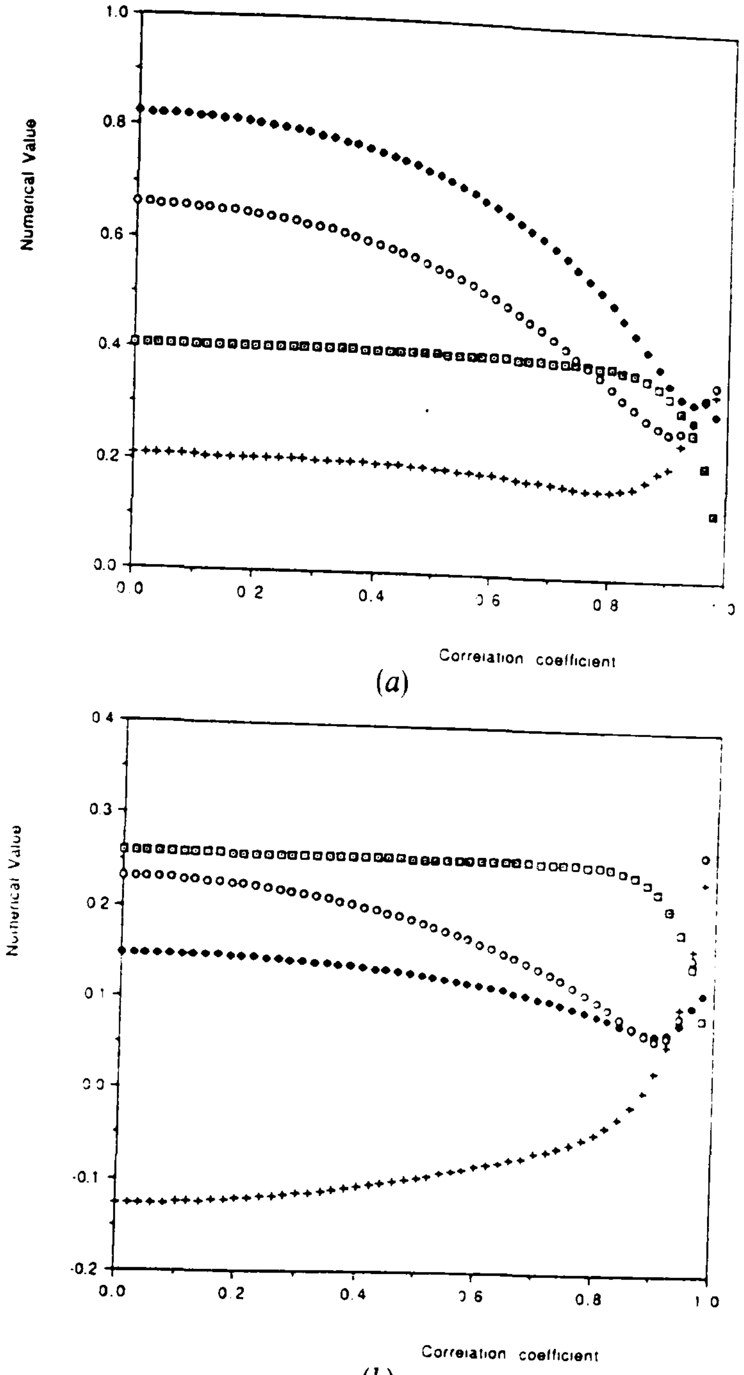
$$P(u)=(1-|\mu|^2)\sum_{k=0}^{\infty}\frac{[(2k+1)!](u|\mu|^2)^kr^{k+1}}{(k!)^2(u+r)^{2k+2}}, \tag{32}$$

$$r=\frac{\langle I_1\rangle}{\langle I_2\rangle}. \tag{33}$$

The chromaticity variables are simple functions of u (equations (22), (23)) and the probability distribution $P(x_1)$, $P(x_2)$ can easily be found from $P(u)$ by dividing over the Jacobian. The result is

$$P(x_1)=(1-|\mu|^2)|AD-CB|\sum_{k=0}^{\infty}\frac{[(2k+1)!][(Dx-B)(A-Cx)|\mu|^2]^kr^{k+1}}{(k!)^2[(Dx-B)+r(A-Cx)]^{2k+2}}, \tag{34}$$

$$P(x_2)=(1-|\mu|^2)|ED-CF|\sum_{k=0}^{\infty}\frac{[(2k+1)!][(Dx-F)(E-Cx)|\mu|^2]^kr^{k+1}}{(k!)^2[(Dx-F)+r(E-Cx)]^{2k+2}}. \tag{35}$$



(*a*)

(*b*)

Figure 10. Theoretical predictions for the various moments of the probability distribution (*a*) $p(x_1)$ of the chromaticity x_1 and (*b*) $p(x_2)$ of the chromaticity x_2, in the central region, as a function of the correlation coefficient $|\mu|$. In fig. 10(*a*) □ mean X1; ◆ variance × 10; + 3rd moment × 25; ○ 4th moment × 50. In fig. 10(*b*) □ mean X2; ◆ variance × 10; + 3rd moment × 100; ○ 4th moment × 500.

The first few moments of these distributions were evaluated numerically as a function of the correlation parameter $|\mu|$. The calculation has been performed for the central 'sub-region 1', where $r = 1{\cdot}045$ and it could be trivially extended into the other sub-regions. In the numerical evaluation the infinite sum has been truncated at $k = 10$. As a test of the accuracy of the truncation approximation the integrals of the probability distributions $P(x_1)$, $P(x_2)$ have been calculated from zero to one. Without the truncation they would be equal to one; with the truncation they were found to be greater than 0·99. The theoretical results of these calculations are presented in figure 10. A comparison of these results with the corresponding experimental values, presented in the table, shows that the value of $|\mu|$ is 0·8–0·9. In table 1 the theoretical predictions for $|\mu| = 0{\cdot}88$ are given in parentheses, so that they can be easily compared with the experimental results.

The speckle noise model presented in this section is able to predict the probability distributions $P(x_1)$, $P(x_2)$ (or equivalently their moments given by equation (8) in terms of only one input parameter $|\mu|$. As explained above, the value of $|\mu|$ describes the correlation between the two lasers at the output and depends on the roughness of the surface at the output of the fibre and also on other imperfections of the fibre.

6. Discussion

The theoretical model presented is based upon electromagnetic analysis of two monochromatic laser signals and subsequent statistical analysis of the speckle noise patterns. The agreement between experimental and theoretical results is good, proving the validity of the model used.

The electromagnetic analysis has been supplemented by speckle noise analysis which predicts the probability distribution for the chromaticity in terms of an input parameter $|\mu|$. This parameter describes the correlation between the two lasers and is a result of the surface roughness and other imperfections in the fibre. However, major defects (e.g. cracks) may not be entirely described by a single parameter. The small differences between experimental and theoretical results may be attributed to these defects as well as descrepancies between near and far field measurements. The former would cause the largest discrepancy between the two sets of results. Although care was taken over polishing the ends of the fibre, there are nonetheless surface cracks which may have propagated into the fibre and which would alter the modal input distribution. There are other factors which effect the speckle pattern besides surface quality, these include small displacements of the fibre caused by vibration, stability of the laser source and fibre imperfections but the statistical analysis undertaken would reduce the effect of the first two and to some extent the last one. However fibre imperfections induced during manufacture or by subsequent handling would change the detailed structure of the speckle pattern to a larger extent. The difference between the near and the far field patterns is thought to be small. Experimentally the camera is 2 mm from the end of the fibre and the effect of the intervening medium, air, is assumed to be small.

Theoretically, modelling of cracks or other major defects is difficult and would vary enormously from one fibre to the next. The best that may be hoped for is experimentally limiting these imperfections. Further modelling can be undertaken to take account of the intervening air and so establish a transfer function relating near and far fields for this experimental arrangement.

Chromatic modulation techniques have over the last six years developed into a powerful technology for use in optical sensing applications [3, 4]. To further develop this technology a more fundamental understanding of the propagation of polychromatic light in a fibre system is required.

The theoretical model presented begins to address this need. Although using two monochromatic wavelengths from two sources, agreement between theoretical and experimental values is good. This establishes the model to be sound and that it is possible to model and input the right experimental conditions.

Such a model can be further developed and used in the design of optical sensing systems. A knowledge of the effects of changes in the modal distribution can be utilized in the design of practical displacement sensors, using microbending, and also to predict the effects of changes to the propagation within a fibre resulting from fibre creep or other induced imperfections. In this way our fundamental analysis can be used for practical optical sensing design.

Acknowledgments

The authors would like to express their appreciation to BNF plc and SERC for funding of this work, and Mr D. Garlic and Mrs. A. Fitzgerald of BNF plc for their advice and help.

References

[1] Williamson, S. J., and Cummins, H. Z., 1983, *Light and Colour in Nature and Art* (New York: Wiley).

[2] Overheim, R. D., and Wagner, L. D., 1982, *Light and Colour* (New York: Wiley).

[3] Jones, G. R., Kwan, S., Henderson, P., and Lewis, E., 1987, *Opt. Laser Technol.*, **19**, 297.

[4] Kwan, S., Beavan, C. M., and Jones, G. R., 1990, *Meas. Sci. Technol.*, **1**, 207.

[5] Kapany, N. S., and Burke, J. J., 1972, *Optical Waveguides* (London: Academic Press); Gloge, D., 1971, *Appl. Optics*, **10**, 2252.

[6] Adams, M. J., 1981, *An Introduction to Optical Waveguides* (New York: Wiley); Marcuse, D., 1974, *Theory of Dielectric Optical Waveguides* (London: Academic Press).

[7] Dainty, J. C., (ed.), 1984, *Laser Speckle and Related Phenomena* (Berlin: Springer); 1976, *Progress in Optics*, **14**, 1; Francon, M., 1979, *Laser Speckle and Applications to Optics* (New York: Academic Press); Special issue on Speckle noise, 1976, *J. opt. Soc. Am.*, **66**.

[8] Pask, C., 1978, *J. opt. Soc. Am.*, **68**, 572.

[9] Piazzola, S., and de Marchis, G., 1979, *Electron. Lett.*, **15**, 721.

[10] Daido, Y., Miyauchi, E., Iwama, T., and Otsuka, T., 1979, *Appl. Optics*, **18**, 2207.

[11] Hjelme, D. R., and Mickelson, A. R., 1983, *Appl. Optics*, **22**, 3874.

[12] Abramowitz, M., and Stegun, I., 1968, *Handbook of Mathematical Functions* (New York: Dover).

[13] Papoulis, A., 1965, *Probability, Random Variables and Stochastic Processes* (New York: McGraw-Hill).