THE UNIVERSITY OF LIVERPOOL

Calibration of
Incoherent Optical Fibre Bundles
for Image Transmission

Thesis submitted in accordance with

the requirements of the University of Liverpool for

the degree of Doctor of Philosophy by

Gregory Francis Dujon

Department of Electrical Engineering and Electronics

January 1991

To my parents

With love

ACKNOWLEDGEMENTS

I would like to thank Professor J. Lucas and Dr. A.B. Parker for their help and support during the course of my research. I would also like to thank Professor W. Eccleston, Head of Department, and the members of staff of the department of Electrical Engineering an Electronics for allowing the use of the facilities at the University. My gratitude is also expressed to the Admiralty, Ministry of Defence, Portsmouth, and in particular Dr. J. Livsey and Mr. A. Thomas, for funding the research grant.

In addition, I would like to thank my colleagues and friends, Simon and Richard, and Sarajane, for their encouragement, help and support during the course of my research and in the compilation of this thesis.

G.F. Dujon, January 1991

ABSTRACT

This thesis examines the use of optical fibre bundles for the transmission of images by non-modulated light, that is, as a pure and unaltered optical representation. Optical fibre bundles are classified into two groups: coherent and incoherent bundles. Coherent bundles maintain the spatial relationship between all the fibres in the bundle, between the two ends of the bundle. An image transmitted by such a bundle will therefore not be corrupted by positional displacement of the intensities that make up the image. An incoherent bundle does not necessarily maintain the spatial relationship of the fibres, and the relative position of the fibres in the bundle will, in general, tend to be totally random at the two ends. An image transmitted by such a bundle will therefore be unrecognisable at the output end of the incoherent optical fibre bundle. The intensities that make up the image are still present at the output end, but randomly located over the output face. This thesis seeks to develop a calibration method that will relocate the output intensities in the output of an incoherent optical fibre bundle, in order to reconstruct a representation of the original input image.

Two calibration methods are developed and implemented, one suitable for low resolution bundles, and the other for high resolution bundles. An electronic hardware system is designed and constructed to perform the reconstruction of the images at video frame speeds to allow moving images to be viewed. The use of incoherent optical fibre bundles for image transmission will take advantage of the lower cost of incoherent bundles, as well as the longer lengths of incoherent bundles that are available. It is envisaged that the calibrated incoherent bundles will find use in remote visual inspection where the presence of electrical signals are undesirable, and in areas where the cost of using coherent bundles would prove prohibitive. Results from both calibration methods are presented to allow judgement of the capabilities of the calibration methods, the quality of images that can be obtained from currently available incoherent bundles, and the necessary requirements for high quality images.

CONTENTS

page

page

GLOSSARY OF TERMS

ADC	- Analogue to Digital Converter
ALT	- Alternative Occam processes
C22V10	- A programmable Logic Device Manufactured by Cypress Semiconductor
CCD	- Charge Coupled Device
CRT	- Cathode Ray Tube
CUPL	- Universal Compiler for Programmable Logic
DAC	- Digital to Analogue Converter
DIL	- Dual In Line
EP600	- A programmable Logic Device manufactured by Altera Corporation
EPROM	- Erasable Programmable Read Only Memory
JEDEC	- Joint Electronic Devices Engineering Council, Standard 3-A, 1986
K	- 1024 Units
Kb	- Kilobytes, 1,024 bytes
LED	- Light Emitting Diode
LUT	- Look Up Table
Mb	- Megabytes, 1,048,576 bytes
MIPS	- Million Instructions Per Second
MS-DOS	- Microsoft Disk Operating System
NTSC	- National Television Systems Committee
OFCS	- Optical Fibre Calibration System
[1]PAL	- Phase Alternate Line
[2]PAL	- Programmable Array Logic
PAR	- Parallel Occam process
PGA	- Pin Grid Array
SEQ	- Sequential Occam process
TDS	- Transputer Development System
VLSI	- Very Large Scale Integration

1. INTRODUCTION

Visual perception is one of the quickest methods of assimilating detailed information about our environment. The great digital processing power currently available from microprocessors and computer systems has led to, and allowed, the development of electronic systems that use visual information to interact with the outside world. The latter half of the twentieth century has seen digital communication come of age, and with the development of the optical fibre light waveguide, the use of light for point to point digital communication is now widespread. Optical digital communication involves the transmission and reception of light pulses of frequency, or range of frequencies, just below that of visible light. The majority of research carried out in this field to date, has concentrated on various aspects of digital optical communication.

This thesis examines the use of bundles of optical fibres for the transmission of visible light. The light will be either reflected from, or generated by, an object or objects in a scene, to form an image. The optical fibre bundles have certain characteristics that affect the transmission of the image, and it is these properties that are detailed, with a view to correcting any adverse effects.

1.1 Optical Fibres And Fibre Bundles

An optical fibre is a waveguide for the ducting of light between two points. It is made up of a core of glass or plastic, with a coaxial cladding. The structure can vary in size, but the diameter is usually of the order of a few tens of microns. The length is dependent on the proposed use, and varies from a few centimetres to tens of kilometres. A more detailed description of optical fibres is given in Chapter 3. Optical fibre bundles are groups of optical fibres, each fibre retaining its individual transmission characteristics. An optical fibre bundle can be characterised by its physical and optical properties. The physical properties include the size, that is, diameter and length, minimum bend radius, density of fibres in the bundle, and the packing arrangement of the fibres. The optical properties include spectral transmission and absorption characteristics and the resolution of the bundle, which affect the brightness of the light transmitted. These features are discussed in detail in Chapter 3. For the purposes of visual image transmission, an optical fibre bundle can be classified as being coherent or incoherent, as explained below.

1.1.1 Coherent Bundles

A coherent optical fibre bundle is one in which the spatial arrangement of the fibres in the bundle is maintained between the end surfaces. If the two end surfaces are imagined to be on Cartesian planes, and each fibre a point on the coordinate system, then any point on one end plane maps to exactly the same point on the other plane. An image incident on one end surface will therefore be transmitted through the fibre without corruption, that is, the intensities of light and colours that make up the image will remain in the same relative position. The output image will therefore be seen as a replica of the input image. Coherent fibre bundles are therefore well suited for the transmission of visual images.

1.1.2 Incoherent Bundles

An incoherent optical fibre bundle is one in which the spatial arrangement of the fibres is not necessarily maintained. The point to point mapping between the ends is totally arbitrary. An image being transmitted by the optical fibre bundle will therefore be corrupted. The output image will bear no visible relationship to the input image, as the distribution of intensities that make up the image will be totally random. An incoherent optical fibre bundle is therefore unsuited, as it stands, to visual image transmission. It does find widespread use in remote illumination and multipath digital communication systems. However, it is essential to note that the information that makes up the input image is not lost, but displaced. It should therefore be possible to rearrange the output of an incoherent fibre bundle to reproduce the input image, within the transmission characteristic limitations of the optical fibre bundle.

1.2 Digital Image Processing

The marriage of vision and computer systems has brought massive computational power to bear on image processing and analysis. Digital image processing, as the term implies, is the transformation or modification of the digital, that is, numerical representation of an image to produce another digital image. Digital image analysis is the extraction of information from a digital image, and can be regarded as a branch of digital image processing. Any effort to correct the output of an incoherent optical fibre bundle, by some calibration method, will involve the analysis of the output image to find a relationship to the input image, and subsequent processing to reconstruct the input. With the current availability of digital processing power, the most obvious method of pursuing the calibration is by digital analysis and processing.

1.2.1 Concepts

A digital image is a two dimensional array of digital values, each of which represents a picture element or pixel. This numerical value indicates the intensity of light at that point in the image. The size of the point is variable, and depends on the required resolution of the imaging system : the smaller the pixels, the greater the resolution of the system. A digital image can be visualised as a grid whose contour or relief represents the image intensity over that area, the higher the relief the greater the light intensity at that point.

The digital image array is of course a matrix of values, and as such, is subject to mathematical manipulation. It can be convolved with other matrices, multiplied and divided by constants etc. Subsections of the array may also be manipulated for interpolation of pixel values, local averaging, filtering and differentiation/integration. The filtering operators can be of several types, among them being high and low pass filters and the Laplacian operator. These operations are used to highlight or suppress features in the image, make physical measurements of objects in the image, and enhance / restore the image, among other image processing calculations. Each pixel in the array has an index, and can therefore be placed at any other point in the array by assigning a new index or address. Relevant image processing and analysis operators and methods are discussed in Chapters 4, 5 and 6.

1.2.2 Systems

Any digital image processing system will have certain key elements. A sensor to convert the light to an electrical signal is required. This may be as simple as a photodiode or for more detail and moving images, a video camera. This electrical signal must be converted from an analogue signal to discrete digital samples by an analogue to digital converter. If the image is to be stored, a pixel indexing value is necessary to locate the pixels in the storage area. A count of pixel samples is a straightforward method of generating a pixel address to facilitate storage and retrieval. The processing system must have access to the digitized video information through an interface. The speed and complexity of the calculations that can be performed on the image depends on the computer system attached to the interface, and the speed of data transfer through the interface. These factors will be considered in Chapters 4 and 5. A visual display of the processed result may be required and therefore a digital to analogue converter is needed to produce an analogue signal from the pixel values and some circuitry to provide the associated timings for a complete video signal. These are only the basic elements of a digital

processing system. In reality these elements can be very complex, the degree of complexity being dictated by the intended application of the system

1.3 Aims Of The Project

There exists, currently, a large price difference between coherent and incoherent optical fibre bundles. Coherent optical fibre bundles are considerably more expensive than incoherent bundles. This is due in large part to the manufacturing processes involved: that for coherent bundles is much more labour intensive, time consuming, and requires far greater precision. As a consequence of these stringent requirements, the length of coherent bundles available is limited to a few metres.

Incoherent optical fibre bundles are not subject to any such constraints. Long lengths are available for digital telecommunication. Optical amplifiers can be used periodically along the length of the bundle to counteract the attenuation of the light in the fibres. This would be extremely difficult to achieve in a coherent bundle because of the need to maintain the relative physical positions of the fibres.

The transmission of the image as a pure optical signal to a remote point, before conversion to an electrical signal, allows the visual inspection of objects where possible electrical interference must be avoided. In instances where access is difficult, the physically smaller optical fibre would be more suitable than a camera system. This advantage finds widespread use in the field of endoscopy. This thesis seeks to expand the possible range of applications of endoscopy. The use of incoherent optical fibre bundles for the transmission of visual images is examined. Achieving this will involve some method of finding a remapping function, or calibration, to relocate the output of the individual fibres in the bundle to the correct position, thus recreating the original image.

1.3.1 Software

The project seeks to design and implement software routines that will perform the required calibration, store the data generated, and execute the remapping function to produce a reconstructed representation of the original image. A brief look at the quantities of data to be handled, 64 Kb of data every 40 milliseconds for a video image of resolution 256 pixels by 256 lines, indicates that software routines may not be capable of continuous remapping of the incoming video image, because of the large amounts of data involved.

The calibration is envisaged as a one-time procedure, the remapping function being peculiar to each bundle of fibres. A user interface requiring very little knowledge of the software algorithms will be provided, to facilitate straightforward calibration of the fibre bundles.

1.3.2 Hardware

Initial testing and development of the calibration routines will be carried out using off the shelf video frame capture/display systems and computer systems. However, dedicated systems will be developed, primarily to perform the remapping of the image output from the fibre. Since the hardware reconstruction system will involve all the elements of a digital processing system, it is envisaged that the same system could be used for the calibration of the optical fibre bundles, further reducing the cost and complexity of calibrating and using incoherent optical fibre bundles

The hardware target is real-time remapping. In this context, real-time indicates one video frame time. The output image is therefore continuously updated as each new video frame arrives from the camera, allowing moving images to be viewed, without loss of information and with minimal time delay. The hardware system for remapping will also be used for the calibration of the fibre bundles, through an interface to a chosen processing system. A non-volatile memory location will be used for storage of the remapping function. The option of using a volatile memory location, and downloading the data from another storage medium, the processing system's hard disk for example, will be provided. As the remapping function is peculiar to each fibre, this would allow easy use of other incoherent fibre bundles by installing the relevant data.

1.4 Review Of Chapters

The following is a brief summary of each chapter, and establishes the layout of this thesis.

Chapter 2 outlines the essential principles of image processing and analysis, in order to show the direction that software algorithms will follow. Similar investigations have been proposed by several other organisations, and have been the subject of many patent applications. Chapter 2 presents a critical examination of four of the most relevant patents. The proposed calibration method is outlined in each case, and the merits analysed. The work undertaken in this project is also the subject of a patent

application, and this chapter establishes the basis for the differences and innovations of the calibration methods developed in this investigation. The other work done in the field of visual image transmission by optical fibres has been confined to the field of endoscopy, primarily in medical applications, using coherent bundles and is not particularly relevant in this thesis, except to indicate of possible sources of application for the completed system.

The theory and practice of optical fibres, fibre bundles and transmission of light is examined in Chapter 3, in order to gain some insight to aid the development of calibration methods, and explain any features that may be observed during such development. Propagation of light along an optical waveguide is described in terms of light rays and in terms of the electromagnetic theory of light. Detailed theory is not put forward as this has been well documented elsewhere, and is outside the scope of this thesis. The transmission characteristics of various types of fibres are examined, with attention to the factors affecting the propagation and attenuation of light through the waveguide. Fibre bundles have optical properties that are essentially the same as those of single fibres, but for image transmission some collective properties are of great importance. Due attention is given to these properties in this chapter. The techniques currently employed in the manufacture of optical fibres and fibre bundles are also briefly described.

Having established the characteristics of optical fibres and fibre bundles, the next chapter describes a calibration method. Chapter 4 describes the calibration of a fibre bundle by location of each fibre in the bundle by the use of a moving spot of light. An explanation of the principle, and a description of the equipment and apparatus used is given. Software development and design is described, with emphasis on the image analysis involved and manipulation of the data generated to produce the required result. The results are cursorily examined to expose the limitations of this calibration method, and to outline the need for a better fibre and calibration method and to establish the requirements for the hardware system.

With an established calibration method, a hardware design is described in Chapter 5. The essential requirements of the system are put forward, with an explanation of the satellite systems that will be interfaced, especially the introduction of a transputer as the host processor. The design philosophy is presented and the construction of the system is detailed (with reference to Appendices B.1 and B.2). To test the system, software to perform the calibration on this hardware system was designed and implemented in Occam 2, the dedicated programming language for transputer

systems. The real-time remapping was tested with several sources of data for comparison. An appraisal of the system's performance is presented in terms of its advantages and limitations.

Chapter 6 describes the implementation of a calibration method that uses the transmission of test patterns through the bundle, and analysis of the output for such transmissions. The principle is explained, and a description of the apparatus used is presented. The use of the transputer as the central microprocessor is described, as well as the change to Occam 2 as the programming language. Three methods of generating the test patterns are described, with their individual advantages and disadvantages outlined. The apparatus used in each case is described, along with the software design and development required in each case. A primary examination is made of the results from each method.

Chapter 7 presents the full results for all the calibration methods. A discussion and analysis is undertaken, with a view to explaining the features of each result. Most emphasis is placed on the results obtained using the hardware designed for this particular application, as a version of this hardware will be used by the sponsors of the project for field testing.

Chapter 8 draws the conclusions from the theories and work detailed in the previous chapters. Further work on the calibration method and system is proposed, as well as outlining possible applications.

2. REVIEW OF RELEVANT WORK

Work relevant to the investigations carried out in this thesis can be classified into three areas: the image processing techniques that will be used to solve the problem, previous work carried out by other parties in the search for a suitable calibration method, and the field of endoscopy which indicates possible applications of the completed work. This chapter is divided into three major corresponding sections. The principles of image processing are examined, a critical appraisal is made of four patented calibration methods, and applications of coherent fibre bundles are briefly described.

2.1 Image Processing

Digital image processing depends on the interface of visual image acquisition systems and digital computer systems. The basic elements of such a system are shown in Figure 2.1. Image processing systems can be classified into two categories: real-time systems and non real-time systems. In image processing terms real-time is measured according to the frame refresh period.

There are currently several television display standards, the most common being the Phase Alternating Line (PAL) and the National Television Systems Committee (NTSC) standards [*1 - 4*], for colour display systems. In any current television system, the display of moving images depends on the periodical display of scenes, each slightly different from the previous one. The difference between the scenes is perceived by the human eye as a continuous movement. In the NTSC standard the scenes, or frames are updated 30 times a second, and in the PAL standard 25 times a second, both of which exceed the detection capability of the human eye [*1*]. An image processing system that can extract information from, or alter the information in, the frame within one frame period (40 ms for PAL and 33 ms for NTSC) is described as a real-time system. The attainment of real-time depends on the amount of processing to be done, whether a software or hardware based design is used, and the operating speed of the hardware system. A software system allows more flexibility in application but requires more sophistication and greater processing power. Hardware processing systems tend to be more robust, but dedicated to a single application. Generally, hardware based operation tends to be faster than software operation, making the achievement of real-time operation easier. Two

software based real-time vision systems are described in [5] and [6]. A more fundamental description of the hardware needed for image processing systems will be described in chapter 5, as well as a system that combines the two approaches to image processing.

Non real-time systems are simpler in concept, tend to be less expensive, and therefore are more common. The elements are those of a general image processing system and an interface to a separate computer system. These systems will be initially used in this application for development purposes, specifically the Data Translation DT2853 [7] and the Matrox PIP-1024 [8]. The common principle of these architectures is to digitize the intensities that represent a single frame of analogue video as the data is obtained from the camera, store the digital information in a buffer and make this data available both to the host computer system for processing and analysis, and to a digital to analogue converter for display of the processed images.

The digital values represent intensity values at sampled points, or pixels, in the image. A two dimensional base area matrix whose element values represent the intensities is used to describe the scene. Various operators may be applied to this matrix [9,10] and these generally take the form of:

1. A point operator, where the output value from the operator is dependent on the input value at the same point.
2. A local operator, where the output value is dependent on the region around the pixel.
3. A geometric operator, where the output value depends on a geometrical relationship with some other pixel in the image, as would be the case in a rotation or translation.

The digital, or numerical, representation of the image is subject to mathematical manipulation. The most common operators are described as convolutions, which produce an output value for the pixel in question that is a linear combination of the input values, with coefficients that are dependent only on the positions of the input values relative to the pixel, and not their absolute positions in the image [9].

A simple example of such a convolution is one that performs a local averaging function, where the output value is the average of the neighbours of the corresponding input point. The function is described by the matrix below.

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

The average is calculated for the value at the centre of the matrix. This matrix is typical of the convolutions used in image processing. The implementation of the calculation can be understood by considering the following subsection of an image

$P_{-2,-2}$	$P_{-1,-2}$	$P_{0,-2}$	$P_{1,-2}$	$P_{2,-2}$
$P_{-2,-1}$	$P_{-1,-1}$	$P_{0,-1}$	$P_{1,-1}$	$P_{2,-1}$
$P_{-2,0}$	$P_{-1,0}$	$P_{0,0}$	$P_{1,0}$	$P_{2,0}$
$P_{-2,1}$	$P_{-1,1}$	$P_{0,1}$	$P_{1,1}$	$P_{2,1}$
$P_{-2,2}$	$P_{-1,2}$	$P_{0,2}$	$P_{1,2}$	$P_{2,2}$

where the elements $P_{x,y}$ are the pixel values at points x,y.

The image, *I*, can be described by the relationship

$$I = f (P_{x,y})$$

where f is a function that describes the spread of intensities in the image. Convolution with the above matrix centred on pixel $P_{0,0}$ yields the following expression for $N_{0,0}$, the new value for $P_{0,0}$

$$N_{0,0} = ((P_{-1,-1} / 9) + (P_{0,-1} / 9) + (P_{1,-1} / 9) + (P_{-1,0} / 9) + (P_{0,0} / 9) + (P_{1,0} / 9) + (P_{-1,1} / 9) + (P_{0,1} / 9) + (P_{1,1} / 9))$$

This is one of the simplest applications of the convolution technique. The uses extend to the implementation of more complex mathematical equations to enhance the image and to extract the required information [*11*]. The use of image processing in the industrial environment includes robotic control and measurement systems [*12*]. The capability of the robotic control system depends on the amount of information that can be extracted from the visual scene, and the accuracy of that information.

Measurement systems depend upon the accuracy of the detail in the image, which in turn depends on the resolution of the camera and quantization levels of the digital system.

Any image processing method used for the calibration of the fibre bundles will involve some form of measurement system, to locate pixels and their output addresses in the fibre bundle's output area. It is therefore necessary to examine some of the factors that affect the accuracy of measurement systems. The basic accuracy is determined by the pixel resolution of the imaging system, that is, the number of pixels used to define the image. There are two sources of limitation of this aspect: the camera and the digitising system. The line and pixel resolution of the camera sets an absolute value for the whole system as it is not possible to acquire information about the scene that is not provided by the camera. Interpolation techniques may be used for calculating inter-pixel values [*13*], but the information is derived from the data provided by the camera, and is thus limited by this fact. The number of digital pixel values used to describe the output from the camera is the other factor that affects the resolution of the system. There is nothing to be gained by the resolution of this system exceeding that of the camera, as no more information will be described. However, the resolution may be less than that of the camera to reduce the memory requirements of the system and the amount of data to be handled.

The lens/object arrangement also affects the accuracy of the system. The size of the image of the object is dependent on the magnification factor of the lens and the distance of the object from the lens. For optimum accuracy, the image of the object to be examined should be as large as possible as can be seen from the following example. Consider a system of resolution of 500 pixels horizontally that is to be used to measure an object 5 cm long and one line thick, and two alternative lens arrangements are used giving image sizes that occupy 200 and 500 pixels horizontally. The accuracies attainable are (5 cm/200 pixels) 2.5 mm per pixel and (5 cm/500 pixels) 1 mm per pixel. Since distances can only be measured to the nearest pixel, the second arrangement provides the better accuracy in measuring the length of the object.

All of the above factors must be taken into consideration in the design of an image processing/visual measurement system. Specific image processing manipulations will be developed and described in the following chapters, as the need to solve the problems encountered in the calibration arises.

2.2 The Purpose of the Calibration

Figures 2.2 (a) and (b) show sections of an incoherent bundle and a coherent bundle of fibres respectively. The random arrangement of the incoherent bundle is apparent, and the effect on any images transmitted will be a corruption of the spatial arrangement of the intensities that form the image.

To perform the rearrangement of the output image of the fibre bundle, some transformation algorithm must first be obtained. Since the arrangement of the fibres within the bundle is totally random, this algorithm will be specific to each fibre bundle. The construction of the bundle is such that the arrangement of fibres is fixed by glueing the fibres together at the ends of the bundle. It will therefore be necessary to derive the algorithm only once for each fibre, if some method of storage is used for the information derived.

A digital representation of an image is usually arranged as a two dimensional array. The coordinate system for identifying the location of a pixel in that image is shown in Figure 2.3. This system will form the reference frame for any calibration and reconstruction mechanism. Having obtained the transformation algorithm, the image that is obtained from the bundle must be rearranged to reproduce the original input image, within the accuracy limitations of the calibration system. A simplified block diagram of this procedure is shown in Figure 2.4. The properties of the necessary algorithm can be understood from the following explanation, and referring to Figure 2.4.

Consider an incoherent fibre bundle where the input points X, Y, and Z are displaced by transmission through the fibre to points X', Y' and Z' respectively. The algorithm must relocate points X', Y', and Z' in the output image back to X, Y, and Z to reconstruct the original.

Thus for the fibre bundle

$X \rightarrow X'$

$Y \rightarrow Y'$

$Z \rightarrow Z'$

and for the algorithm

$X' \rightarrow X$

$Y' \rightarrow Y$

$Z' \rightarrow Z$

This concept forms the basis for any remapping that can be used to reconstruct the original image. There are various methods for the realisation of this algorithm. This thesis develops two such methods; the first is derived from a proposal by Light Optics Limited [*17*], and the second a totally original procedure.

2.3 Other Calibration Methods

The general concept of using an incoherent fibre bundle for the transmission of optical images has been the subject of previous research. The principle of using stored information about the fibre to reconstruct the image by remapping the displaced points to their correct locations is common to all the methods that have been examined. The research into the concept has been patented by several bodies. The following is a critical examination of four of the most workable and relevant patents.

2.3.1 First Patent

This patent application, number 8317907 [*14*], details the use of incoherent fibre bundles for use in document scanners. It is not a general application, but is directed at a particular fibre bundle arrangement. The bundle is constructed so that one end is of a rectangular shape and the other a circle, or square. The concept is one where a line of text is condensed into a tighter arrangement for acceptance by a camera or other vision system for electrical transmission, as shown in Figure 2.5. The line of text is optically scanned by the rectangular end of the fibre bundle, and rearranged into the more compact formation determined by the geometry of the bundle.

The reconstruction of the image follows the principles described in section 2.2. The method of obtaining the data to perform the reconstruction is not detailed in this application. However, the patent does introduce the aspect of the limited resolution of optical fibre bundles. Brief mention is made of the density of fibres that will be required to accurately decode the characters being transmitted by the fibre bundle. This indicated that special attention should be paid in the development of this project to the required image quality and the complexity of images that could be resolved by the fibre bundle.

The images to be transmitted in such an application consist of two intensities: the black text and white background. The detail in the image is relatively low, and the objects of interest, the characters of text, forms a significantly large part of the image,

a representation of which is shown on Figure 2.6, as well as the typical output for such an image from an incoherent bundle. Definition of the characters need not be absolute to be recognised. Such an image would therefore be tolerant of a low resolution imaging system. Images containing a larger sample of intensities would require higher resolution to preserve the detail in the original image.

2.3.2 Second Patent

This patent, number 8203417 [*15*], offers a proposal for calibrating the fibre and performing the reconstruction. The reconstruction is essentially similar to the previous procedure. To perform the calibration, the use of a single beam Cathode Ray Oscilloscope (CRO) is proposed. The beam from this device is focused to a small point, to provide an approximation to a point light source for the calibration procedure as shown in Figure 2.7. This spot of light is intended for the illumination of single fibres in the bundle. The position of the spot is varied by applying voltages to the deflection plates of the CRO. By finding the output point for each fibre in the bundle, and knowing the input points for the location of the CRO spot from the voltage applied, a table of corresponding input and output transmission points is obtained. This table can then be used in the reconstruction of the original image from the output of the fibre bundle.

Theory of the CRO [*18*] shows that the position of the spot of light can be moved over the display screen by varying the voltage applied to X and Y plates of the CRO. The accuracy of control over the movement of this spot depends on the size of the voltage steps used to effect this movement. The finer the voltage steps, the smaller the movement increments. The deflection coefficients of the CRO, for the X and Y directions will also determine the degree of deflection of the spot of light. For a typical light guide (Chapter 3), containing fibres of 50 μm in diameter, to illuminate one fibre only would require a light source whose area of illumination did not exceed the surface area of the individual fibres in the bundle (section 3.5.2.1). Larger spots of light would cause multiple fibres to be illuminated. By the nature of the phosphor screen used to produce the light output from the CRO, where the total light output is dependent on the size of the spot [*18*] and the speed at which the electrons impinge on the screen, the spot of light will tend to be comparatively large, and it was not expected that a light spot small enough and bright enough could be obtained from a standard CRO. The effect of attempting to illuminate single fibres with a relatively large beam is shown in Figure 2.8. This method of calibration was therefore not pursued further, but it did establish the principles for the first calibration method that was implemented.

2.3.3 Third Patent

This patent, number 8229495 [*16*], describes a particular calibration method. The principle of the reconstruction method is identical to the previous descriptions. The calibration depends on illuminating rows and columns of fibres in the bundle. The illumination source is a narrow band of light, or strip, of predetermined width, and is moved across the bundle in steps of one fibre diameter as shown in Figure 2.9 (a) and (b). This stepping action is performed in both the horizontal direction to obtain a value for the x coordinate of the entry points, and the vertical direction to obtain the y coordinate for the entry points. The output point for each of the fibres illuminated is determined by calculating the x and y coordinates for all the fibres that are illuminated in the output image, by the horizontal and vertical strip, respectively. Thus the coordinates for the input points are obtained for groups of fibres illuminated by the strip of light at each entry coordinate, providing a table of corresponding input and output locations.

Closer examination of the principle and practicalities involved showed that implementation of this method would be difficult. The primary reason for this difficulty stems from the fact that the method assumes that the fibres are aligned to a square grid. The location of the fibres would then correspond to the steps of the strip of light. It was observed from close examination of the fibres in an incoherent light guide that the arrangement of fibres was not to a square packing arrangement (section 3.5.1.1). The arrangement was essentially random, but could be approximated by a triangular packing arrangement (section 3.5.1.1.). This indicated that some fibres would be partially illuminated by the strip of light as in Figure 2.10, and they would produce an output when not exactly aligned to the centre of the strip of light. Differentiation of the correct output fibres for the input strips of light would be impossible when the problem is compounded by its existence in two dimensions, and involving a variable and unknown number of fibres that would be illuminated for each x and y entry coordinate. Selection of the fibres that should correctly correspond to any input coordinate would prove difficult and subject to large errors.

It could be argued that by reducing the width of the strip of light, the spread over fibres that should not be illuminated would be reduced. Consideration of the size of fibre in the bundle, about 50 µm, and referring to Figure 2.11, shows that the width of the strip of light should be less than half that of the fibres' diameter, or smaller than 25 µm. The effect of misalignment with the light source is shown for various possibilities in Figure 2.11. A strip of light of width 25 µm would still produce erroneous output points, due to the almost random arrangement of the fibres in the

bundle. The problem of choosing the correct output points would still remain, but with fewer possible errors.

The principle of calibration showed promise in that a relatively small number of sample positions over the face of the fibre would be have to be taken, keeping the amount of data to be handled small. However, the apparatus to perform a test of this method was not available, and the difficulty foreseen in implementing the image processing software to extract the correct information indicated that this method should not be pursued further.

2.3.4 Fourth Patent

Patent number 8118952 [*17*] presents a calibration method and the typical reconstruction procedure. The proposed method of calibration uses a single spot of light to illuminate single fibres in the bundle. A simple analysis of the nature of optical fibres shows that the spot should be no larger than half the diameter of a single fibre in the bundle (see chapter 3). The spot of light is generated from a light source and focusing lens arrangement. The proposal states that the face of the fibre should be scanned by this light spot by means of an electronically controlled rotating mirror. The output of the fibre bundle for each entry point is digitally examined to locate the output point. The resulting output for various alignments of the spot of light and the individual fibres is shown in Figure 2.11. With the knowledge of the input point of the light source and the calculated output point, a mapping table can be built up for the whole fibre.

Further examination of the principle of transmission of light by optical fibres and fibre bundles, detailed in chapter 3, showed that this method was the most practical, and relatively easy to implement. Section 2.1 described the importance of the resolution of the imaging system. A typical high quality digital image acquisition system may contain over a quarter of a million picture elements (pixels). If a similar resolution is required from the fibre bundle, a large number of sample points will have to be taken to calibrate such a bundle of fibres. The quantity of data produced by such a calibration method would be very large and would probably require a long time to extract the relevant information.

An approximation of the area occupied by a bundle of fibres can be obtained from the following consideration. A bundle of 250,000 fibres, each 50 µm in diameter will have a total fibre area of

$$N \times \pi \times r^2$$

where N is the number of fibres in the bundle

r is the radius of each fibre, giving

$(2.5 \times 10^5 \times \pi \times (2.5 \times 10^{-5})^2)$ m^2 or

5×10^{-4} m^2

If a circular bundle is assumed, this area will be that of a bundle 2.5 cm in diameter. While this may not seem excessively large, the significant advantage of the small size of optical fibre imaging systems is compromised. Smaller fibres, 10 µm or less are available, which would significantly decrease the overall size of the bundle, to an approximate diameter of 5 mm for the same number of fibres. The above calculation does not take into account the packing arrangement of the fibres in the bundle. Obviously a real bundle will have a larger surface area than the sum of the individual fibre areas, because of the inevitable gaps between the circular fibres in the bundle.

Calibration of fibre bundles made up of 10 µm fibres by this method would prove extremely difficult because of the small size of the light spot that would be required, and the accuracy of location of the light source. It was decided however, that the implementation of a calibration system based on this patent would be useful, primarily to develop the necessary image processing algorithms and to gain a practical knowledge of the optical light transmission characteristics of optical fibres.

2.4 Endoscopy

The concept of non-modulated light transmission by an optical fibre bundle is generally known as endoscopy [*19*]. The purpose of this thesis is to design a method for the use of incoherent fibre bundles as endoscopes. Endoscopes find wide use in areas where optical examination is necessary, but access to the object to be viewed is difficult. The main use of endoscopes to date has been in the medical field, where internal inspection of the human body, through various orifices, is desirable. The primary advantage is the flexibility of the endoscope, and the possibility of applying the properties of endoscopes to the industrial environment is being probed.

The main areas of study and application are for inspection of objects that are difficult to view directly. However the possibility of using endoscopes where the presence of an electrical signal, as in a camera based system, would be undesirable is also being

investigated. The image can be optically transmitted to a remote location, a few metres away, before being transformed into an electrical signal. The manufacture of endoscopes is an expensive process because of the great accuracy required in the physical assembly of coherent fibres, and the complex guidance mechanisms required for controlling the object end of the fibre. Coherent fibre bundles are difficult to manufacture in great lengths, and since most of the development of optical fibres is targeted at optical telecommunication frequencies, and not visible light, the absolute length of the endoscope is limited by the attenuation of the light energy at visible frequencies.

Incoherent bundles need not suffer from this limitation, as optical amplifiers [*20*] may be placed periodically along the length of the bundle to boost the light energy levels. If the calibration method is independent of the transmission medium, then this will not affect the transmission, as long as the calibration is performed with the amplifiers in place. Coherent bundles cannot be used in this way, as any interruption of the transmission of the light would negate the accurate alignment of the fibres in the bundle.

Illumination of the object to viewed is usually achieved by means of an incoherent bundle of fibres. This is a low resolution bundle whose path parallels that of the endoscope. Light is ducted to the object via the light guide from a remote source. Bundles of fibres may also be used for purposes other than the point to point transmission of light. By manipulating the shape of the ends of the bundle, and the diameter of the fibres, circular images may be converted to a rectangular format, and may be magnified or reduced [*21*].

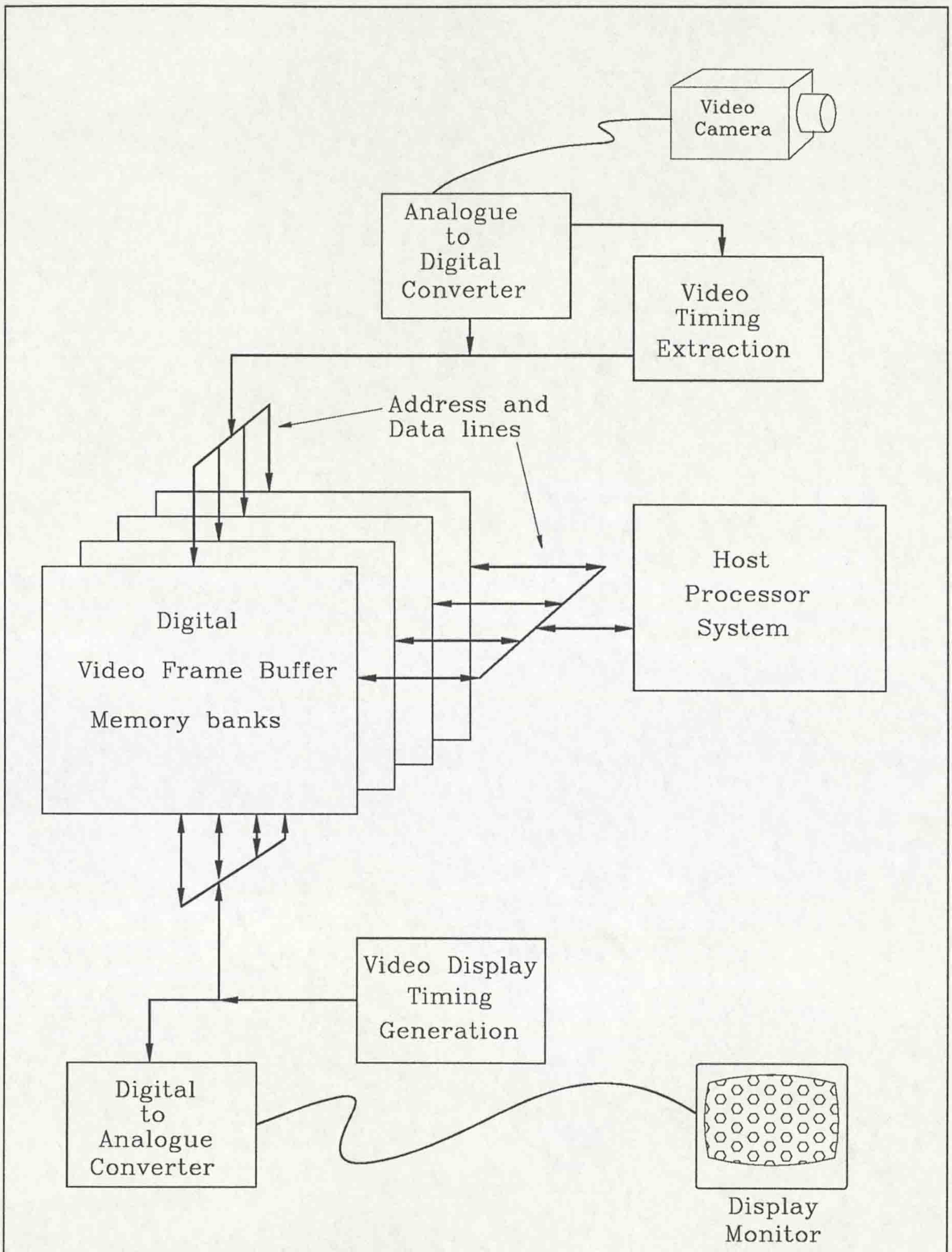


Figure 2.1 – A basic digital image processing/analysis system

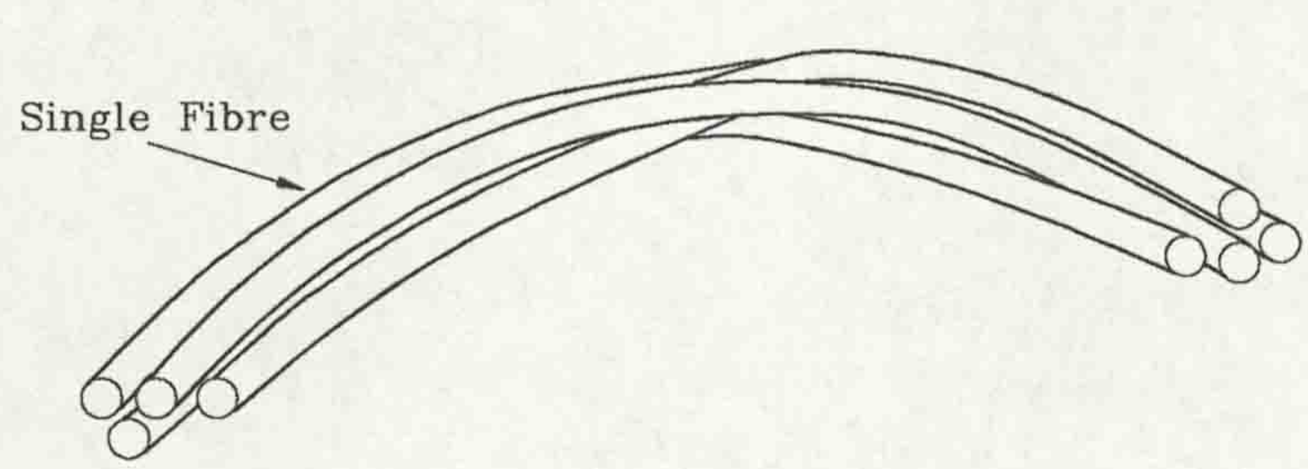


Figure 2.2 (a) – An incoherent optical fibre bundle

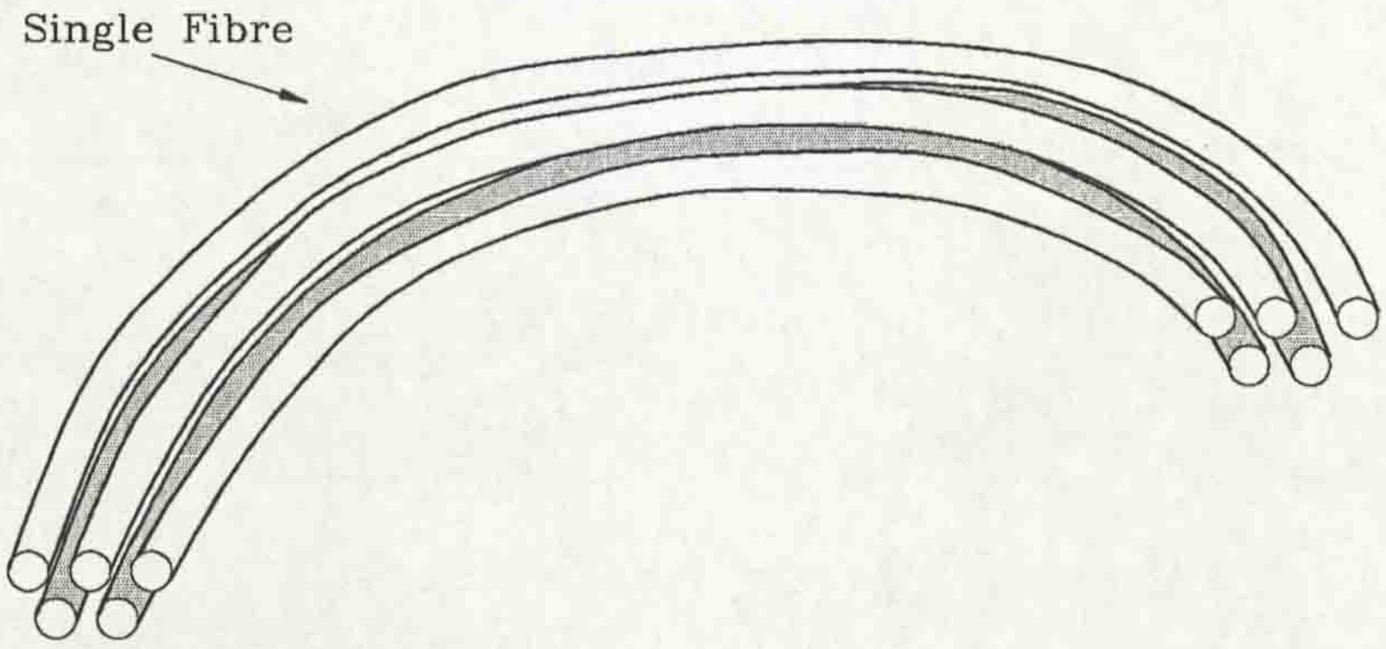


Figure 2.2 (b) – A coherent optical fibre bundle

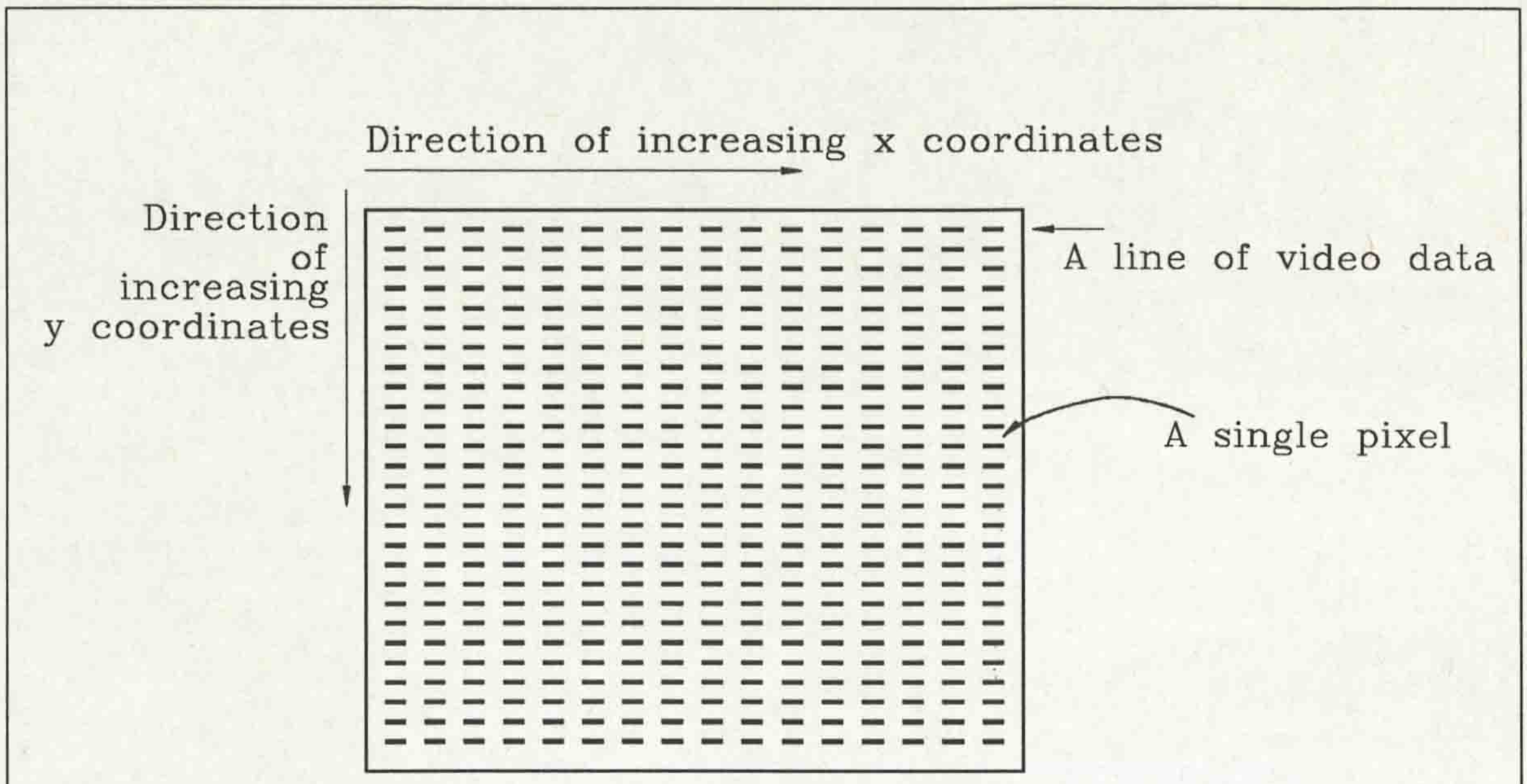


Figure 2.3 – The image coordinate system

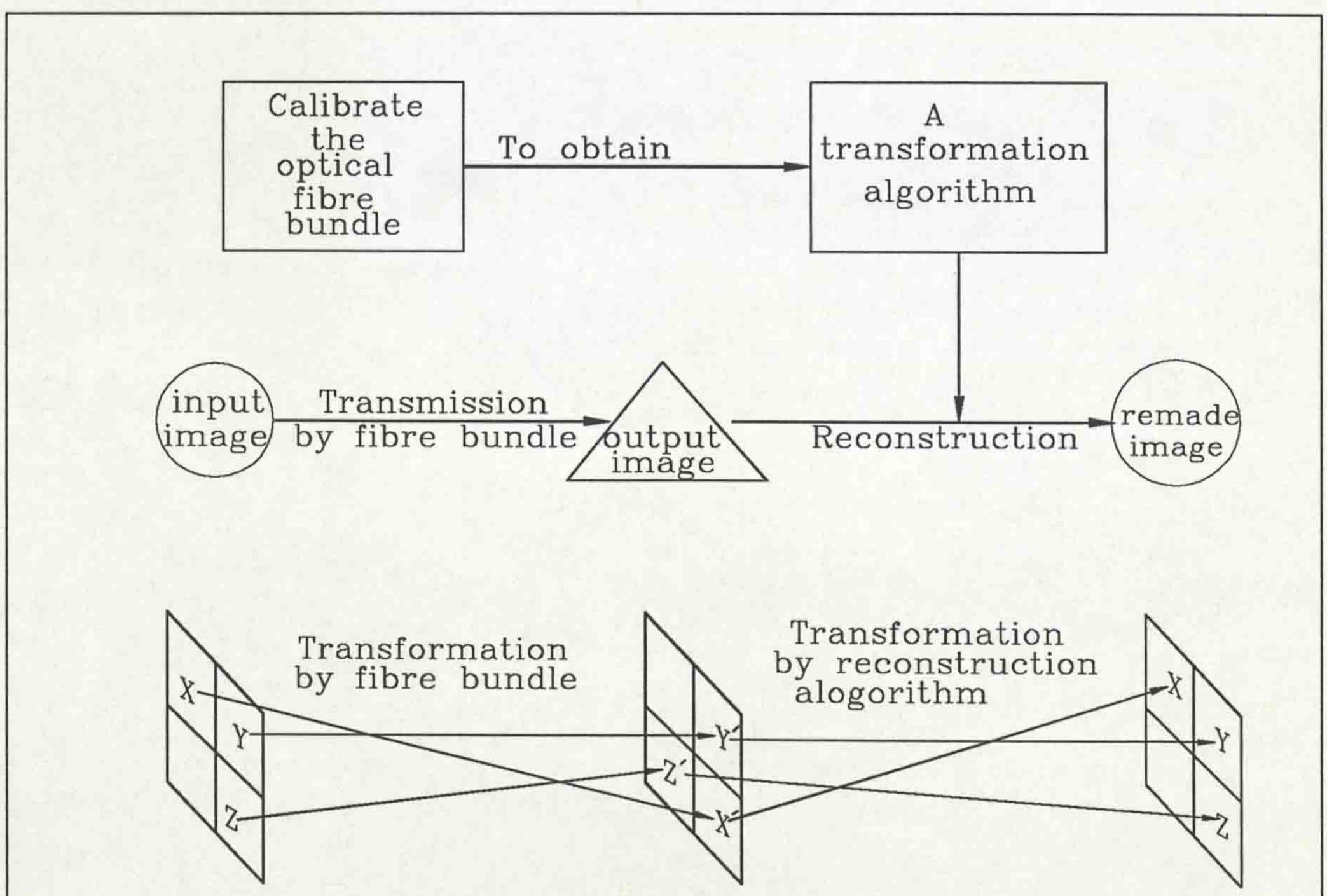


Figure 2.4 – The purpose of the calibration

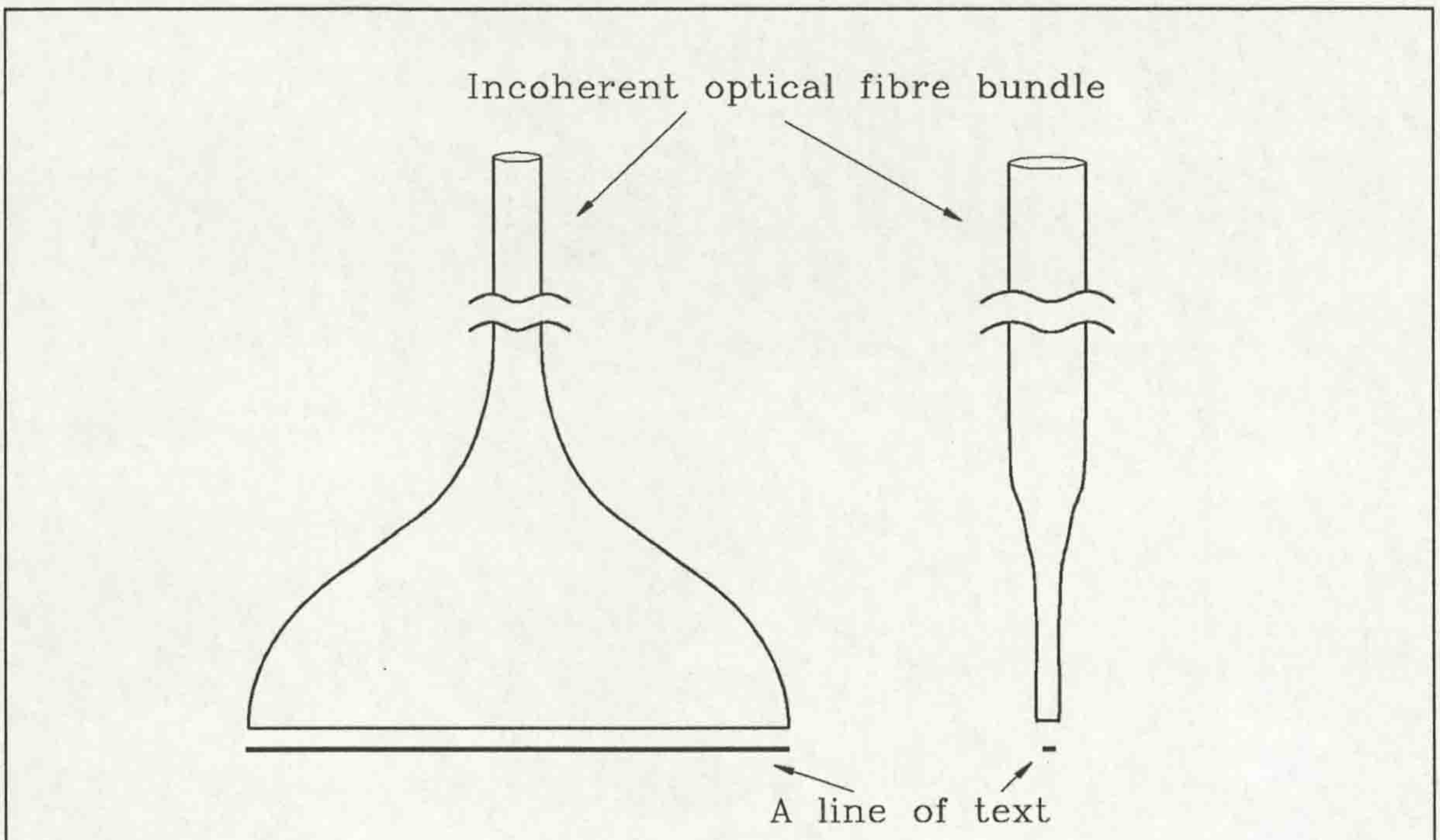


Figure 2.5 – A condensing incoherent fibre bundle for text scanning

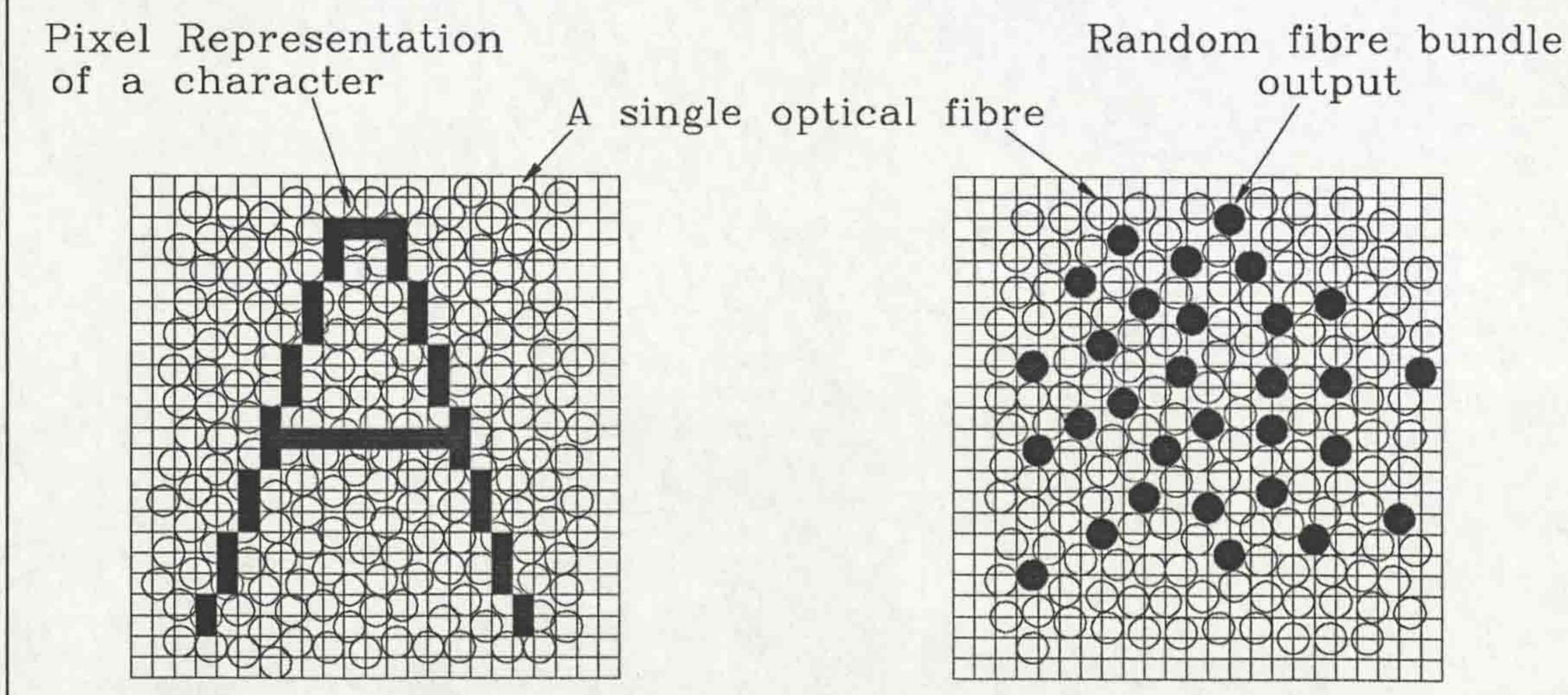


Figure 2.6 – The random transmission of text by an incoherent optical fibre bundle

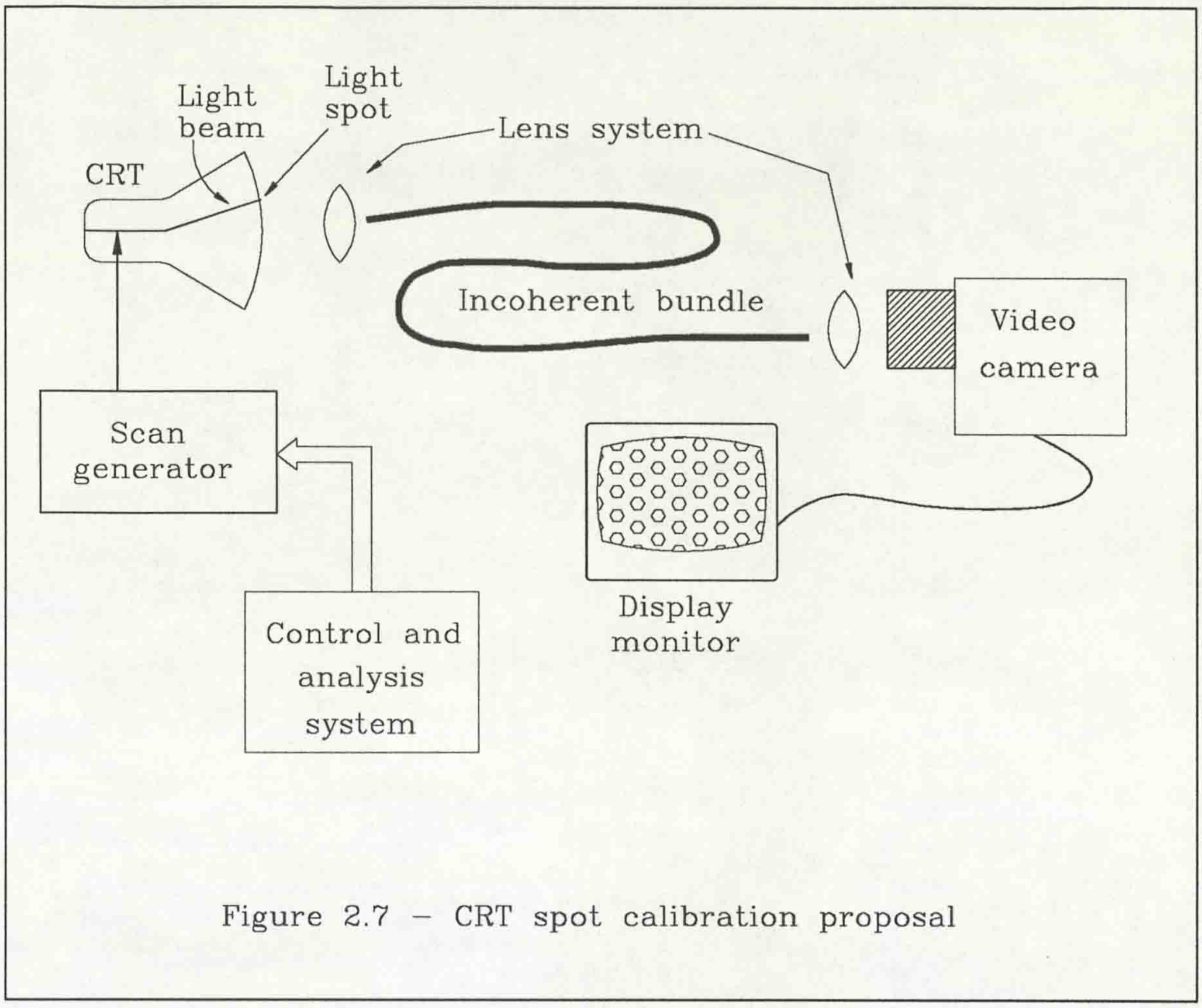


Figure 2.7 – CRT spot calibration proposal

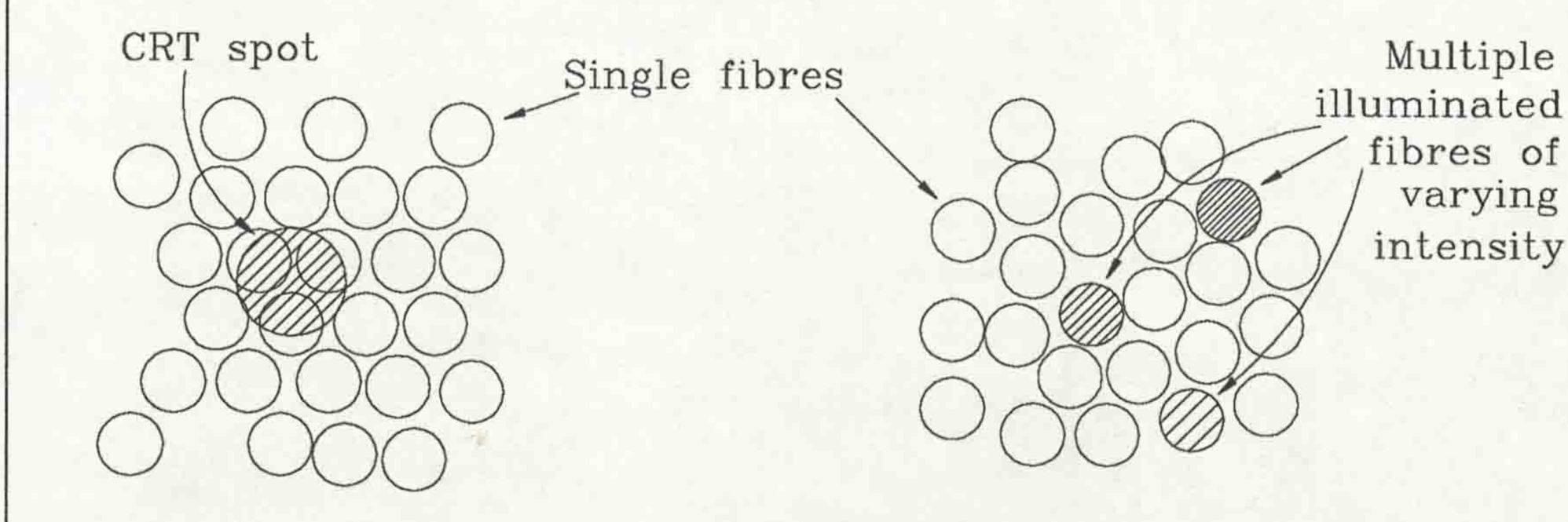


Figure 2.8 – The multiple illumination of fibres by a single large input light beam

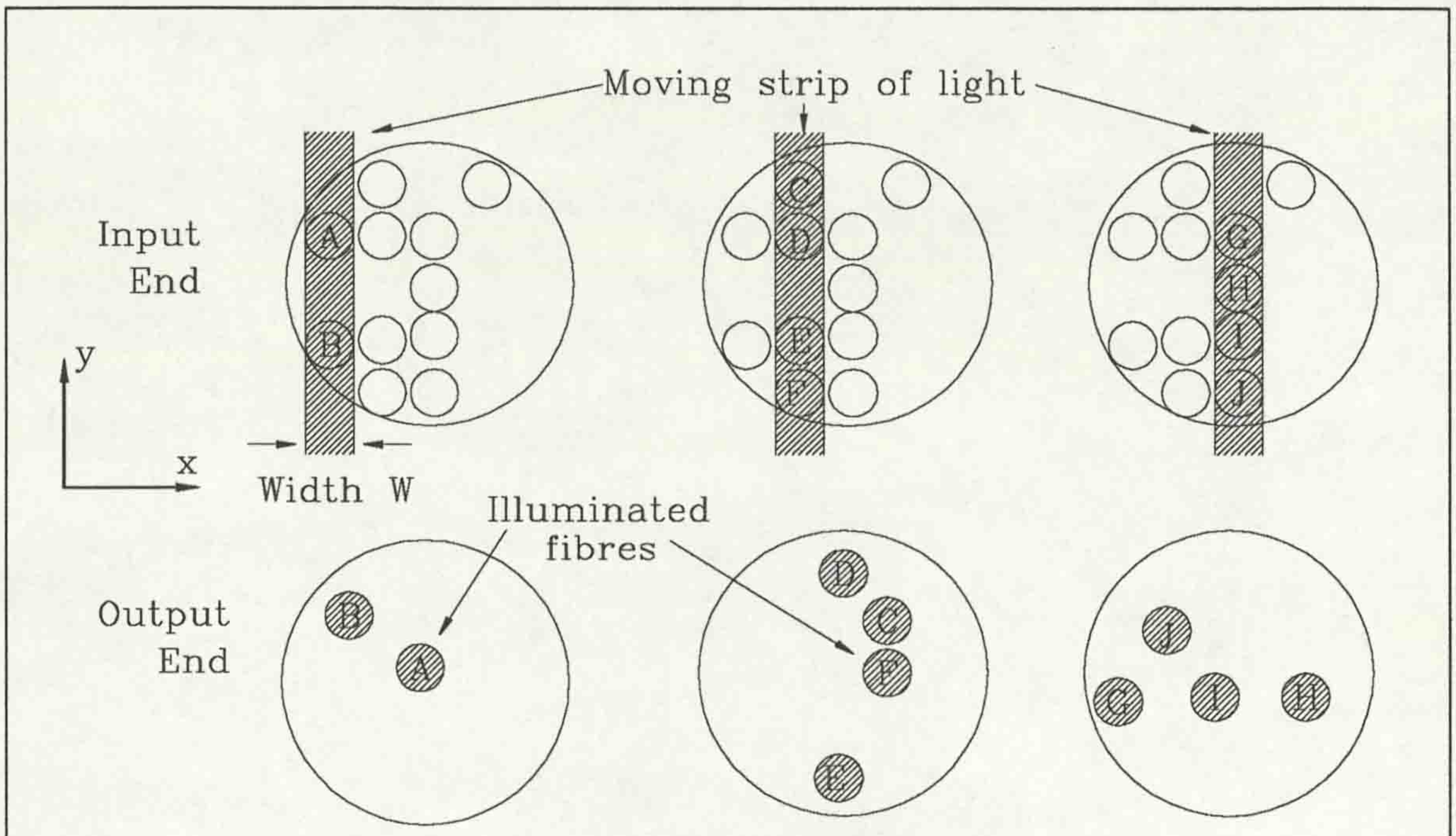


Figure 2.9 (a) – The location of y coordinates by moving strip of light

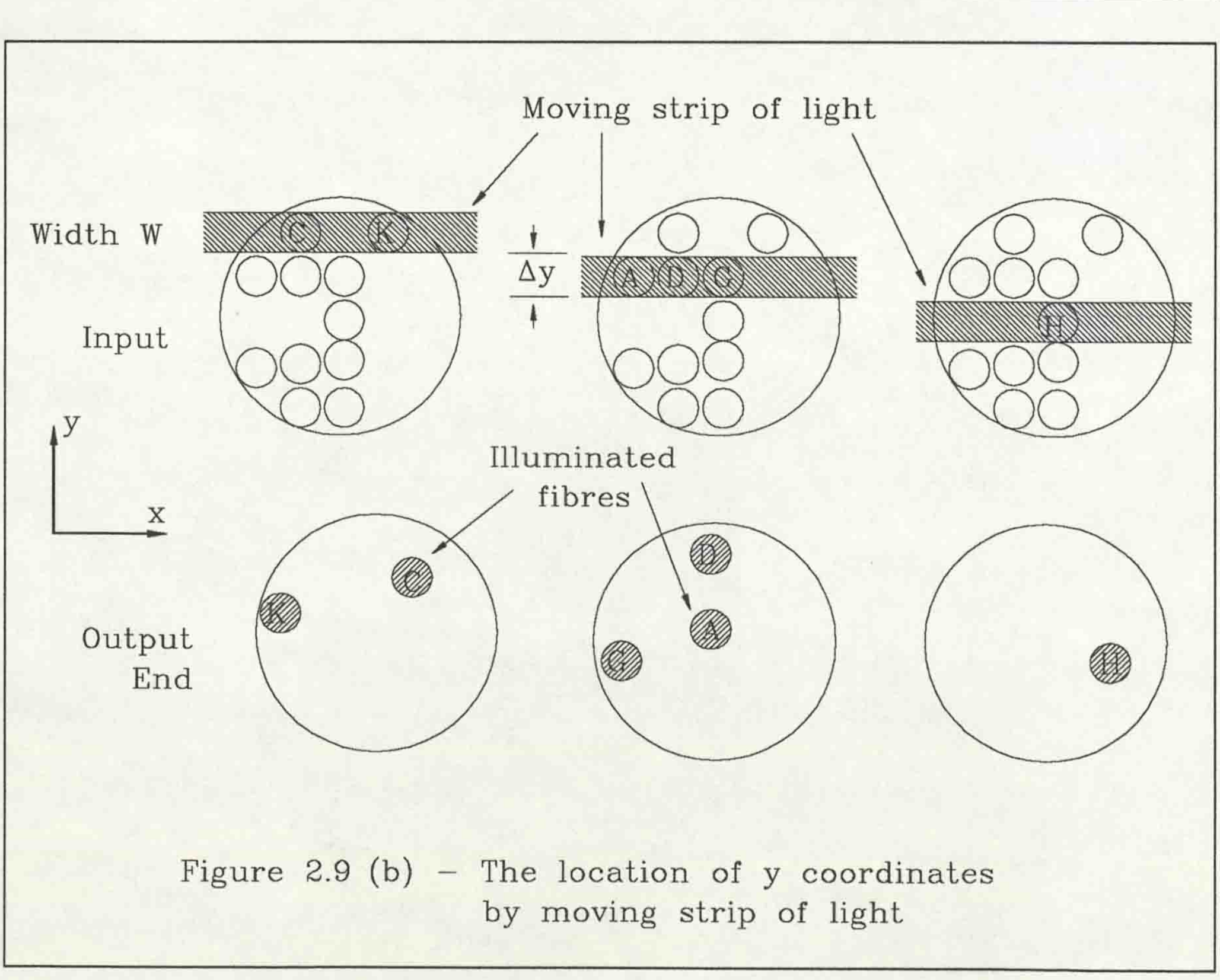


Figure 2.9 (b) – The location of y coordinates by moving strip of light

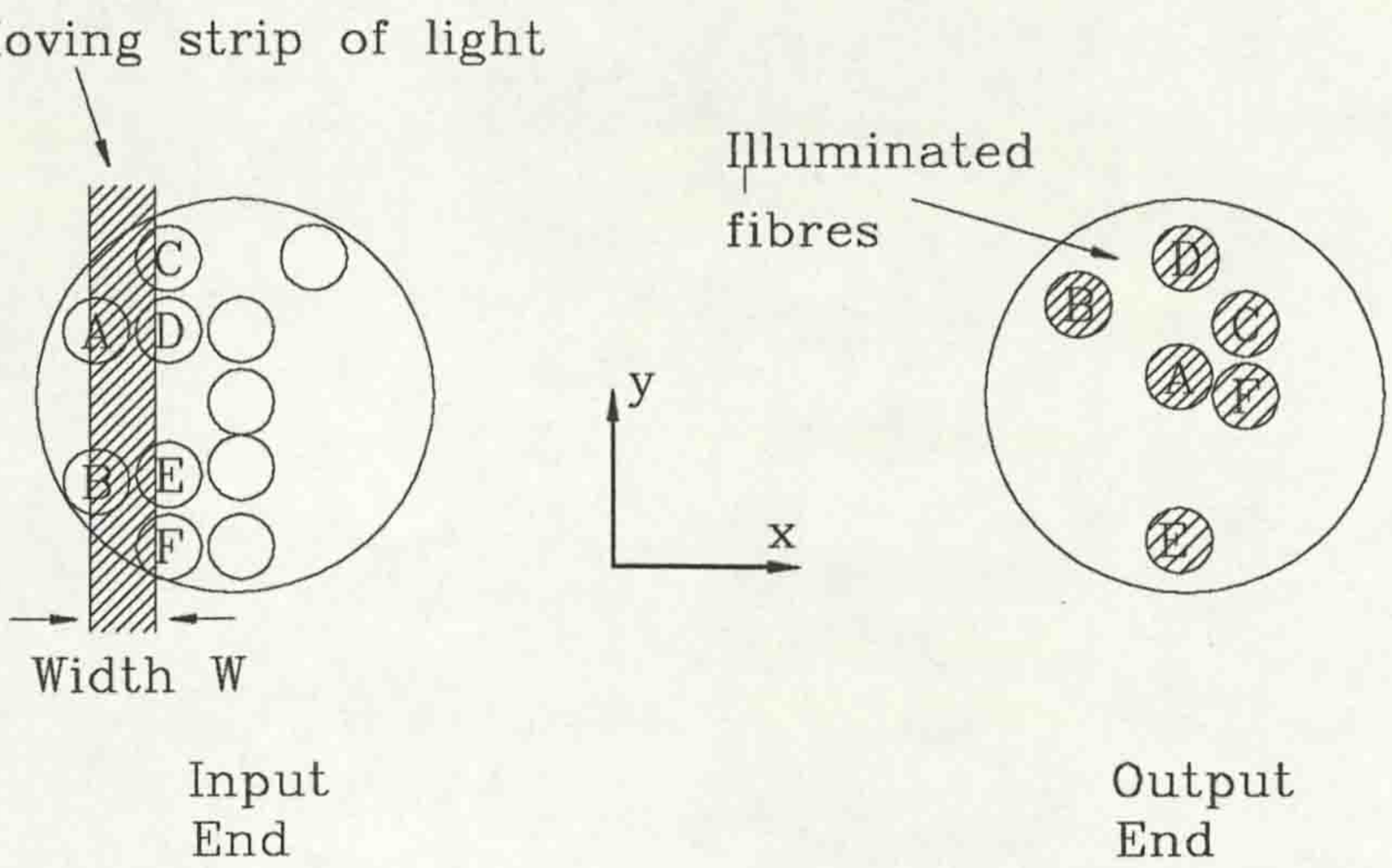


Figure 2.10 – Multiple illumination of fibres by misalignment to light strip

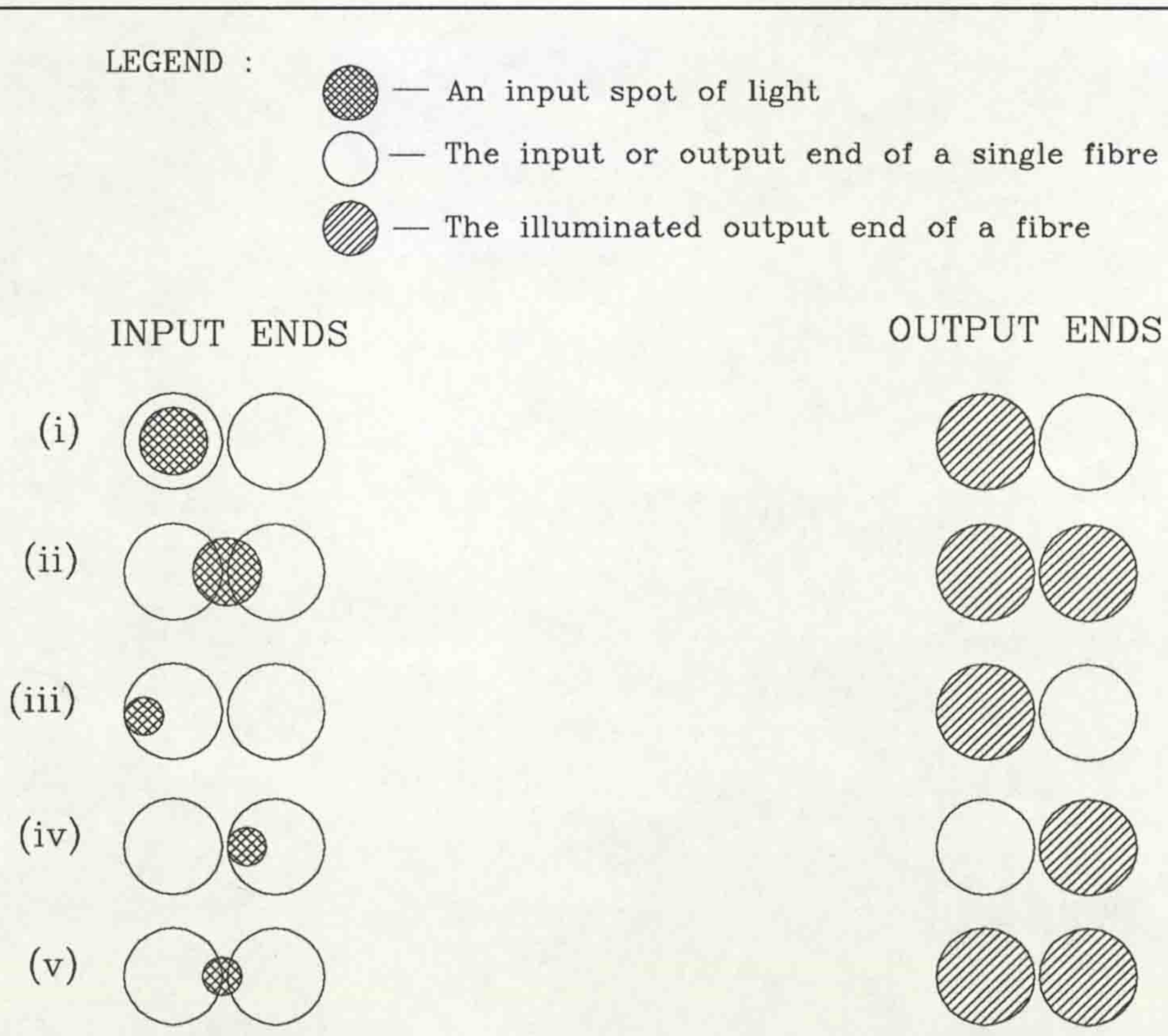


Figure 2.11 – The relationship of size and position of the light spot to the illumination

3. OPTICAL WAVEGUIDE THEORY

The principles of operation, and the optical and physical characteristics of optical waveguides are presented and discussed in this chapter. The propagation of light along an optical fibre is explained in terms of light ray theory and electromagnetic wave propagation theory, as some aspects are better regarded from one or the other viewpoint. A brief historical review of the development of the optical fibre is necessary to understand the current uses that optical fibres are put to, and the path of current research and development. Total internal reflection, on which the essentials of light ray ducting is based, is first described, with the related properties analysed with respect to the propagation of visible light.

3.1 Historical Review Of Optical Waveguides

Optical telecommunication is as old as man's ability to see. The use of light for audio communication was first proposed by Alexander Graham Bell in the latter half of the 19^{th} century. Development of the optical fibre has been mainly targeted at digital telecommunication, and thus parallels the progress of light sources for this purpose, particularly the semiconductor laser. The first lasers were demonstrated in 1960 to 1961 [*22*].

One medium considered for the transmission of optical information was a gas filled pipe, with lenses placed at regular intervals (of the order of 100 m) to recollimate the light beam [*23*]. This system produced very good performance in terms of available bandwidth and repeater separation, but was exceedingly complex and difficult to build. The initial use of long continuous glass fibres for the transmission of optical information is credited to K.C. Kao and his colleagues at Standard Telecommunication Laboratories in Harlow, England [*24*]. The techniques involved were an extension of those employed in short lengths for endoscopes, and gave rise to severe attenuation of approximately 1000 dB per kilometre at visible frequencies, over longer lengths. By reducing the impurities in the glass, mainly hydroxyl ions (OH^-), the level of attenuation was lowered to 2 dB per kilometre by 1975, but at frequencies just lower than visible light, in the near infrared range [*25*]. Since then, the attenuation has dropped to 0.2 to 0.5 dB per kilometre at the lower frequency of 2×10^{14} Hz, with the visible frequencies attenuated at a greater rate [*25*].

These large improvements have come about as a result of the enormous potential afforded by optical/digital communication systems and the resulting interest and research in this field. The focus of this thesis is on the use of these fibres to transmit visible light, which is subject to greater attenuation, but the uses are expected to be over tens of metres not kilometres.

3.2 Light Ray Theory Of Optical Waveguides

The ray theory of light energy propagation and distribution is an approximation that represents the limit in which the wavelength of the light tends to zero in comparison with the dimensions of the medium in which these aspects are to be examined. The electromagnetic field is then supposed to be that of a plane wave, and the path is the trajectory perpendicular to the plane of oscillation as shown in Figure 3.1. Optical fibres may have core diameters of 1 mm, in which case the approximation would be reasonably applied. The core diameter is more likely to be of the order of a few microns, typically 50 μm for digital communication fibres and 10 μm for coherent optical fibre bundles. With such fibres, the ray theory might be thought to reach the limits of its applicability, but is useful in establishing the basic operation of optical fibres in relatively simple terms. The electromagnetic theory is put forward to explain the more complex aspects of optical fibres.

A further dimension to be considered is that of the physical structure of the fibre, particularly the grading of the refractive index across a diameter of a cross-section of the fibre. These factors are considered in more detail in Section 3.4.1. The following analysis applies to fibres of uniform core refractive index, in a bid to establish the basic principles of operation.

3.2.1 Total Internal Reflection

Total internal reflection describes the complete reflection of a light ray at the interface of the medium in which it is travelling, with a less optically dense medium. According to the laws of reflection, the angle of reflection will be equal to the angle of incidence (Figure 3.2(d)). Reflection occurs because of the refraction, or bending, of light rays that occurs at the interface of the two dielectric mediums [*26-29*]. The refractive index of a dielectric medium is defined as being the ratio of the velocity of light in a vacuum to the velocity of light in the medium concerned. The refractive index can be taken as a measure of the optical density of any dielectric medium [*26,27*].

A ray of light incident on the interface between two dielectric mediums of different refractive indices will therefore undergo a change in velocity as it passes from one to the other. The ray will bend either towards the normal of the interface, or away from it, as it leaves one medium and enters the next. If the second medium is of lower refractive index, the ray will bend away from the normal to, and towards the tangent at, the interface, and vice versa if the refractive index is higher, shown in Figure 3.2 (a) and (b).

3.2.1.1 Snell's Law and the Critical Angle

Consider two adjacent mediums as in Figure 3.2 (c) and (d), of two different refractive indices n_1 and n_2, where n_2 is greater than n_1. The light ray IS will be refracted and follow path SR as it enters the medium of refractive index n_1. Snell's law governs the angle of refraction and can be stated in the form

$$n_1 \sin\theta_r = n_2 \sin\theta_i \qquad [26]$$

where θ_r and θ_i are the angles of incidence and exit respectively, to the normal at the point of incidence.

At a given angle of incidence, θ_r will be perpendicular to the normal, and the refracted ray will travel along the tangent at the point of incidence. This angle of incidence is called the critical angle, θ_c. For angles of incidence greater than θ_c, the light ray will be totally reflected back into the medium of refractive index n_2. Total internal reflection is then said to occur. For angles less than θ_c, partial reflection also occurs, but the percentage of light reflected is low compared to the high efficiency at total internal reflection (about 99.9%) [26]. Numerically, the critical angle can be calculated from Snell's Law, given that θ_r is 90° and therefore $\sin\theta_c$ is unity giving

$$n_2 \sin\theta_c = n_2 \text{ and thus}$$

$$\sin\theta_c = n_1 / n_2$$

3.2.2 Light Ray Paths Through A Fibre

Consider a cylindrical glass fibre of refractive index n_2 surrounded by an outer cladding of refractive index n_1, where n_2 is greater than n_1, as shown in cross-section in Figure 3.3. The following explanations assume a perfect boundary between the two mediums and an abrupt, or step, change in the refractive indices of the two mediums. The end faces are also assumed to be cut at right angles to the longitudinal axis of the fibre.

3.2.2.1 Meridional Rays

A light ray entering the glass such that it impinges on the core - cladding interface at an angle less than θ_c will undergo a series of internal reflections, potentially an infinite number, at the interface of the two mediums, see Figure 3.3. Such a ray is known as a meridional ray. A meridional ray is defined as a ray that travels in a plane that includes the fibre axis, passing it twice in one period of undulation. If a perfect fibre is assumed, then a meridional ray will be propagated along the fibre without attenuation. The angle at which a meridional ray emerges from the other end face, assuming perpendicular end faces, is the same as the original angle of incidence. Clearly the light rays propagating along the fibre will follow multiple paths, each of different lengths. Considering a straight length of fibre as in Figure 3.4, the shortest path is the followed by the ***axial ray*** AA. Over an axial distance *l*, a meridional ray such as the most oblique ray BB, will have travelled a distance of $l / \cos\theta_m$ and thus have a different path length. The eventual path difference obviously depends on the number of internal reflections that the meridional ray undergoes.

3.2.2.2 Acceptance Angle

For a light ray to be propagated along the fibre core, it must enter the core at an angle less than the critical angle. However, unless the light comes from a medium of identical refractive index to the glass core, it will undergo refraction as it enters the core through the end face. The geometry for injecting a light ray for propagation, into an optical fibre is shown in Figure 3.5. If the light comes from a medium of lower refractive index such as air, the light ray will bend towards the longitudinal axis of the core, thus reducing the angle at which it enters the core. The acceptance angle, θ_a, is the maximum angle to the end face at which a light ray can be launched into the fibre. At angles larger than θ_a, the light ray, BB in Figure 3.5, will be refracted at the glass-cladding interface but still continue into the cladding. It may undergo total internal reflection at the cladding-air interface or be lost into the air. If it is totally reflected it will probably be absorbed by the cladding, or be lost through radiation, as it cannot re-enter the core at a sufficiently shallow angle for propagation within the core [*22 - 28*].

3.2.2.3 Numerical Aperture

A more generally used term to describe the input characteristics of an optical fibre is its Numerical Aperture [26]. This relates the refractive indices of the mediums concerned, that is, the core, the cladding and the medium from which the light is injected, usually air. Let the refractive indices be n_{core}, n_{clad}, *and* n_{air}.

Referring to Figure 3.5 and [*22 - 28*], the end face is assumed to be normal to the longitudinal axis of the fibre. At the air - core interface, according to Snell's Law

$$n_{air}\sin\theta_{air} = n_{core}\sin\theta_{core}$$

and at the core - cladding interface

$$n_{core}\sin\theta_{core} = n_{clad}\sin\theta_{clad}$$

If n_{air} is taken as being unity, then

$$\sin\theta_{air} = n_{core}\sin\theta_{core}$$

By the laws of trigonometry, in the right angled triangle ABC

$$\sin\theta_{core} = \cos\alpha$$

and since $\sin^2\alpha + \cos^2\alpha = 1$

$$\sin\theta_{air} = n_{core}\sqrt{(1 - \sin^2\alpha)}$$

In the limiting case for total internal reflection, the value of θ_{air} becomes the acceptance angle, θ_a, and the value of α becomes the critical angle for the core - cladding interface, θ_c.

Thus

$$\sin\theta_a = n_{core}\sqrt{(1 - \sin^2\theta_c)}$$

From previous discussion

$\sin\theta_c = n_{clad} / n_{core}$, yielding

$$\sin\theta_a = \sqrt{(n_{core}^2 - n_{clad}^2)}$$

The numerical aperture (NA) is defined as

$$NA = \sin\theta_a = \sqrt{(n_{core}^2 - n_{clad}^2)}$$

Incident meridional rays subtending an angle θ_i at the end face of a fibre will be propagated if $0 \leq \theta_i \leq \theta_a$.

If the relative refractive index difference, Δ, between the core and cladding is defined by the equation

$\Delta = (n_{core}^2 - n_{clad}^2) / 2n_{core}^2$, which is approximated by

$n_{core} - n_{clad} / n_{core}$ if Δ is much less than 1 (typically 0.01).

The numerical aperture can then be expressed in the form

$$NA \approx n_{core}(2\Delta)$$

This expression for the Numerical Aperture of an optical fibre shows that the ability of a fibre to collect light is independent of its core diameter. It must be remembered that these equations were developed on the ray theory of light, and are not a complete explanation of the behaviour of optical waveguides. These approximations hold well for core diameters greater than 10 µm. For smaller diameters the theory of electromagnetic waves must be employed for a fuller understanding.

3.2.2.4 Skew Rays

Thus far, only propagation of light rays which cross the longitudinal axis has been considered. It is possible for a light ray, not incident on the end face at the centre point, to also be propagated along the fibre. These rays follow a helical path along the fibre and are called Skew Rays. For larger diameter fibres, the skew rays injected into the fibre greatly outnumber the meridional rays, since they have a greater area for possible entry to allow propagation. It can be shown that skew rays tend to travel near the outer region of the core, and are subject to greater scattering from the core into the cladding, at bends in the fibre [*23,26,29*].

To understand the propagation of skew rays along a fibre, a three dimensional model is needed. Figure 3.6 is a representation of such a model. The effects at the core - cladding interface is the same as that for meridional rays, but the path followed is more complex. Skew rays are also more affected by the index grading of the fibre. In a step index fibre, a skew ray will follow a helical path, but in graded index fibres the path tends to be more complex.

The light ray theory analysis of skew rays is therefore complicated, and is dealt with under the electromagnetic theory of optical waveguides. It is worth noting that the point of emergence of skew rays is dependent on the number of reflections that they undergo. For a non-uniform light input, as in an optical image, skew rays tend to cause a random distribution of the light output over the output face of the fibre. This is observed as a blurring of the image that is transmitted by the fibre bundle.

3.3 Electromagnetic Wave Theory Of Optical Waveguides

An electromagnetic wave consists of two oscillating components, an electric field **E** and a magnetic field **H** [*22,26*]. These two fields oscillate in planes that are perpendicular to each other as shown in Figure 3.7. The strengths of these fields is

measured in terms of flux densities, the electric flux density **D** and the magnetic flux density **B**. These quantities have magnitudes and directions, and can therefore be expressed as vectors and manipulated as such. This section presents a primarily qualitative description of the theory of electromagnetic waves, since detailed analysis and derivation of mathematical solutions is beyond the scope of this thesis and has been well documented [*22*]. The basic mathematical concepts of electromagnetic wave propagation are given in Maxwell's equations [*22*] which relate the above quantities in the curl equations

$$\nabla \times E = -\partial \mathbf{B}/\partial t \text{ , and}$$
$$\nabla \times \mathbf{H} = -\partial \mathbf{D}/\partial t$$

and the divergence equations

$$\nabla \cdot \mathbf{D} = 0 \text{ , and}$$
$$\nabla \cdot \mathbf{B} = 0.$$

Proven theory relates the flux densities to the fields in the equations

$$\mathbf{D} = \varepsilon \mathbf{E} \text{ , and}$$
$$\mathbf{B} = \mu \mathbf{H}$$

If a rectangular coordinate system is assumed, any vector in that space can be defined by the combination of the unit vectors ***i, j, k*** in the x, y, z directions respectively. The electric and magnetic fields associated with the wave can then be envisaged to be oscillating in the x and y planes respectively, and the wave to be travelling in the direction of the z-axis. Manipulation of the above equations [*31*] yields an equation of the form

$$\nabla^2 \Psi = (1/v_p)(\partial^2 \Psi/\partial t^2)$$

Ψ represents a component of the electric or magnetic field and v_p is the phase velocity. This equation has a solution of the form

$$\nabla = \nabla_0 \exp j(\omega t - \mathbf{k} \cdot (x,y))$$

where ω is related to the frequency of oscillation, f, by the equation

$$\omega = 2\pi f \text{ [22]}$$

k is the propagation vector that indicates the direction of propagation and the rate of change of phase with distance. The magnitude of **k**, k, is referred to as the propagation constant. (x,y) describes the point at which the calculations are being made. This equation can be separated into two equations, one for each of the two fields involved [*31*].

$\mathbf{E} = \mathbf{E}_0(x,y) \exp j(\omega t - \mathbf{k}\bullet(x,y))$ for the electric field, and

$\mathbf{H} = \mathbf{H}_0(x,y) \exp j(\omega t - \mathbf{k}\bullet(x,y))$ for the magnetic field.

The phase velocity, v_p, defined as being the velocity of propagation of a point of constant phase in the wave, is given by the equation

$$v_p = 1/\sqrt{(\mu\varepsilon)}$$

If the wavelength in a vacuum is denoted by λ, then the magnitude of the propagation vector, k, is given by

$k = 2\pi/\lambda$ and k is referred to as the free space wave number.

The speed of travel of the wave is given by

$$v = f\lambda$$

where v is the speed, f the frequency of light and λ the wavelength [*22*].

Assuming a constant frequency of light, as the speed is increased, the wavelength must decrease. In a region of refractive index n_1, the wavelength will decrease to λ/n_1 if n_1 is greater than the refractive index of a vacuum. The propagation constant will also change to a value given by n_1k. These equations describe mathematically the electromagnetic wave confined by the optical waveguide. The propagation constant, **k**, is analogous to the ray representation of the light transmitted by the fibre, in that it indicates the direction the light wave is travelling. The components of the electromagnetic wave, the electric and magnetic fields, have been shown to be described by equations of the same form, and will therefore be subject to the same effects in the waveguide. This similarity is examined in the description of some of the properties of the electromagnetic wave in the optical waveguide in the following sections.

3.3.1 Propagation Modes

For simplicity, the explanation of propagation modes, phase velocity and group velocity will be given with reference to planar guides of the form shown in Figures 3.5 and 3.9 . This consists of a rectangular layer of a dielectric medium sandwiched between two layers of a medium of slightly lower index. Propagation will then take place in the plane which is perpendicular to the direction of propagation (the z-axis), and includes all three layers of dielectrics. This section links the ray and electromagnetic wave theory of the propagation of optical energy along an optical fibre.

Consider an electromagnetic wave travelling in the direction of the z-axis indicated in Figure 3.8. The associated propagation constant can be resolved into components in the x-direction and the z-direction [*22, 31*]. The value of the component in the z-direction, Φ_z, is given by the equation

$$\Phi_z = n_1 k \cos\theta$$

The component in the x direction is given by the trigonometric relation

$$\Phi_x = n_1 k \sin\theta$$

where θ is the angle between the direction of the propagation constant and the central axis of the waveguide.

This resolution of the propagation constant into its components applies equally to either the electric or magnetic fields. The component in the x-direction will be reflected at the boundary of the two dielectric mediums. This component will undergo a phase change (see section 3.3.1.1). This component will therefore travel transversely across the planar waveguide as the wave propagates along the optical fibre. Interference will occur as this component moves across the waveguide. For different wave vectors travelling along the fibre, there will also be phase differences between them which depend on the distance travelled along the waveguide (see Figure 3.4). When the total phase difference is equal to $2m\pi$ radians, where m is any integer, constructive interference will occur and a standing wave is obtained in the x-direction. Thus only some wave vectors will be supported for transmission along the waveguide. Each value of m corresponds to a **mode**, which represents the field distribution in the x-direction. The field also has the normal periodic variation in the z-direction as a component of a sinusoidal electromagnetic wave travelling in the z-direction. The lowest value of m corresponds to the situation where the field is at its maximum intensity at the centre of the fibre and decreases towards the boundary of the two mediums. The light ray paths are shown in Figure 3.9 for various values of m.

The diagram shows three light ray paths, and one component of the fields associated with these paths will be varying in the x-direction with the component in the z-direction being zero. In the instance when the non-zero field is the electric field the modes are said to be **T**ransverse **E**lectric (TE) and if it is the magnetic field, **T**ransverse **M**agnetic (TM). In a circular cross-section fibre, the electromagnetic wave is also bounded in the y-direction, and the Transverse modes are then described by two integer values, one for each direction. In these fibres, the z-direction component of the field is not always zero, and a hybrid standing wave

format is obtained. This condition corresponds to the propagation of skew rays. The analysis of these hybrid waveforms has been undertaken in [*22*].

3.3.1.1 Effects at The Dielectric Boundary

In previous discussion, it has been assumed that total internal reflection occurs at a point on the boundary of the two dielectric mediums. It was first noted, by Goos and Haenchen [*26*], that a lateral displacement in the z-direction occurs at the interface. The effect is similar to that which would occur if the reflecting plane was further into the medium of lower refractive index, as shown in Figure 3.10 (a). The displacement is small, but results in a phase change which is essential to the support of propagation modes in an optical waveguide.

It was therefore reasoned [*26*] that the electromagnetic field extends into the cladding. This field is known as the **evanescent field**, and decays exponentially into the dielectric medium of lower refractive index, the cladding, as described by Figure 3.10(b). It therefore becomes clear that the cladding is used in the transmission of optical energy. The cladding must be of sufficient thickness to allow the evanescent field to decay, otherwise energy would be lost through the cladding. The evanescent field decays rapidly and so does not require a relatively thick cladding, but does define a minimum thickness of the cladding.

3.4 Transmission Characteristics of Optical Waveguides

The transmission characteristics of optical fibres depend on many factors, which are usually tailored to best suit the intended application of the fibre.

3.4.1 Fibre Types

A fundamental characteristic of optical fibres is the index profile across a cross-section of the fibre. These are usually of a step profile, a graded profile or a 'W' profile [*26,29*] (see Figure 3.11 (a) to (c)). The effect of these profiles are examined in sections 3.4.1.1 to 3.4.1.3. Another important characteristic of optical fibres is the number of modes of propagation that the fibre will support. They are usually classified into multimode fibres which support several modes, or monomode fibres which support only one mode. Monomode fibres are of great importance in telecommunications, since multimode fibres allow several signal paths, each of potentially different lengths, which materialises as a spreading of the signal that was originally transmitted.

The number of modes supported by a fibre depends on several factors, including the diameter of the core, the numerical aperture and the frequency of light to be transmitted. A value, the V number of a fibre, has been defined to enable an estimate the number of modes supported by a fibre [*26*]:

$$V = (2\pi dNA)/\lambda$$

where λ is the wavelength of interest

d is the diameter of the core

and NA is the numerical aperture

The number of modes supported, M_s, is estimated from the relationship

$$M_s = V^2/2$$

The differences in propagation mechanisms are shown in Figure 3.9. When the number of modes propagated is one, only the axial ray is propagated. This essentially gives a monomode fibre independence of the frequency of light to be transmitted, disregarding the other attenuation and dispersion factors which are to be considered later. The simplified explanations presented of multimode and monomode fibres shows that the monomode fibre is ideally suited for transmission of the relatively wide bandwidth of frequencies that make up a colour image, since transmission is less dependent on the frequency to be propagated. The multimode fibre promotes dispersion of each frequency, and supports different numbers of modes for each frequency. Monomode fibres have a smaller core diameter than multimode fibres, typically 5 - 10 µm, which makes them more difficult and therefore expensive to manufacture. Multimode fibres have core diameters of around 50 µm or larger [*26-29*].

3.4.1.1 Step Index

The stepped index fibre profile is shown in Figure 3.11 (a). It has been the object of the above explanations, as a typical optical fibre waveguide. The refractive index of the core is constant across the diameter, and is slightly higher than that of the cladding, which is also constant. Of much greater importance is whether the fibre is monomode or multimode.

3.4.1.2 Graded Index

The graded index fibre has a profile as shown in Figure 3.11 (b). As the light ray, or propagation vector travels towards the cladding, it is refracted gradually, so that it impinges on the interface at a much more shallow angle than it would have done in a medium of uniform index.

It can be shown that the optimum index profile is a near parabola [*26*]. The main advantage of a graded index fibre is reduced dispersion of the transmitted frequencies. This occurs because the electromagnetic waves that travel further from the axis, do so in regions of ever decreasing refractive indices and hence increasing speed, as they move towards the circumference of the core. Thus the light rays with the longest paths, returning to geometric optics, travel at the greatest average speed. The parabolic profile tends to normalise the path length versus speed variation to maintain a constant group velocity.

Skew rays tend to follow exceedingly complex paths in graded index optical fibres, generally of a helical description, as shown in Figure 3.6.

3.4.1.3 W-Type Index

A W profile core has a variation of refractive index as shown in Figure 3.11 (c). These fibres have characteristics that lie between step index and graded index fibres. The core dimensions are those of a multimode fibre. The primary mode of propagation is supported for transmission, while other modes tend to penetrate into the intermediate layer and are lost through the region of higher refractive index surrounding it. The advantages of this method lie in a greater numerical aperture, and attendant light gathering capacity, and the higher bandwidth capabilities of a monomode fibre. This type of fibre is currently the most expensive to produce, but would suit the purpose of this project admirably, as most images do not have the high light intensities of digital telecommunications sources, and thus require as high a numerical aperture as possible, while retaining the bandwidth of a monomode fibre.

3.4.1.4 Plastic Fibres

The above discussions have been concerned primarily with glass fibres. Other materials have been used for the manufacture of optical fibres. The most commonly used alternative materials are plastics, usually a polystyrene core of refractive index of about 1.60, and a methyl methacrylate cladding of refractive index 1.49 [*25,29*]. The main advantages of plastic fibres are their greater physical strength compared to glass fibres, and lower costs. They do however have much greater attenuation characteristics, typically 10 times larger than glass, which restricts them to uses where long lengths are not required. Because of their greater flexibility, they can have core diameters that are substantially larger than glass fibres.

3.4.1.5 Phase and Group Velocity

Any electromagnetic wave will have points of constant phase. An imaginary line joining points of constant phase is called a wavefront , and the speed and direction of travel of this wavefront is described by the propagation vector or equivalent light ray. The speed of this wavefront, the phase velocity, has been defined in section 3.3. This definition holds quite well for a monochromatic light source. In a complex image, however, there is a relatively wide range of frequencies present, and those frequencies in a close range tend to travel along the fibre in a packet or group [*22*]. The image will therefore appear to travel along the fibre with a group velocity v_g, expressed by the equation

$$v_g = \partial\omega/\partial\Phi$$

The group velocity appears to have a linear relationship to the spread of frequencies in the group, $\partial\omega$, providing the spread is small.

3.4.2 Attenuation

The factors affecting the attenuation of the electromagnetic waves will now be briefly examined. Greater detail about any of the following aspects can be obtained from [*22-29*]. These factors can be grouped under the headings of absorption and scattering losses, as well as the losses that occur as the fibre is bent. Each of these factors affects different frequencies to different extents.

3.4.2.1 Absorption Losses

Absorption of electromagnetic energy by any material occurs primarily at the atomic level, since the energy is absorbed as photons [*22*], to excite the electrons into a higher energy band. Thus the amount of absorption that will occur depends on the transmission medium. The glass used for the manufacture of optical fibres is generally made up of the oxides of silicon, sodium, calcium and boron, and tends to have a great absorption in the ultraviolet region. Some infrared absorption also occurs, as theory predicts [*22,26*] that energy is absorbed by the intermolecular bonds in the glass structure. Impurities in the glass, mainly hydroxyl ions from water, also give rise to absorption losses.

3.4.2.2 Scattering Losses

This loss method accounts for the larger part of attenuation by optical fibres. The scattering comes from two sources. The first is not dependent on frequency, and attributed to the nonhomogeneous nature of the glass. The inhomogeneity arises

from imperfect manufacturing methods and impurities in the glass. Variations in stress and strain on the glass structure also cause scattering of the electromagnetic waves.

The other scattering mechanism, Rayleigh scattering, is frequency dependent, and varies as the inverse of l^4 [*22,26*]. It arises as a result of minute density fluctuations in the glass, as glass is a non-crystalline structure. The variations in density are usually caused as the glass is frozen into a solid during manufacture. The natural Brownian motion of the molecules means that they will come to rest in a non-uniform arrangement to create different densities of molecules. When these irregularities approach one tenth of the wavelength to be transmitted [*22,29*], they tend to act as secondary points of radiation for the electromagnetic wave.

It must be noted that some non-linear scattering effects can give rise to optical amplification, but a discussion of these effects are beyond the scope of this thesis and are not of significant interest in this application of optical fibres.

3.4.2.3 Fibre Bend Loss

This loss is associated with the evanescent field in the cladding. As the fibre is bent, the part of the field in the cladding on the outside of the bend has to travel faster than the part in the core. The energy in this field must therefore theoretically travel faster than the speed of light in the cladding at some critical bend radius, and as this impossibility cannot arise, the energy is lost by radiation from the cladding [*26*]. The critical bend radius can be quite small, less than 1 mm, and therefore is not usually a constraining factor. Optical fibre bundles are restricted by more physical constraints before this critical bend radius is approached.

3.5 Properties Of Optical Fibre Bundles

The electromagnetic properties of optical fibre bundles are essentially the same as that for a single fibre, as expected. However, they have some properties, mainly physical, that are of paramount significance in this application.

As introduced in section 1.1, optical fibre bundles can be grouped into two classes: coherent and incoherent bundles. For the transmission of optical images certain properties are required. The intensity of light transmitted must undergo minimal absorption to aid visual perception or to provide sufficient illumination from a light

guide. The numerical aperture of the fibres in the bundle determines the efficiency of the fibre at collecting light from the source.

3.5.1 Physical Properties

The physical properties of the bundle such as its length and flexibility are significant since the primary use of optical fibre bundles is for the transmission of light to and from remote, and often difficult to reach places. The bundle of fibres is usually enclosed in a protective cover for strength and protection. This cover can be a plastic jacket for protection from scratches and to retain maximum flexibility, or a metal coil for better strength but reduced flexibility. The ends of the bundle are usually held rigid in some form of end ferrule as shown in Figure 3.12. These ferrules help in maintaining the spatial arrangement of the fibres, which is of great importance in coherent bundles.

3.5.1.1 Packing Fraction

The shape and size of the fibres in the bundle determine the density of fibres in the bundle. Figure 3.13 (a), (b) and (c) shows three typical packing configurations for fibre bundles. These packing configurations apply more to coherent bundles, since the arrangement of fibres is less important in incoherent bundles and generally tends to be of a random rather than ordered structure.

If the core diameter is a_1 and the cladding diameter is a_2, as shown in the diagram, then the packing fraction, f_p, or ratio of core area to total area is given by the following relationships [*19*]

$f_p = [(\pi/3)\ (a_1/a_2)^2]$ for square packing Figure 3.13 (a)

$f_p = [(\pi/(2\sqrt{3}))\ (a_1/a_2)^2]$ for triangular packing Figure 3.13 (b)

$f_p = (a_1/a_2)^2$ for hexagonal packing Figure 3.13 (c)

As expected, the packing fraction increases as the ratio of core to cladding diameter increases.

3.5.1.2 End Surface Reflection

Another important aspect for consideration is the efficiency of acceptance of light from the source into the fibre. The numerical aperture indicates the fibre's ability to accept light, but some reflection also takes place at the end surface so that not all the light within the acceptance angle is injected into the fibres. This property is called

Fresnel Reflection [*26*], and this reflectance value, R_f, is given by

$$R_f = [(n_{core} - n_{air})/(n_{core} + n_{air})]^2$$

For typical values of refractive indices, $n_{air} = 1$, $n_{core} = 1.60$, just over 5% of the energy is reflected. This reflection occurs at both ends which means that more than 10% of the light energy is lost. This loss can be reduced by applying an antireflection coating. This is a layer of transparent material, one quarter of the wavelength to be dealt with, evaporated onto the end of the bundle. Destructive interference occurs between the light reflected from the two boundaries [*26*]. In practice, either a multilayer coating can be applied to deal with a range of frequencies, or the thickness of the layer is taken to be a quarter wavelength of the central frequency in the range to be transmitted.

3.5.2 Image Quality

The overall quality of a visual image can be expressed in terms of the sharpness, detail, range of colour, and intensity of colours [*9*]. The first two properties are affected by the resolution of the transmission/display medium, and the second two by the spectral transmittance of the transmission medium. In this thesis, the image sources (see chapters 4,5 and 6) will be in grey levels, so the spectral transmittance is not as significant, but is worth consideration.

3.5.2.1 Resolution

The resolution of a visual imaging system is a measure of the system's ability to provide detail about the image. It is difficult property to quantify, but describes the 'distinguishability' of small close objects in the image, and is of a primarily subjective nature. An informal measure of resolution can be obtained by the consideration of an image consisting of equally spaced black bars on a white background. This will be perceived as an alternating set of black and white bars, Figure 3.14. If a line pair is defined as a pair of adjacent black and white bars, and the width of a single bar is *w*, then the resolution R, of the imaging system can be said to be defined by the relationship

$$R = 1/(2w_{min})$$

where w_{min} is the minimum width of bars than can be distinguished accurately [*29*].

Consider a rectangular array of light sensitive elements in which each element has the same aspect ratio as the array and there are no gaps between the elements, as shown in Figure 3.14. The value of w_{min} will be the width of one element in the array.

If the array does contain some gaps, as in a typical optical fibre bundle, the resolution is no longer as great and some other method must be found for quantifying the resolution. Two values have been developed for such a purpose, the spot resolution and the line resolution [*9*].

For a single optical fibre, a spot of light incident on the fibre end face will be either fully or partially transmitted depending on the angle of incidence at which the rays of light in the spot impede on the face, and the acceptance angle of the fibre. Considering the previous explanations of the ducting of light along an optical fibre, the spot of light will be diffused across the output face of the fibre, occupying the entire end face area. This is a result of the different paths followed by the light rays through the fibre and the multiple exit angles at which the light rays emerge from the fibre as explained in section 3.2.2 and [*26,27*].

Consider a bundle of triangular packing, with fibre dimensions a_1 and a_2, as shown in Figure 3.15. Simple trigonometrical manipulation yields the minimum width of a spot of light a_s, for the spot to always fall at least partially on a fibre in the bundle.

Let the centres of the fibres be joined by the equilateral triangle ABC, whose sides are of length of a_2, and the included angle at each vertex is 60°. The largest circular spot that can occupy the region as shown in the diagram, without being partially transmitted will have its centre at the centre of the triangle, O. The distance from O to the centre of any of the fibres, OA, is derived as follows

$$(a_2/2)/OA = \cos 30°$$

$$(a_2/2)/OA = (\sqrt{3})/2$$

giving $OA = a_2/\sqrt{3}$

and the minimum radius of the spot as $(OA - a_1/2)$

and minimum diameter of $(2OA - a_1)$ or in terms of fibre dimensions

$$a_s = ((2a_2/\sqrt{3}) - a_1)$$

This equation gives a measure of the minimum width of a spot of light to ensure that the spot will always fall on a fibre. It can also be shown [*9*] that for two spots to be distinguished they must be separated by the distance $(a_1 + a_2 + a_s)$. These two relationships quantify the spot resolution of a triangular packing arrangement of fibres in a bundle.

Quantification of the line resolution is more involved, but can be visualised if a line is considered to be a linear array of points. Such consideration shows that the minimum width of line is $((\sqrt{3}a_2 / 2) - a_1)$ [*9*] for transmission regardless of position and

orientation. The effects of various orientations can be found in [9]. A minimum spacing of lines, each of width a_l, is also defined by the equation $(a_1 + a_2 + a_l)$.

These relationships apply to a perfect triangular packing of fibres. In practice, this in not fully achieved, but the equations give a fairly good approximation of the limits of real fibre bundles. A few less important factors affect the resolution of a fibre bundle, including defects in the bundle, dark spots and voids. Defects tend to arise from the manufacturing process, and repeated handling while in use causing breakages of fibres in the bundle. The breakages are seen as dark spots since the broken fibre does not transmit any light. Voids are caused by non-uniform packing, leaving gaps in the array. The voids are similar to dark spots, but tend to be irregular and spread over a larger area as a misplaced fibre tends to cause further displacements in its locality.

3.5.2.2 Spectral Transmittance

For uniform transmission of a range of frequencies, a very pure glass must be used to prevent the absorption and dispersion of different frequencies within the fibre. For sufficiently bright images, a high numerical aperture for greater light collection is required. This means that the glass must be of high refractive index, but in general the higher the refractive index, the higher the critical angle, resulting in the need for some design compromise. Since most of the development of optical fibres has been centred on optical communication frequencies, the research and development of fibre bundles with the best spectral response for optical images has thus far been limited.

However, for the applications considered and the work done, a grey scale of image intensities was considered adequate. The spectral response of the components involved will be discussed in the ensuing chapters.

3.6 Manufacture Of Optical Fibre Bundles

There are various methods for the manufacture of optical fibres [25]. The essential requirements of any manufacturing system are some method of producing the core and cladding, and producing the correct size of fibre.

Production of the glass requires high precision to achieve the correct refractive index and maintain low levels of impurity. The silicon oxide required is obtained from other silicon compounds, usually silicon tetrachloride or other silanes [25]. This is heated

by Radio Frequency (RF) induction, with a supply of oxygen to produce the silicon oxide. The oxide, in vapour form, is then deposited on a rotating pedestal and fused to form a rod, or preform, of the same cross-section of refractive indices as is required for the fibre, but of much larger diameter [*25*].

This preform is then heated and drawn out to form the fibre. The thickness of the fibre depends on the softness of the preform as it is being drawn out, and the speed at which it is being drawn out. Checks on the thickness of the fibre are carried out as it is drawn by examining the refraction of light across the fibre, and using the information to adjust the speed of drawing in a closed loop control system. The fibre is drawn onto a rotating drum. This process takes advantage of the fact that the cross sectional relationship of the core and cladding does not change as the fibre is drawn from the preform.

The length of fibre on the drum must then be assembled into bundles. For coherent bundles each length of fibre must be cut and laid out in a precise arrangement, which requires a high degree of mechanical accuracy. This process accounts for most of the great cost difference between coherent and non-coherent fibres. Incoherent fibres are assembled by grouping the required number of fibres and clamping or glueing the two ends to hold the group together. A protective cover is then installed to give strength and a degree of protection to the bundle of fibres.

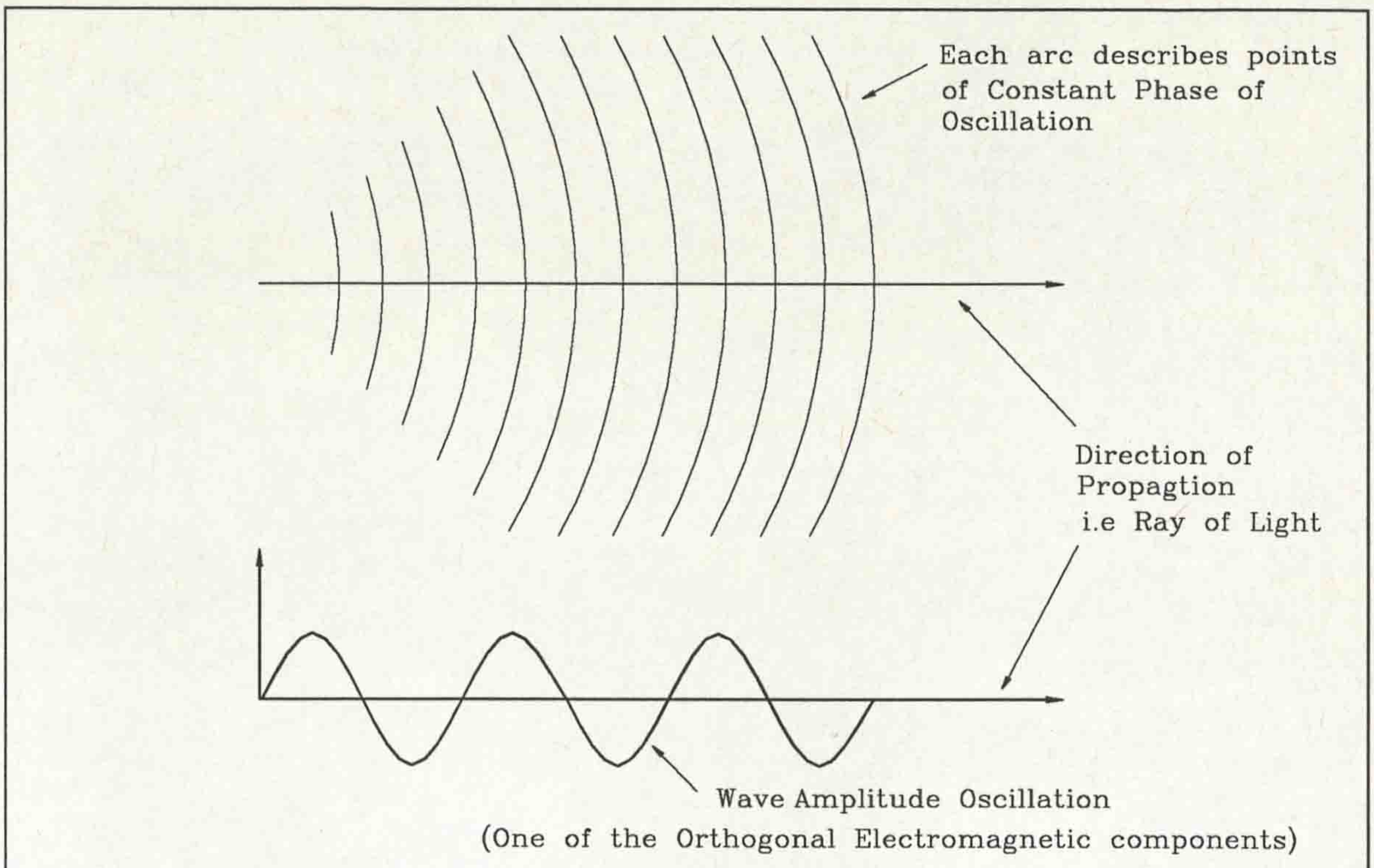


Figure 3.1 – The relationship between a ray of light and the electromagnetic wave

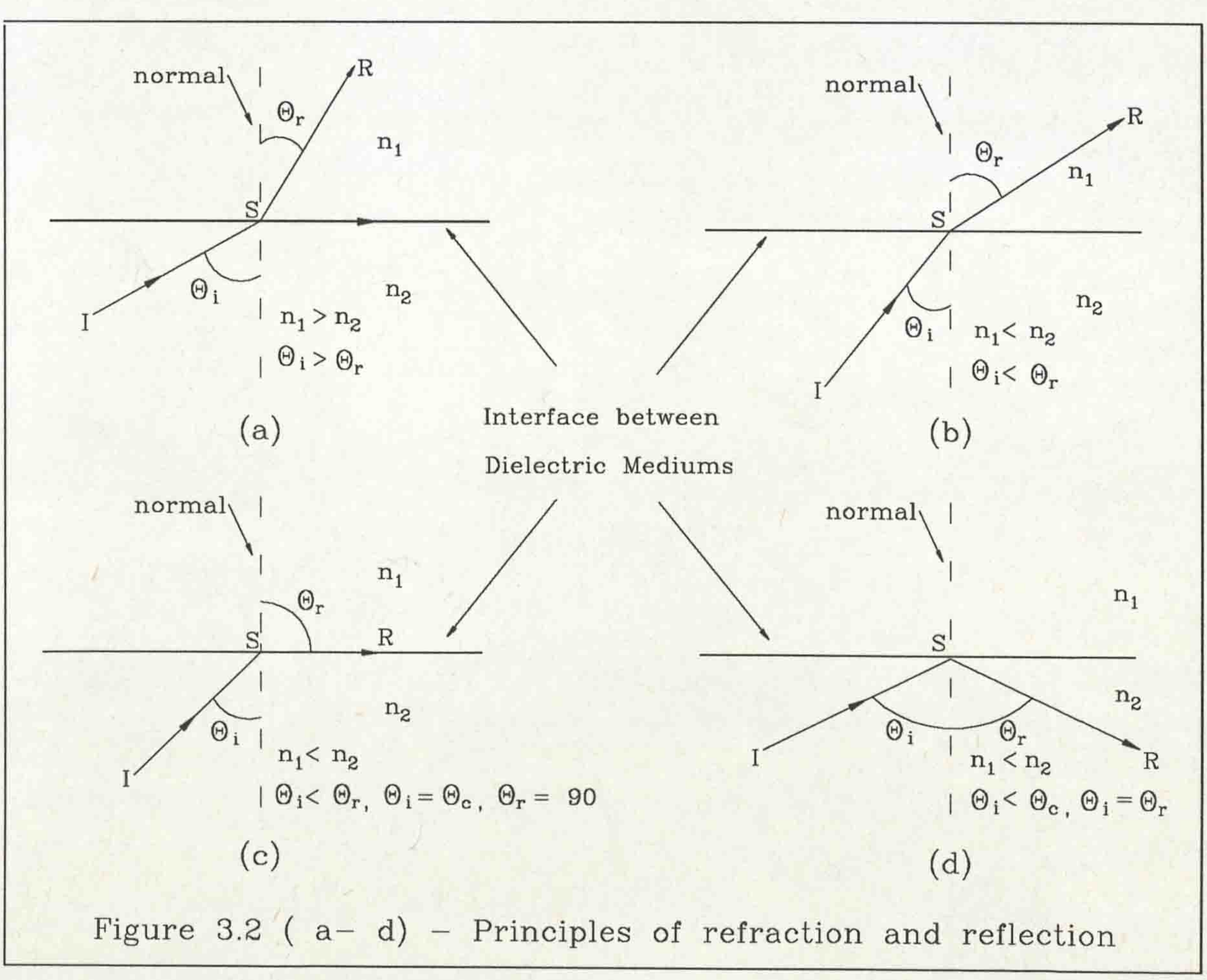


Figure 3.2 (a– d) – Principles of refraction and reflection

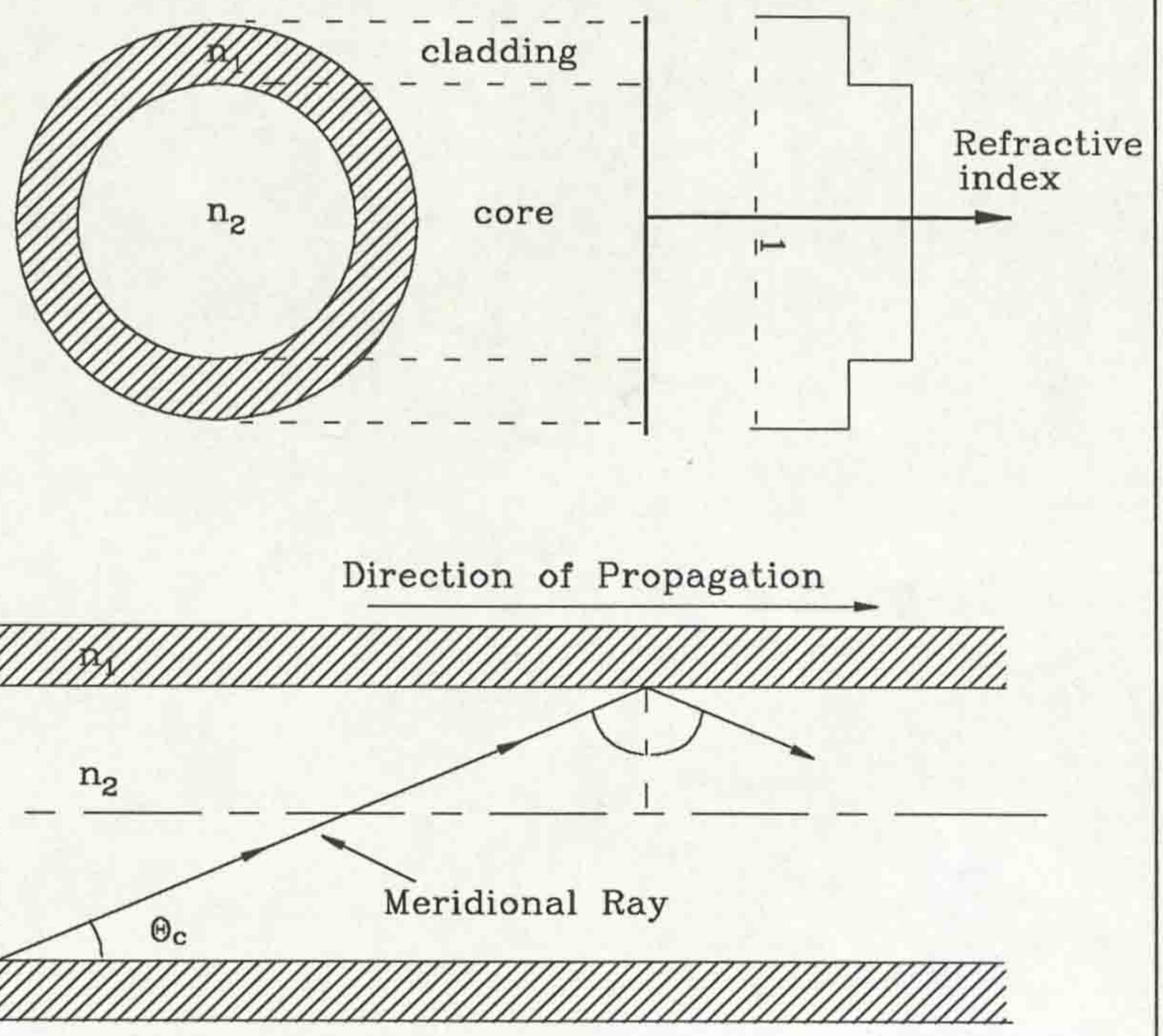


Figure 3.3 – Propagation of a Meridional light ray

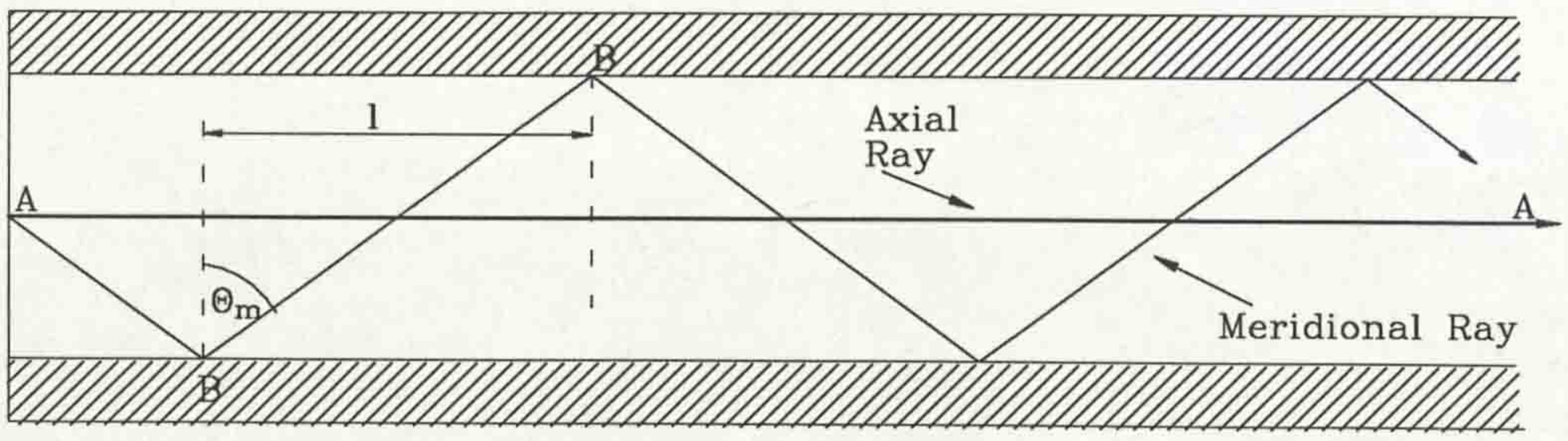


Figure 3.4 – Path length difference between Axial and Meridional ray

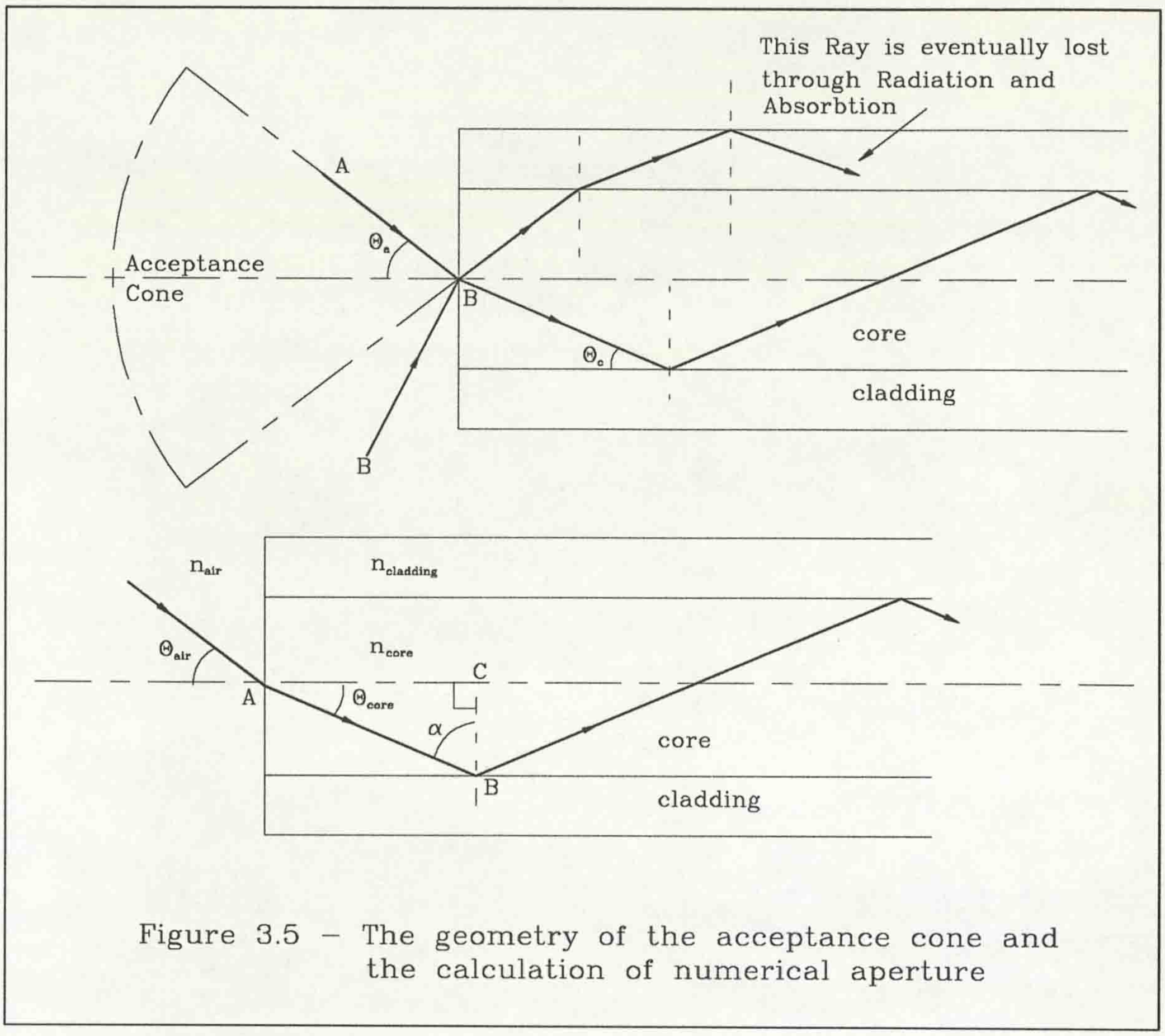


Figure 3.5 – The geometry of the acceptance cone and the calculation of numerical aperture

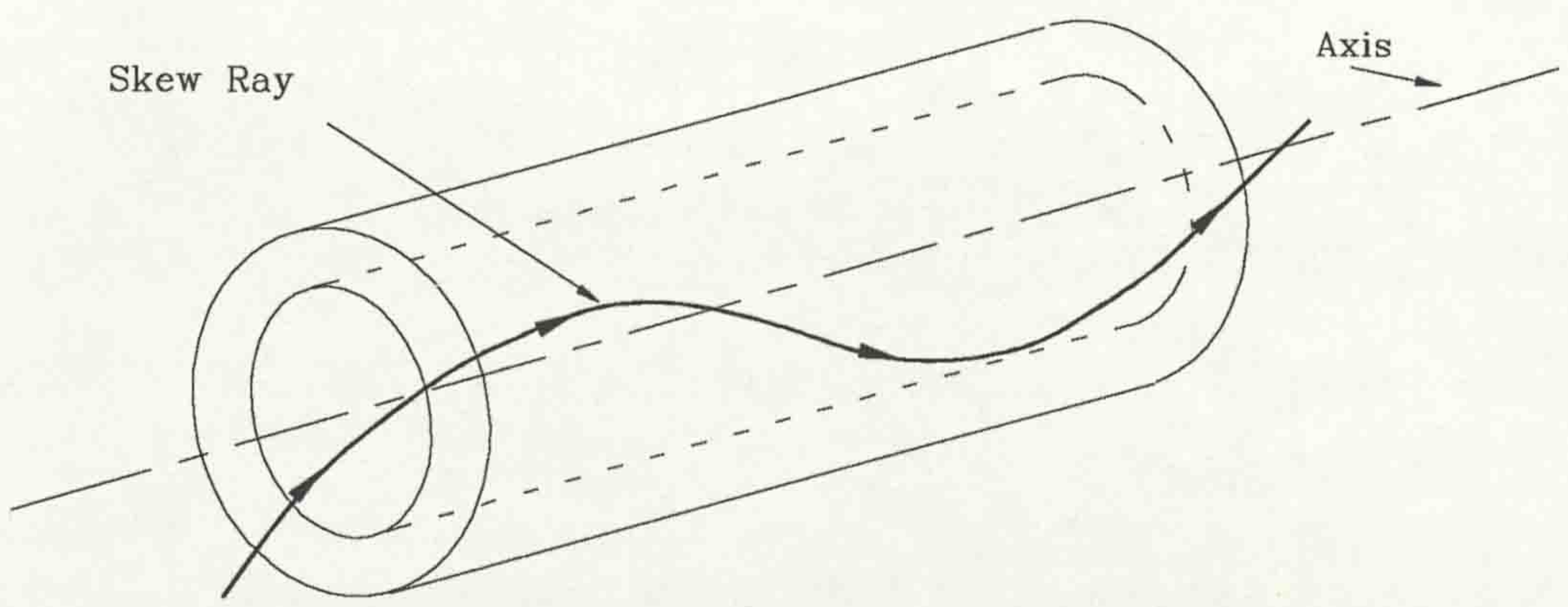


Figure 3.6 – Skew ray propagation through a single fibre

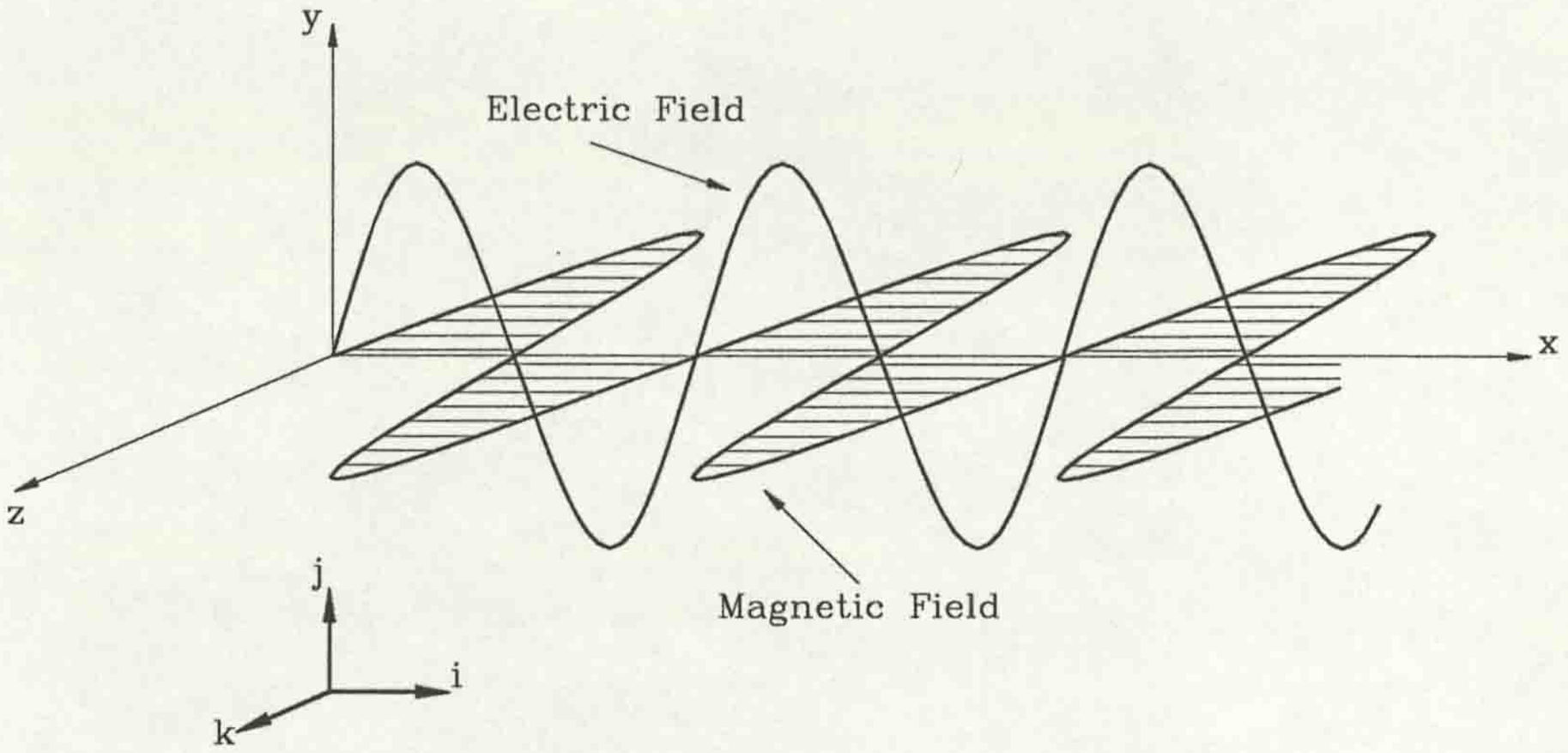


Figure 3.7 – Components of an electromagnetic field

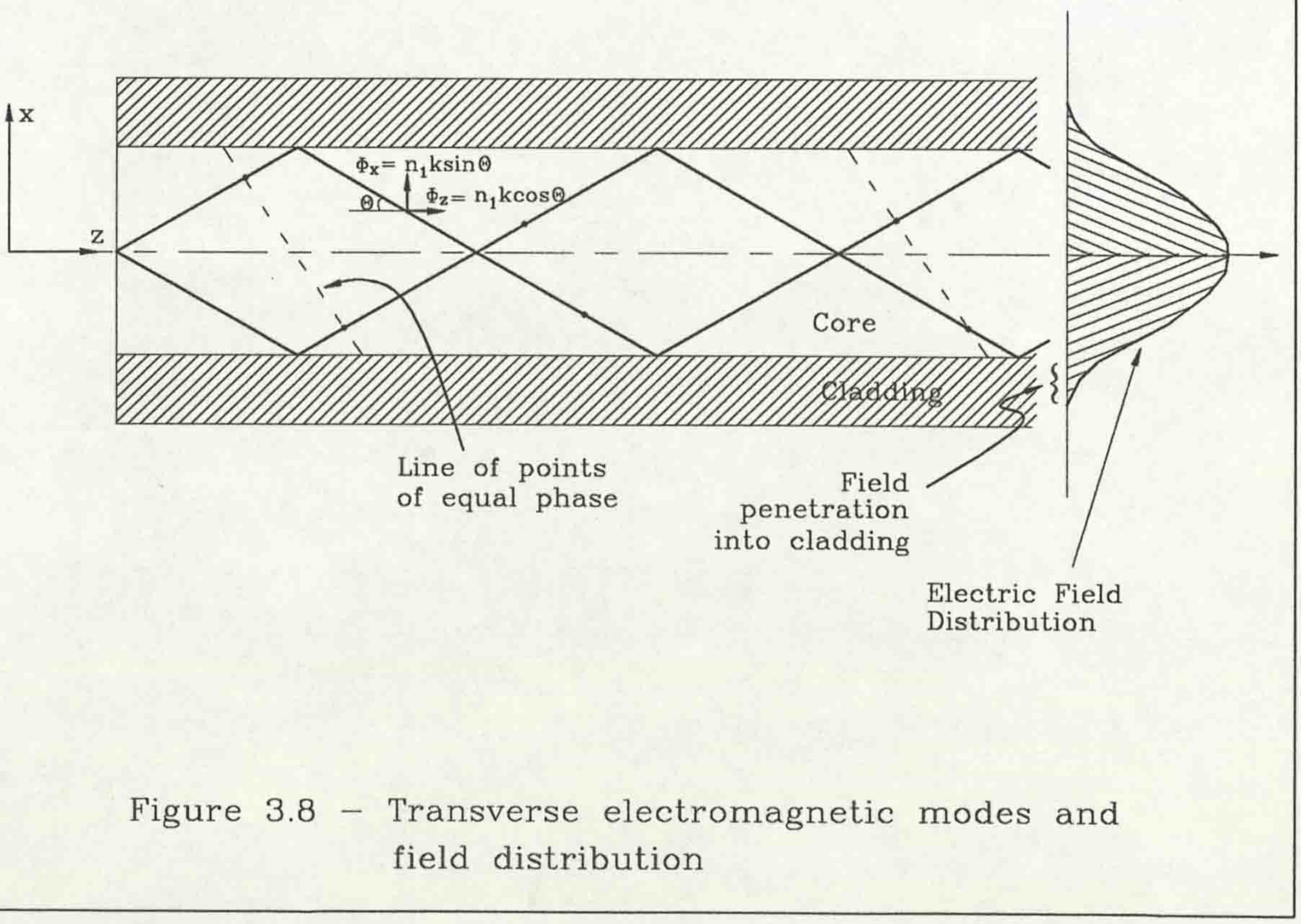


Figure 3.8 – Transverse electromagnetic modes and field distribution

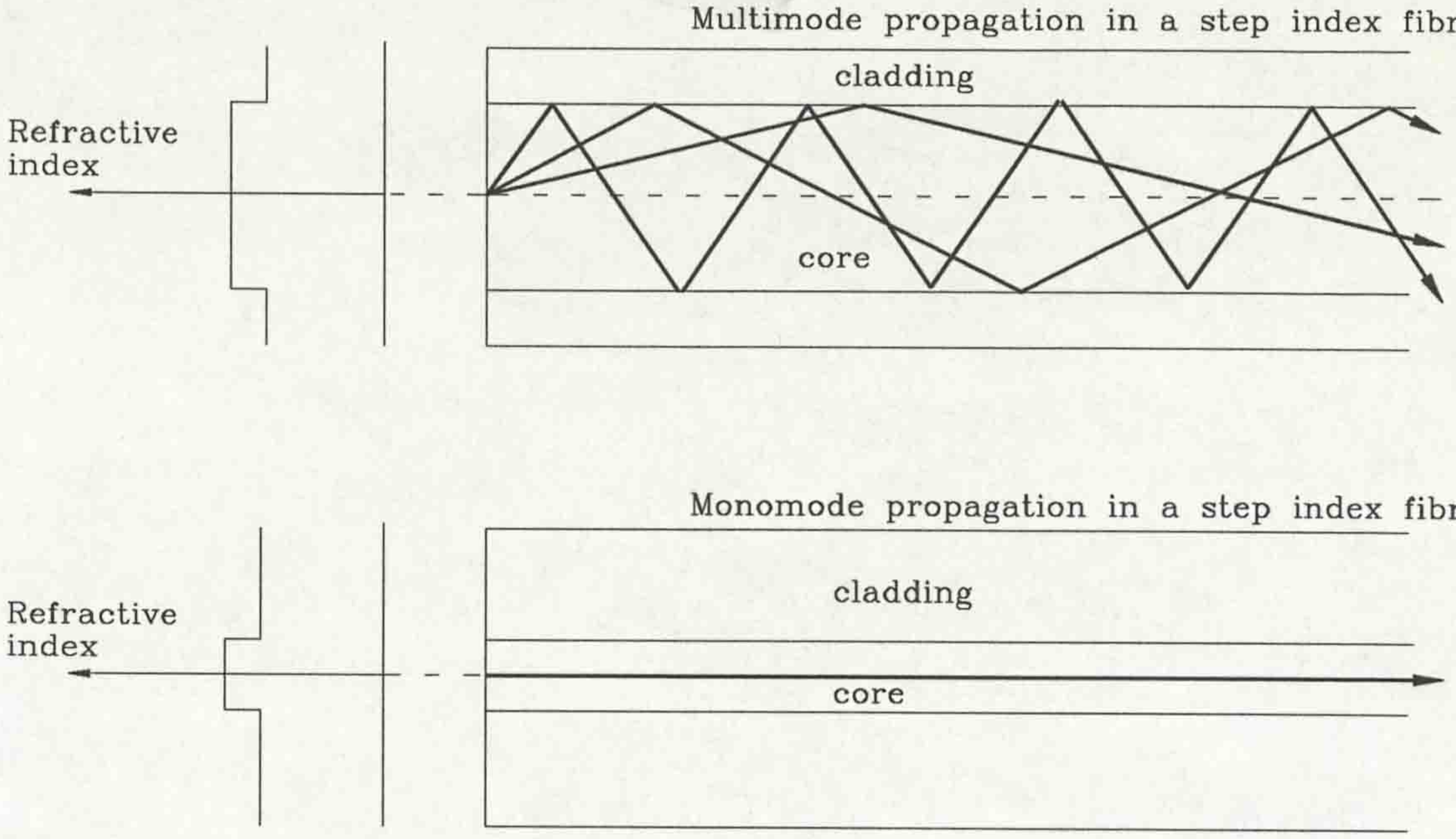


Figure 3.9 – Modes of propagation in an optical fibre

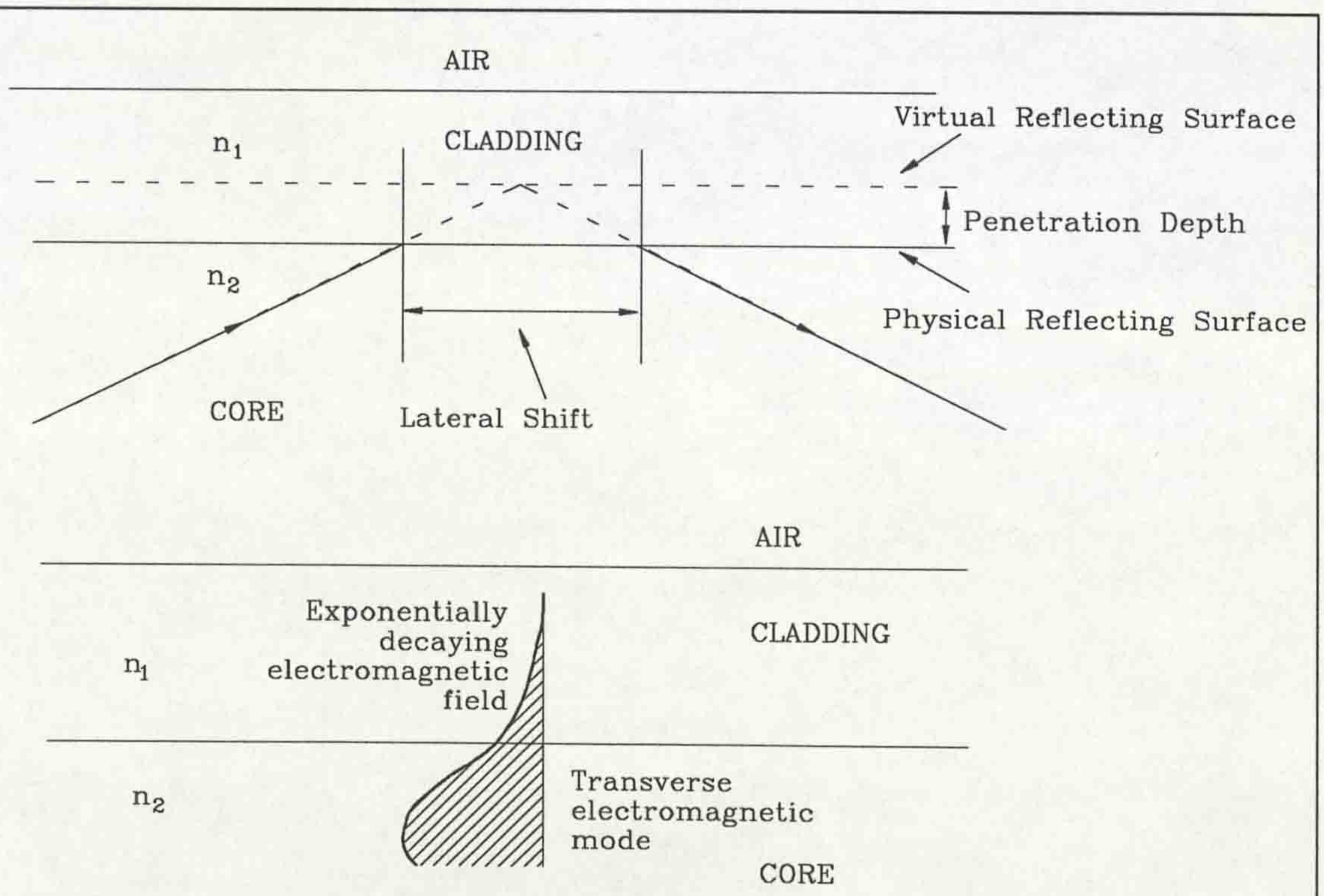
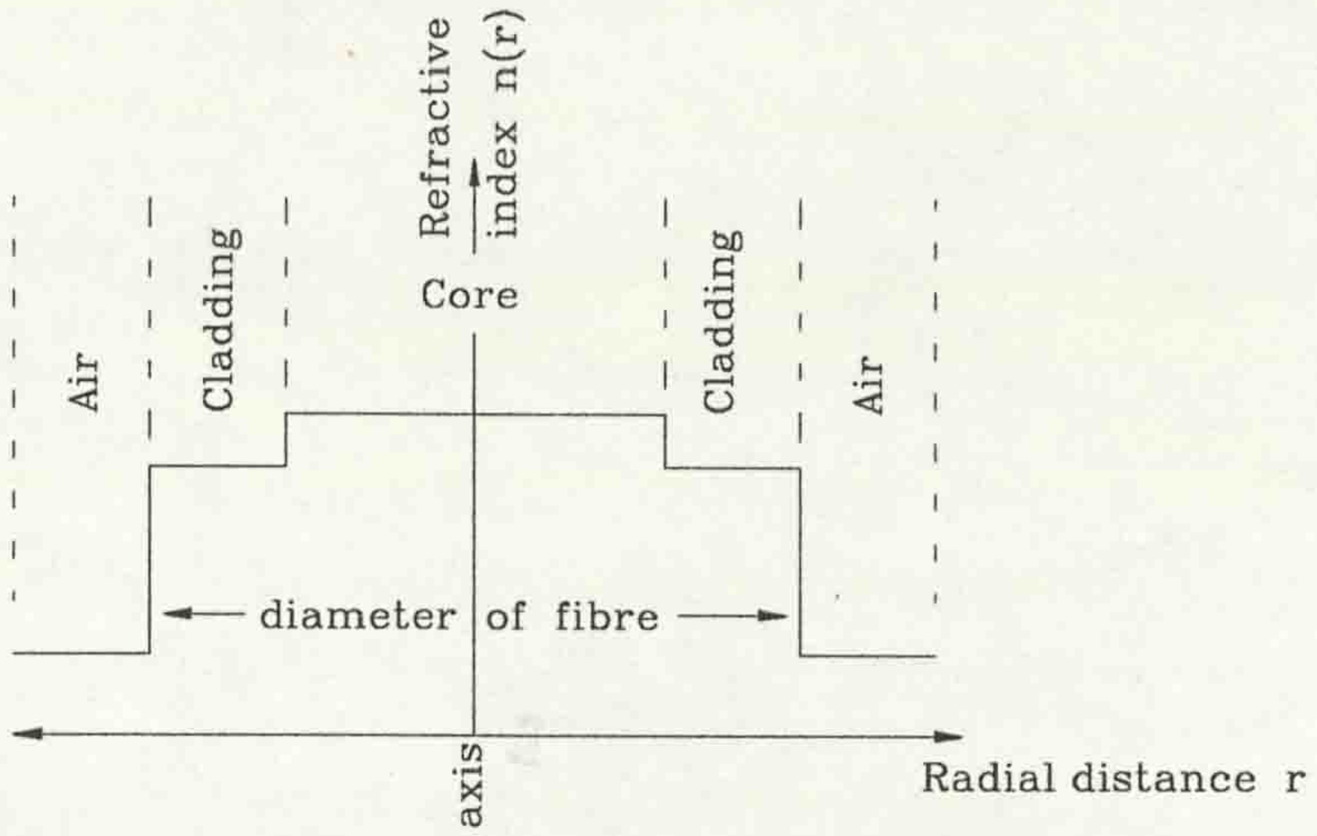
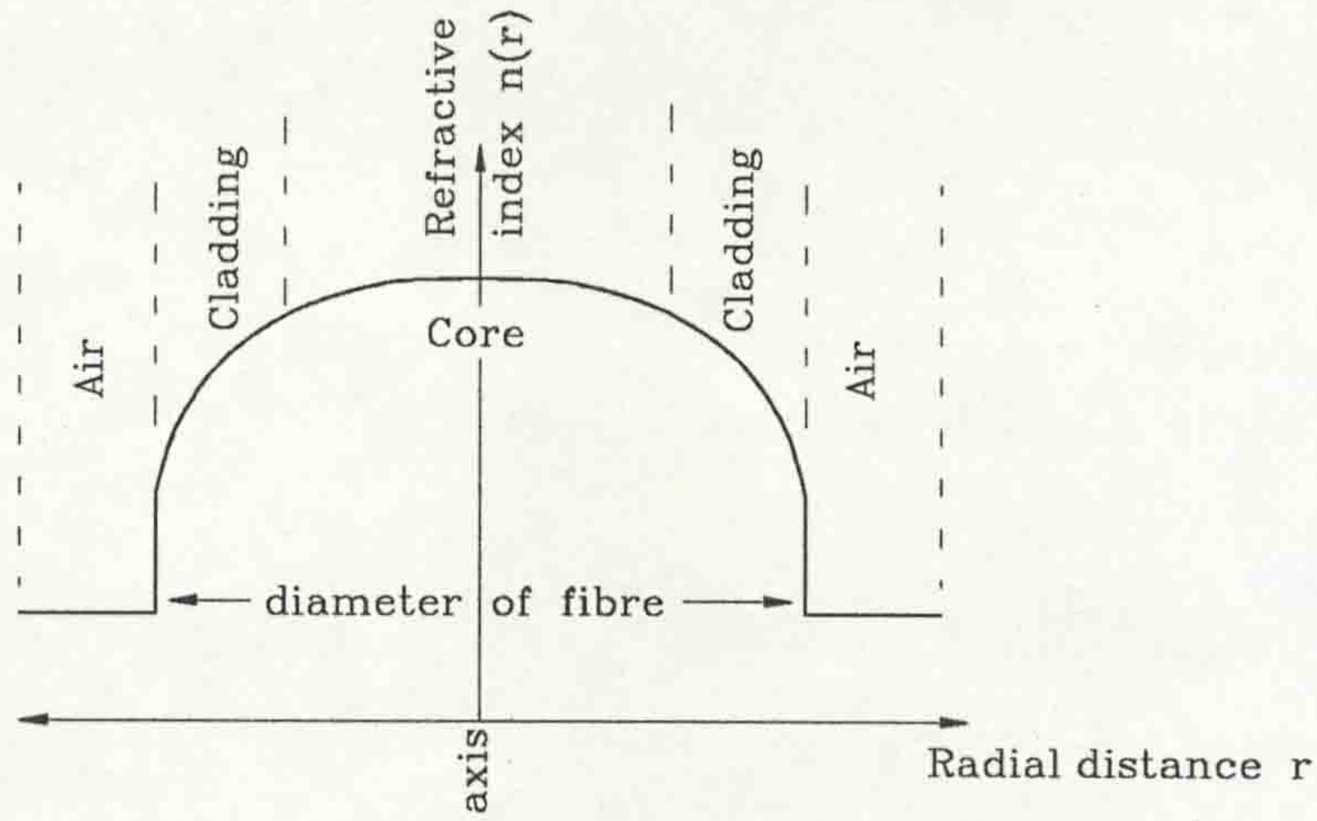


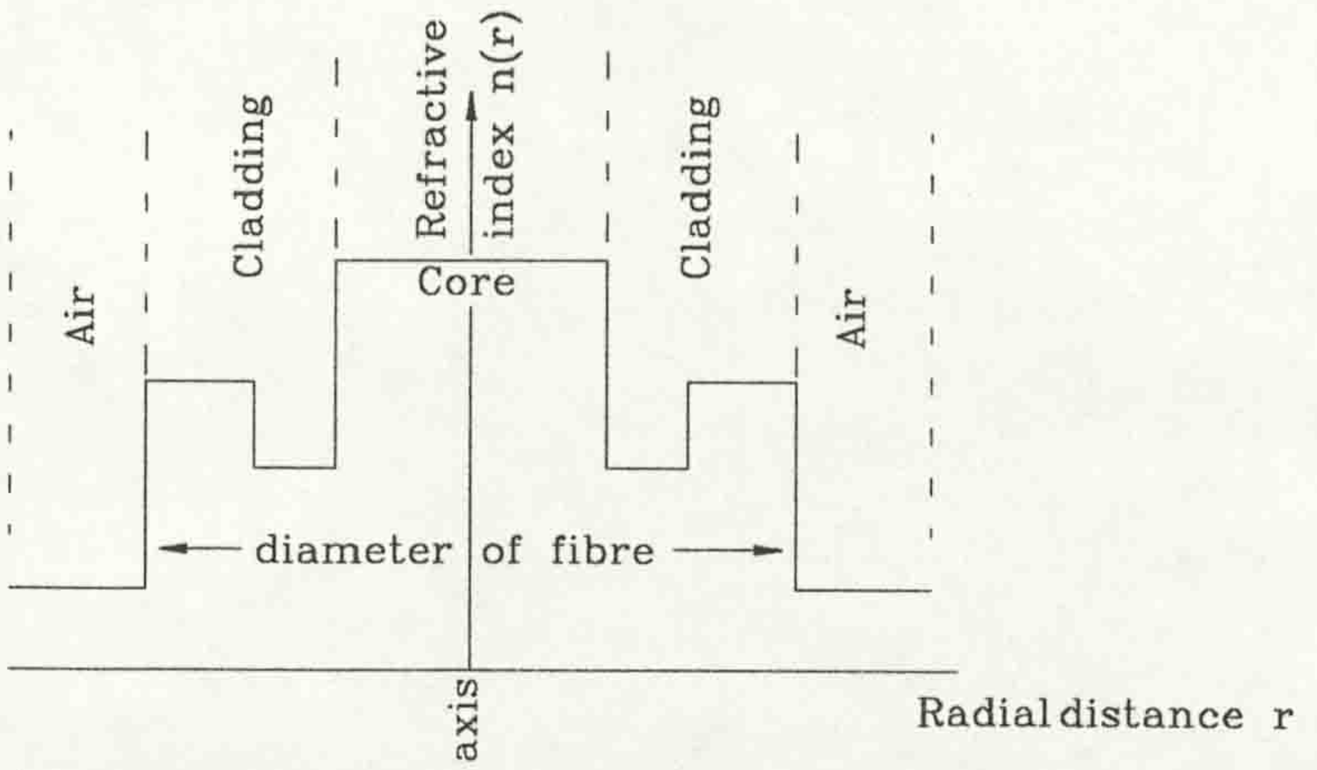
Figure 3.10 (a) and (b) – The Goos–Haenchen shift and the evanescent field



(a) A step index profile for an optical fibre



(b) Graded index profile for an optical fibre



(c) W-type profile for an optical fibre

Figure 3.11 (a) – (c) – Step, Graded and W-type refractive index profiles

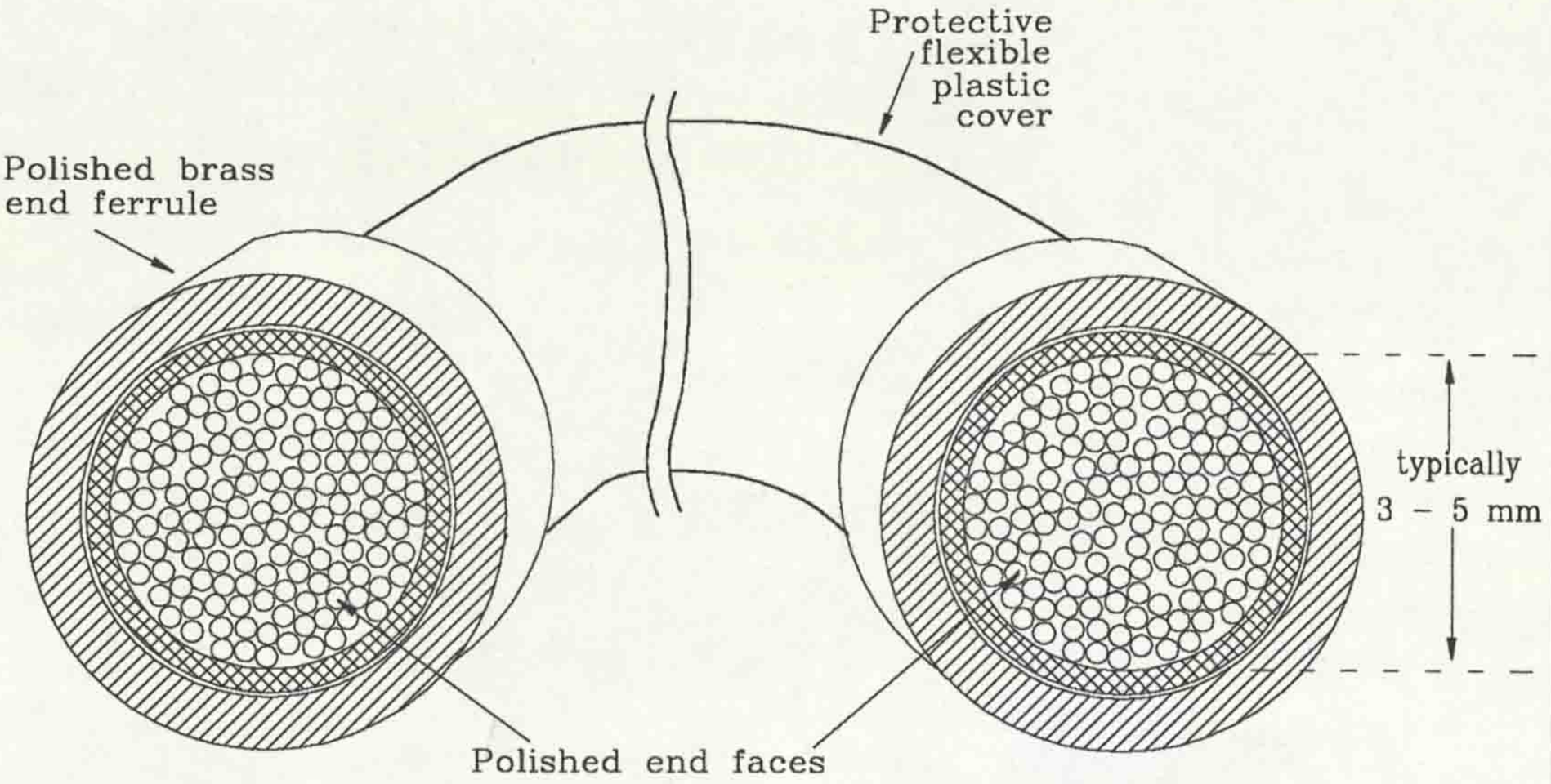
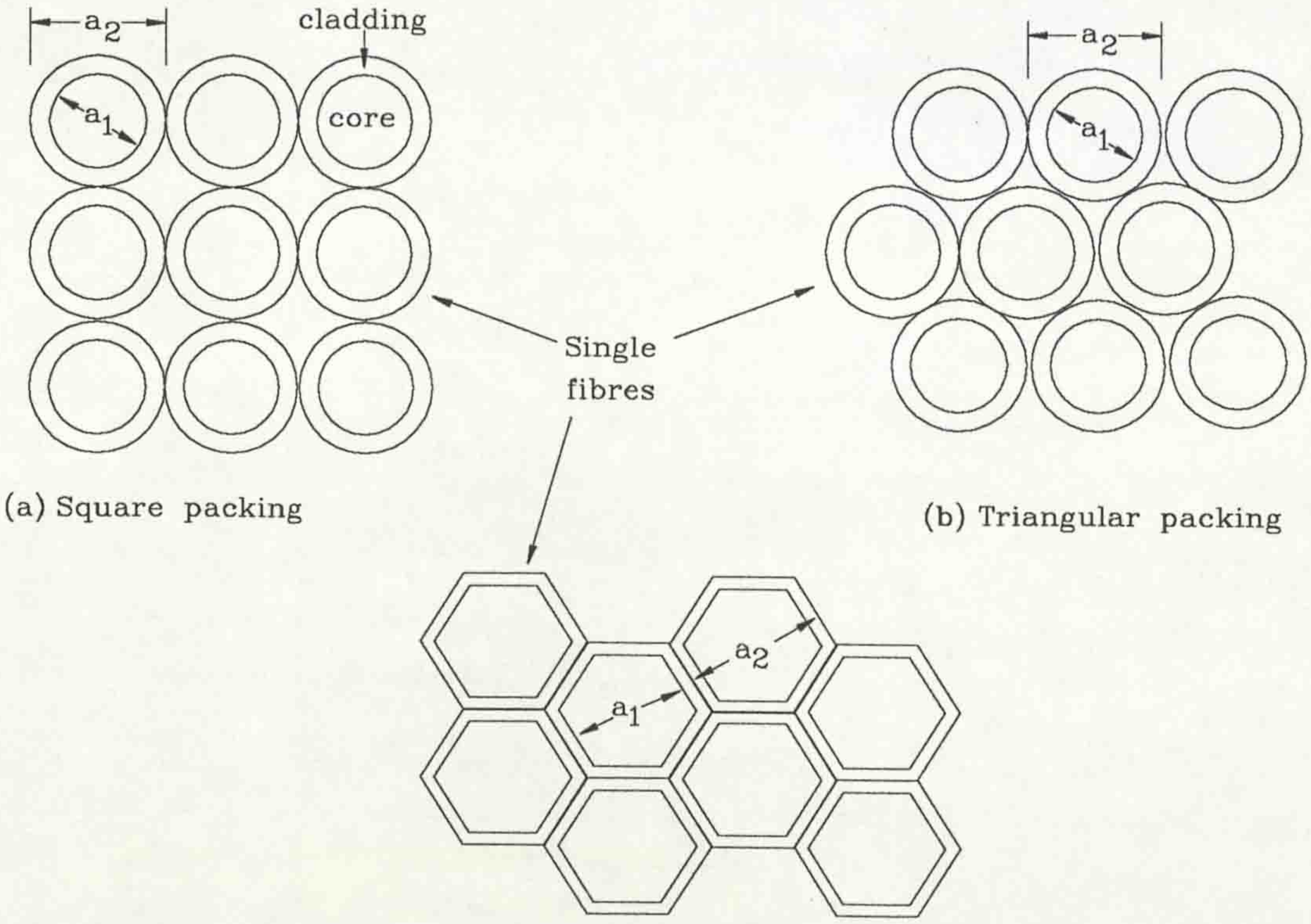


Figure 3.12 – A typical incoherent optical fibre bundle



(a) Square packing

(b) Triangular packing

(c) Hexagonal packing

Figure 3.13 – Three packing arrangements of fibre bundles

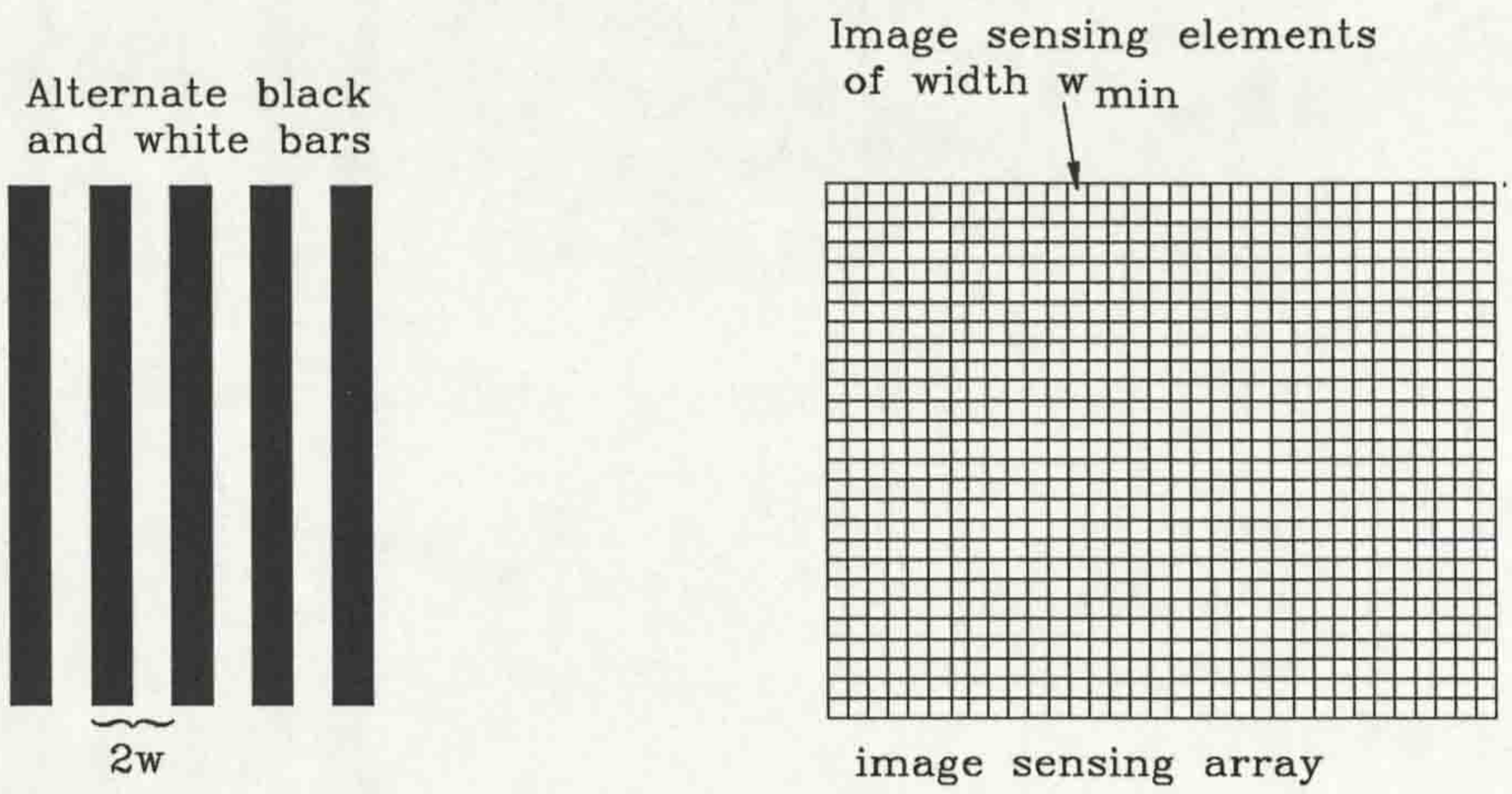


Figure 3.14 – Estimation of image resolution

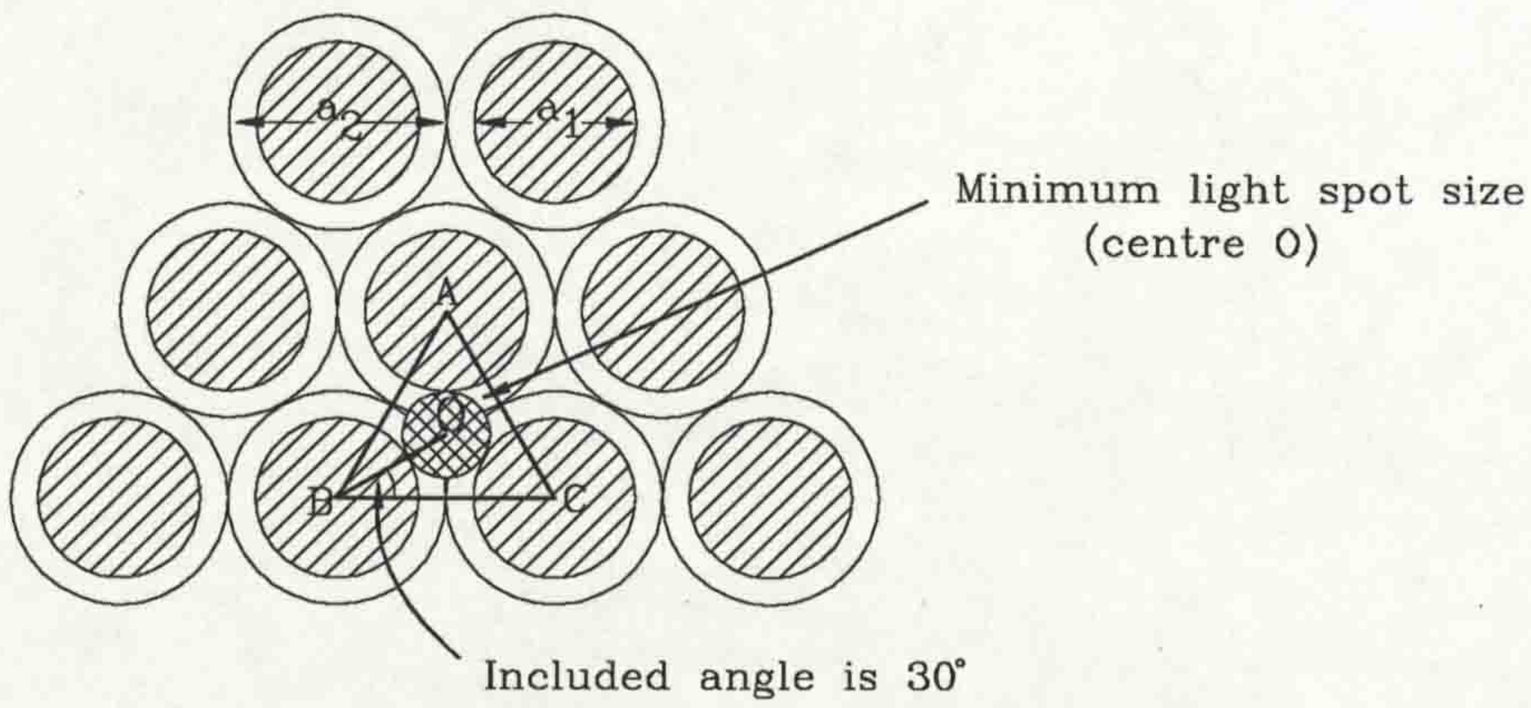


Figure 3.15 – Minimum spot resolution of optical fibre bundle

4. CALIBRATION BY SINGLE FIBRE ILLUMINATION

This chapter describes the first calibration method that was tested. All the development was carried out on standard 'off the shelf' image processing apparatus. As will be seen, this posed some difficulty, particularly in the speed of data transfer between the image analysis and the image storage systems, but did serve to establish the possibility of the use of incoherent optical fibre bundles for the transmission of optical images.

4.1 Principle Of Calibration Method

The essential principle of the calibration method is simple: illuminate each fibre in the bundle in turn, locate the output from that fibre, and knowing the input point and the output point for that particular fibre, build up a table of corresponding input and output points for the entire bundle. The principle is shown in Figure 4.1. As explained in Chapter 3, the input of light at one point into the fibre results in the illumination spreading across the entire output end face of the fibre. In this case therefore, one input point will map onto an output area, assuming that the fibre's output area occupies more than one pixel in the output image. Some method of obtaining a one to one mapping will have to be obtained from this data. The primary reason for testing this calibration routine was to gain some insight into the quality of images that could be obtained from incoherent optical fibres and relevant image processing methods.

4.2 Description Of Apparatus

4.2.1 The optical fibre bundle

The optical fibre bundle used for this experiment was a light guide manufactured by Pilkington Electro-optic Materials Ltd [*21*] . The physical specifications of the bundle are: a length of 1 m, a diameter of 2.5 mm, and an estimated 2200 fibres in the bundle, each of core diameter 50 μm and cladding thickness of 1.5 μm. The estimated figure was arrived at by assuming triangular packing and using the formula described in chapter 3, giving the ratio of core area to end surface area as 0.81. The core area of each fibre is 2×10^{-9} m^2 and the total end surface area is 5×10^{-6} m^2,

yielding a first estimation of the number of fibres in the bundle as 2500. Examination of the end face of the fibre shows that a loose, approximately triangular packing arrangement was used, and the initial estimation was too high. A value of 2200 was therefore settled upon as being a reasonable estimate of the number of fibres.

The length of the bundle of fibres was enclosed in a metal coil sheath for strength, and entirely covered in a plastic cover for added protection. The ends were encapsulated in metal ferrules to maintain the arrangement of fibres in the bundle (Figure 3.12). The intended use of such a fibre bundle was as a light guide, and particular attention was not paid to their optical qualities, in manufacture. The end surfaces were not polished flat, and this would cause non-uniform injection of light from an image source. However, to test the theories proposed, and obtain first round results, they were considered adequate.

4.2.2 Opus PC V

The OPUS PC V is an IBM PC AT compatible personal computer [*32,34*]. It is based around the Intel 80286 microprocessor, with an Intel 80287 mathematics co-processor. The microprocessor has a 16 bit data bus and 24 bit address bus. The width of the address bus limits the system to a physical address space of 16 Megabytes (Mb). To maintain compatibility with previous versions of the IBM PC family, which had an address limit of 1 Mb, the 80286 in this application normally uses only 20 of its address lines, to address the lowest 1 Mb of memory in its address space. To enable full addressing of the memory space, the processor must be put into 'virtual' mode, where the full 24 bits are utilised. The lowest 1 Mb of the memory is divided as shown in Figure 4.2. Memory below the 640 Kilobyte (640 Kb) boundary is normally present, that between the 640 Kb boundary and the 1 Mb boundary is referred to as expanded memory, and above the 1 Mb boundary as extended memory.

There are several memory drivers for addressing the various memory configurations, and the particular machine used was fitted with 640 Kb of base memory and 383 Kb of memory starting at the 1 Mb address. The extended memory was accessed via the Microsoft Ramdrive virtual disk manager [*33*]. The Opus PC V was equipped with a 32 Mb hard disk, two floppy disk drives, and two serial input/output ports that communicate to the RS232 standard. The entire system was operated under the Microsoft Disk Operating System (MS-DOS, version 3.3) [*33*], which provided the basic commands and controlled the filing functions and manipulation of the system's

data. MS-DOS has a memory addressing limit of 640 Kb, hence the need for other memory drivers to access other memory locations.

The intrinsic operation of the system is determined by the Basic Input Output System (BIOS) [*34,35*], which is resident in Read Only Memory (ROM). This is essentially a storage location of software routines that is automatically read on system power up. Among the software in BIOS are the time keeping functions, and memory management software. Some of the functions will be used to address the frame store used.

4.2.3 Data Translation DT2853 Frame Store

The architecture of the DT2853 frame store is shown in Figure 4.3. It has two frame buffers, each of 256 Kb in size and capable of storing a single video frame of resolution 512 pixels by 512 lines. The input analogue video signal is digitized to an 8 bit value, and this value used to address one of eight look up tables of 256 by 8 bit values to allow on-line transformation and scaling of the input pixel values [*7*].

For display of the contents of the frame buffers, the data is read from the specified buffer and supplied to another look up table of 256 values each 24 bits wide. This 24 bit value is made up of three contiguous blocks of 8 bits, each representing a red, green, and blue value, which can be combined to produced a colour output. These values are converted to analogue signals by three separate digital to analogue converters (DACs), which are mixed with the correct video timing signals to generate a display image.

The DT2853 interfaces to the Opus PC V address and data buses, to allow software control of its operation and to facilitate data transfer between the two systems. The two frame buffers form one continuous address space of 512 Kb. This memory area can be mapped onto a location starting at a point above the 1 Mb memory boundary of the Opus PC V. The operation of the DT2853 is controlled by eight 16 bit hardware registers mapped onto the input/output ports of the Opus PC V. The address of these ports is physically set by the insertion or removal of jumpers on the board, which tie the relevant control lines to logic 1 or 0. The base address location of the frame buffers is also controlled by a similar method. The addressing of individual pixels is achieved by reading the value at the offset position from the base, of that pixel.

4.2.3.1 Operating Principles

To operate the DT2853, the board must be first be set up with the right base addresses for the frame buffers and the control registers. The frame buffers are memory mapped into an area above the 1 MB boundary of the OPUS PC V, to avoid contention with the system memory. The exact location is user configurable. The control registers are mapped into input/output (I/O) bus of the OPUS system. The references to all addresses in the following discussion are made in hexadecimal notation for ease of manipulation and demonstration of the principles involved. All the registers must be initialised before usage as there are no default values.

The registers, addresses, functions, and their abbreviated names are shown in Table 4-1. The following is a brief description of the function of each register.

INSCR1: the lower 8 bits are all read/write locations, and the upper 8 bits are all read only. The lower eight bits determine which of the eight input look up tables will be accessed, say whether the frame store is busy or not, and tells the frame store whether to stop at the end of the current operation.

INSCR2: The upper 8 bits are read only, and the lower eight bits are read/write. These lower 8 bits determine which buffer will be accessed. They also mask off some of the data bits to prevent overwriting during write accesses to a frame buffer. Three of these bits set the mode of operation of the frame store. The mode can be one of the following:
FEEDBACK: the output of one buffer is folded back to the input of the other.
VIDEO IN: normal operation where the video data is digitized and stored in a specified frame buffer.
LOAD LUT: where data can be written to the specified look up tables of the frame store.
TRIGGER IN: the operation specified is performed as a trigger signal goes to logic 1, that is on the positive edge of the signal.

OUTCSR: the upper 8 bits inform about the state of the video timing, that is which field is being processed, and whether vertical sync is active. The lower 8 bits control the display operation, turning the display on, selecting the buffer for display and the output look up table, and turning a cursor on or off.

CURSOR: this register contains the pixel and line address of a cross hair cursor. The actual value of the pixel and line locations are obtained by shifting the lower 8

bits up by one for the pixel location and similarly with the upper 8 bits for the line location.

INDEX: the lower 8 bits of this register contain the address in the particular look up table to be accessed.

INLUT: this register contains the value to be written to the input look up table in the location indicated by the value in INDEX.

REDGRN: the upper 8 bits of this register contain the value to be written into the green , and the lower 8 bits the value for the red, output look up tables as indicated by the value in INDEX.

BLUE: the lower eight bits of this register contain data for the blue output look up table.

0	Video Input Control/Status Register 1 (INSCR1)	Controls the video input
2	Video Input Control/Status Register 2 (INCSR2)	Control the video input
4	Video Output Control/Status Register (OUTCSR)	Controls the video output
6	Cursor(CURSOR)	Controls the cursor position
8	Index (INDEX)	Controls the LUT index
A	Input Look Up Table entry (INLUT)	Controls the LUT entry
C	Red and Green output LUT entry (REDGRN)	Entries for the Red/Green LUTs
E	Blue output LUT entry (BLUE)	Entry For the Blue LUT

Table 4-1 - The Offset Address, Name, and Function of The DT2853 registers

For the purposes of this experiment where only grey scale images are involved, only the green output will be used. The input look up table will be selected and initialised to produce a replica of the input video data, that is a uniform and continuous grey scale, as will the output look up table. The cursor is of little use and will be turned

off. The mode will be set to Video in, for normal operation. Having set these values on initialisation, they can be left unchanged during the whole operation. The only interaction with these buffers will be to turn on the capture and display of the frame buffers. The software required to do that is listed in Appendix A.

4.2.4 Oriel Translator System

The calibration method requires knowledge of the location of the input ends of the fibres in the bundle. To achieve this, a sufficiently small light source, to illuminate each fibre in turn, is required, and some method of determining the position of this light source. A pinhole light source, driven by two stepper motors, and manufactured by Oriel Scientific Ltd. [*36,37*], was chosen for this purpose.

4.2.4.1 Stepper Motors and Pinhole Holder

A stepper motor is a device which translates electrical pulses into mechanical movement, as the following describes in simplified form. A permanent magnet is located at the centre point of a cross of four electromagnets [*36*]. By energising pairs of electromagnets, the central permanent magnet can be made to rotate, in either direction, in a series of steps. The arrangement of the electromagnets, and the distribution of the magnetic fields yields a 15° rotation per step.

The stepper motors are arranged so that they lie in the same plane, but at right angles to each other. The motors each wind or unwind a screw the other end of which pushes against the pinhole holder. The arrangement is shown in Figure 4.4. The pinhole holder is a solid disk 4 cm in diameter and 1 cm thick with a slotted, 1 cm diameter, concentric aperture for the pinhole. This is located in another disk, 6 cm in diameter and 1.5 cm thick, which has a central aperture 3 cm in diameter. This disk is U-shaped in cross-section to allow movement of the pinhole holder inside it. There are also two holes in the side, positioned at right angles to each other, to allow the screw from the stepper motors to push against the pinhole holder. At the other end of the diagonal where the screws push against the holder, there are two leaf springs which offer some resistance to the movement of the stepper motor screws. When the screws are unwound, these springs push against the pinhole holder, returning it to a position determined by the new location of the screws. Thus forward and backward motion is achieved in the direction of two perpendicular axes.

The stepper motors each consist of a four phase variable reluctance motor. They are capable of half step operation, and the speed is variable from 20 to 1000 half

steps per second. When coupled to the screw used in this application, the linear resolution is 1 µm over a range of 25 mm. At maximum torque, the spindle exerts a force of 68 N along the axis of travel.

The arrangement proposed for injecting light into the fibre is shown in Figure 4.5. The light source was a high intensity red Light Emitting Diode (LED). To minimise the spread of light from the pinhole before it entered the fibre, the pinhole should be as close as possible to the fibre end face, and the light source as far as practically possible from the other side of the pinhole. This was in an effort to ensure that the light rays would not be diverging to a great degree as they exited the pinhole and entered the fibre. The spread of light over the end face of the bundle would therefore be minimised, making the illumination of single fibres easier. The cone of light would also be more likely to fall within the numerical aperture of the fibres in the bundle making for a greater efficiency of injection of light and therefore brighter illumination of the fibre. A range of pinhole sizes are available for the system, and a 10 µm unit was chosen for this application for the reasons described in section 3.5.2.1.

4.2.4.2 Stepper Motor Driver

The stepper motor driver provides the pulses that drive the stepper motors. The Oriel 20010 is a two motor driver which receives commands from a host computer and translates these to the correct series of pulses to drive the motors. The interface to the host is made by a RS232 data link connection as described in section 4.2.4.2.1.

The driver accepts a series of characters which form the command, performs the required operation and/or communicates a result back to host. The commands have a special protocol which is explained below for the relevant commands. Further detail can be obtained from [*37*]. The commands can be grouped into translator, data/status, control data, and direct control commands. The most used commands are those which enable and disable the motors, and request travel for a specified number of steps or half steps. The driver has two pairs of on board digital registers that store the number of steps travelled by, and the absolute location in steps of each motor. These registers can be read from or written to. The unit operates in one of two modes: terminal or computer mode. In terminal mode every character sent to the unit is echoed back to the source, along with a character sequence that indicates completion of the command. In computer mode, only the character sequence is sent to the source to indicate the completion of the command.

A typical sequence of commands, after power up, would be:

1. Clear the location registers
2. Clear the step counter registers
3. Enable the motors
4. Issue the stepping commands
5. Wait for the command completed sequence of characters.
6. Disable the motors.

There are two variations of the stepping command, T and G. T is incremental, that is, increases or decreases the number of steps in the registers by a specified value. G is an absolute command which directs the motors to a specified count of steps in the registers. In this experiment, to aid repeatability, an origin was predetermined, and the motors were always parked at that position at the end of each test. The origin was taken to be the centre of the range of travel. Thus on resumption of the experiment, the absolute location registers must always be clear. These stepping commands are followed by a direction indicator and a number up to eight digits long, that specify the number steps to be taken. All commands are ended by a comma. A list of the commands is given is Table 4.2.

COMMAND	DESCRIPTION
D	Disables the motor winding current
E	Enables the motor winding current
F	sets Full step mode
H	sets Half step mode
+	Clockwise operation
-	Anti-clockwise operation
I	Inquire about the translator status
Q	Query the status of a move
X	Absolute number of steps for motor A
Y	Absolute number of steps for motor B
C	Clear the absolute registers
G	Go to an absolute destination
R	set the step Rate
T	set the Travel register
A	data/command to motor A
B	data/command to motor B
=	start both motors
@	stop both motors

Table 4.2 - The Oriel Controller commands

The entire assembly, that is, the stepper motors, driver, pinhole holder and the camera system is shown in Plate 4.1

4.2.4.2.1 RS232 Interface

The RS232 interface is an electrical standard for the transmission of digital data over relatively long distances (up to a few hundred metres), while offering good noise immunity. It can operate at several predetermined speeds, measured in baud. A rough guide to baud rates is given by the relationship :

1 baud is equivalent to 10 characters per second (CPS) [*37*].

The information is transmitted as a series of characters, each a sequence of bits in the form:

start	information	stop

The information is encoded into ASCII form for transmission. The connections at both ends are usually made through 25-way D-type connectors, and for this particular application the relevant connections are shown in Figure 4.6.

It was proposed that the Oriel 20010 be under the control of a top level program that determined the time, direction and length of step required by the image analysis algorithms involved in extracting the required data. For this purpose, a special software routine has to be written to enable top level programs to pass data to the serial communication port of the Opus PC V for transmission to the driver, and is listed in Appendix A.

4.2.5 Camera/Lens Imaging System

The output of the fibre must be transformed into an electrical signal for image analysis. To view the entire output area of the fibre, a camera system was chosen. A lens system was required to project the image of the end face of the fibre onto the camera sensing element. Preferably, a method that excludes ambient light from the image would be used. A normal camera lens system works as shown in Figure 4.7 [*2,4,38*]. The resulting image is a reduced version of the original. Generally, to extract the required information from any image requires that the detail be sufficiently apparent, that is, large. This implies that the image of the end face of the fibre must be as large as possible without spilling over the sensing area of the camera. Thus

some form of magnification was required. The proposed solution involved reversing the camera lens to achieve the reverse effect of its intended method of operation. The explanation of this effect is also shown in Figure 4.7. The resulting magnification factor is dependent on the distance of the lens from the camera's imaging system, and the focus is controlled by altering the distance between the end face of the optical fibre bundle and the lens [*3,39*].

A camera in which the light sensing element is a Charge Coupled Device (CCD) was used [*40*]. This type of camera is more sensitive to low light levels and less prone to 'blooming' [*41*], than the vidicon type cameras. The CCD is a semiconductor based device where the elements accumulate charge proportional to the intensity and duration of light falling on them. These charges are then clocked out as an electric current and combined with the necessary video timing signals to provide a video image. The chosen camera, a Pulnix TM-460 [*42*], has a CCD resolution of 512 pixels and 512 lines. This was considered adequate to provide a system for measuring the location of the spot corresponding to light output from an individual fibre. The CCD is however susceptible to residual noise. This occurs because the elements are not perfect and do not completely discharge between frames, nor do they respond identically to light stimulus. For most purposes this effect can be ignored as it is not visible. However when only a small portion of the light sensitive area is illuminated, the rest of the image appears as a relatively noisy background. This effect must be dealt with by any image analysis software.

The theory of video image display systems is discused in detail in section 5.1. However, for a complete understanding of the following explanations, a few of the principles concerned are briefly described here. A video frame is one complete still 'snapshot' of the scene under examination. The frame data is updated and displayed 25 times per second. Each frame is made up of two fields of data in an interlaced display. The two fields are called the odd and even fields, and as expected are displayed alternately to make up one video frame. For a camera such as the TM-460, each field is made up of 256 lines to give a total resolution of 512 lines per frame.

Plate 4.2 shows the output from the incoherent optical fibre bundle, when the 10 µm pinhole is used as the source of illumination. Plate 4.3 shows the imaging assembly (camera/lens system/fibre bundle) used for examining the output of the bundle. A frame was designed to hold the lens system rigidly to avoid movement during the calibration procedure.

4.3 Software Design And Calibration Procedures

This section describes the software routines that were necessary to implement the calibration. A listing of the software is provided in Appendix A. All of the software was written in the C programming language [*43*] or 8086/80286 assembler code [*44*]. These two languages are easily interfaced to each other and were used for driving the Oriel stepper motors, accessing the DT2853 frame store, and the image processing. To create executable code, that is, a series of instructions to be performed by the microprocessor, the programs must be compiled to check the protocol and then linked to provide the required machine code instructions [*35*]. The essential calibration method was to start at the top of a fibre, and scan the surface in steps as shown in Figure 4.8. At each step, the driver was interrogated to ascertain whether the full step distance had been reached, a frame was then captured, analysed to determine the location of any spot or spots, and the information stored in file for post-analysis.

4.3.1 The Light Source Driver

As described in section 4.2.4.1, the pinhole could be driven in two orthogonal directions in one micron steps. It will be seen in the following section, that the analysis of each image is relatively time consuming. It was therefore necessary to reduce the number of frames to be searched. Referring to Figure 4.8, it is obvious that the minimum search area is a circle covering the entire face of the fibre bundle. Software was written to incorporate a function which calculated the horizontal limits of the bundle at each vertical step, knowing the nominal diameter of the bundle. A step size had to be determined which would allow every fibre to be located, but to minimise the number of searches that had to be implemented.

To obtain some idea of the distances involved, small areas of the fibre were examined by manually stepping the pinhole, via a keyboard to driver software interface, over the area. The resulting images were captured and test measurements were made to establish the number of fibres that were illuminated, the spot size, intensity spread over the spot, and typical distances between the spots of light. It was observed that the fibres were loosely packed in an approximation to triangular packing. It was also noted that the typical distance between fibre centres was 60 µm. This measurement however, was subject to great variation between a minimum value of 35 µm and 100 µm. It was estimated that a 33% reduction in the typical value would allow adequate oversampling to compensate for the imperfect

alignment of the light source and the fibres, and therefore a step size of 40 μm was decided upon in both the x and y directions.

An initialisation of the stepper motors was required, since the number of steps in each direction would provide the x and y coordinates of the input points. The motors were driven to the both limits of each axis to obtain the maximum travel in the x and y directions. Half this distance was then traversed from one end of each axis, which would locate the centre of the pinhole holder approximately in the centre of the fibre bundle. The bundle was inserted into the holder, all the counters and registers of the Oriel driver cleared. The light source was then driven, from the keyboard, across the face of the fibre until it just went off the end of the face in both directions of each axis. To check that each of these locations was the limit of the fibre bundle, a small search was conducted perpendicular to the axis, at the limit as shown in Figure 4.8. If more fibres were found then the limits of the fibre were redefined, and the original centre point modified accordingly. This produced an accurate measure of the diameter of the fibre, and its centre point. The light source was then returned to the centre position and all counters cleared.

4.3.2 Data Transfer

Before the image could be analysed, it had to be made available to the OPUS PC V system. As described previously, the frame store was mapped above the 1 Mb boundary. On invoking the executable code the necessary memory area requirements are allocated by MS-DOS. This makes it necessary to make available the data from the image in a memory location somewhere in the lowest 640 Kb area for access by the programs. To facilitate this data transfer a BIOS routine has been provided. This routine causes an interrupt (INT 93) which halts the processor operation, puts it into virtual mode, copies the contents of the memory location above the 1 MB boundary to a new location below 640 Kb, and then returns the processor to the next instruction in the program [*34*]. This routine must be provided with the base address of the memory block to be copied, and the base address of the location to which it is to be copied. The size of the memory block to be copied, from 1 byte up to a limit of 32 Kb per transfer, must also be supplied. Several registers and flags must be stored before the interrupt is generated, and they must be restored before operation of the processor can be continued from the correct point in the program. A listing of this program is provided in Appendix A.

The overhead time, that is, the time to stop and restart the processor at the relevant point in the main program, is the same independent of the number of bytes copied. It

was therefore determined that for the maximum efficiency, the largest blocks of 32 Kb of data should be copied for each access to the frame store whenever the entire frame of data was required. The entire frame could therefore be obtained by eight sequential accesses, each with a base memory location 32 Kb above the last, to read the entire 256 Kb of each frame.

If individual pixels were required, then they could be accessed by specifying a block size of 1 byte. To gain an idea of the time required to move one frame of data to the system memory a program was written which started a clock, copied a frame of data one hundred times to the same location in the system memory, and then stopped the clock. This yielded an average time for the transfer of one frame of 2.5 seconds. The implications of this value are discussed later in this chapter.

Other software routines were written to initialise the frame store, control the capture and display operation, and to access lines and columns of data in the frame as well as any 32 Kb block. This data had to be transferred to a location in the system software, and to ensure that this data did not overwrite any existing data that was being used by the system, a request for the required memory area has to be made to MS-DOS. A program, which returned an address value of the base of the area granted was written to perform this function. The memory allocation also had to be released on completion of the program, to enable future use. Having copied the data to a system memory location, access speeds were determined purely by the speed of the processor and program efficiency. All the programs accessed the data by performing a simple calculation of the new address of the pixel value required, knowing the base location on the system memory.

4.3.3 Feature Extraction

The image expected from the fibre output is a few illuminated fibres for each input point. As explained in section 3.5.2.1, there is a minimum spot resolution. Any spot larger than this value will tend to illuminate more than one fibre in the bundle. Since only one output point is required per input point, a decision will have to be made as to which of these illuminated fibres to select as the best output point. Depending on the magnification factor (see section 4.2.5) of the output area required to fill the camera image, each individual fibre will occupy an approximately circular area of several pixels. The centre point of this area would provide the required output point, as described by Figure 4.9. In this instance the image could be imagined to consist of the required feature and a background. This proposed method poses three main problems: finding the output areas of the fibres, determining the best choice among

those areas found, and finding the centre point of this area. There are various image analysis methods for solving these problems [*45,46,47*]. The relevant features of the analysis methods are discussed below.

4.3.3.1 Histograms

A histogram provides a two dimensional graphical representation of the spread of pixel intensities [*48*]. This information can be utilised to establish further analysis parameters. Consider an image of N pixels per line and M lines, and pixel intensities in the range 0 to *i*. A histogram H, can be expressed as a function of the number of pixels at each intensity, P_i, that is

$$H = f(P_i)$$

and in general terms [*48,49*]

$$H = \sum_m \sum_n \sum_{x=0}^{x=i} P_x$$

This yields a profile of the spread of the number of pixels at each intensity. For an image as expected from this experiment, a typical histogram is expected to have two peaks, one at the value of the background intensity in the image and a much lower one at the intensity of the pixels illuminated by the spot of light. These peaks are expected to be conical and centred at the background and illuminated fibre intensities, since a small spread of intensities is expected [*50,51*]. The region of intensities between these two peaks, often referred to as the valley, ideally should contain no pixels, however practicalities have determined that while the region will not be of null entries, the number of pixels therein would be much lower than the peak values. The analysis of the intensity histogram is discussed further in Chapter 6.

4.3.3.2 Thresholds

The simplest method of differentiating an object from its background is by identifying a pixel intensity value which is between the intensity values in the background and the feature [*52*]. By setting all the intensities above the threshold to a particular value, the maximum brightness for example, and those intensities below the threshold to another value, the minimum intensity for example, a sharp differentiation between the feature and the background could be obtained. In a histogram such as the one described above, this threshold value exists somewhere in the valley, between the two peaks. In general a thresholding method produces a new value of pixel intensity based on the current actual pixel intensity. If the new value is determined solely on the intensity at that point, the threshold method is said to be point dependent. If the

surrounding values are used in determining the new intensity, the method is said to be region dependent. A global thresholding technique applies one threshold to the entire image to determine the new intensities. However in more complex images, the choice of the threshold value becomes more complex [*45*].

P.K. Sahoo et al [*53*] undertake a review of the various methods of chosing a threshold. Among those examined are the p-tile, the mode and various entropic methods. The p-tile and mode methods were deemed most suitable to this application. Briefly the p-tile method chooses a threshold which puts a specified number of pixels above the threshold, while the mode method finds a threshold in the valley.

The p-tile method was rejected on the grounds that different numbers of pixels would be illuminated at each input point, and some information might be lost as the number of illuminated fibres varied and exceeded the specified value. This method could also include background pixels in the feature, if too few fibres were illuminated, as it tried to meet the required number of pixels above the threshold.

The mode method was not suitable for this application as the separation of the peaks was large, and therefore provided a wide possible range of values for the threshold. The choice of threshold by the mode method works best when a narrow band or ideally a point can be determined as the valley between the peaks. The wide range of values in this case meant that the choice of threshold was almost arbitrary, and consistent results would not be obtained.

Examination of the problem lead to the conclusion that a specific method must be developed for this application. A statistical approach was adopted: average values for the background and for the illuminated pixel intensity would be obtained, and a threshold would be chosen that was the median of these two values. This would ensure that the background was always effectively subtracted from the image to be analysed, leaving only the required features. Variations in the background occur because of imperfections in the CCD light sensing element of the camera, and its sensitivity to thermal noise [*40*]. However, an average over twenty-five frames was seen as adequate to obtain a sample of the background. The light source was positioned at various points over the fibre bundle face and the resulting histogram was plotted and examined to obtain the spread of intensities resulting from the fibres' output. The results show that a satisfactory threshold, providing consistent differentiation of the illuminated fibres, could be obtained by this method.

4.3.3.3 Measurements

Having located the illuminated fibres in the output, some method of determining the fibre whose input centre was nearest the centre of the input spot of light must be developed. It was reasoned that the brightest and largest illuminated area would be the required fibre. A value was assigned to each individual area that was a relative measure of the amount of light transmitted by each fibre. This value was the product of the number of pixels in each area and the average intensity of light in that area. It was reasoned that the fibre that should be chosen as the input and output medium would be the brightest and cover the largest area, as it should be transmitting the largest amount of the injected light energy.

To identify the output from a single fibre, the illuminated area must be in one solid area. If there were gaps in the illuminated area, the light could be the output of two adjacent fibres. It is therefore necessary to identify each separate area, and ensure that the identified block of pixels is from one fibre. To find the illuminated pixels, a search must be implemented over the entire frame.

The simplest search method would be to scan every line in the frame from start to end, checking for pixel intensity values above the chosen threshold. The locations found would be stored, and then analysed to decide the number of fibres found. This method was implemented, but found to be relatively time consuming, taking approximately 14 seconds to search one frame. It did however, show that the typical area covered by a single fibre's output, at the chosen magnification, was 30 to 40 pixels in a circular arrangement as in Figure 4.9, which meant that the diameter of this circle was about 7 pixels long. Extrapolating the time to search one frame, and expecting at least one frame search per input point, the following relationship expresses the total analysis time for a bundle of fibres

$$T_{total} = N_f (T_{fa} + T_{ft})$$

where

T_{total} is the total time to calibrate a whole bundle

N_f is the number of sample inputs to cover the bundle

T_{fa} is the frame analysis time

T_{ft} is the time to transfer the frame to the system memory

For a 2.5 mm bundle of fibres, and a 40 µm step size, this equation gives a search time of 17.5 hours, including some sampling beyond the nominal area of the bundle to ensure that all fibres were included. While this value is not beyond consideration for the calibration of a single bundle, it was considered impractical.

T_{ft} is essentially a fixed value, at 2.5 seconds, so the frame analysis time would have to be reduced. Since the fibre output occupied an area of about 35 pixels, it was decided to search every other pixel on every other line to locate an illuminated area, and then conduct a full search around this pixel. This would reduce the area to be searched to very nearly one quarter of the frame. The analysis time would therefore decrease by a similar amount. In practice, this reduced the analysis time to approximately 6 seconds, since some the programming overheads were still incurred. This still represented a significant gain, and meant a total calibration time of 8 hours for a bundle of 2200 fibres.

To ensure that the illuminated areas were not missed by this search method, the search was alternately started on pixel 0 and pixel 1, producing a checker board search pattern.

4.3.4 Data Manipulation

The search pattern produces information about the spots of light found in the image, which was stored in a data file, labelled LUT_A.DAT. This information consisted of the coordinates of the pixels in the spot and the intensity at each point, as well as the input coordinates derived from the number of steps taken by the pinhole drivers. The coordinates, of the form (x,y), represent the pixel number in the line and the line number respectively. A sample section of this data is given below:

Input		Output and intensity		
55	1	230	67	80
		231	67	76
		229	68	85
		230	68	80
		231	68	81
		232	68	77
		230	69	77
		231	69	80
		229	70	77
		230	70	80
56	1	180	120	91
		181	120	90
		182	120	77
		180	121	80
		181	121	85
		182	121	80
		183	121	80
		179	122	77
		180	122	80
		181	122	85
		182	122	85
		183	122	80
		180	123	81
		181	123	80
		182	123	78

Input		Output and intensity		
57	1	80	179	75
		77	180	80
		78	180	81
		79	180	80
		80	180	83
		81	180	82
		79	181	80
		80	181	80
		81	181	81
		79	182	85
		80	182	80

To check whether the spot was one solid block, continuity of the coordinates of the pixels in the spot must be validated. This was simply accomplished by scanning

each line from the first pixel in the spot to the last on that line, and repeating the procedure for each line in the spot. This would also differentiate between adjacent spots of light as long as they were separated by at least one pixel or line. With the spots of light differentiated from each other, the correct spot must be chosen. As explained above, the spot containing the most light energy will be chosen. A weighting method was implemented to compare the spots of light. This method involved summing the intensities of light in the spot. The spot of greatest intensity or covering the largest area would be chosen as the best output.

For the next step in producing the look up table, the centres of the areas needed to be calculated. The mathematical relationship between circles defined on a rectangular array, as in a video display system, is analysed in [*54,55*]. A derivation of the methods described was developed for this application. This was simply done by finding the average of the left and right, and the top and bottom, extremes of the spots of light to give an x and y coordinate respectively, for the centre point, as shown in Figure 4.10 (a). The data from this step was stored in a file labelled CENTRES_A.DAT, a sample section of which is given below. It shows the multiple outputs for some input locations, as for input location `62 1`.

Input		Output centres		Average Intensity	Number of Pixels
60	1	332	391	39	17
60	1	190	418	44	19
60	1	199	427	35	10
61	1	199	426	41	13
62	1	297	335	36	12
62	1	250	364	40	13
62	1	199	425	39	14
63	1	249	363	57	32
64	1	249	363	59	30
64	1	232	448	40	17
65	1	249	364	38	17
65	1	232	448	42	14
67	1	227	429	40	20
68	1	119	292	36	18
68	1	258	357	50	30
68	1	227	429	42	19
69	1	120	291	53	31
69	1	258	357	54	30
70	1	120	291	51	29
72	1	285	352	52	29
73	1	284	351	57	39
73	1	356	361	43	20
74	1	284	351	40	19
74	1	355	361	44	20
76	1	224	433	48	27
77	1	223	433	58	25
77	1	236	445	38	12
78	1	128	381	46	25
78	1	224	433	47	27
78	1	236	444	35	12
79	1	224	272	37	12
79	1	128	380	57	31

It was observed that some adjacent input points produced the same or very nearly the same centre points, as at input positions `62 1`, `63 1`, and `64 1` above, if the fibre was not completely cleared by the pinhole step in either the x or y directions. For complete accuracy in the location of every fibre's single output centre point, this meant that the duplicates would have to be weeded out, again using the same criterion for choosing the best input point/output point relationship. The method of stepping across the face of the fibre produced a linear arrangement of the input

points, each line was scanned to give increasing x values for the y coordinate, and then the y coordinate was incremented. However because of the incoherent arrangement of the fibres, the output points are totally random, which makes the procedure for finding duplicate output points complicated and exceedingly time consuming.

A sorting method was therefore required which arranges the output points into a linear array of increasing y coordinates and as a subset of each y coordinate, increasing x coordinates. The procedures for numerical sorting such as Shell's method and the Heapsort technique, are well documented [*56*] but for the large amounts of data involved, would be very time consuming. It was decided that in view of the limited range of the coordinate values (0 to 511) and with the prior knowledge that the x coordinates are to be sorted within the single y values, a dedicated numerical sorting routine would be most efficient. A straight insertion method was used, where all the y coordinates from 0 to 511 were sequentially found and placed in order in another file, CENTRESY.DAT. The x coordinates were then sorted by the same method within the range of each y coordinate. This data, along with the respective values of total intensity were then written to another file CENTRESX.DAT.

The data from CENTRESX.DAT was then searched for duplicates. Since the centres were not always exactly duplicated, because of misalignments with the fibre centres at the input, a tolerance was specified to determine when two output points indicated the same fibre output. A typical fibre's output was 7 pixels or lines in diameter. The chosen value was arrived at by taking into account the following aspects.

The typical minimum separation of the two centres was 7 units. If the spots are on a horizontal axis, then the units are measured in pixels, and on a vertical axis, in lines. If however, they are placed as shown in Figure 4.10 (b), then the separation *d*, can be expressed in terms of pixels and lines. For simplicity a one to one aspect ratio of pixels to lines was assumed. If *p* was the number of pixels, and *l* was the number of lines then

$$d^2 = l^2 + p^2$$

If *p* is equal to *l*, and *d* is taken to be 7 units, then *d* and *l* have a value of about 5 units.

Integer mathematics were used to calculate the centre points of the circles. This meant that circles whose diameter were an odd number of units would have the centre points displaced by half a unit. For two adjacent such circles, the minimum

displacement could therefore be reduced by one unit to 4. The minimum separation of coordinates in the centres was therefore taken to be 4 units, that is either 4 pixels or 4 lines. To partially compensate for the effect of integer mathematics, the tables were purged of all centre coordinates whose x and y coordinates were within one unit of each other. The results of this purge were written to the file CENTRES1.DAT, which was then reorganised into numerical order as before.

To decide whether two centres were too close, the x coordinates were examined in turn, and compared with all other x coordinates whose y coordinate was within 4 lines of the y coordinate value of the location under examination. If the x coordinates were also within 4 pixels of each other, a duplication was deemed to have occurred, and the spot whose average intensity was greater was chosen as the correct entry. As the entries in CENTRES1.DAT were in numerical order, only a limited search was needed - as the x coordinate values went out of the range of 4 pixels, the search for duplicates of that value could be stopped.

Repeated applications of the duplication removal were implemented as prior removals brought possible duplications within the search range of 7 units. The final result was a table of (x,y) coordinates for the input points and corresponding (x,y) output points. From an initial data set of approximately 6000 points, 2000 pairs of input/output points were obtained that were taken to represent the calibration table for that particular fibre. A sample of this data is shown below.

Input	Output	Input	Output	Input	Output	Input	Output
61 0	297 335	37 65	305 189	47 67	205 276	88 138	336 76
64 0	203 408	39 65	265 76	48 67	311 234	90 138	165 34
66 0	158 371	41 65	374 334	53 67	249 67	94 138	201 40
68 0	257 357	43 65	203 31	57 67	257 90	96 138	199 46
70 0	186 409	47 65	311 234	59 67	256 97	98 138	222 33
72 0	284 351	49 65	338 178	62 67	346 360	101 138	173 21
76 0	224 433	53 65	342 233	66 67	248 441	51 139	311 71
78 0	190 341	55 65	240 89	67 67	330 321	54 139	391 185
80 0	293 424	56 65	264 112	70 67	308 282	59 139	327 79
82 0	253 410	60 65	321 266	76 67	189 270	62 139	347 50
85 0	211 401	62 65	257 427	78 67	186 243	64 139	383 122
86 0	253 374	64 65	282 406	80 67	191 349	67 139	349 56
57 1	219 380	65 65	269 388	84 67	173 193	82 139	362 77
59 1	331 391	67 65	236 408	86 67	155 192	83 139	265 161
60 1	190 418	70 65	240 426	87 67	137 188	90 139	158 32
61 1	199 426	71 65	154 194	88 67	129 192	91 139	164 35
63 1	249 363	74 65	159 230	89 67	107 305	92 139	172 27
67 1	227 429	75 65	160 266	90 67	114 193	93 139	201 40
69 1	120 291	79 65	175 211	91 67	106 306	97 139	222 32
74 1	355 361	80 65	228 199	94 67	124 214	48 140	317 128
79 1	128 380	83 65	122 134	99 67	121 140	53 140	271 30
81 1	293 424	84 65	134 158	102 67	139 105	58 140	332 47
82 1	236 359	86 65	145 243	105 67	84 314	61 140	364 96
83 1	253 410	88 65	121 196	112 67	123 122	66 140	350 56
85 1	253 374	90 65	144 261	115 67	318 271	69 140	258 22
87 1	211 408	92 65	119 104	117 67	281 173	70 140	351 67
89 1	232 369	93 65	130 195	119 67	324 206	72 140	167 28
91 1	210 307	98 65	123 188	120 67	112 232	74 140	174 9
55 2	232 413	100 65	115 127	128 67	313 313	77 140	241 14
57 2	273 395	105 65	89 255	131 67	326 182	79 140	359 96
66 2	228 430	107 65	131 220	137 67	372 340	81 140	362 77
68 2	119 291	108 65	180 256	143 67	128 284	83 140	230 14
71 2	318 411	110 65	354 300	147 67	175 47	85 140	380 128

4.3.5 Image Reconstruction

To reproduce a representation of the original input image to the fibre, the output must be reorganised into the correct order. The process has been labelled as image reconstruction for the purposes of this thesis. The above data manipulation techniques produce a table of entry points and corresponding output points, a look up table (LUT). The list of entry points in this LUT does not represent pixel coordinates but the input coordinates of the pinhole light source. These entry points must be converted to corresponding values for pixels before a reconstruction could be performed. Since the magnification arrangement chosen to view the fibre bundle's output allowed it to cover nearly all of the input area of the camera system, the minimum values of the x and y coordinates were chosen to be the zero pixel address values and the maximum values of x and y, the maximum pixel address values of 512. Since the DT2853 did not display and capture square pixels, but those with an aspect ratio of 4 : 3, x : y, and the input coordinates were on a square grid, this produced an image whose proportions would be distorted by a similar ratio. This was corrected by scaling the x input coordinates by a factor of 3/4 to restore the same proportions as the DT2853 frame store images.

To achieve the image reconstruction, a video frame of the output from the fibre must be captured, and the intensities at the known output points, that is, the centre points of the fibres' outputs, written to the corresponding input location. This is diagrammatically represented in Figure 4.11. Therefore it is theoretically possible, if the corresponding relationship between every input and output pixel is known, for the entire image to be reconstructed from the incoherent output of the fibre. However with the fibre bundles used for this experiment, with approximately 2200 fibres per bundle, a one to one fibre to pixel ratio for the entire image could not be achieved - an image of 512 pixels and 512 lines contains 262,144 pixels. It has been explained that unless a one to one mapping exists between the individual fibres and pixels, a fibre bundle is incapable of resolving the intensities at each pixel location, so that the output from each fibre spreads across several pixels of the video image, illuminating them almost evenly.

Thus, if the known output points in the video image are assumed to be near the centres of the fibres, these points can be reasoned to contain the intensity that is spread over the surrounding area. 2200 fibres therefore allows only a limited resolution of an image. Using only these centre points to reconstruct the image produces an output image that does not fully allow a visual evaluation of the original image, as there is not enough information in the image for the brain to fill in the gaps to create a recognisable image for complex structures, although simple images are

readily recognised as shown in Figure 4.11. This meant that some method of improving the image quality had to be implemented.

4.3.6 Image Enhancement

Image enhancement describes the techniques employed to improve the perceived quality of an image [*45*]. The quality of an image can be judged by the factors of resolution, contrast and sharpness, continuity/smoothness, and the range and resolution of intensities present in the image [*10,45*]. The resolution of the image, in this application, is limited by the density of fibres used to transmit the image, and cannot be changed by image processing methods. Image processing cannot provide detail that was not originally present in the image. The apparent detail can be increased however, by various interpolation methods and these will be discussed below. The range of intensities is determined by the hardware in the image acquisition system, but can be remapped by various arrangements of capture and display look up tables. Generally, the greater the range of intensities and the higher the resolution of these intensities, the greater the perceived detail.

4.3.6.1 Filters

In image processing terms, a filter is an operator that selectively transforms the intensities in the image [*45*]. This principle could be used to spread the pixel values over a wider range for smoothing, and to highlight or suppress certain features in the image as described by [*45,57*]. For example, a high pass filter would accentuate areas where there are rapid changes in intensity. This effect can be understood by examining the Fourier spectrum [*58*] of a periodic waveform. Theory shows that the squarer the waveform, that is the greater the rate of change of amplitude, the higher the frequencies contained in the Fourier spectrum of the waveform [*59,60*]. A high pass filter would therefore indicate areas where these high spatial frequencies are present. Conversely, a low pass filter would tend to attenuate these areas of high spatial frequencies. Visual noise would appear as a significantly different intensity from the surrounding values, and would therefore create a rapid change in intensity, giving rise to high frequency components. In practice, these filters are described by a matrix, usually of dimensions 3 by 3 units. This matrix is convolved with the image to obtain the result (see section 2.1). The matrix for a high pass filter operation is shown in Figure 4.12 (b). The mathematical calculation can be expressed by considering a 3 by 3 area of pixel in the image as shown in Figure 4.12 (a). The output value P_n, for the pixel at the centre of this area is calculated as follows.

$$P_n = \sum_{n=1}^{n=3} \sum_{m=1}^{m=3} (HP_{mn} \times P_{mn})$$

where HP_{mn} are the filter values and

P_{mn} are the pixel values at location mn

This will not alter the intensity of the pixel in question if the pixel is in an area of uniform intensity. Otherwise, the pixel will be highlighted either by reducing its intensity further if it is less bright than the surrounding area, or increasing its intensity if it is brighter.

A commonly used filter is the Laplacian filter [*45*], which is based on a two dimensional second order differentiation. It is used for edge detection and is based on the same principle as the high pass filter. Mathematically, the Laplacian operator is defined by the equation

$$\partial^2 F/(\partial y \partial x) = \partial F/\partial y + \partial F/\partial x$$

where F(x,y) describes the distribution of intensities in the image.

The matrix to perform this operation is shown in Figure 4.12 (c). Its similarity to the high pass filter is obvious, but in areas of uniform intensity, the pixels are set to zero. This results in highlighting the boundaries, or edges, between light and dark areas in the image, that is, an edge detection.

4.3.6.2 Smoothing

Section 4.3.5 showed that the output image after reconstruction contained only 2000 points. To improve the perception of the image represented by these points, an image smoothing algorithm had to be applied. This would involve filling in the areas in the image between the known intensities. A most widely known method of interpolation is the cubic spline technique [*56*]. It has been widely used for one dimensional interpolation and for two dimensional interpolation as described by [*13,61*]. A two pass adaptation was tried for this application, where the interpolation technique was applied twice, once in the x direction and then the y direction, and the average of the two taken to be the best approximation of the pixel intensities. On examining the results, it was noted that this method yielded only a very small variation in the pixel intensities between the known points. The non-mapped points were being filled with the surrounding known pixel intensities. Various other methods have been documented for the purposes of image smoothing [*62,63*], and the principles embodied in those methods were modified to suit this application.

To simplify the process of smoothing the image, a quicker method was developed whereby the blank areas were filled by a series of small spirals centred at the known points. Any pixel of zero intensity was assigned the intensity of its nearest known mapped point. This yielded results almost indistinguishable from the cubic spline interpolation method, and was quicker by a factor of 20. The principle of this filling method is shown in Figure 4.13.

4.4 Calibration System Limitations

Examination of the results obtained by this calibration method (see Chapter 7), shows that the resolution of the system was severely limited by the number of fibres in the bundle. The accuracy of the calibration method was relatively high, with most of the fibres being located and in the right place. The images obtained from the reconstruction were recognisable only if the detail required to recognise the source image was not high, as in black and white images found in large text transmission. The smoothing technique employed greatly improved the overall quality of the image but tended to obscure finer detail.

The only way to improve the inherent quality of the image was to increase the resolution of the fibre bundle used for the transmission of the image. In ideal circumstances, to avoid any detail loss through transmission by a fibre bundle, each pixel in the image would be mapped onto a single fibre in the bundle. For a 512 pixel by 512 line image, over a quarter million fibres would be required. Digital image processing system applications have shown that an image more than adequate for visual inspection can be obtained from an image of 256 pixels by 256 lines, equating to 65536 pixels in total. Some detail resolution is obviously lost in comparison to the 512 by 512 systems, but unless great accuracy is needed, the lower resolution system can be used satisfactorily. It was estimated from the theory of triangular packing, that 65000 50 µm fibres would form a bundle 1.5 cm in diameter. At this size, the major advantage of small size and manoeuvrability of optical fibre bundles over small camera systems would be negated.

Current technology can produce 10 µm single mode fibres for telecommunication purposes. From the triangular packing theory, it was estimated that a quarter million 10 µm fibres would form a bundle less than 1 cm in diameter. This is still considerably smaller than most camera lens systems while maintaining the flexibility of optical fibres. A bundle of 10 µm fibres required to resolve a 256 pixel by 256 line

image (that is 65536 points) would be about 4.5 mm in diameter, considerably smaller than any currently available lens/camera system.

Using the previously described calibration routine, several problems become apparent. The first is the great amount of time that would be required to locate 65536 fibres. Extrapolating the values for the fibre used in the previous experiment, a calibration time of over 100 hours is obtained. This was considered to be impractical. The second problem was the injection of light into each fibre for identification of its input and output points. A 10 µm pinhole was used for a 50 µm fibre, and assuming the same ratio for a 10 µm fibre, a 2 µm pinhole would be required. With stepper motors capable of moving in half micron half steps, the centre of the pinhole would be located to an accuracy of 25% of its diameter. This accuracy, together with the problems of differentiating the multiple fibre outputs at some input points, was considered to be too difficult to achieve. For these reasons a calibration routine capable of calibrating a fibre of higher resolution was needed, and its development and implementation is described in Chapter 6.

However, the tests carried out up to this point indicated that it was possible to transmit optical images through calibrated incoherent optical fibre bundles. The next step in the attainment of the goals of this project was to accomplish the reconstruction in real-time and then to pursue further the calibration of higher resolution fibre bundles.

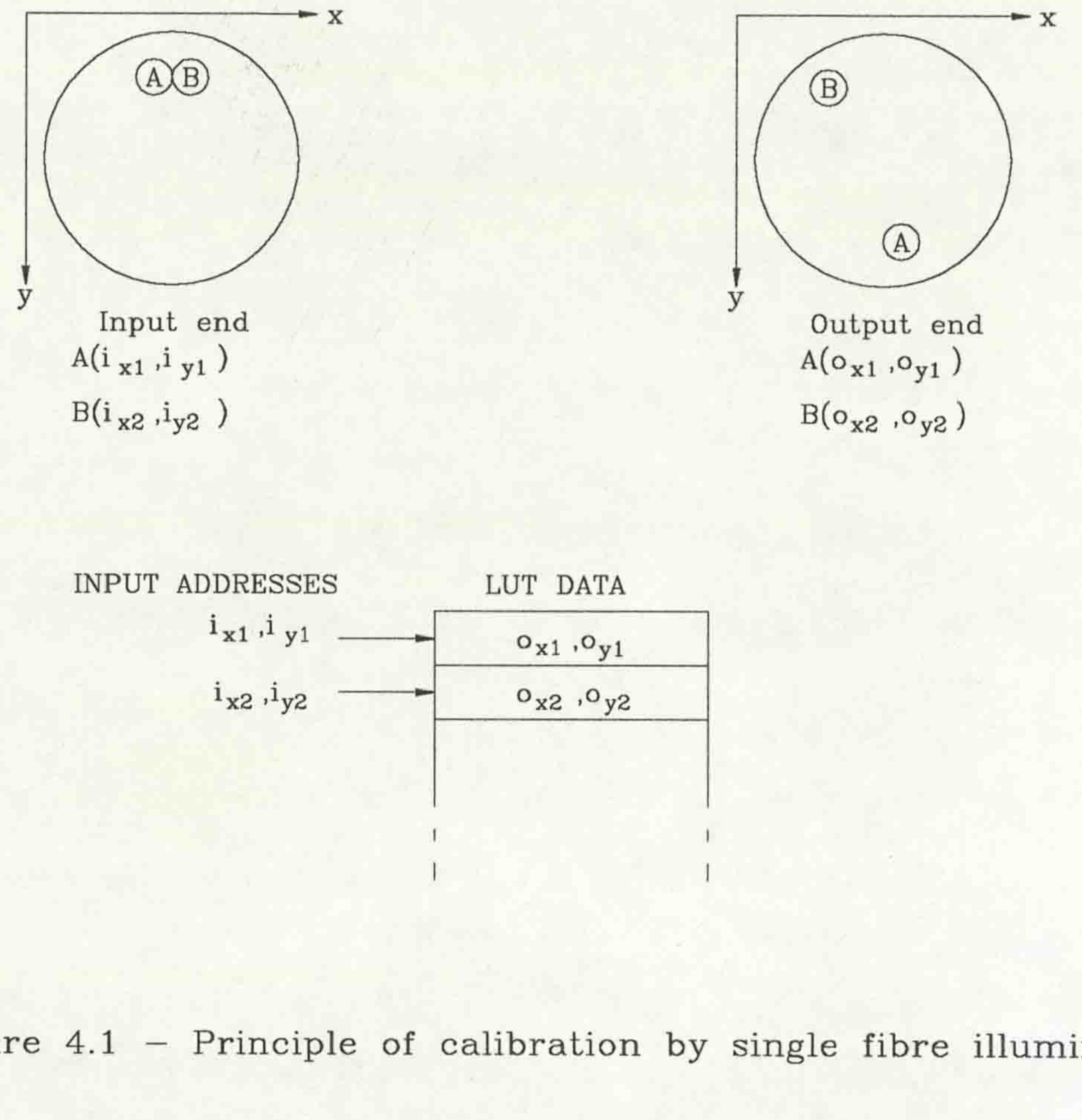


Figure 4.1 – Principle of calibration by single fibre illumination

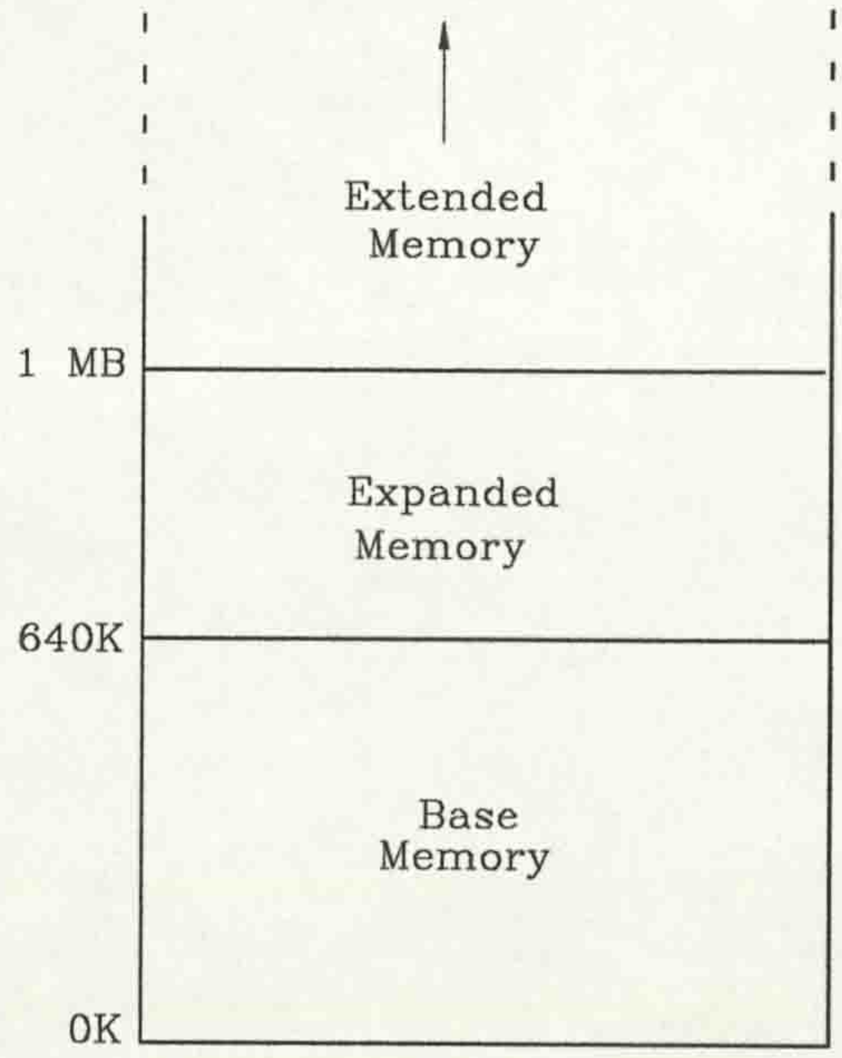


Figure 4.2 – IBM/DOS system memory map

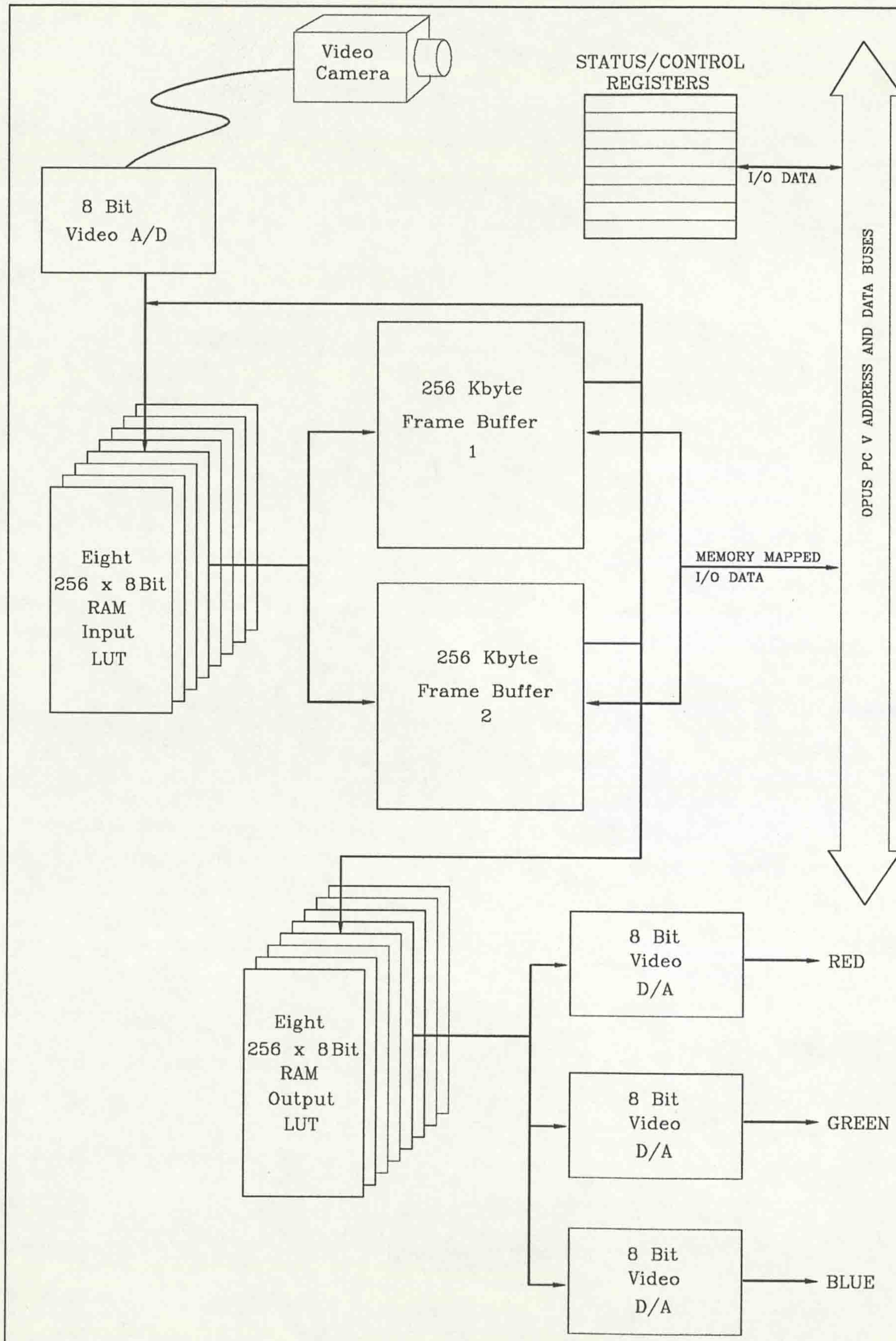


Figure 4.3 – The Data Translation DT2853 frame store

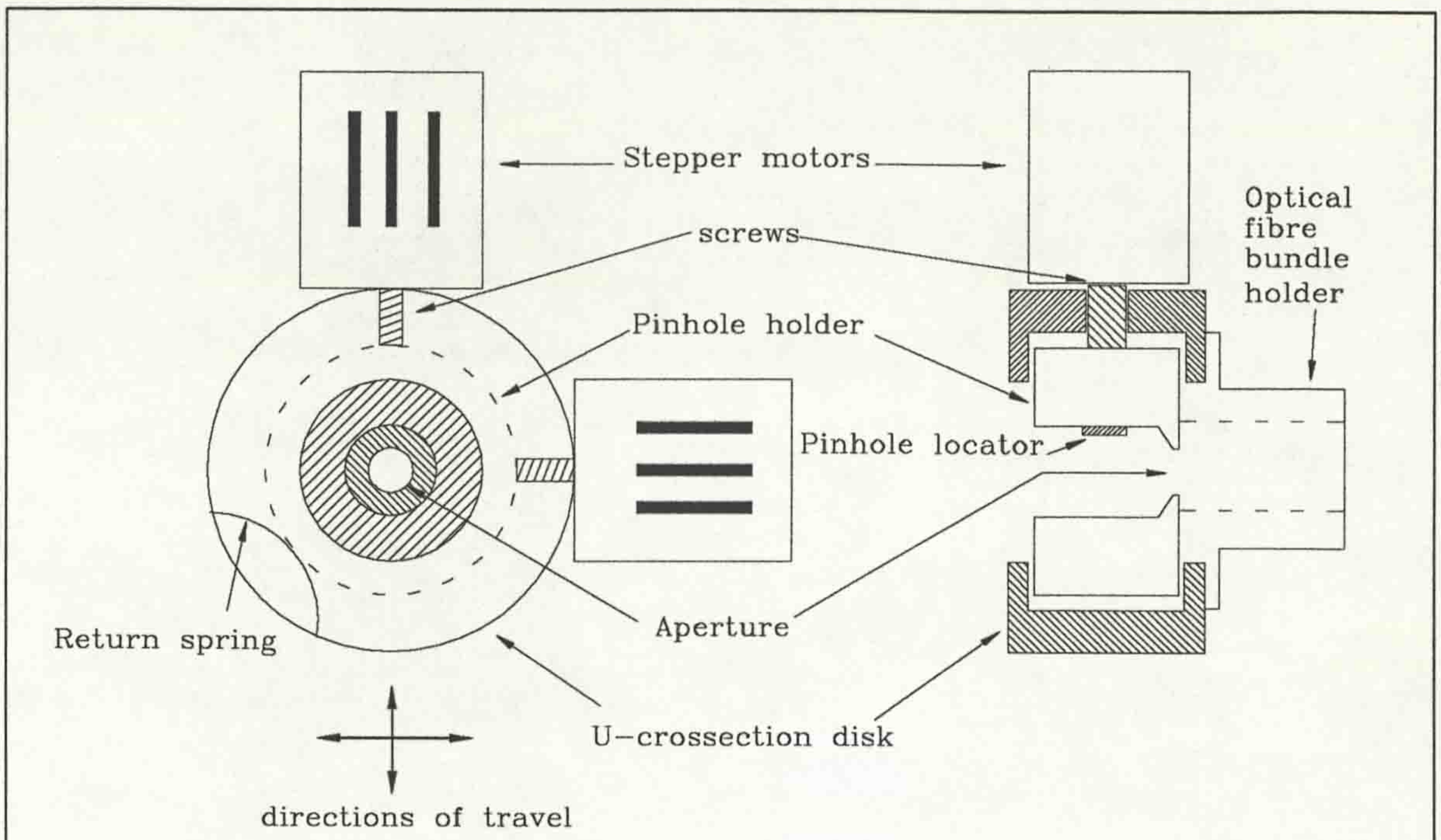


Figure 4.4 – The Oriel stepper motors and pinhole holder

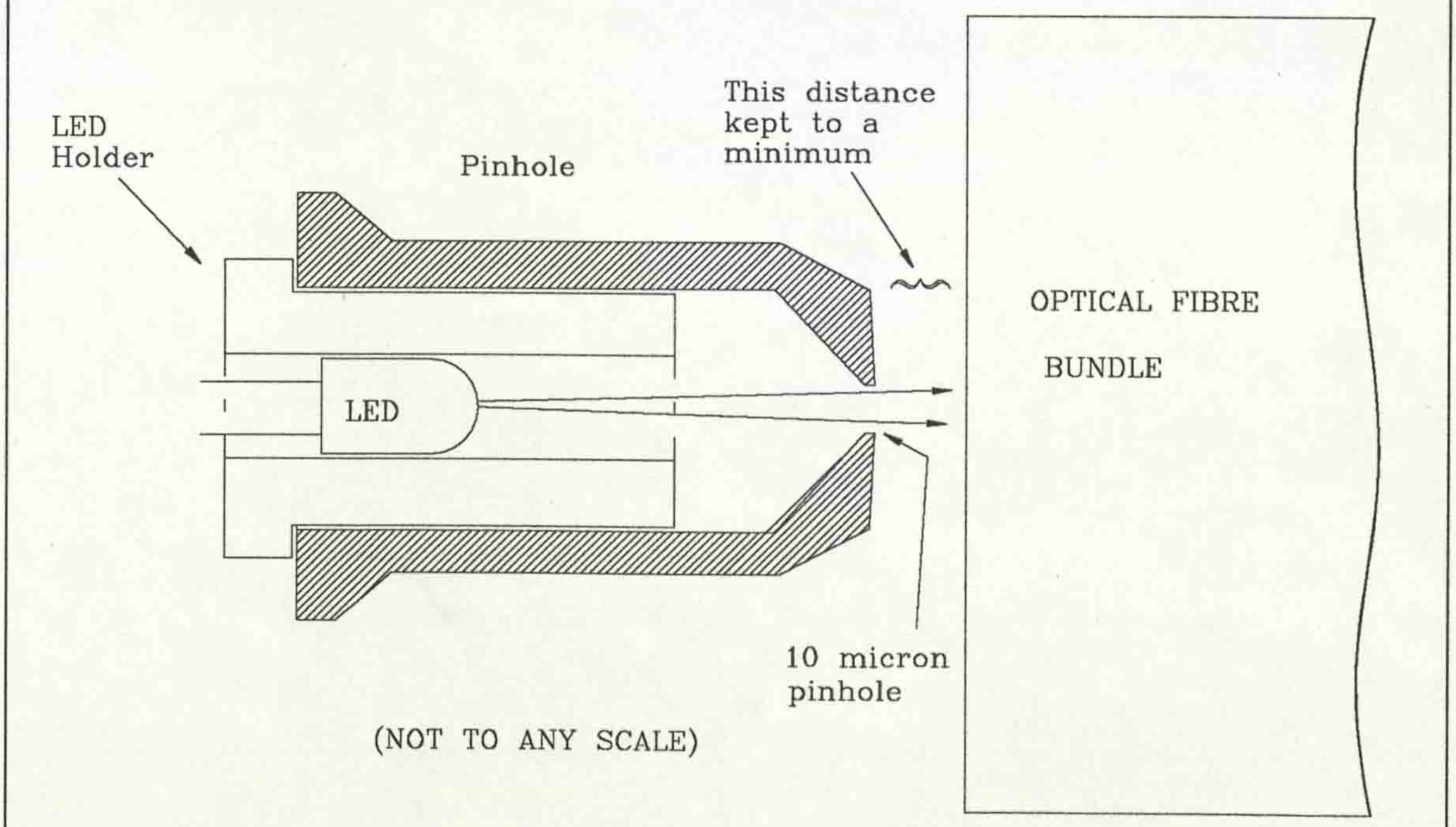


Figure 4.5 – Injection of single spot of light into fibre bundle

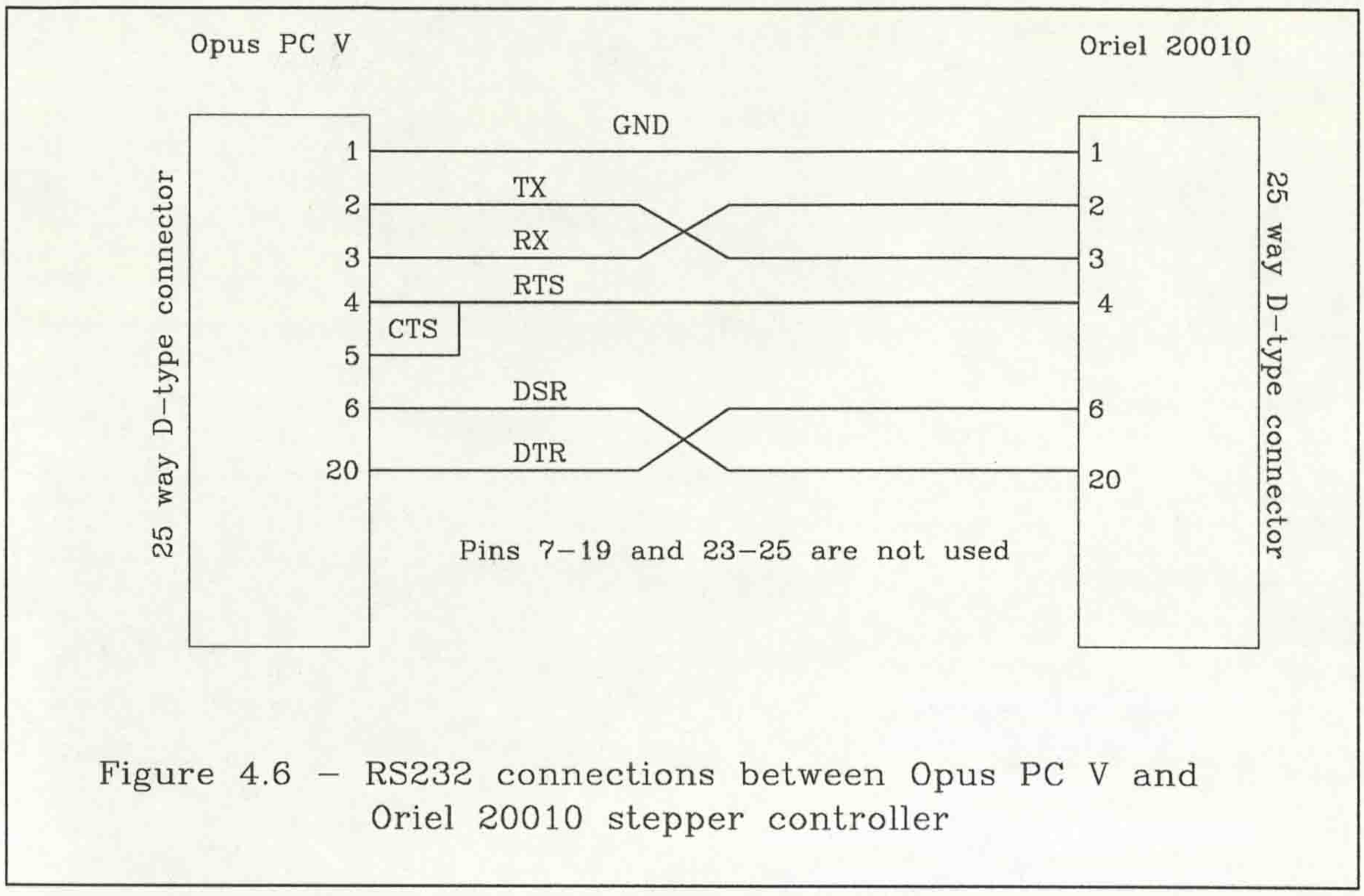


Figure 4.6 – RS232 connections between Opus PC V and Oriel 20010 stepper controller

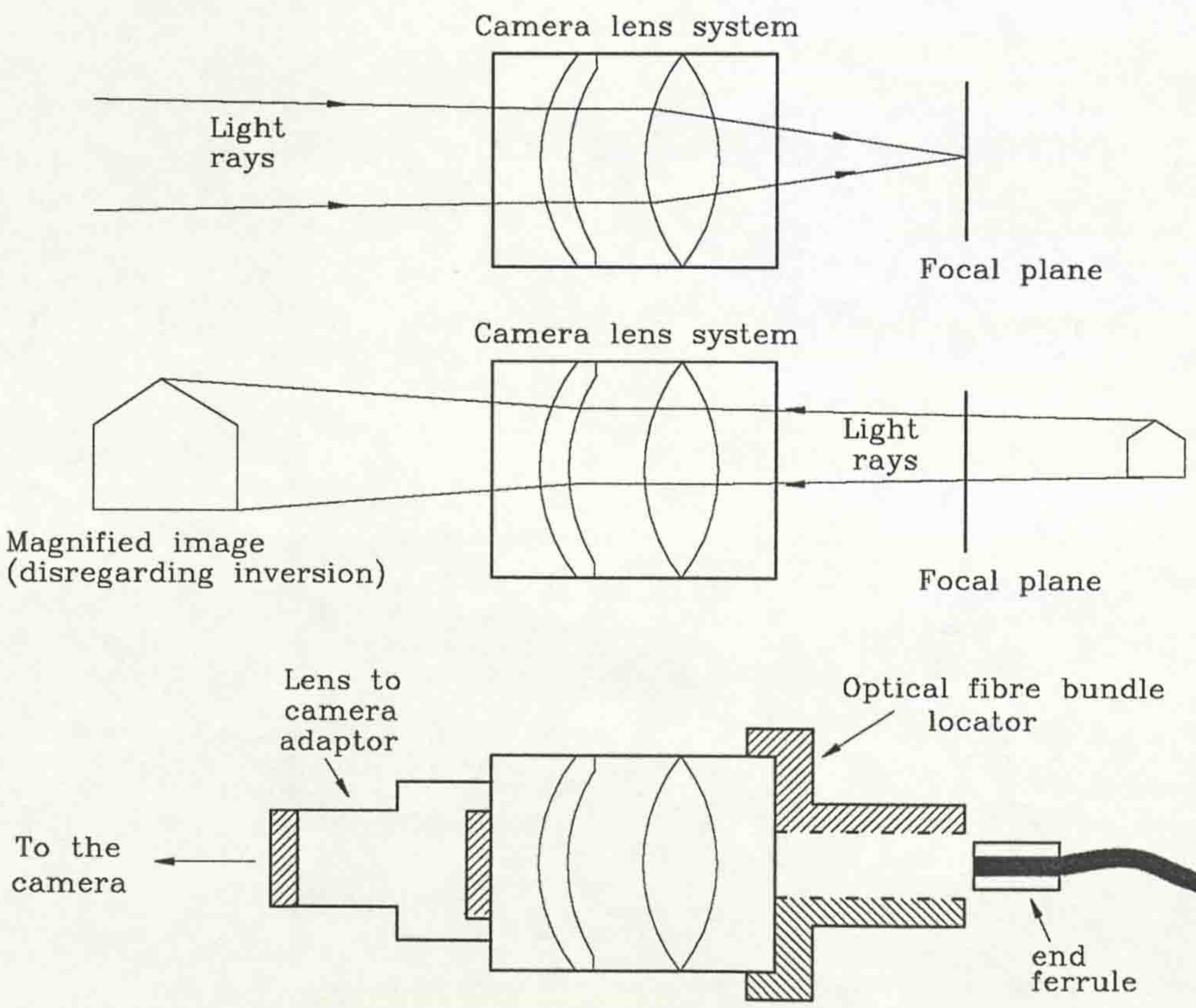


Figure 4.7 – Normal, reversed use of camera lens and the arrangement used for experiment

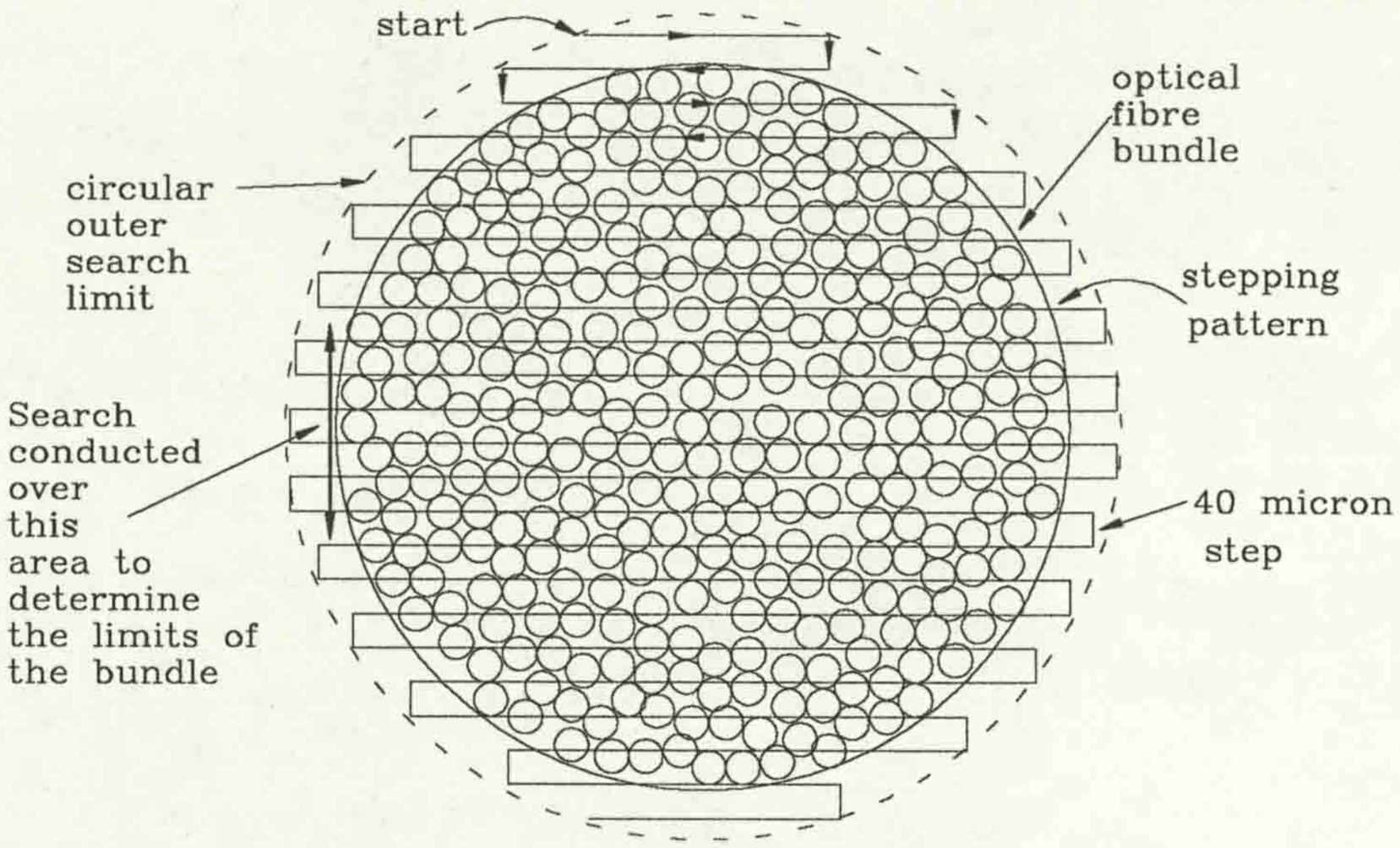


Figure 4.8 – Search path for single fibre illumination

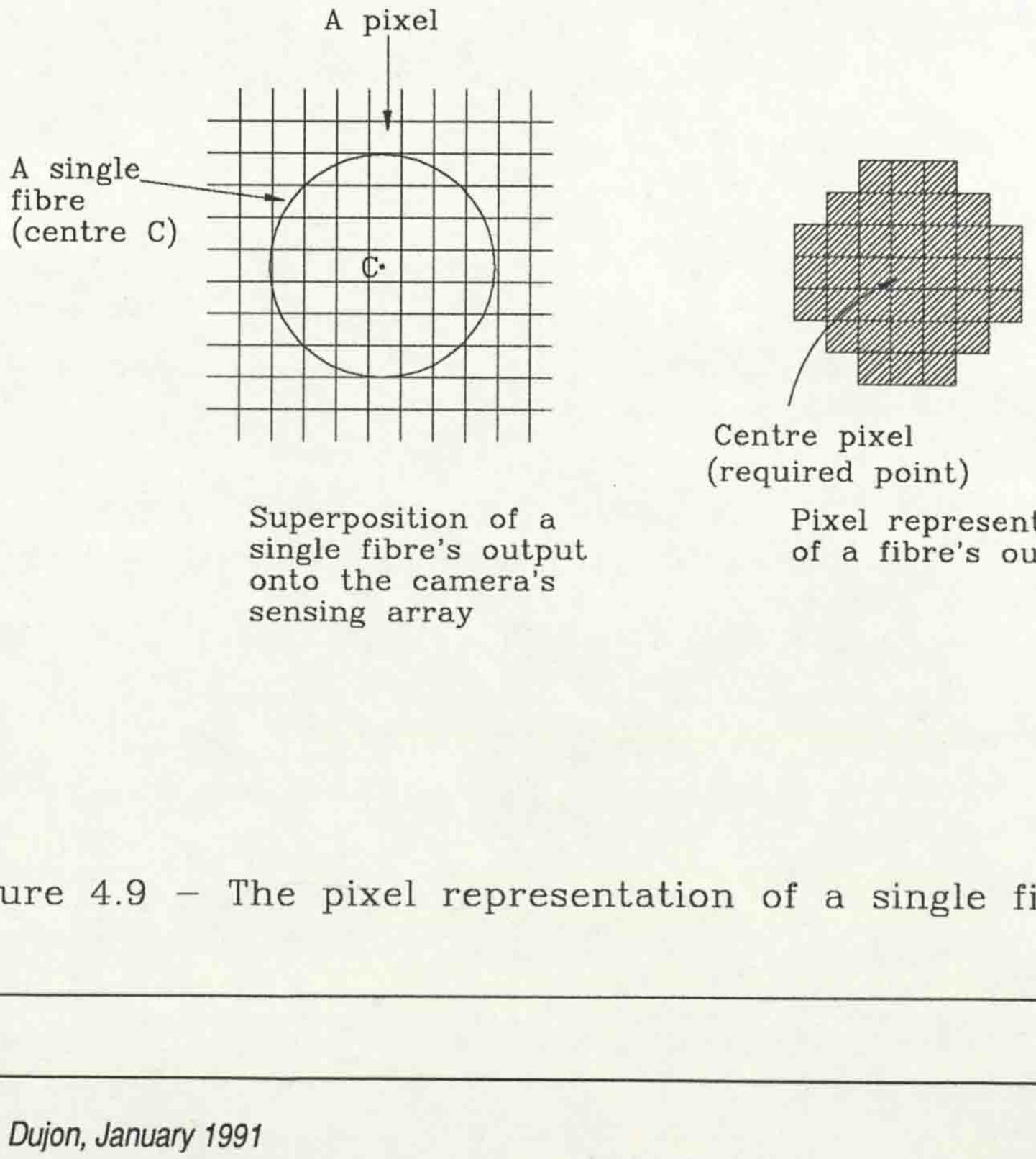
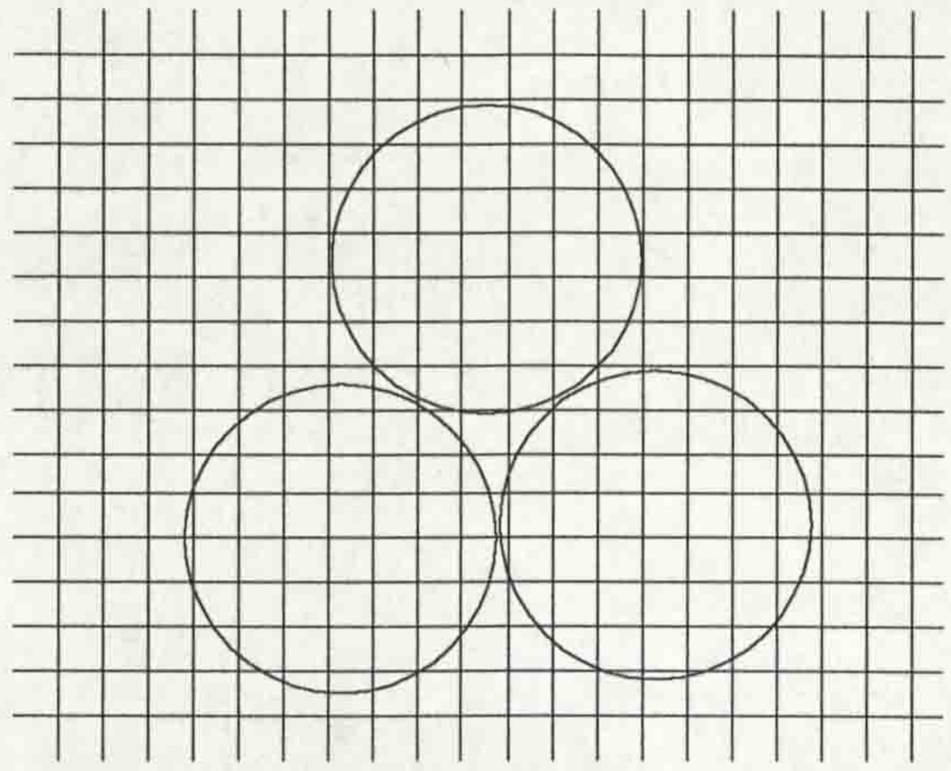


Figure 4.9 – The pixel representation of a single fibre's output



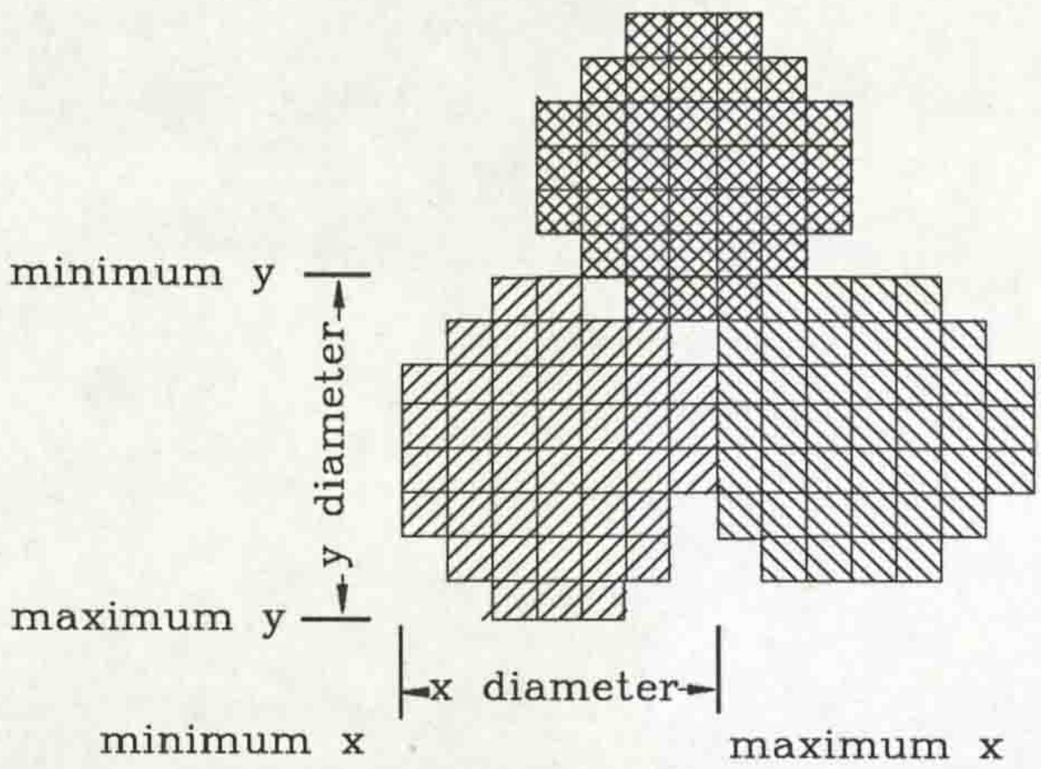


Figure 4.10 (a) – Fibre output measurements

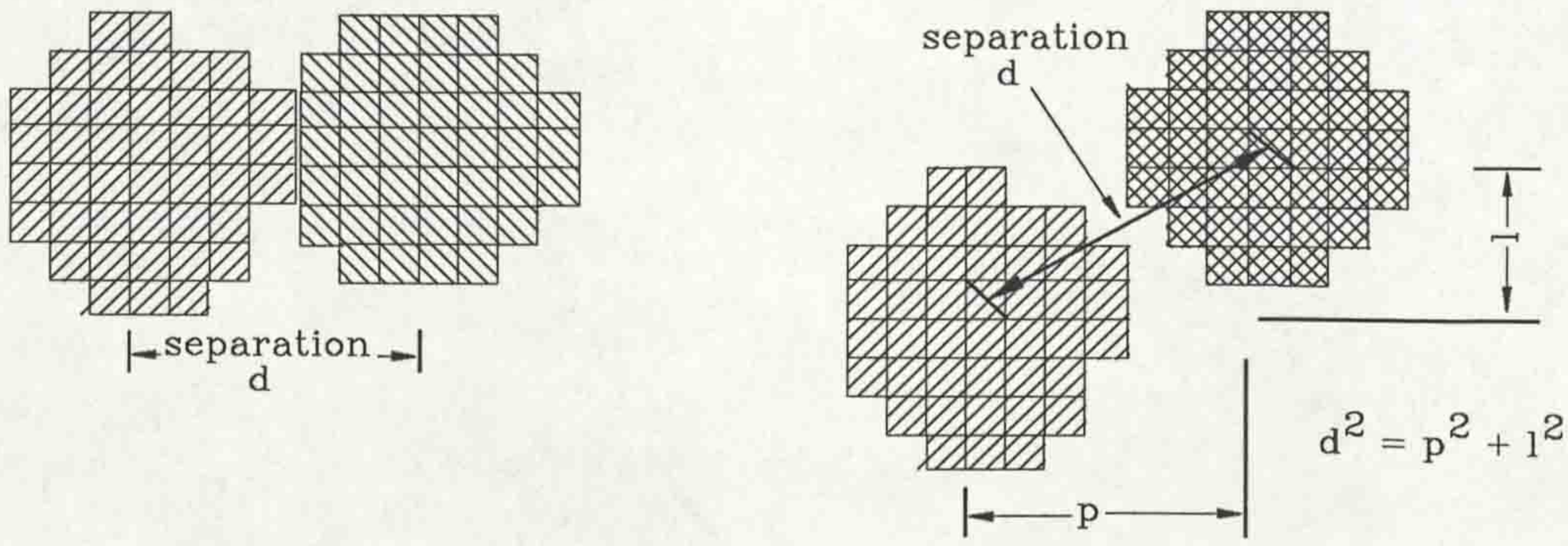


Figure 4.10 (b) – Measurement of the separation of centres

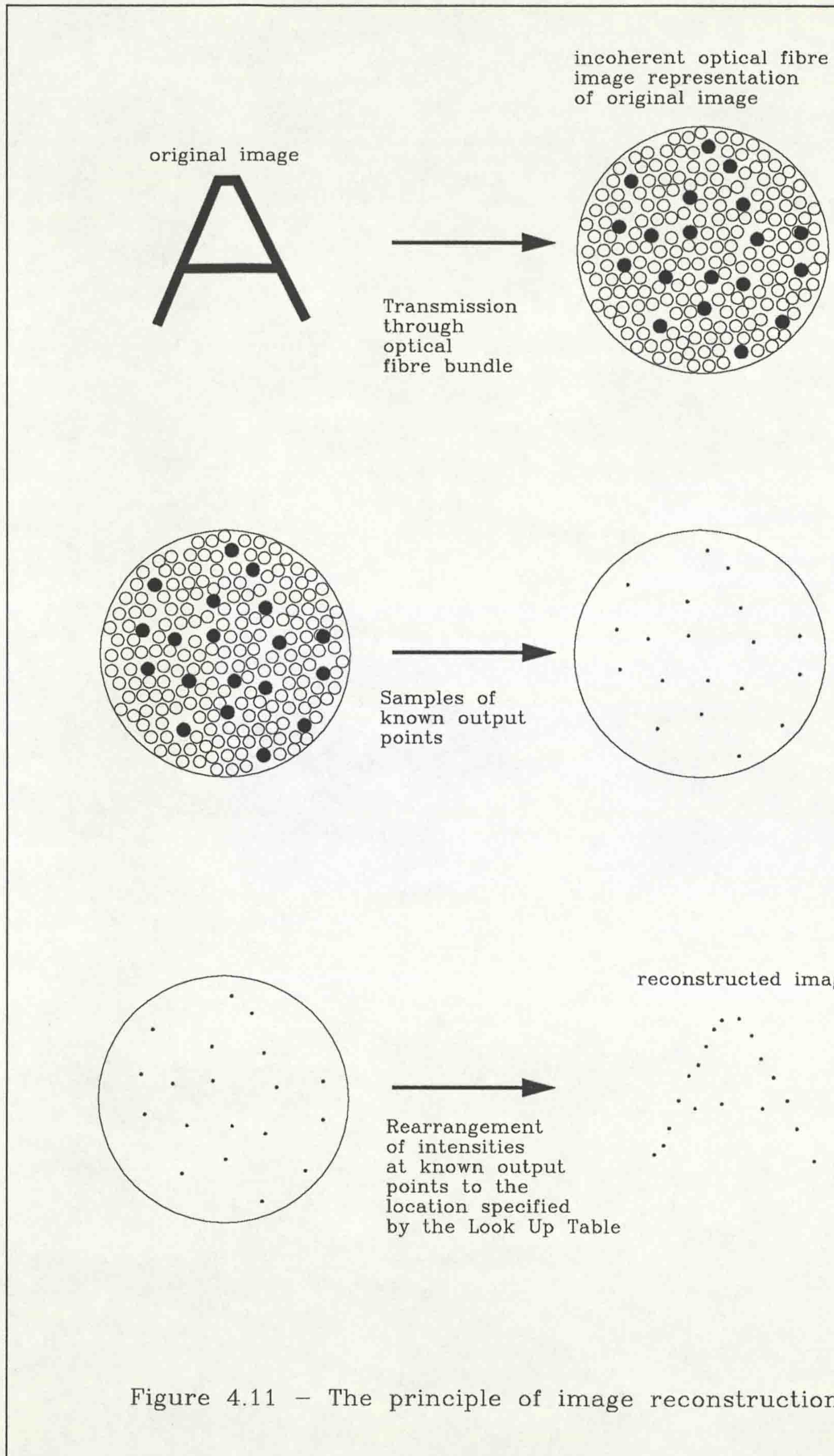


Figure 4.11 – The principle of image reconstruction

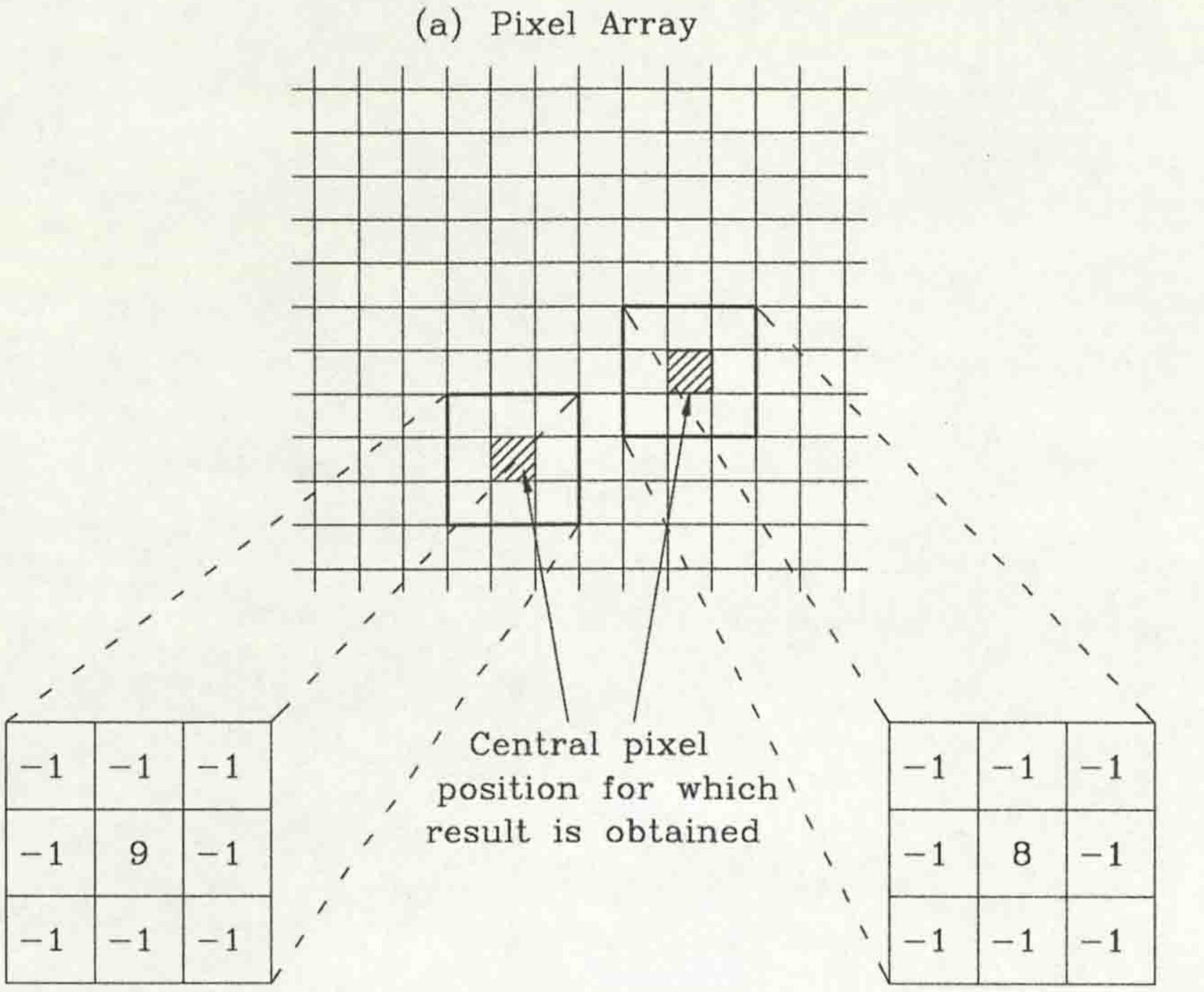


Figure 4.12 (a) – (c) – The operation of convolution filters in image processing

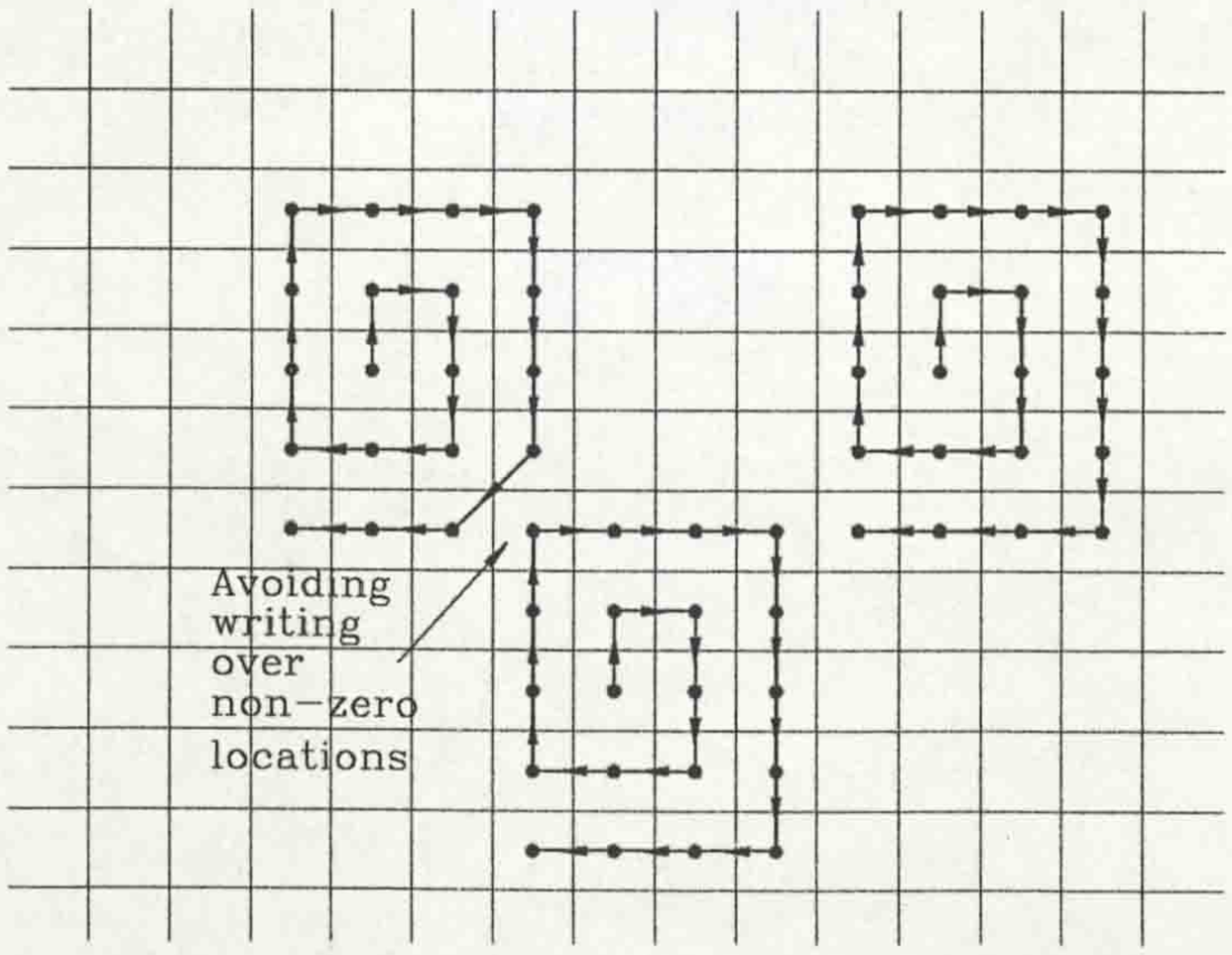


Figure 4.13 – The pixel spreading algorithm

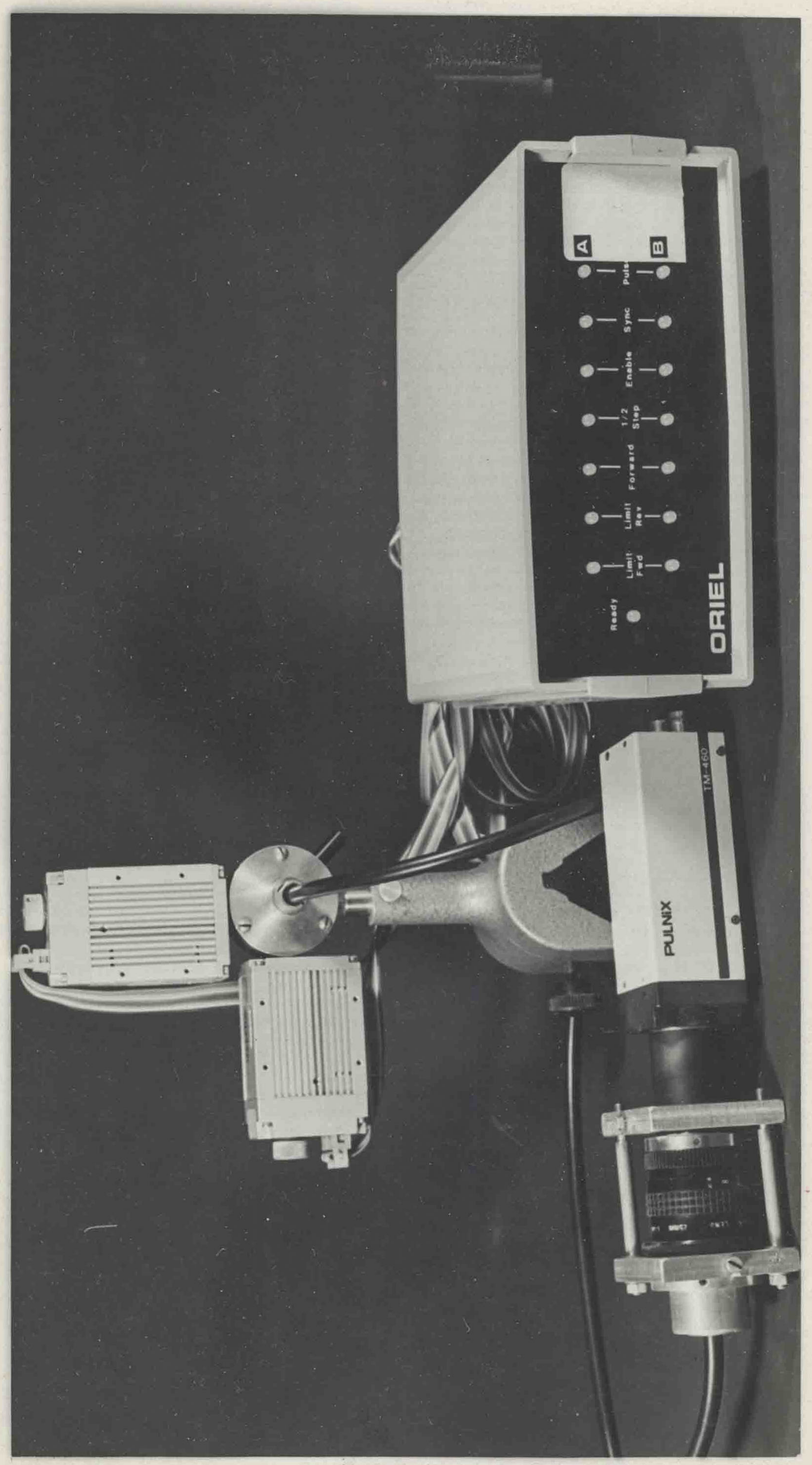


Plate 4.1 - The Oriel Pinhole holder, stepper motor system, camera and optical fibre bundle

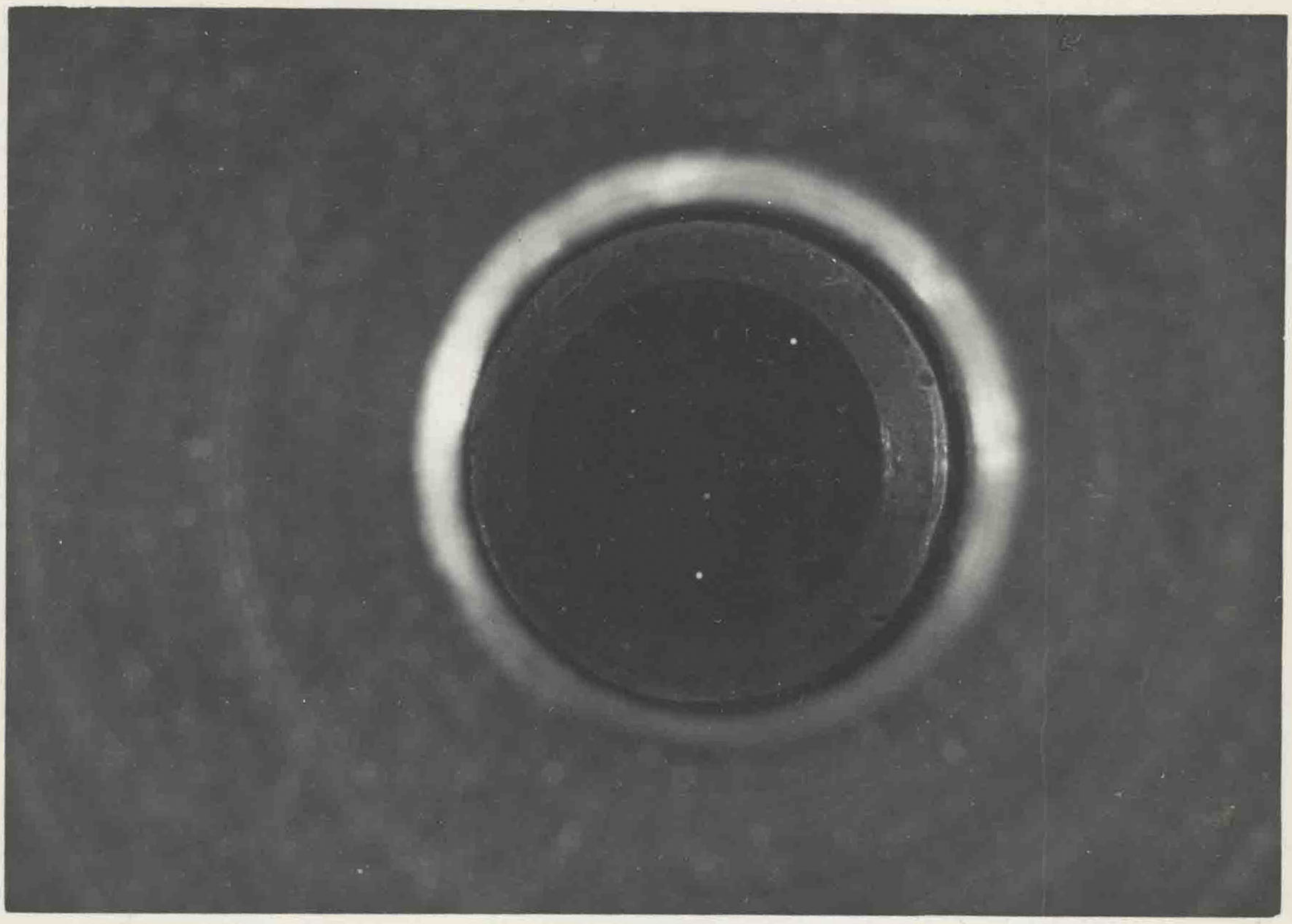


Plate 4.2 - The Output of an Incoherent Bundle Showing Multiple Illumination of the Fibres

Plate 4.3 - The Camera and Lens Systems for Image Transmission

5. HARDWARE DESIGN

This chapter describes the development of a hardware image processing system to perform the calibration of incoherent optical fibre bundles, and reconstruction of the output images from the bundles. A calibration method has already been implemented and proven to work within certain limits. This method will be implemented on the following hardware system, and the system will also be used for further investigations into other calibration mechanisms.

The input into the system will be an analogue video signal from a camera, which represents the output of the optical fibre bundle. This video signal will be analysed to obtain the information required for the calibration, using the methods described previously. The calibration information will then be used by the hardware system to perform a real-time reconstruction of the fibre bundle's output to produce a representation of the original input image to the fibre bundle. The following sections are the descriptions and development of the elements required to achieve these aims.

5.1 Video Image Display Systems

The basic objective of any video system is to electronically produce and display moving images that can be perceived as such by the human eye. The specification of such a system will therefore be determined by the characteristics of the human eye. Extensive investigations have shown [*1,3*] that the human eye interprets a sequence of images displayed at a rate of 15 frames per second or greater, as a continuous movement (section 2.1). A video frame can be visualised as a still photograph of the scene to be displayed, and is therefore a description of the scene at one instant in time. The human eye however can detect a flicker in the image until frame rates approach a frequency of 40 Hz.

The use of a cathode ray tube as the display medium dictates that a two dimensional image must be reproduced by a beam focused to a point. The two dimensional picture information must therefore be displayed as a function of a one dimensional coordinate system, that is, time. The combination of these two requirements dictates that some scanning method must be used to produce the sequence of frames on a cathode ray tube system. The principle involved, as described by the RS-170

standard [*1-4*], is essentially the same for all television transmission systems, with variations in the bandwidth of the transmission system and coding of the colour information. The electron beam in the cathode ray tube is scanned across the screen from left to right, as the screen is viewed, while subject to a constant vertical motion, which produces a slight tilt to the horizontal lines. The aspect ratio of the frame, for domestic television systems, has been defined as 4 : 3, width : height. The frame is described by two scanned fields: the odd and even fields, to give an interlaced display as shown in Figure 5.1 (a). At the end of each line, a horizontal retrace signal is applied to the electron beam to start the next line at the correct position. This signal produces a much faster retrace than the scanning signal. A frame rate of 25 frames per second, according to the PAL standard, is used. This is aligned with the mains electricity supply frequency of 50 Hz, to reduce the effect of any power line interference to a stationary pattern. This frame rate allows a field rate of 50 Hz, above the flicker detection capability of the eye.

The above description applies to grey level, or non-colour, television systems. The principles for colour displays are similar. The hardware system to be designed is concerned with non-colour systems, and further explanations will concentrate on such systems. The camera that will be used, the Pulnix TM-460 [*42*], has an CCD imager [*41*] resolution of 512 pixels per line horizontally, and 512 lines vertically, produced in two fields of 256 lines each. An examination of the PAL standard shows that each frame occupies a time period of 40 ms. The standard also requires 625 lines per frame, giving a line period of 64 µs. The TM-460 produces the video information to the same line frequency and frame rate. At the end of each field, several equalisation pulses are produced for synchronization to the start of the next field. The TM-460 outputs these pulses for a longer period than the normal PAL standard to compensate for the fewer number of lines in each field.

A typical video signal, shown in Figure 5.1 (b), is made up of analogue data for each line, separated by horizontal synchronization pulses, and two fields of lines, each field separated from the next by a number of equalisation pulses that determine whether the next field will be even or odd. Interlaced fields are used in order to reduce the total bandwidth of the transmission system [*64*]. If a single field was used, the time to fill the entire display area from top to bottom would take twice as long. The use of interlaced fields allows a full screen to be displayed in half a frame period, while using the second field in combination with the persistence of the display apparatus to improve the resolution of the system. While the eye will perceive a scene changing at 25 frames a second as continual motion, if the light levels

between successive frames changes significantly, a flickering of the image will be perceived. The doubling of the number of fields to 50 by the interlaced scanning method reduces this effect [*1-4*]. The time periods shown in Figure 5.1 (b) determine the operating specifications of the image processing system to be designed.

5.2 Principle Of Operation

The principle of operation of the hardware system was defined by its intended function. The main functions of the system are to perform the calibration of the fibre bundle and to then use this information to perform a real-time reconstruction of the video images acquired from the optical fibre bundle. The envisaged method of operation for the system would be in the following sequence:

1. Initialise the system by clearing all the memory areas and writing to control registers, to define the intended mode of operation of the system.
2. Calibrate the fibre bundle, if necessary, to obtain the reconstruction data for that bundle and store that data in a file for later retrieval, and/or
3. Read the data for the previously calibrated fibre bundle from the storage file and transfer that data to the LUT prior to initiating the reconstruction operation.

5.2.1 Calibration

The first objective, the execution of the calibration, will be examined. To achieve this, the system must implement all the features of a standard image processing system. The analogue video data from the camera must be digitized and stored in an area of memory that must then be made available to the host processor for analysis. The communication path with the host must be bidirectional to allow the data obtained from the analysis and the result of any image processing, to be displayed. It would therefore be advantageous, but not essential, to have two frame buffers, one to hold the original data and the other to hold the processed result. Two frame buffers are not essential for this processing operation if the host processor has enough available memory to store multiple images in its local memory. However, in some image processing applications it may be necessary to display intermediate results while retaining the original data in another frame buffer. The digital information in the frame buffers must be displayed for a visual appraisal of the image processing results, and therefore a digital to analogue conversion system is necessary. To perform the required analogue to digital conversions and vice versa, and to store and retrieve the values thus generated in the correct positions, a time dependent relationship to the video timing signals must be obtained. Therefore

some method of extracting the timing signals from the composite video information must be implemented. This establishes all the elements for a basic image processing system.

5.2.2 Real-Time Reconstruction

Two paths may be followed to achieve the second objective of real-time reconstruction. The first uses the calibration information to relocate each pixel in the display as the intensity for that pixel is obtained from the camera. However, because of the sequential nature of the scanning display, it is not possible to dynamically relocate the input pixels directly to their output points in the display. To do this would involve a random movement of the electron beam in the cathode ray tube, which is not compatible with the scanning method of display for moving images. The second method of achieving real-time reconstruction is to relocate the pixels one field at a time, as the information for that field becomes available from the camera, while displaying a previously reconstructed field. If the field being reconstructed is consecutive to the one being displayed, there will be a time delay of one field time, 20 ms, between the output of the fibre bundle and the image being displayed by the hardware system. It was decided that this method would be adopted as it would satisfy most requirements of image processing applications, unless the information about the scene was needed less than 20 ms after an event that changed the information in the field.

To achieve this method of reconstruction, two field buffers will be needed: one to hold the digitized and relocated information from the camera, and one to make the information in a field, that has already been reconstructed, available to the display system. This ties in with the two buffer system for image processing as described above. An area of memory will be required to hold the information for relocation. This area will be called the Look Up Table (LUT), and will be of the form shown in Figure 4.1 and Figure 5.3 (a) to (d). Read/Write access to this memory area from the host will be necessary to load the LUT with data sets for different fibre bundles and to check the data in the LUT. The calibration data will be read from this area of memory to provide the new location for the data from the video image representing the fibre bundle's output.

5.2.2.1 The Look Up Table

Closer examination of the principle of calibration and the theory of reconstruction shows that the LUT generated from the calibration contained duplicate points, that is,

several input points were mapped to the same output point (see sections 4.3 and 4.4). The initialisation procedure of the calibration assigned all the output points to a null value before the calibration was executed. Any output point not assigned an input would therefore contain a null value. An image processing algorithm was described (section 4.3.6.2) to partially compensate for this effect. A similar result could be achieved by using the duplication of output points to locate a source for the displayed data at each pixel location. This possibility is investigated next.

The implementation of the reconstruction could be achieved in two ways. The first would be to find the output point for the data from each input point in turn, that is answer the question: What is the known output point for the data from this input point? Thus the data from each input point is directed to the new location dictated by the calibration. The other method would be to answer the question: Where should the data for this output point come from? Using this method, the data for each output point is selected from the original, unaltered array of data, at a location pointed to by the LUT. These two reconstruction methods correspond to different locations of the LUT in the real-time reconstruction method. To understand this aspect, it is necessary to examine the organisation of data in the LUT.

The previous calibration, the single spot method, produced a LUT where each entry consisted of two points, an input and output location, each of two coordinates. The software to perform the reconstruction used the input points, converted from the stepper motor coordinates to a pixel coordinate system (section 4.3.5) to relocate the output data back to the corresponding input position. Since it was envisaged that an automatically incrementing addressing system would be used to extract the data from the LUT in the hardware system, the need for both sets of coordinates to be stored could be negated if the storage address represented one of the pairs of coordinates. The address and the data stored at that address are an interchangeable pair, providing a unique mapping, and as such it is unimportant which pair of coordinates form the address and which the data. The LUT can then be visualised as a two dimensional array, each entry containing the relocation address for the input address of that entry. To form this LUT from the data from the previous calibration, the stepper motor input address must first be converted to a pixel coordinate system. This new data format can then be used either as the storage data or the address for the output points corresponding to the stepper motor input locations as determined by the calibration.

With reference to Figures 5.2 (a) and 5.2 (b), two locations are shown for the LUT. It can be placed, as in Figure 5.2 (a) between the address generator for the incoming pixels from the camera and the destination buffer. The digitized pixels are then relocated in the buffer to their correct position, ready for display, when they can be sequentially accessed in the correct order. This method of operation can be described as the answer to the question: What is the correct output point for the data from this input point? The other arrangement, Figure 5.2 (b) places the LUT between the address generator for the display pixels and the source buffer. This arrangement answers the question: Where should the data for this output point come from? These two locations of the LUT correspond to the use of the output point as the data and the input point as the address for the data (Figure 5.2 (a)), and the use of the input point as the data and the corresponding output point as the address for the data (Figure 5.2 (b)), in the LUT.

In the previous method of reconstruction, based around software procedures and the DT2853, the data for the frame to be reconstructed was read into the host processor's memory from the DT2853. This data was then relocated to its correct output position and then transferred back to the DT2853 for display purposes. This method corresponds to the block diagram representation shown in Figure 5.2 (a). With the apparatus used for the software reconstruction it was not possible to interfere with the data between the source buffer and the display mechanism. The results from the software method of reconstruction, presented in Chapter 7, showed that many gaps were present in the reconstructed image, before the smoothing algorithm was applied. The presence of these gaps was due to the nature of the data in the LUT and the fact that many input points were mapped to the same output point. During reconstruction, the data values from several input points were written to the same output point, and the image that was finally displayed is made up of the last value that was written to each output point. Since some output points, not illuminated by fibres, have not been assigned to an input point, and have been left at the null value from initialisation, data is not written to these locations, and these points show up as gaps in the image.

In section 4.3.6.2, a smoothing procedure was described that filled in the gaps in the images reconstructed from a bundle of only 2200 fibres, 2000 of which were assigned input/output coordinate pairs. This method filled in the gaps with the value at known points in the output. It was decided to implement this algorithm in this design. By placing the LUT between the address generator for the display mechanism and the source buffer, as in Figure 5.2 (b), the data from one output point

in the fibre bundle's output area can be written to the several input locations assigned to it. To achieve this, the LUT would redirect the address from the display address generator to locate the pixel which should be displayed, as dictated by the calibration data. Since all the original data is stored in the source frame buffer, multiple accesses can be made to each pixel value to spread its value to all the duplicate locations determined by the LUT. The gaps in the image would then be filled to give a more complete image. Figure 5.3 shows various possibilities in the structure of the data in the LUT. Figure 5.3 (a) shows a normal, random association of input and output points. Figure 5.3 (b) shows null points where certain locations have not been assigned an output. Figures 5.3 (c) and (d) show duplication of output points occurring at adjacent input points and points further away from each other.

5.2.3 The Analogue to Digital and Digital to Analogue Converters

The arrangement of the principle elements of the hardware system have thus been established, and is shown in Figure 5.4. The other main decision to be made concerns the resolution of intensities in the images. It was decided that for better calibration accuracy, the greatest number of bits possible should be used to describe the intensities. This would allow a more precise definition of the intensity at each pixel. For a given total number of pixels, 65536 in this case (a 256 line by 256 pixel display), spread over a certain range of intensities, the use of more digital values to describe the intensities would mean fewer pixels at each intensity because of the finer intervals between intensities. Small errors in the choice of threshold for each input point would therefore displace fewer pixels (see section 4.3.3.2). Examination of the devices available that would be suitable for such an application, showed that 6 and 8 bit devices were most prominent. It was therefore decided that an 8 bit intensity resolution would be used.

5.3 Design Criteria

This section describes the technical development and design of the hardware system. The overall concept was explained in the previous section and the detail of each subsection will be presented. The design method will also be described in order to clarify the operation of the system. The final design is presented in the schematic diagrams in Appendix B.2, along with the custom design elements. The system has been labelled the Optical Fibre Calibration System (OFCS), and will be referred to as such in the remainder of this document.

5.3.1 Design Principles

In the interest of low cost, it was decided that as far as possible, standard digital integrated circuits should be used. The design was envisaged as a group of elemental blocks, each to be controlled by specially designed logic systems and linked by control signals that determined the overall operation of the hardware system.

A synchronous method of operation would be best suited to this design philosophy as it would allow accurate and predictable control of the elemental blocks. The algorithmic state machine (ASM) design method [*65*] was adopted for the development and implementation of the controlling hardware operation, as this assured synchronous operation. The algorithms were developed using the CUPL software package [*66*], which converts the state machine description of the desired operation into a logic equation description for each output. The resulting JEDEC [*67*] files defining the logic operation were transferred to appropriate Programmable Array Logic (PAL) devices using a STAG logic programmer [*68*].

5.3.1.1 Programmable Array Logic (PAL) Devices

PAL devices [*65,69,70*] are integrated circuits with several inputs and outputs. The inputs are fed into an array of logic gates that are interconnected by fuses. The fuses are blown to a pattern determined by the information in the JEDEC file to implement the desired hardware operation. The outputs from this logic array are either directly available at the pins, or can be directed through registers, driven by a clock, to hold the required outputs for feedback into the array for the next stage of operation [*70*]. The devices are available in erasable format, where the fuse connection can be reformed either by an electric current or ultraviolet light.

As with all electronic devices, PAL devices have a finite propagation delay, nominally between 15 ns and 50 ns, depending on the device specification. A design using PAL devices would have to take this delay into account, as it is greater than the typical single gate delay and can therefore form a significant percentage of the duration of the signals that are used in the design. The operation of the devices required in this application will be described in the relevant subsections following the description of each elemental block.

All PAL devices can only implement a finite number of logic equations, of limited complexity, and with a predefined maximum number of inputs and outputs. The device must therefore be chosen to meet the required speed of operation, be able to

provide the number of logic gates for the minterm expression [71], and have enough inputs and outputs for the desired operation.

5.3.2 The Host

The host processor system will initially be the Inmos B004 transputer board [72], which is based around the Inmos IMS T414 transputer. This system was chosen for several reasons. Primary among those was the high system operating speed when used with the Occam language [73]. The ease of interfacing external systems directly to the processor system via the serial communication links would also be of great use in this application.

The basic architecture of the T414 is shown in Figure 5.5 [74]. This device contains a 32 bit processor, 2 Kb of memory, a 32 bit external memory interface and four bidirectional serial communication links. The entire family of transputer products operate from an externally supplied 5 Mhz clock, independent of device type, processor speed and transputer word length. The required clock frequency for internal operation is derived by a phase locked loop from this clock. The electromagnetic transmission problems associated with the distribution of a high speed clock signal is thus avoided. The interface to the hardware could be achieved by two methods: including the memory areas for the buffers, LUT and controlling registers as part of the host's memory area, or using the available links for serial communication between the two separate systems. The links were chosen as the more suitable interface primarily because of the physical difficulty of performing the memory interface to the Inmos B004 board that was being used in this application. This design was laid out on a full length, single slot IBM format expansion board, with some of the processor's address and data lines, as well as the links, available for interfacing via a 37 way D-type connector. These address and data lines would have to be extended approximately 30 cm to the new hardware system. It was reasoned that transmission difficulties may arise over that distance, and since the first prototype was to be hand wired, the complexity of manipulating the data and address lines, and the greater sophistication of the interface, was best avoided.

The main advantage of a memory interface would be the greater speed of transfer of data between the processor and the frame buffers and LUT. However, these data transfers would only occur during calibration, and a slower interface would not affect the real-time operation of the reconstruction. In the implementation of this application, it was found that the use of two Inmos IMS-B004 would be advantageous. The arrangement will be discussed and described further in the next

chapter, where software development for the calibration is investigated. The separation of the IMS-B004 systems running the TDS and the program makes it easier to analyse the state of the program if and when the system was halted on error.

The use of the links for the interface would allow more general connection to any transputer system, as all Inmos transputers have at least one link connection, and all support the same communication protocol. The maximum speed of communication along the links, between transputers is at least 10 Mbits/s [*72*]. These links are handled as hardware channels in Occam, and can be directly accessed from the software. The communication protocol and devices for interfacing peripheral systems not based around a transputer, to transputer host systems, will be discussed in the following sections.

5.3.2.1 The Transputer

A faster image analysis system would speed up the iteration time for any software development which greatly eases the task of analysing the result. The transputer is a microcomputer with its own memory, and communication links for point to point connection of the transputer to other transputers and interfacing to other electronic systems. This is all contained on one VLSI circuit, in a Pin Grid Array (PGA) package. The IMS T414 was the first in a family of transputers, and is a 32 bit microprocessor, with 2 Kb of on chip memory, and the now standard 4 serial communication links. The T414-20 is capable of operating at speeds up to 10 Million Instructions Per Second (MIPS), with a 50 ns processor cycle time. This compares favourably with 0.9 MIPS for an Intel 80286, operating at a clock speed of 10 Mhz [*34*].

The design architecture of the transputer is based on a system of interconnected processes. A process, in this instance, is a unit of design that performs some task. It is defined by the data it sends and receives, and communicates to other processes via point to point software channels. The design of the transputer allows many processes to be run concurrently, with the processor sharing its time between processes. Detailed description of the hardware design and principles of operation of the transputer is outside the scope of this thesis, and therefore only a brief introduction is given here to facilitate understanding the implementation of the calibration method. References [*72,74,75,76*] provide more in depth information about the transputer.

The software required to perform the calibration was developed on the Inmos B004 board which is based around the T414 microcomputer. This board provides an additional 2 Megabytes of program memory for the transputer and differential line drivers for the links. The Inmos links are bidirectional, point to point, serial communication channels. They allow direct connection of transputers to each other without the need for additional circuitry and work at data transfer rates up to 20 MBits per second. The IMS B004 is available as a standard IBM format expansion board, and occupies a single 8 bit expansion slot.

The software programs were written in Occam (section 5.3.2.2) [*73*], and compiled and developed under the Transputer Development System (TDS). The TDS is essentially an operating system for transputer systems, and provides several functions that enable the IMS B004 to use the host IBM PC or compatible for storage filing and screen/keyboard interfaces. It takes up over 100 Kb of memory on the transputer board. A more detailed description of the TDS can be obtained from [*76*]. The interface to the host machine, an OPUS PC V in this instance, is accomplished via one link from the transputer and some on board circuitry which adapts the link communication protocol to that of the 80286 microprocessor. This allows the transfer of data between the two systems, the speed of transfer being limited by the 80286 bus architecture and the speed of its peripherals such as the hard disk and screen graphics adaptor, since the transputer link operates at a much higher speed. The entire system will be transferred to a Toshiba T5200 portable computer for on-site use, and a 2 MB transputer board designed at the University of Liverpool [*5*]. The 2Mb transputer board is identical in operation to the Inmos B004.

5.3.2.2 Occam

Occam [*73,74*] is a high level programming language that was developed together with the transputer, as a dedicated language. While compilers of the commonly used languages such as C and Fortran are available for the transputer, to make full use of the processing power of the transputer, it was decided that Occam should be used. An introduction is given to the principles of Occam for an understanding of the software algorithms described later.

5.3.2.2.1 Concurrency

The essential processing power of the transputer lies in its ability to execute processes concurrently, and the point to point communication allowed by the links. When the processes are executed on a single transputer, the total processing time is shared between them. However, when a network of transputers is used, the TDS

allows the processes to be assigned to individual transputers, each communicating the result of its process to the other transputers via its links. The transputers can be arranged into several network configurations to facilitate various data communication paths that may be suitable for a particular application [*75*].

5.3.2.2.2 Constructs

Occam processes can be combined to form a construct, which is itself a process. The are four essential types of construct, namely the sequential (SEQ), parallel (PAR), conditional and repetitive constructs. The SEQ construct indicates that the processes following it are to be performed sequentially, and the PAR construct that they are to be executed concurrently. The conditional constructs, IF and ALT, determine which process will be executed, and the repetitive constructs indicate how many times the processes will be executed [*72*].

5.3.3 Input/Output Interfaces

The choice of using the serial communication links for the interface has been explained. The data is transmitted one byte at a time. The transfer of the data is not synchronised to any clock, so can operate freely between independently clocked systems. The protocol for the transmission of data takes the form of a pair of start bits, 8 bits of data, and a stop bit. The source of the data waits for an 'acknowledge' packet from the destination before sending the next information packet. The transmission of the acknowledge signal is multiplexed with the transmission of the data, and the destination device may acknowledge the receipt of a data packet as soon as the transmission starts. There may, therefore, be a continual stream of data from the source, with no delay between data packets [*72*].

The data from the link is received in serial bit form, and must therefore be converted to parallel byte form. There is a device available from Inmos, the IMS-C011, which does exactly that, as well as conforming to the link transmission protocol and providing handshaking signals to the destination system for the bidirectional transmission of data to and from the host transputer. A block diagram representation of the device is shown in Figure 5.6, and a complete explanation of the operation can be found in [*72*]. The speed of transfer of data between the host transputer and the hardware system is affected by three aspects: the rate of transfer along the link, the speed of operation of the IMS-C011, and the capability of the hardware peripheral system.

The rate of transfer of data along the link sets an absolute limit on the system. Assuming a 10 MBits/s data rate, and each byte of data forms an 11 bit packet of information (two start bits, 8 data bits and one stop bit), the rate of transfer of bytes will be 0.91 Mb/s, or 1.05 µs per byte. Examination of the specifications for the IMS-C011 [*72*] shows that the maximum data rate through this device is 1.3 µs per byte. If the link is operating at its maximum rate, then the IMS-C011 will set the operation limit. The speed of the peripheral hardware is yet to be decided.

The protocol for the transmission of the data to and from the hardware system must be defined before the specification of the interface is complete. The data could be transmitted with a destination address for each byte. If each byte requires a 16 bit address, this would triple the amount of data to be transferred and thus the time taken. After examination of the nature and amount of data to be required, it was decided that blocks of data would be transferred, with a counter included in the design to generate the destination address of the data. The data block for the frame buffers and the LUT is 64 K (64 x 1024 units) long, so the same counter can be used for the addresses. The data for various control registers that might be needed would be of variable length, and this problem will be dealt with later. This disadvantage of writing the data in complete blocks is that individual memory locations cannot be read from or written to in isolation. However, it was not envisaged that access would be required to individual pixels and LUT entries, so this should not prove to be of great consequence.

5.3.3.1 Related PAL Designs

The required controlling logic design should conform to the parameters defined above. The first problem addressed was the on board generation of the destination addresses for the data sent to the hardware system. It was decided that a 16 bit counter would be used for this purpose. There are several dedicated 8 bit counters available, and two of these would be required, used in cascade to provide the full 16 bit address. On examination of the cost of these items, it was realised that they were not significantly cheaper than a PAL device that could be used for the same purpose. It was decided that the PAL device should be used to provide the maximum flexibility. The device chosen for this purpose was the PAL-C22V10-25 [*69*]. This is available in a 24 pin dual in line (DIL) package, with 22 input/output pins, of which up to 10 can provide registered outputs. The outputs can be isolated from other signals by tri-state buffers. The device has a propagation delay of 25 ns, and can operate at clock speeds of up to 40 Mhz.

Thus only two devices must be used for a 16 bit counter, with the ripple carry out from the lower 8 bit counter providing the clock for the upper 8 bits. A counter clear signal would be necessary to reset the counter after each data transfer operation. The ripple carry out from the upper 8 bit counter will indicate the end of the operation. Thus each device must have two inputs: a clock and reset, and 9 outputs: 8 address bits and a ripple carry out. Since the outputs will be used in a bus system, it is desirable to be able to isolate them from other signals on the bus, making the use of an output enable signal necessary. The logic description for these devices is given in Appendix B.1. Each device has 1 output pin free and 9 input pins free. These could be used to implement any separate logic equations that may be required.

The controlling device (labelled IO_CNTRL) must provide the clock for the address counter. The IMS C011 accepts the serial data from the links and presents it in parallel form with the appropriate handshaking signals: QVALID for data from the links, and IACK for accepting data to the links. In response to these signals, QACK and IVALID, respectively, are expected from the peripheral hardware system. IO_CNTRL must perform the handshaking with the IMS C011, and derive the clock for the counters from these signals.

The communication protocol for the transfer of data between the transputer and the hardware system was defined as a byte of control data to determine the operation that was to follow, followed by the required data either to or from the hardware system. The control data byte must be stored to continue the desired operation. The chosen storage device was an 8 bit transparent latch, labelled the operation control latch. IO_CNTRL must also provide the clock (CTLCLK) for this latch at the correct time. The data in the control byte will therefore determine the future operation of IO_CNTRL. The complete operation of IO_CNTRL is specified by the flow diagram in Figure 5.7, and the ASM listing is provided in Appendix B.1.

5.3.4 The Video Data System

The primary principle of operation of this system is the manipulation of video image data. It is therefore essential that an accurate digital representation of the image is obtained. From the method of calibration, 256 pixels will be used to describe each of the 256 lines of the image. The details of the video data have been explained (sections 2.1 and 5.1) and this section deals with the extraction of the digital values from the video signal.

The time dependent relationship of the signals used to generate a display from analogue video data are shown in Figure 5.1 (b). The duration of each line of data is 64 μs, 4 μs of which is the line synchronization signal, and two more 4 μs segments for the front and back porches [*1-4*]. The total amount of video data in the line occupies 52 μs. This segment must be sampled at 256 regular intervals to obtain the digital representation. This corresponds to a sampling frequency of 5 Mhz. The camera outputs two interlaced fields of 256 lines each. There is consequently very little difficulty in chosing one or other of the fields to provide the image, at a reduced resolution.

The video fields are physically located at different points in the image, vertically displaced from each other by a distance of one video line. To ensure the accuracy of calibration is not compromised, the same field of data, either odd or even, will have to be used for the raw calibration data. During the reconstruction both fields will be used to provide a non-flickering display. The field currently being supplied by the camera will have to determined, as well as when the field starts, and the start of each line within the field. There are devices available which separate the component timing signals from the composite video signal. One such device is the National Semiconductors Corporation LM1881 [*77*]. By using various analogue integration and differentiation methods, this device supplies the field and line synchronization signals and the field type being displayed. The device requires a specific analogue input voltage base level, and the output from the camera must be terminated according to transmission line used, in this case a 75 Ω coaxial cable. Proper termination will ensure the maximum power transfer to the receiving circuit. The design to achieve this is shown in Appendix B.2. It should therefore be relatively straightforward to detect the lines and fields in the image from the information provided by the LM1881.

It is essential that the digitisation of each line begins at the same point relative to the start of the line. If this condition is not adhered to, the digital samples of each line in the image may be horizontally skewed relative to the other lines, providing an inaccurate representation of the image, which will lead to errors in the calibration. Some method must therefore be found for synchronising the sampling clock, that is, the pixel clock, to the start of each line. The most common method of synchronising two asynchronous oscillating signals is to make one the input to a D-type latch whose clock is driven by the other signal, as shown in Figure 5.8 (a). In this case the clocking signal would be the pixel clock and the input to the D-type, the line synchronization signal. This method has one disadvantage. Since the output of the D-type does not appear until the rising edge of the clock, the input signal may have

changed state at any time during the previous clock cycle, resulting in an uncertainty as to the exact moment of change of state. In most applications this uncertainty is of no consequence, but it was decided that the highest possible level of accuracy was required in this application. The method chosen to achieve the necessary synchronization used digital delay lines.

A digital delay line delays the output value relative to the input value by a specified and variable length of time [78]. The devices were used to provide an oscillator system whose cycle could be lengthened so that the rising edge of a cycle coincided exactly with the rising edge of the end of the horizontal synchronization signal. The principle is shown in Figure 5.8 (b) and (c). The delayed output is inverted and applied back to the input of the delay line to create an oscillator. The period of oscillation T_{osc}, is given by the relationship

$$T_{osc} = 2 \times (T_d + T_{pd})$$

where T_d is the delay provided by the delay line and

T_{pd} is the propagation delay of the invertor.

The synchronization is achieved by using the line synchronizations signal to stop the inversion and therefore the oscillation, just before the rising edge of the line synchronization. The clock will therefore be held low until the synchronization signal is high again. The signal to stop the inversion is derived from the line synchronization signal and a slightly delayed, inverted sample of itself as shown in Figure 5.8 (c). The frequency of the synchronized clock obtained by this method was chosen to be 20 Mhz to drive the various ASM designs that would be required. The pixel clock was obtained from this clock by dividing the frequency by 4 to obtain the required 5 Mhz pixel clock.

The video timing signal obtained from the LM1881 were also used to generate the display signal, since for reconstruction the display of the output images and capture of the input images will have to be simultaneous during continuous operation. To accurately digitize the input video, the base voltage levels of that signal were clamped to a predetermined reference level. The circuits to perform this function are shown in the schematic diagram in Appendix B.2. The ADC chosen for this application was the Hitachi HM19209C [79,80]. This device has an 8 bit resolution and a maximum conversion rate of 30 million samples per second (msps). It required two voltage reference levels which determined the base voltage level and the range of voltage conversion. It was decided to make these levels variable by the

user to provide the maximum resolution of the input voltage level range from the camera. The design is shown in Appendix B.2.

5.3.4.1 Related PAL Designs

The controlling PAL for this section, a PAL-C22V10 labelled TIMING, must use the information provided from the LM1881, and the data in the operation control buffer to:

1. Decide when valid data is present for subsequent operations.
2. Provide a clock signal for the counters that address the frame buffers to read and write the video data, and other controlling signals for the counters.
3. Decide when to capture and display video frames.

Counters will also be needed to address the memory for the video frame data. The basic design of the counters was the same as that for the interface. However, to decide when valid video data was present, variable lengths of time will have to be measured from various reference points. The first time period will be from the end of one field to the start of the next, which establishes a reference point for the start of the next frame. The LM1881 provides the vertical synchronization and odd/even field signals, whose relationship will specify when the next field starts [*77*]. There are a number of equalisation pulses that must be counted before data will be present on the camera's output. The PAL device, VCOUNTER, a PAL-C22V10, performs the function, and in addition to the previous outputs, asserts a signal labelled DONE, when the specified number of equalisation pulses have been counted to indicate that the field's video data is now valid. It then reverts to counting each video line to provide the higher 8 bits of the pixel address. After 256 lines, ripple carry out is asserted to indicate the end of the required data from that field. The relationship between these signals is shown in Figure 5.9.

PAL device HCOUNTER performs a similar function, but counts the time gap between the end of the horizontal synchronization signal and the start of the video data on each line, that is, the front porch period [*77*]. A ripple carry out is asserted at the end of each line when 256 pixel locations have been sampled. The clock for both of the counters is derived from the 5 Mhz pixel sampling clock, as determined by TIMING. The operation of TIMING is completely specified by the flow diagram in Figure 5.9.

5.3.5 The Frame Buffers

There are three parameters that define the frame buffers: the size of memory required for each buffer, the speed of operation, and the method of access between each elemental block and the buffers. These parameters will determine the design of the bus system to perform the communication and allow specification of memory devices that may be used for the frame buffers.

The size is defined by the resolution of the acquisition and display systems, that is, 256 pixels square of eight bits each, to give 64 kilobytes. At a sampling rate of 5 Mhz, each pixel will be present at the input for 200 ns. The total access time to read or write each pixel cannot therefore exceed this value. For the display of a pixel an address must be generated and the LUT read to give the new source address. This is expressed by the relationship

$$T_{lut} + T_{mem} + T_{add} \leq 200 \text{ ns}$$

where T_{lut} is the time to access the LUT

T_{mem} is the memory access time

T_{add} is the time to generate the address

The value of T_{add} is equivalent to the propagation delay of the address counters, 25 ns. This leaves 175 ns for the two memory addresses. Assuming the same memory devices are used for the LUT and frame buffers, the read and write cycle times for the memory cannot exceed 87.5 ns.

The interface to these memory devices must be accomplished through tri-state buffers to separate the read and write addresses from each block of the system, to allow the control sections to access the various elements simultaneously. The frame buffer memory will have to be accessed by the host system, the video system for writing the data into the memory and the LUT for displaying the data. The Hitachi 62256LP-85 [*81*] meets these criteria. It is a low power consumption static Random Access Memory (RAM) device, 32K x 8 bits in size, with a minimum access cycle time of 85 ns.

Static memory was chosen over dynamic or video memory in order to reduce the complexity of the timing signals required for operation. Dynamic and video memory devices require periodic refresh control signals to maintain the memory storage. They are slightly less expensive in comparable sizes and are available in faster access times. It was decided that the increased complexity the use of such devices

demanded did not outweigh their advantages in this application, in view of the relatively small amount of memory required.

5.3.5.1 Related PAL Designs

The controlling PAL for this memory system would have to ensure that the timing constraints of accessing the memory devices was adhered to. This involved ensuring the right address was presented to the device for the data read operation, and the right address and data for a write operation. The control was divided into two fields of operation: input and output accesses and video accesses. Two PAL devices, MEMIO and RUN respectively, were assigned control over the two operations, with the transfer of control determined by the value in the operations control latch.

MEMIO directs the data from the links and the IMS C011, to and from the memory areas via the data and address bus buffers. This device is an EP-600 programmable component [*70*], available in a 24 pin DIL package, with two clock inputs, and 20 input/output pins. In write operations, it also provides the strobe signal to transfer the data into the memory. For a read operation, the system assumes that the data will be present at the outputs of the memory devices within the specified access time. Since two devices (32K apiece) are required for each frame buffer, it also controls the switching between the devices, to provide full 64K addressing.

RUN, another EP-600, controls the flow of data to and from the memory when video operations are required. The video system operates in one of several modes:

1. Capture a single video field
2. Display a single video field
3. Capture a field into one buffer while synchronously displaying the other
4. Continual reconstruction, where the data for every video field displayed is obtained from the locations specified by the LUT, while the data from the next field is captured into the other buffer.

To achieve the first objective, the addresses from the line and pixel counters must be applied to the specified buffer, and the data from the analogue to digital converter corresponding to the pixel location is written into the buffer at the correct time. For the second objective, the addresses from the pixel and line counters must be applied to the LUT to obtain the correct data for that display point. The data from the LUT is then applied to the frame buffer as the new output address. The data thus obtained from the memory buffer is directed to the digital to analogue converter for display.

Since the display and capture of video fields is performed at times determined by the same video signals, the achievement of the third goal is a combination of the first and second goals. The address for the frame being captured is applied from the counters to the specified buffer, and the address from the LUT is applied to the source buffer for the display data.

The main purpose of the design is to operate in continual reconstruction mode. The capture and display operations are the same as described above. The destination and source buffer is determined by the field, even fields are written to buffer 0 and odd fields to buffer 1. The controlling signal for this operation is derived from the logic level of the ODD/EVEN output of the LM1881. The software listings for RUN and MEMIO are provided in Appendix B.1.

5.3.6 The Look Up Table

The LUT must hold a 16 bit address for each of the 65536 pixels in the image, giving a total size of 128 Kb, arranged in a 64 K by 16 bit block. This arrangement requires 4 Hitachi 62256-8 devices. It must be accessed by the video counters and the input/output counters. The two address buses must be isolated from each other by tri-state buffers. The data from the LUT must be available to the input/output interface as consecutive bytes, and to the frame buffer memory as a 16 bit word.

The data will be arranged in the LUT such that each 16 bit word is made up of an upper and lower byte. When access is required for input/output purposes, the lower bytes are read first and then the upper bytes. For video purposes both bytes are placed on the data bus to the address inputs of the frame buffers. The pin out for the Hitachi HM62256 devices is identical to 32 K by 8 byte Erasable Programmable Read Only Memory (EPROM) devices. This meant that EPROMs could be used as a direct replacement for the static, volatile memory. The data for a particular fibre bundle could therefore be stored in EPROMs, and need not be loaded before reconstruction, in keeping with the one time calibration principle. The system could therefore be used as a stand alone reconstruction system if the restoration of the LUT data did not have to come from a host.

5.3.6.1 Related PAL Designs

The method of data communication between the LUT and the input/output interface is the same as for the frame buffers as the devices are identical. Therefore MEMIO was used for this purpose, with a controlling input bit to determine the path of the

data. When data is sent to or read from the LUT, the devices have to be selected at the right time to enable the data transfer to take place. The memory area was divided in four areas: high and low bytes and upper 32 Kb or lower 32 Kb subsections. The subsections were selected on the basis of the logic level of the highest address bit, and the high and low bytes as determined by a controlling bit sent by the host.

In video operations modes, the source address for display is applied to the LUT, again via bus selection and isolation tri-state buffers. The data from the LUT is continually addressed according to the pixel sampling clock, and is directed to a frame buffer as determined by a controlling bit from the host, or whether the frame is even or odd. To ensure that the address and data timings conformed to the specifications required by the HM62256-8, the address from the LUT is stored in a D-type latch for one extra clock cycle to ensure its presence at the address inputs of the frame buffers for the correct length of time. All the control signals for the video mode operation of the LUT are provided by RUN.

5.3.7 The Display Mechanism

The display of the digital pixel values must involve their conversion to an analogue value. The video timing signals must also be included to create a composite video signal to drive the display monitor. Since there are two possible sources of display data, the flow of data from the two frame buffers into the digital to analogue converter must be selective. Selection of the data source will be determined by controlling signals from RUN.

The device chosen as the display driver was the Inmos IMS G178 [*82*]. This device has three 8 bit digital to analogue converter display drivers. It also contains three read/write look up tables, each 256 bytes long. These look up tables are addressed by the digital pixel value applied from the frame buffers. Each LUT produces an output value for that address. The three digital values are applied to the analogue to digital converters to produce three analogue output voltages. These values can then be used to drive the red, green and blue (RGB) inputs of a colour monitor to provide pseudo colour from grey level inputs. The colour balance can be altered by changing the values in the three LUTs. In a grey level display system, any one of the DAC outputs can be used to provide the display data.

The device also has inputs to provide the timing for the video synchronization signals. The pixel data and the synchronization pulses are mixed to provide a video

signal to the RS-170 standard. The video timing signals were the same as those used in the rest of the design and derived by the LM1881 from the input video signal. Default values for the three LUT entries are not specified, and therefore must be initialised before a display can be obtained. Access to the IMS G178 must be given to the host for this purpose. The device has two controlling bits that determine the destination for any data presented at its inputs. A control system was therefore required to ensure correct operation of the IMS G178.

5.3.8 Related PAL Designs

The control system was developed around a PAL-C22V10-25 device, CLUTCTRL, and a data latch. The control byte to initiate access to the IMS G178 disables IO_CNTRL and enables CLUTCTRL. The next byte will be latched to define the destination of the following data. The IMS G178 has an automatically incrementing address system, so there is no need to provide an input address. CLUTCTRL performs all the handshaking operations required for data transfer to and from the IMS C011. When the transfer of data to the IMS G178 is complete, CLUTCTRL hands control back to IO_CNTRL. The system will then be ready for the next step in operation.

5.4 Construction And Test Procedure

This section describes the construction method of the system and the test procedures used to ensure correct operation of the design.

5.4.1 Construction

For prototype construction of electronic logic circuits, there are two predominant methods: the wire wrap method and the speedwire method. Both methods rely on the forced displacement of insulation from the wire to make the electrical connection. The speedwire method allows several points in the circuit to be connect by a continuous length of conductor, whereas the other method requires a separate length of wire for each connection. The wiring pins in the wire wrap method also project to a length of about 12 mm from the board and thus requires more space than the speedwire method, which has wiring pins about 5 mm long. The wire wrap method does however, produce a more physically robust construction.

The speed wire method was chosen for this application as it would allow faster construction of the circuits, and since field testing would not be carried out on this first prototype, robust construction was not a high priority. The board chosen for the layout was a Verospeed IBM format expansion card with a ground and power plane. These two planes have lower power rail interference than construction methods that use individual power lines to every device in conjunction with high signal switching speeds. The entire assembly, with a 2 Mb Transputer Board, was placed in a metal enclosure for physical protection as well as greater immunity from external electromagnetic radiation noise. The assembly can be seen in Plate 5.1.

5.4.2 Test Procedure

The system used for the following test procedures is shown in Figure 5.10. The use of the two Inmos IMS B004 boards is to make the analysis of terminal problems easier, since the main unit, running the TDS is not affected by problems at the lower levels of the hierarchy, that is the hardware system and the second Inmos B004 [*72*]. The following aspects of the design had to be tested to ensure correct and accurate operation of the system:

1. The operation of the input/output system,
2. The video system components, ADC, DAC, and timing signals,
3. The operation of the LUT,
4. The ASM designs had to be verified, and
5. The video operation in capture, display and reconstruction modes.

To check for correct operation of these features several test routines were written. The software routines to execute the testing are provided in Appendix B.3.

The input/output system had to direct the data to the correct location. Therefore every byte location had to checked. To achieve this a program was written to write a sequence of different bytes to consecutive locations, and the stored data was read back into the host and checked against the original data. This operation was performed on the frame buffers, the LUT and the control byte locations. To ensure durability the test was repeated 1000 times for each location, which allowed the system to reach an equilibrium, or working temperature. The relevant timing signals were inspected during the test procedure, using a 60 Mhz cathode ray oscilloscope and a Hewlett Packard 1630D logic analyser. After small timing problems, mainly due to noisy signals, were corrected, the system was again tested as before, until correct operation was reliably obtained.

Testing the video operation was more complicated because of the different elements involved, and the qualitative nature of the assessment of some of the results. The testing was separated into three areas: the quality of the digitisation procedure, the read/write operations into and out of the frame buffers, and the display quality of the output image.

Before the input could be tested, in order to display the test results, the display mechanism had to be tested. This was achieved by initialising the IMS G178 with direct LUT entries, that is, each address in the LUT contains the value of that address, which would not alter the data applied to the digital inputs. A memory buffer was then filled with a test image where every line in the image consisted of pixels containing the line number. The first line was therefore made up of pixel value 0 corresponding to absolute black, to the last line with pixel value 255 for white. The displayed image should therefore be a continually increasing intensity from black at the top to white at the bottom. The display was then visually examined to ensure that the expected output was obtained. The overall level of brightness of the display could be adjusted by varying the current supplied to the reference input of the IMS G178. This was varied until an adequate level of brightness was obtained with good definition of the shades of grey in the image. The test was repeated with the image varying from black to white, from the left to right, to check the alignment of the pixels between lines.

With the display mechanism verified, the input could be tested. To test the quality of the digitisation of the camera's output, the output from the ADC was connected directly to the digital pixel inputs of the IMS G178. The video image from the camera was displayed on a monitor identical to that used for the output from the IMS G178. The voltage range and the base reference level for the HM19209 were then adjusted until the best possible match was obtained between the two displayed images. The test was repeated with the data from the HM19209 stored in a frame buffer and then read for display by the IMS G178. This would show any pixels that had been written to the wrong locations because of incorrect video timing and pixel address alignment.

Displaying pixels with addresses from the LUT was the next aspect to be tested. The LUT was first filled with a direct table, which should not affect the location of any data in the frame buffers. Having verified that this was so, sets of data were generated that mirrored, inverted, and swapped diagonal quadrants of the image. These would check the left to right and up to down operation of the LUT, and implement an almost

random address output to check for any interference that may occur when all the address lines are oscillating between logic levels at the same time.

After the display system was checked, the feature of the LUT which would copy pixel values to multiple locations to provide a full display image from the calibration data was simulated. This was implemented by creating a data set where every address value at an even line and pixel address was duplicated by its neighbours at memory address locations one pixel higher and one line higher. Every odd pixel address location will therefore be a copy of the previous even address, which yielded three copies plus the original memory location address value, as shown below, where x and y are the entry points and A_1 is the address at that location.

(x,y), (x+1,y)

A_1	A_1
A_1	A_1

(x,y+1), (x+1,y+1)

This halved the number of lines and pixels available for display, and therefore reduced the resolution of the system by a factor of 4.

To ensure that erroneous operation would not occur if non-functional values were sent to the controlling registers and irreversible operational states were entered on power up, the PAL devices had been designed to revert to a 'wait for command' mode in such an eventuality. This was checked by deliberately writing values that did not perform any operation or instructed conflicting operations, to the controlling registers. Any errors that occurred from this procedure were corrected by modifying the PAL designs, until the system performed as required.

The display, capture and reconstruction operations were checked by displaying various test patterns to ensure that the pixels were in the right location, capturing test card patterns that were drawn by an Apple LaserWriter® [*83*] from postscript programs [*83*] that duplicated the display patterns, and the reconstruction was verified by writing data sets to the LUT to provide the various effects as described above.

To test the simultaneous display and capture of the same image which will be needed for the next calibration method, the camera was focussed onto a monitor displaying the test patterns, its output was captured, and the original and captured data were compared. None of these tests used optical fibre bundles or the data associated with them. This was deliberately done to ensure that the basic system

functioned correctly before introducing the variables associated with the transmission of optical images through optical fibre bundles. These tests showed that the system functioned as intended, and could be used to implement the calibration of an incoherent optical fibre bundle and the reconstruction of the images obtained from it. Plate 5.2 shows a mirror image transformation on the input image, performed by the OFCS.

The entire group of test routines were assembled into one software package. The tests were arranged to allow the specific test required to be chosen from an on screen menu. This method would enable complete testing by any other party who was not familiar with the design, but who could appreciate the basic operation of general video systems and the intended use of this particular design.

5.5 Use and Operation of the System - Software Routines

The following is a brief introduction to the actual workings of the system and how it is to be programmed. Sample listings, in Occam, of the procedures are provided to demonstrate various points. A more compete listing is provided in Appendix B.3.

The following is a list of the standard procedures that have been written, all in Occam. They include all the display and capture functions as well as data transfer functions between the host and the system.

```
reset.board()
init.G178()
read.buffer(INT buffer.number,[256][256] BYTE buffer,BOOL OK)
write.buffer(INT buffer.number,[256][256] BYTE data,BOOL OK)
read.lut([256][256][2] BYTE lut)
write.lut([256][256][2] BYTE lut)
display.buffer(INT buffer.number,BOOL OK)
capture.buffer(INT buffer.number,BOOL OK)
open.file.write(CHAN OF ANY screen,keyboard,from.filer,to.filer)
open.file.read(CHAN OF ANY screen, keyboard,from.filer,to.filer)
close.file(CHAN OF ANY from.filer,to.filer)
file.buffer(CHAN OF ANY screen,keyboard,from.filer,to.filer,
file.lut(CHAN OF ANY screen,keyboard,from.filer,to.filer,
get.buffer.file(CHAN OF ANY screen,keyboard,from.filer,to.filer,
get.lut.file(CHAN OF ANY screen,keyboard,from.filer,to.filer,
```

Before the board could be used the following initialisation sequence which sets up the display mechanism and prepares the OFCS for the next command must be invoked:

```
SEQ
  reset.board()
  SEQ count = 0 FOR 8
    init.G178()
    ... normal LUT -- This invokes the following sequence
```

The following is a typical method for generating a LUT from a software procedure, and downloading it to the OFCS:

```
INT count1,count2 :
BYTE data :
SEQ
  count1 := 0
  WHILE count1 <= 255
    SEQ
      count2 := 0
      WHILE count2 <= 255
        SEQ
          data := BYTE(count2)
          LUT[count1][count2][0] := data
          count2 := count2 + 1
      count1 := count1 + 1
  count1 := 0
  WHILE count1 <= 255
    SEQ
      count2 := 0
      WHILE count2 <= 255
        SEQ
          data := BYTE(count1)
          LUT[count1][count2][1] := data
          count2 := count2 + 1
      count1 := count1 + 1

  write.lut(LUT)
```

To capture and display a frame of video data, the typical sequences of commands would be:

```
capture.buffer(buffer,OK)
IF
  OK = TRUE
    SEQ
      OK := FALSE
      read.buffer(buffer,image,OK)
  TRUE
    SEQ
      goto.xy(screen,10,1)
      write.full.string(screen,"Did NOT capture !")

display.buffer(buffer,OK)
IF
  OK = TRUE
    SEQ
      OK := FALSE
      display.buffer(buffer,OK)
  TRUE
    SEQ
      write.full.string(screen,"Cannot display!")
```

All of the above procedures were placed into a library of routines, and are made available to all the Occam programs that call upon the use of this library [*76*]. This simplified the software programs that had to be written to carry out a calibration on the OFCS.

5.6 Calibration Using the OFCS

The next step in the development of the project was to test the previous calibration method with the hardware system. The principles would remain the same, but new software would have to be written to handle the new data communication paths and protocols. The system used for this calibration is shown in Figure 5.11, as is the flow of data between the elements of the calibration system. The only common components with the previous trial were the Oriel stepper motors and driver, the pinhole and holder, as shown in Figure 4.4, and the Pulnix camera.

5.6.1 Software Design

The software routines to communicate with every element of the system were already available from the test algorithms. The implementation of the calibration was also available from the previous trials, but written in 'C'. It was therefore a straightforward proposal to rewrite the software, combining the two sets of routines, to produce a complete optical fibre bundle calibration system. However, several new concepts would be implemented, primarily in the image analysis procedures and the data handling algorithms.

Use is made of the parallel nature of Occam to divide the program into two main, concurrent, processes: the motor control process and the image analysis process. This would allow the motors to be stepped while the image analysis was carried out. [*36*] indicates that the maximum step speed for the motors was 500 steps per second. For a step size of 50 μm, this meant every movement of the motors to a new position would take one tenth of a second. A typical number of steps for a 3 mm diameter bundle would be about 8000, or 800 seconds taken to move the motors, before any image processing could be performed. This represents a significant overhead of time, which could therefore be put to productive use by performing the image analysis for the previous location in parallel with the movement.

The following is the outline of the spot calibration on the OFCS. The information was extracted from the TDS program via the fold editor [*76*]. This editor allows the program to be grouped under titled headings that describe the code contained within and then collapsed to leave just the headings. Such headings are preceded by three dots, as can be seen below. A more detailed listing is available in Appendix C. Further description of the Occam procedures and processes can be obtained from [*73,74*].

```
...  Libraries
...  values
...  stepper motor variables
...  LUT
...  timer vars
...  Chans
SEQ
  ...  set up OFCS
  ...  main screen
  ...  get.fibre.dimensions
  ...  clear output array
  ...  clear buffer
  PAR
    ...  start time
    ...  calculate circle of operation and window size
    ...  initialise oriel commands
  SEQ
...  initialise motors
CHAN OF BOOL arrived,captured,end :
CHAN OF INT coordinates :
PAR

  ...  Drive Motors

    finished := FALSE
    ...  send B direction
    ...  the top half
    ...  the bottom half
    finished := TRUE
    end ! finished

  BOOL snap.image, done :
  BOOL stop :
  INT buffer.no :

  ...  image processing vars

  SEQ
    ...  screen
    buffer.no := 0
    total.output := 0
    stop := FALSE
    WHILE (stop = FALSE)
      SEQ
        ALT
          arrived ? snap.image
            SEQ
              ...  get input coordinates
              ...  image procs
          end ? stop
            SKIP

  ...  park motors

...  end time

...  filing operations
```

The method used for finding the outputs of the single fibres was changed to enable the image analysis to be executed more quickly. The principle used was to centre a two dimensional matrix on every pixel whose intensity was above a chosen threshold, and then to find the sum of intensities of every pixel within the matrix. The centre of the matrix whose sum of intensities was greatest, was taken to be the output corresponding to the known input point. When several fibres were illuminated for a single input point, this method would, theoretically yield the brightest spot as the output fibre, and the centre of that fibre as the chosen output point. The principle is shown in Figures 5.12 (a) and 5.12 (b).

Figure 5.12 (a) shows three fibres illuminated by the same input source, to varying intensities because of different overlaps between the source and fibre, resulting in

varying acceptance of light by each fibre as determined by the acceptance cones (section 3.2.2.2). The pixel resolution of such an image is shown in Figure 5.12 (b), which also shows four locations for a 5 by 5 matrix. The size of the matrix is calculated by the program, given a typical pixel area size for the output of a fibre in the image. An odd value is chosen for the matrix dimensions to ensure that only one point could be chosen as the centre, with the use of integer mathematics. The size of the matrix is also chosen such that it's dimensions equal or exceed the typical diameter of the output of a single fibre, so that as large a sample of the fibre's output is chosen in assessing its total intensity. In the diagram, location A would be chosen at the brightest output area, and the centre of that area as the corresponding output point.

In the previous calibration method (Chapter 4), a step size of 40 µm was chosen for the movement of the pinhole. In this method, a smaller step size would yield a greater number of sample input points, and although greater duplication of output points would occur, because of the spreading nature of the operation of the LUT (section 5.2.2.1), the greater the number of sample points, the more complete the output image would be. A step size of 20 µm was chosen after test runs with step sizes from 12 µm to 40 µm. For a 3.5 mm fibre bundle, over the circular area of that bundle, approximately 24,000 sample points would be taken. This value is obtained by equating the radius of the bundle to the number of steps taken to traverse the radius (175 in this case), and using this value to compute the 'step area' of the bundle. Not all sample points would be assigned an output, as not every location would fall on a fibre in the bundle. However, a significant percentage would, and test runs showed that approximately 18,000 pairs of input and output points would be obtained.

Apart from the increased speed of the processor, the new method would yield a faster analysis time as all the checks to assign the output pixels to spots, and then to ascertain whether the output locations were uniform or not, as in the previous trial, need not be conducted. Since the data for each frame was reduced by a factor of 4, a gain in the analysis time was also expected, but not by a similar factor as the previous search routines used a reduced area algorithm (section 4.3.3.3). Since a greater number of sample points would be taken, the overall analysis time was not expected to be dramatically less than the system based on the DT2853 and Opus PC V. The amount of data to be handled for each frame was reduced by a factor of 4, but the number of samples was increased by a similar factor. Any decrease in calibration time would be due to the faster processor and more efficient algorithms.

After analysis, the data generated for the LUT was stored in a file on the host system's hard disk for later use. Prior to all reconstruction procedures, this data must be read back and transferred to the LUT. The results for such a calibration are presented in Chapter 7. During reconstruction, despite the errors, moving images could be clearly perceived, showing the LUT and display mechanism were functioning as required. However, because of the compensating nature of the operation of the LUT, the perceived quality of the image was improved, as expected (section 5.3.2.1).

5.6.2 System Performance Analysis

Since the system met all the design specifications, any judgement of its performance will depend heavily on the picture quality obtained from an incoherent optical fibre bundle. The specification defines the resolution of the images as 256 pixels square, to give a total of 65536 points to define the image. This exceeds the resolution of any optical fibre that was obtainable for the calibration trials, and indicated that the optical fibre bundle would set the limit on the imaging resolution. However the results showed that 2200 fibres in a bundle was adequate to resolve simple images such as text and large objects against plain backgrounds, as will be seen in the results in Chapter 7.

The first obvious gain was the increased speed of data transfer to the host transputer. At a data rate of 1.3 µs per byte, the data for a full frame could be transferred in 85 ms. This meant that the entire data set (24,000 frames) for the calibration could be transmitted to the host in approximately 2000 s. The maximum time taken to capture a field of video data is 40 ms, as a compete field must be present within one frame time, giving a total acquisition time of approximately 900 seconds. Timed tests, using the transputer's clock system [*74*] showed that the image analysis of a field of video data took approximately 0.5 seconds, extrapolating to 12000 seconds for 24,000 fields. The expected calibration time obtained from the sum of these values would be in the region of 15,000 seconds. In practice a larger value, 3.7 mm, was used as the diameter of the bundle to ensure that none of the bundle was uncovered by the scanning of the input light source. This would increase the calibration time. As the commands to step the motors were sent while the image analysis was carried out, this was not expected to significantly affect the calibration time. The overall result was that the calibration took 5.5 hours (19,800 seconds) compared to 14 hours previously. While this is a more practical value, a faster calibration method would be advantageous in an industrial environment, should the system be adopted for commercial purposes.

Having shown that the calibration of incoherent optical fibre bundles is possible and that real time reconstruction is feasible, the calibration of a higher resolution bundle will be investigated next, as well as reducing the calibration time, which with the previously described method, increases linearly with the number of fibres in the bundle.

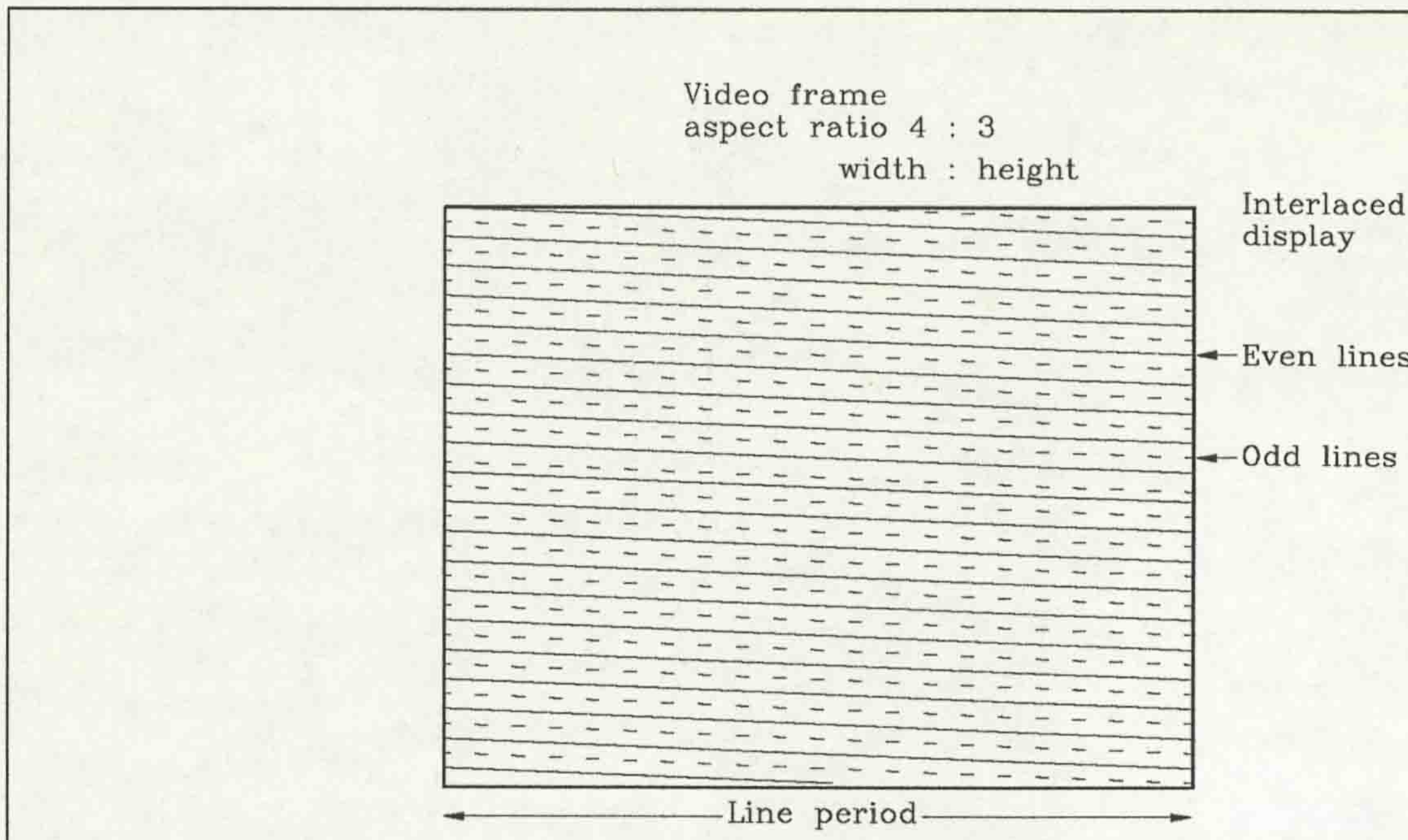


Figure 5.1 (a) – The interlaced display

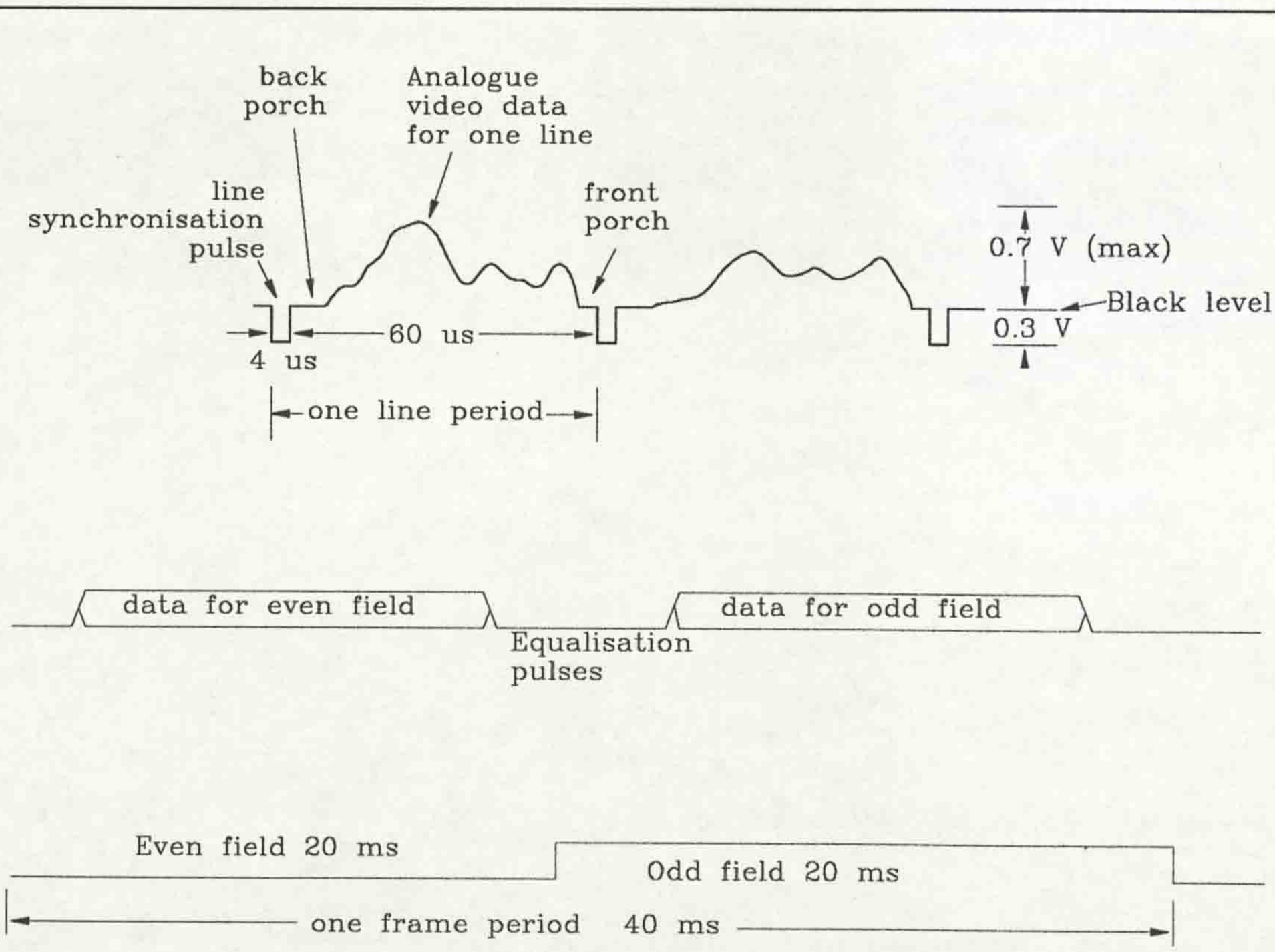


Figure 5.1 (b) – RS170 video timing

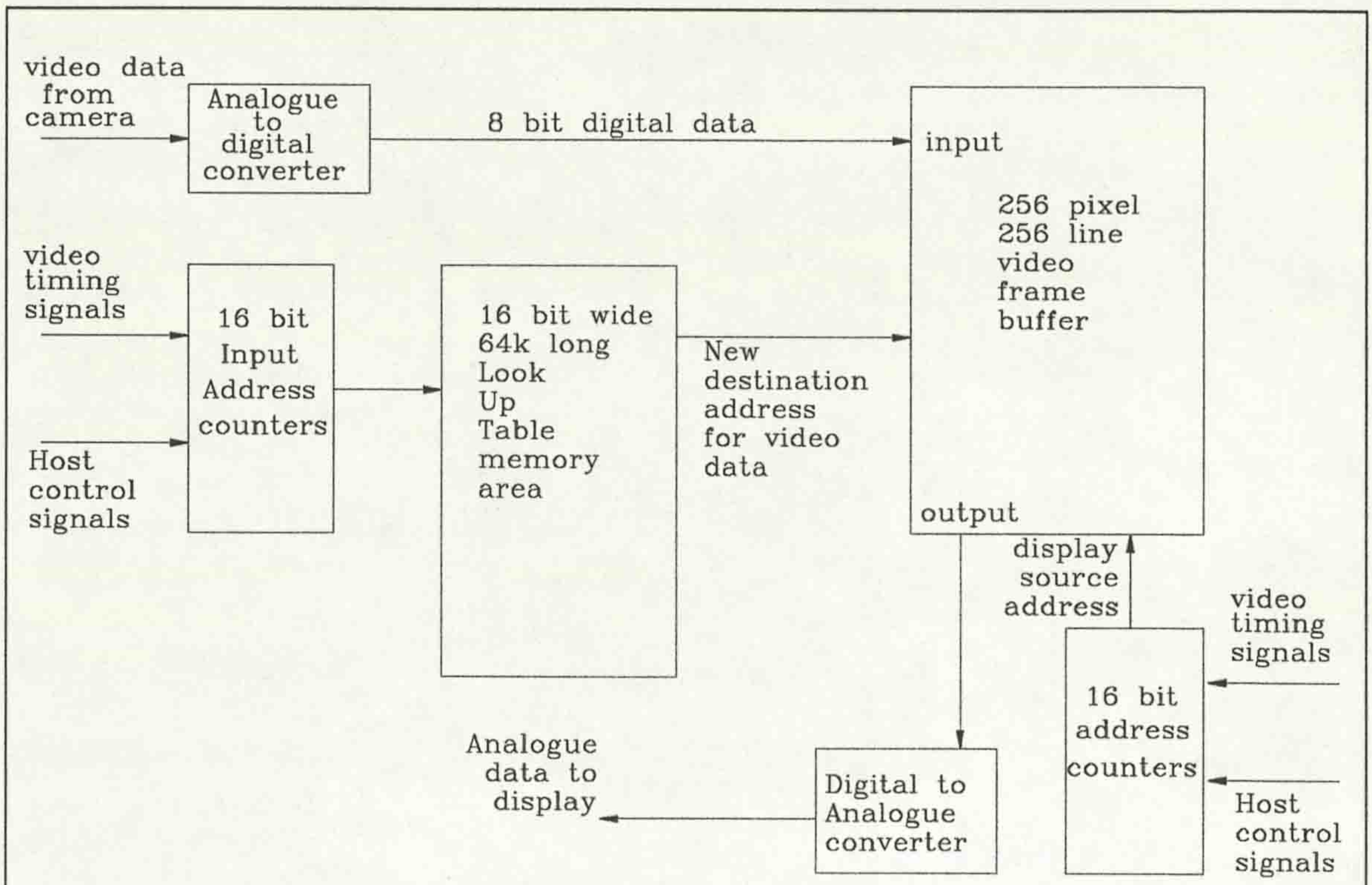


Figure 5.2 (a) – Location of the LUT between the incoming data and the destination buffer

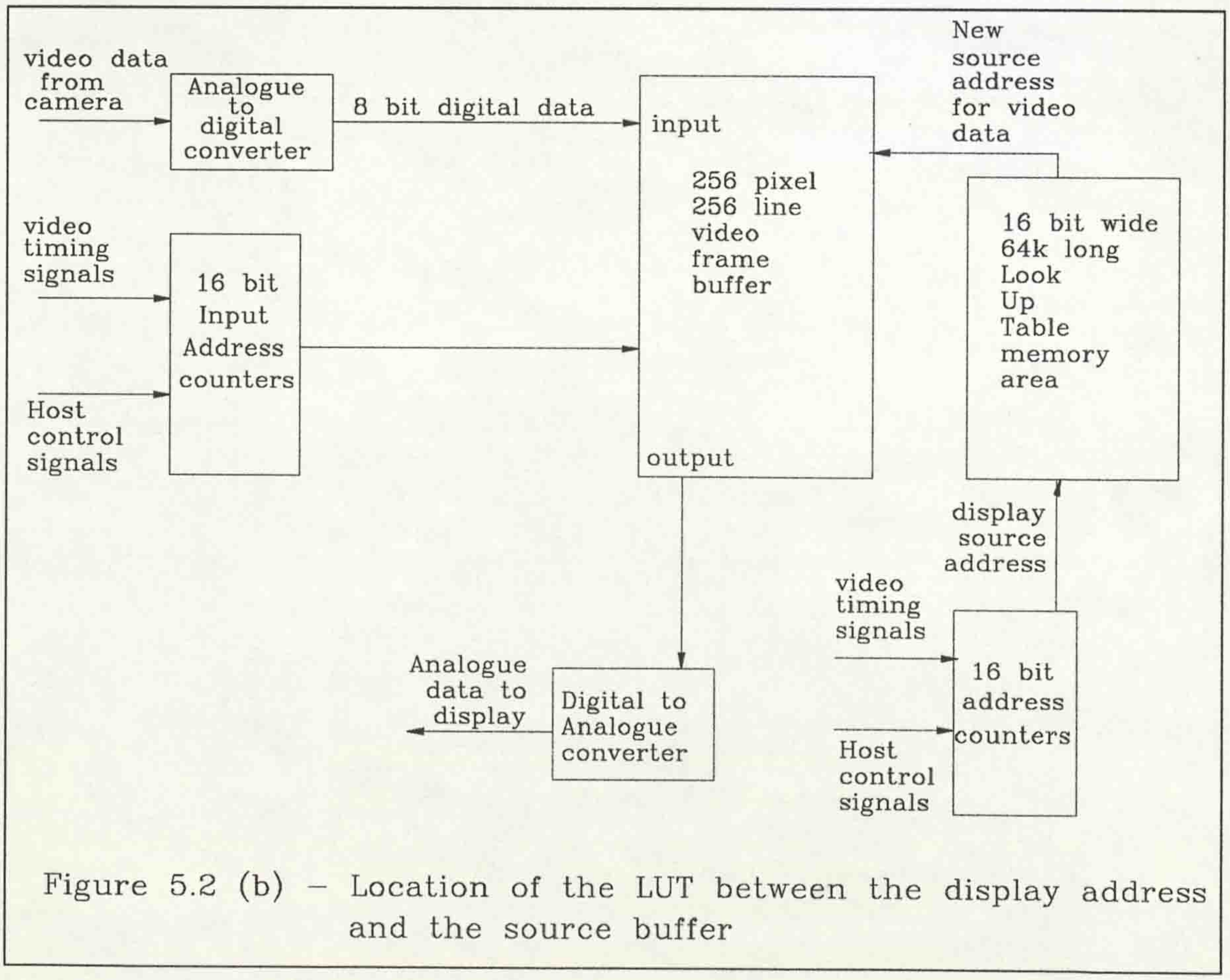


Figure 5.2 (b) – Location of the LUT between the display address and the source buffer

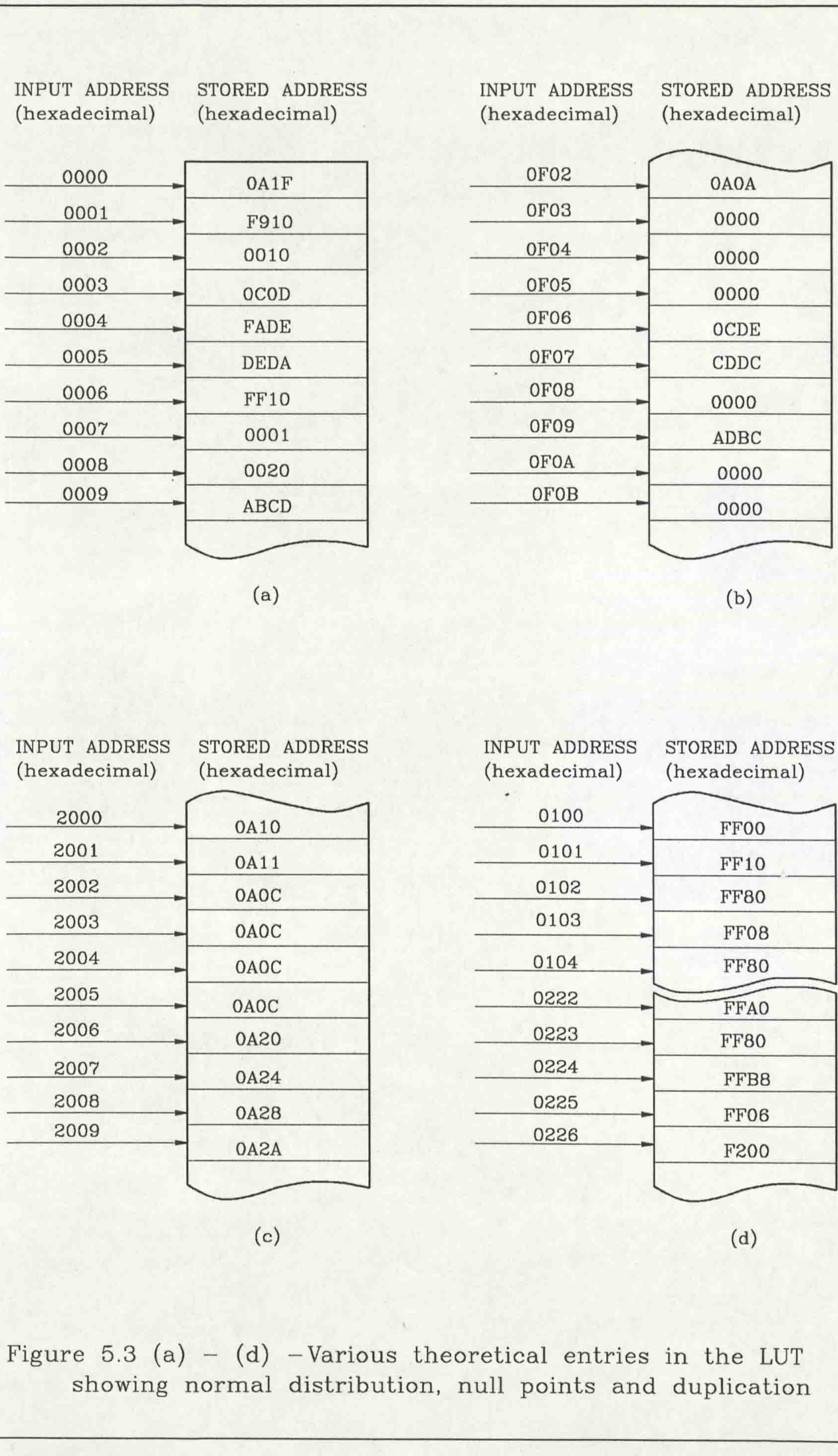


Figure 5.3 (a) – (d) –Various theoretical entries in the LUT showing normal distribution, null points and duplication

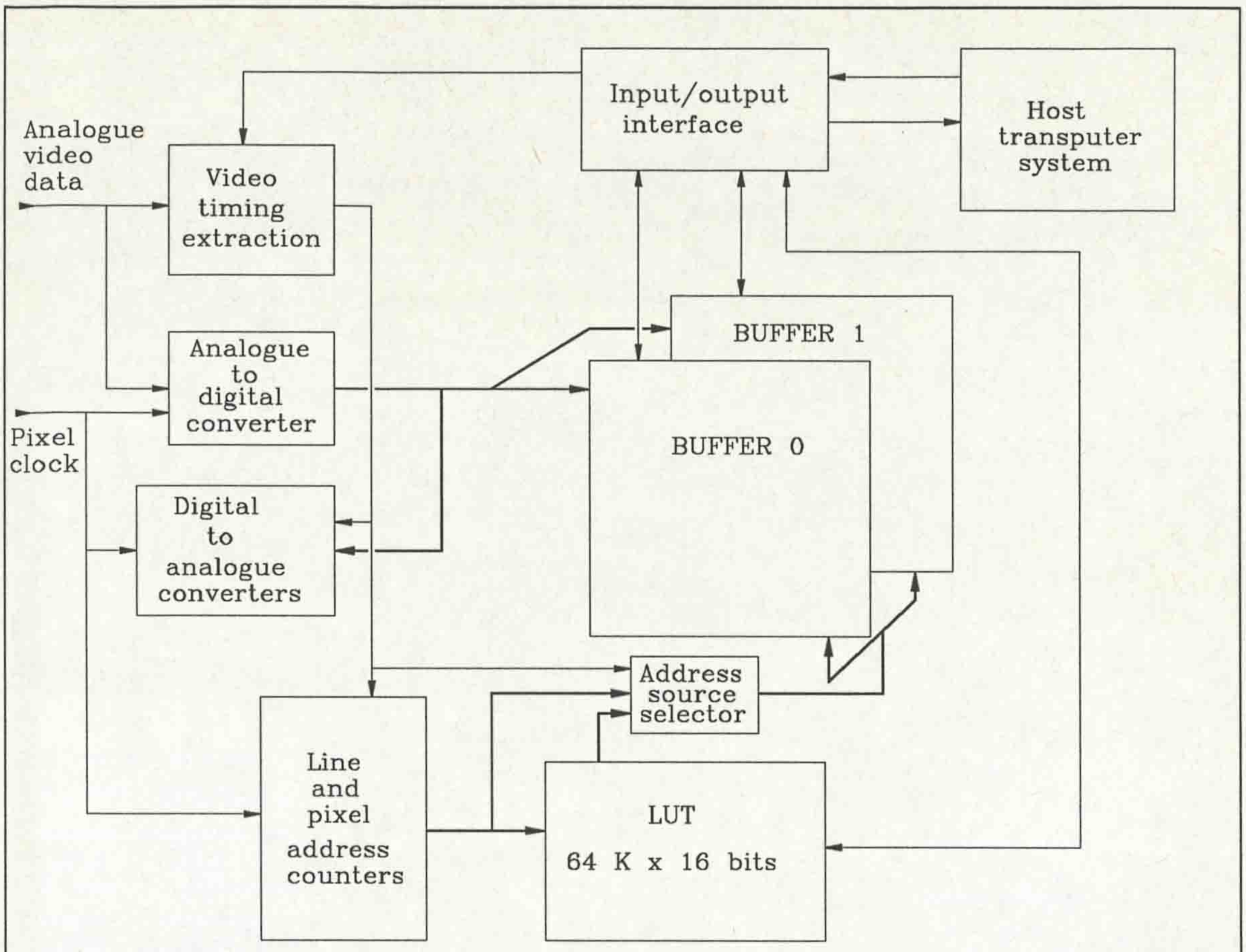


Figure 5.4 – The system block diagram

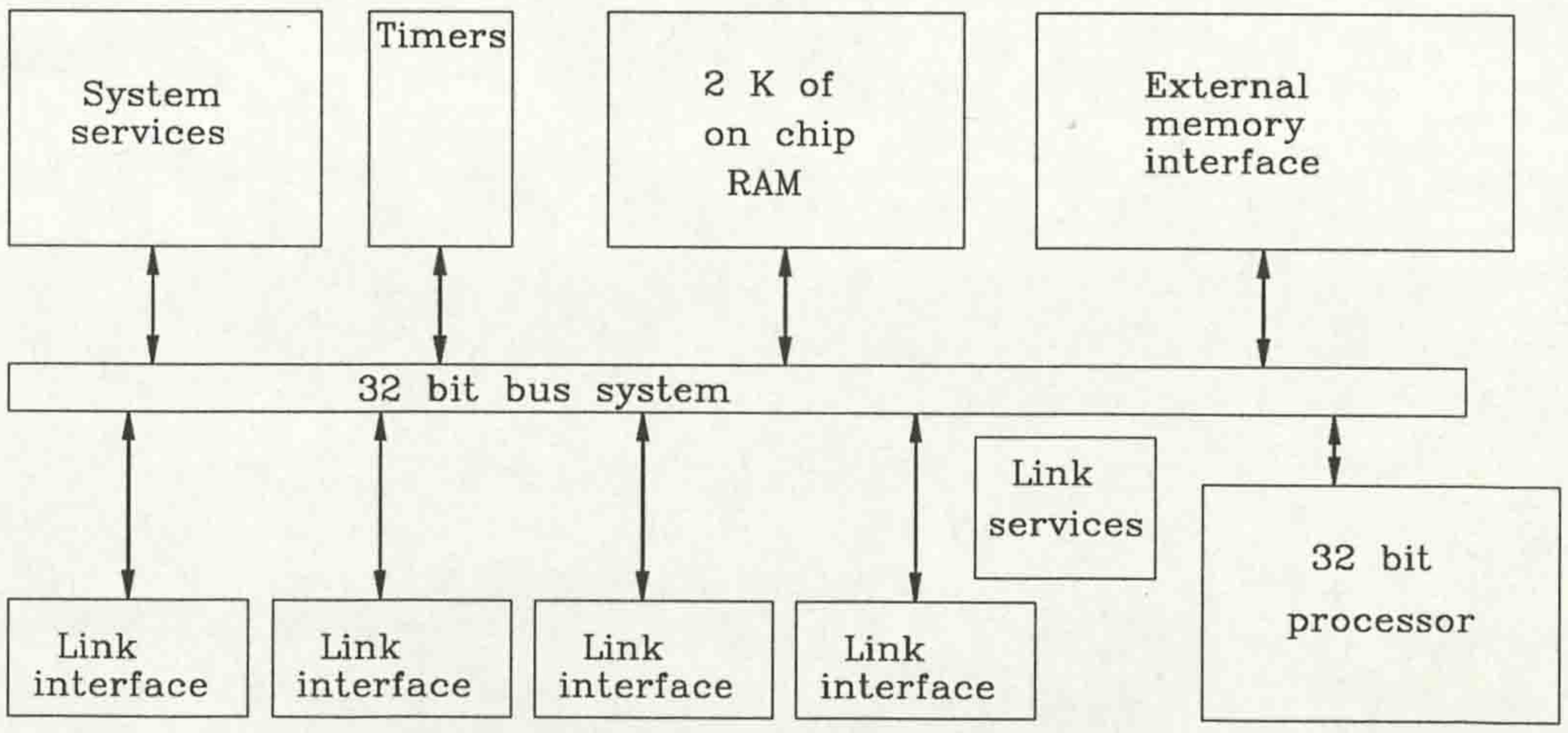


Figure 5.5 – The internal architecture of an Inmos T414

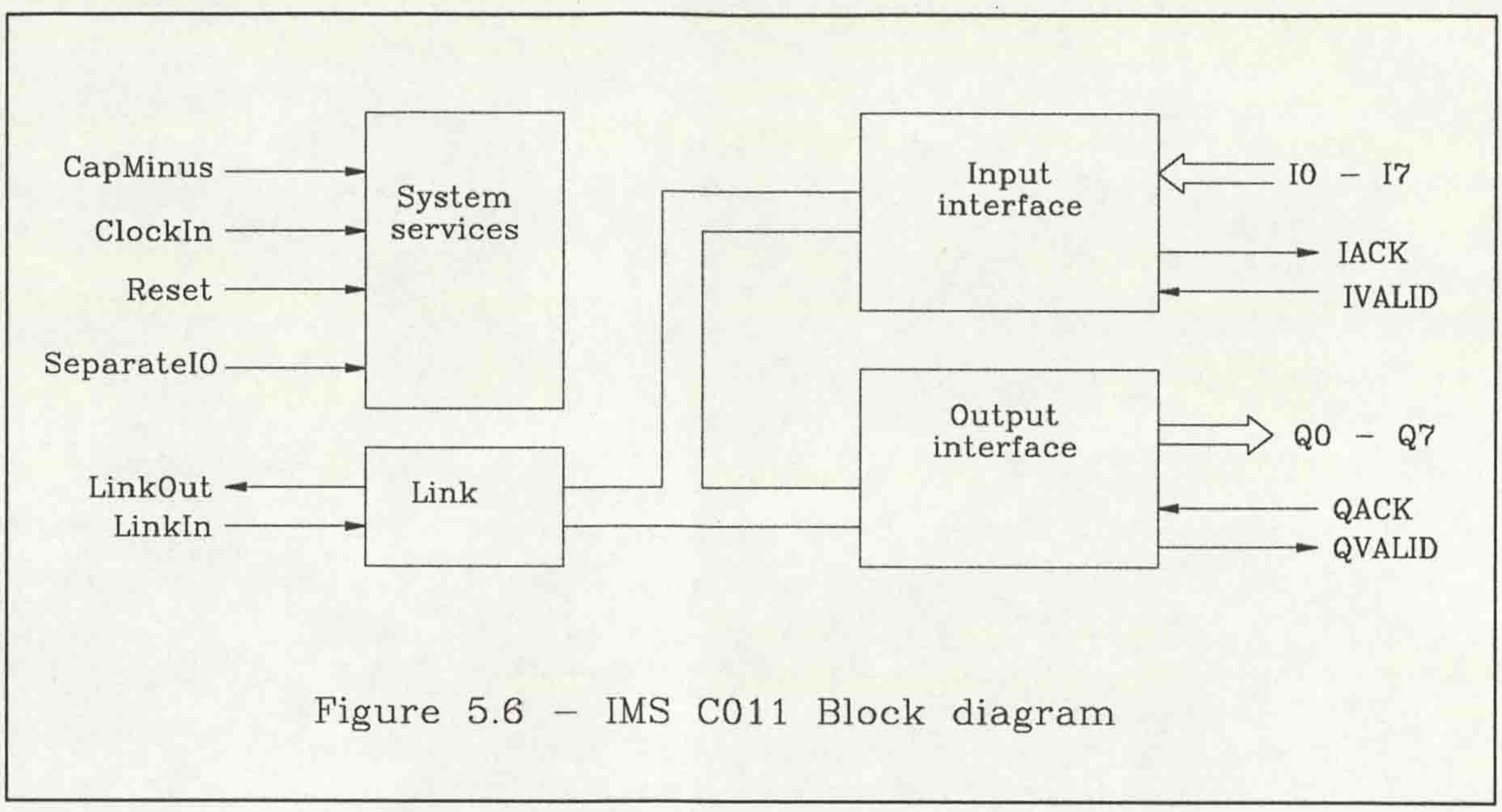


Figure 5.6 – IMS C011 Block diagram

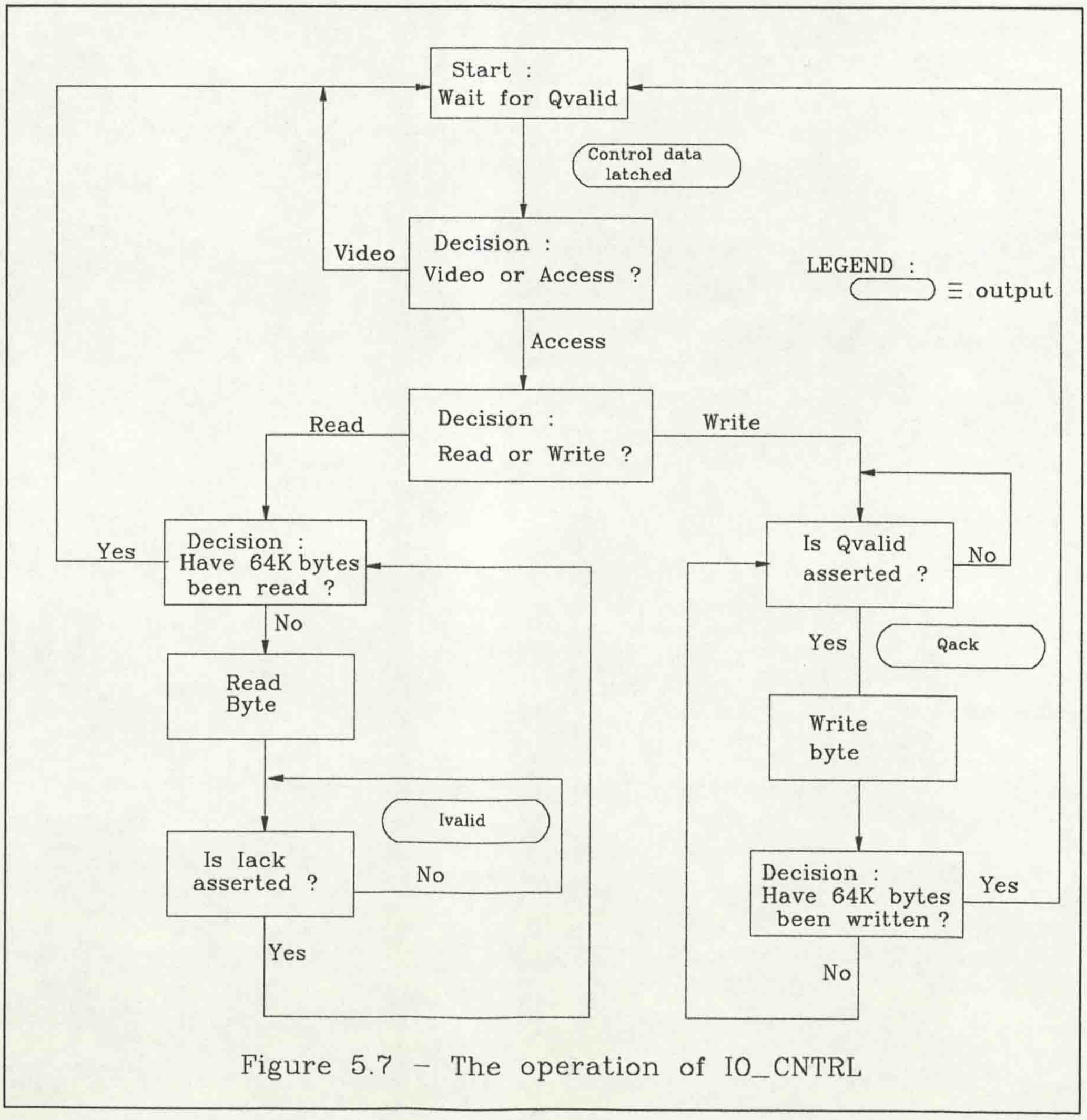


Figure 5.7 – The operation of IO_CNTRL

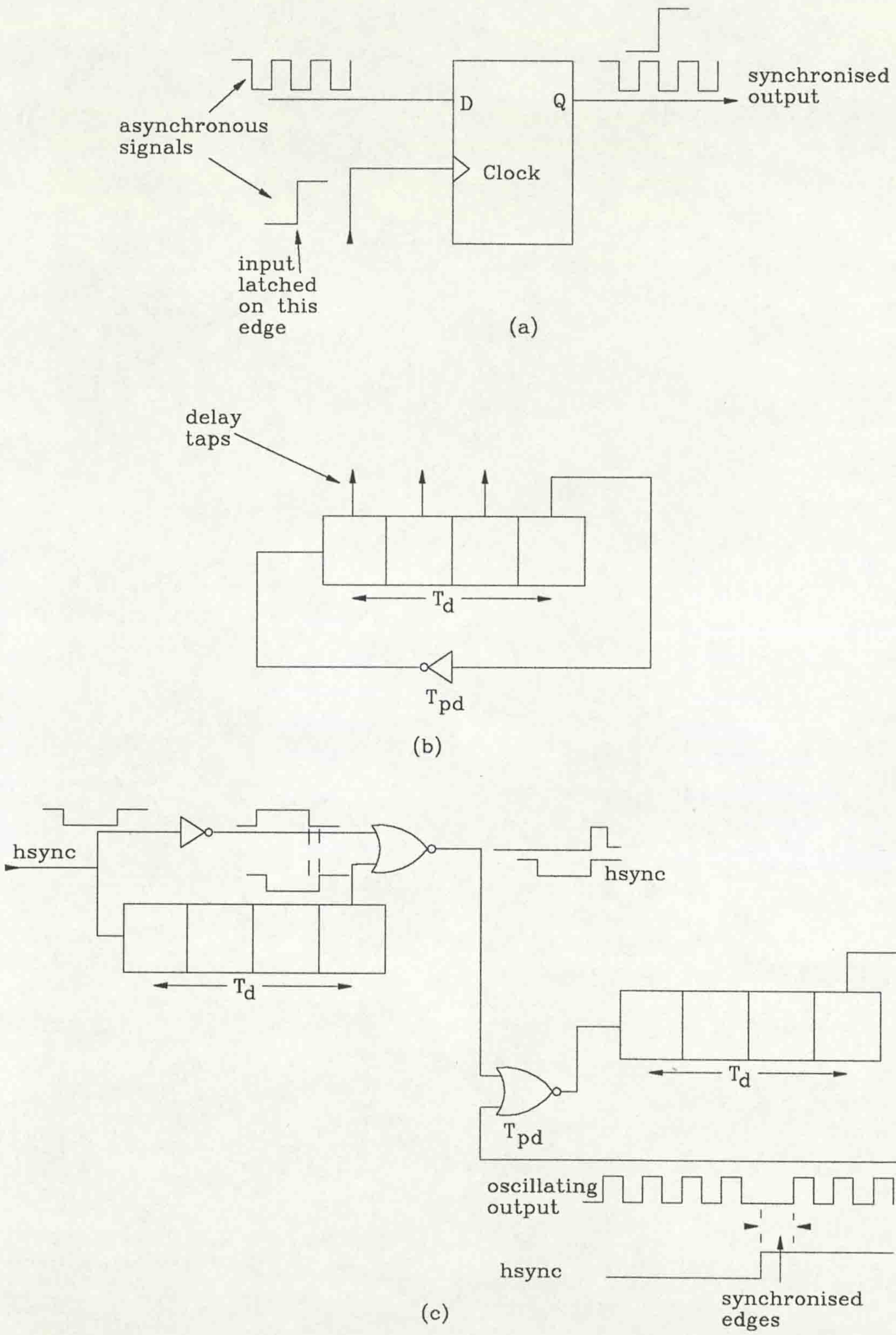


Figure 5.8 (a) – (c) – Conventional synchronisation, oscillating delay line and pixel clock generation synchronised to HSYNC

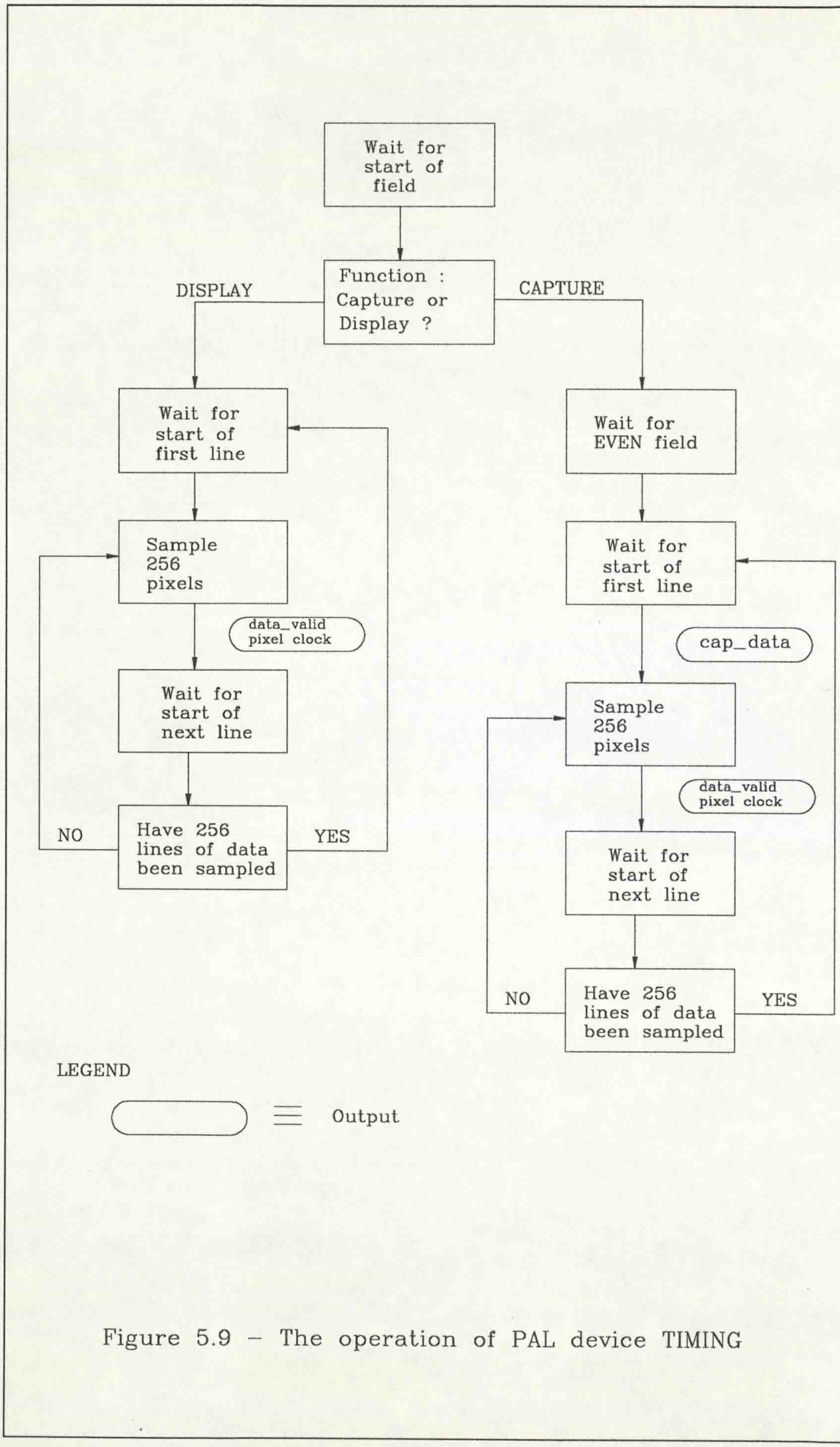


Figure 5.9 – The operation of PAL device TIMING

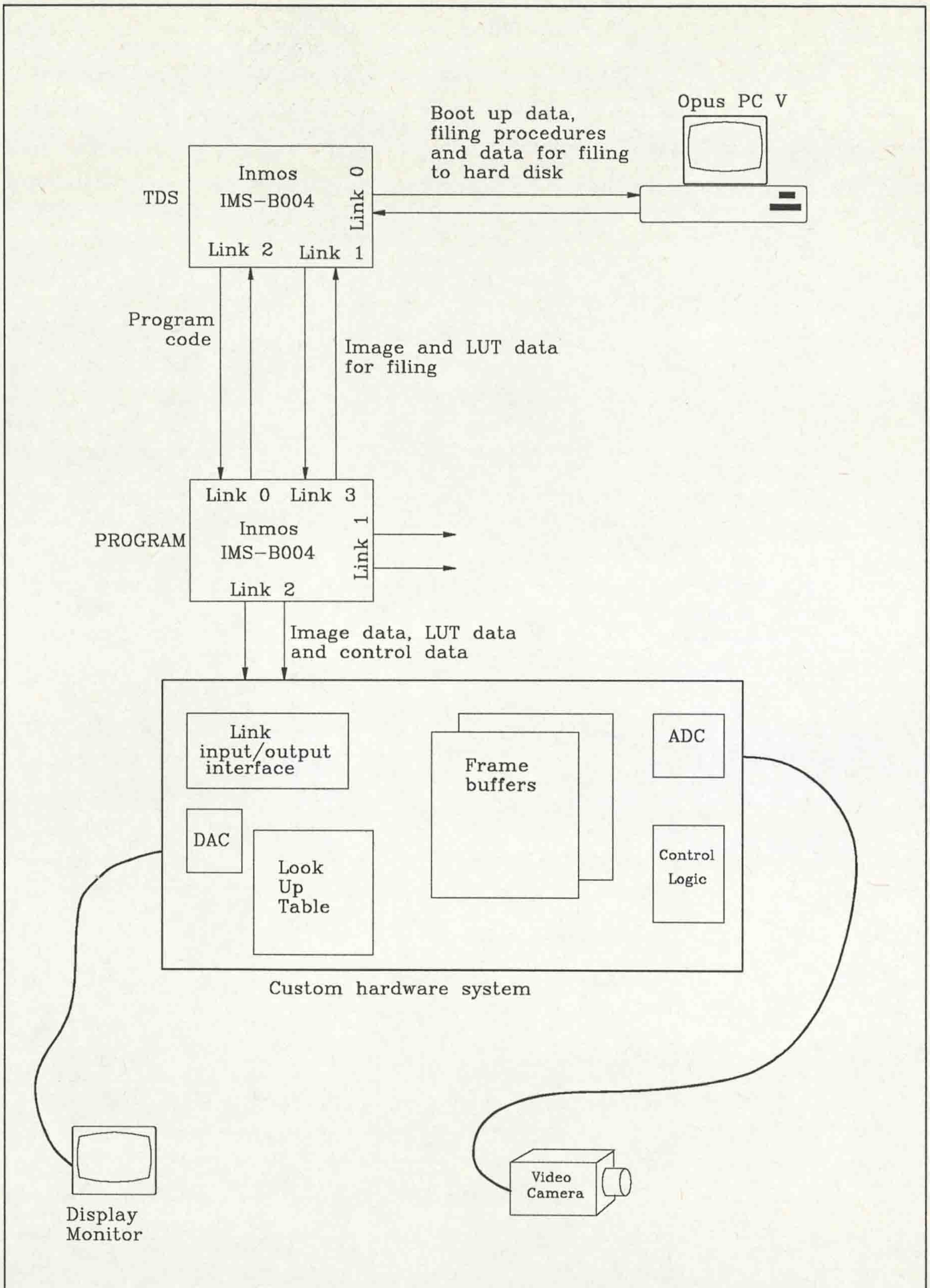


Figure 5.10 – The system used for operational verification

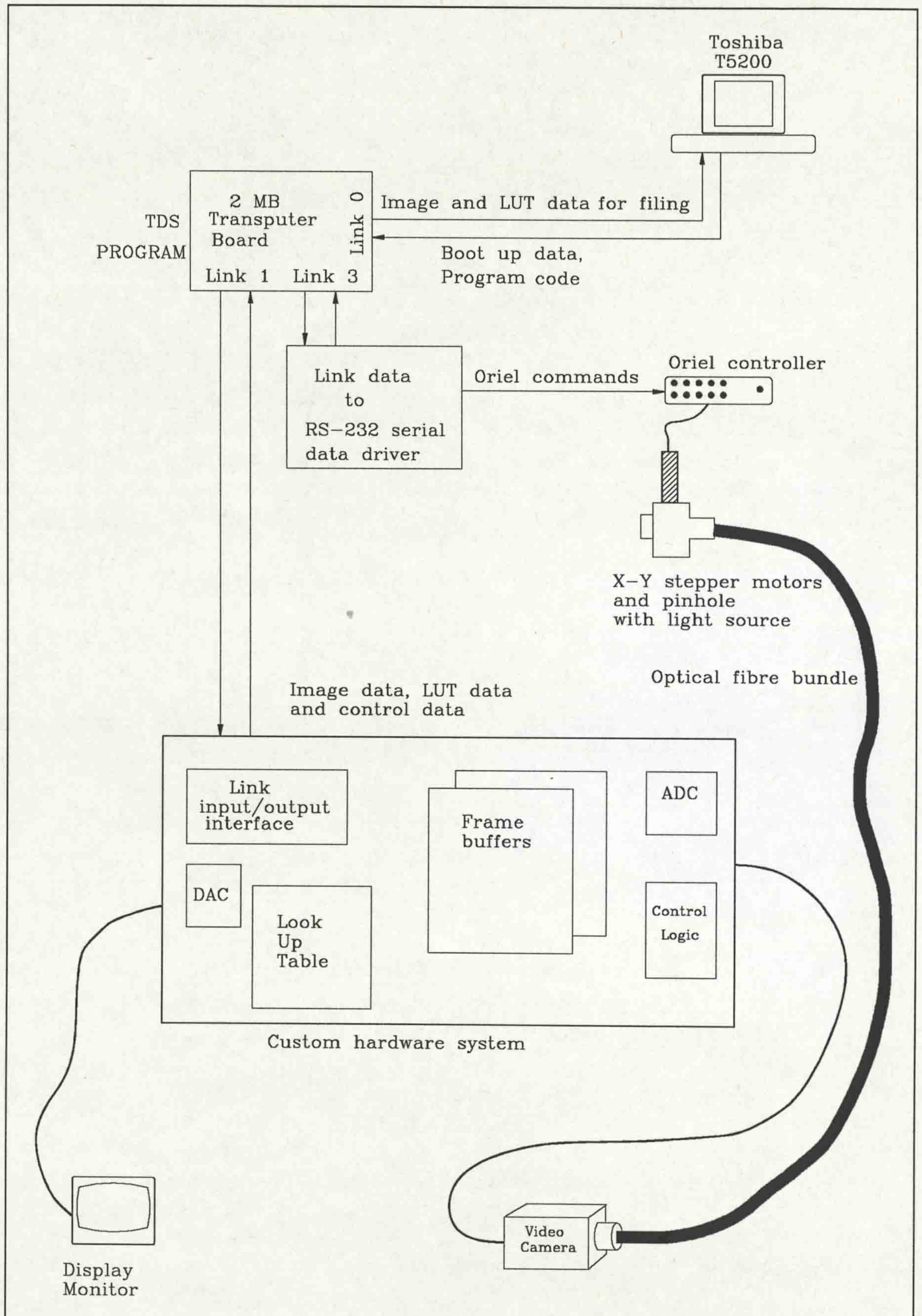


Figure 5.11 – The spot calibration system

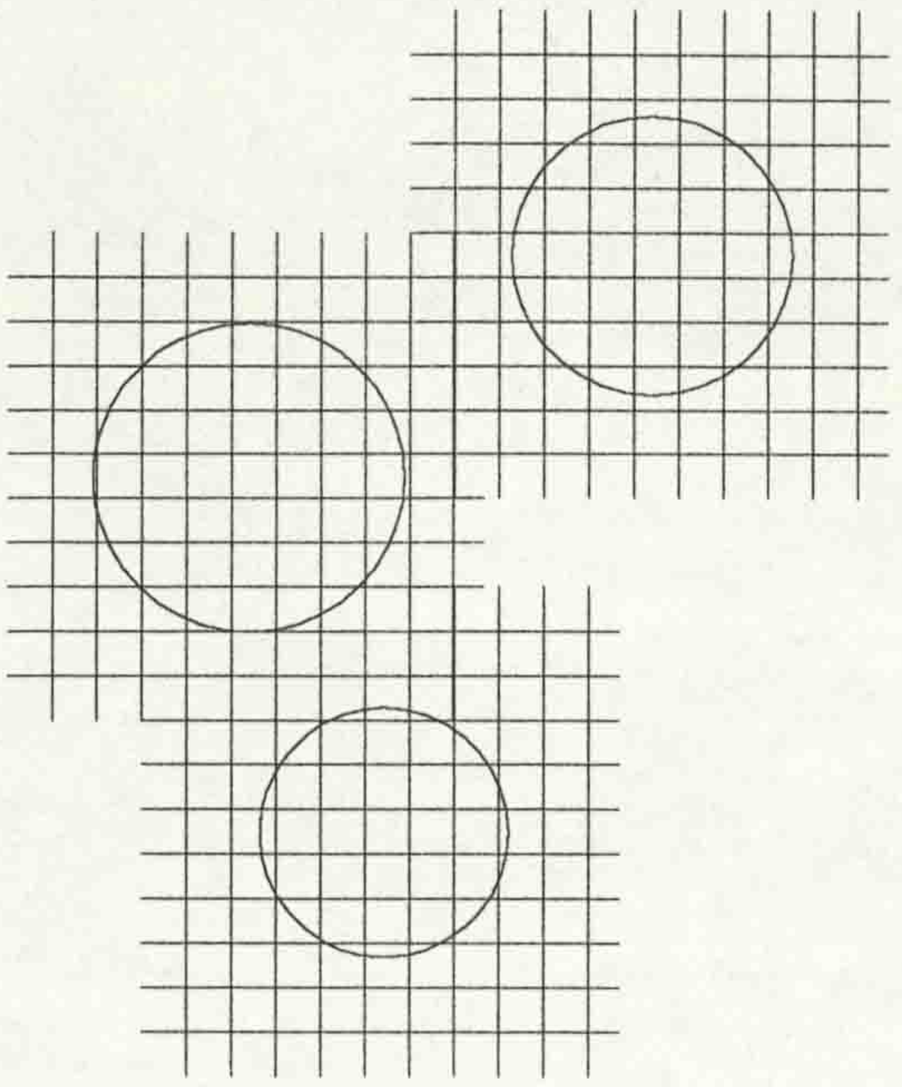


Figure 5.12 (a) – The illumination of three fibres to varying intensities

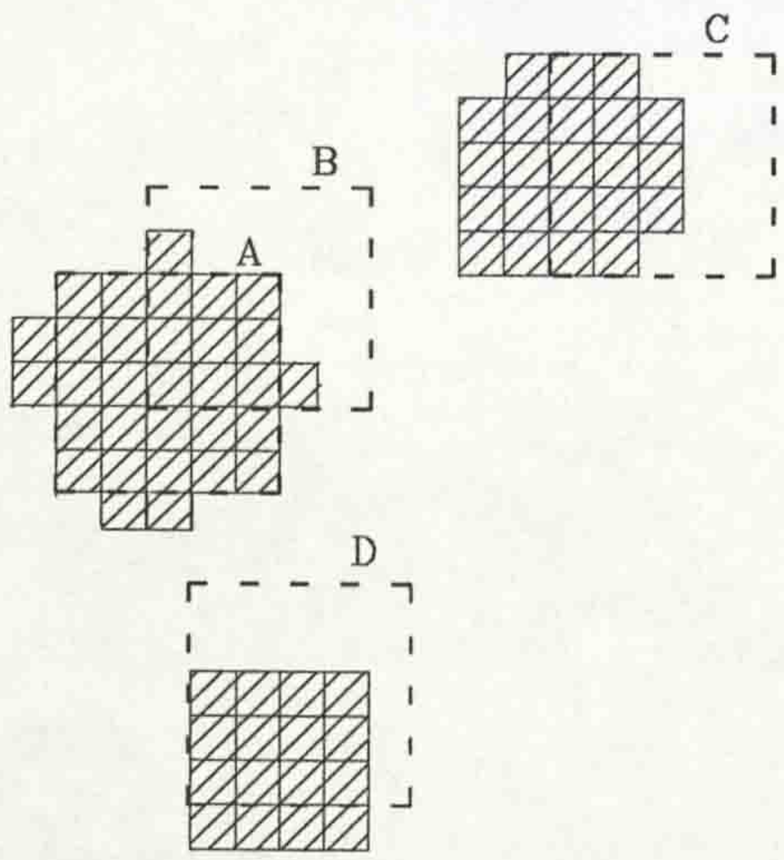


Figure 5.12 (b) – The algorithm to find the brightest area

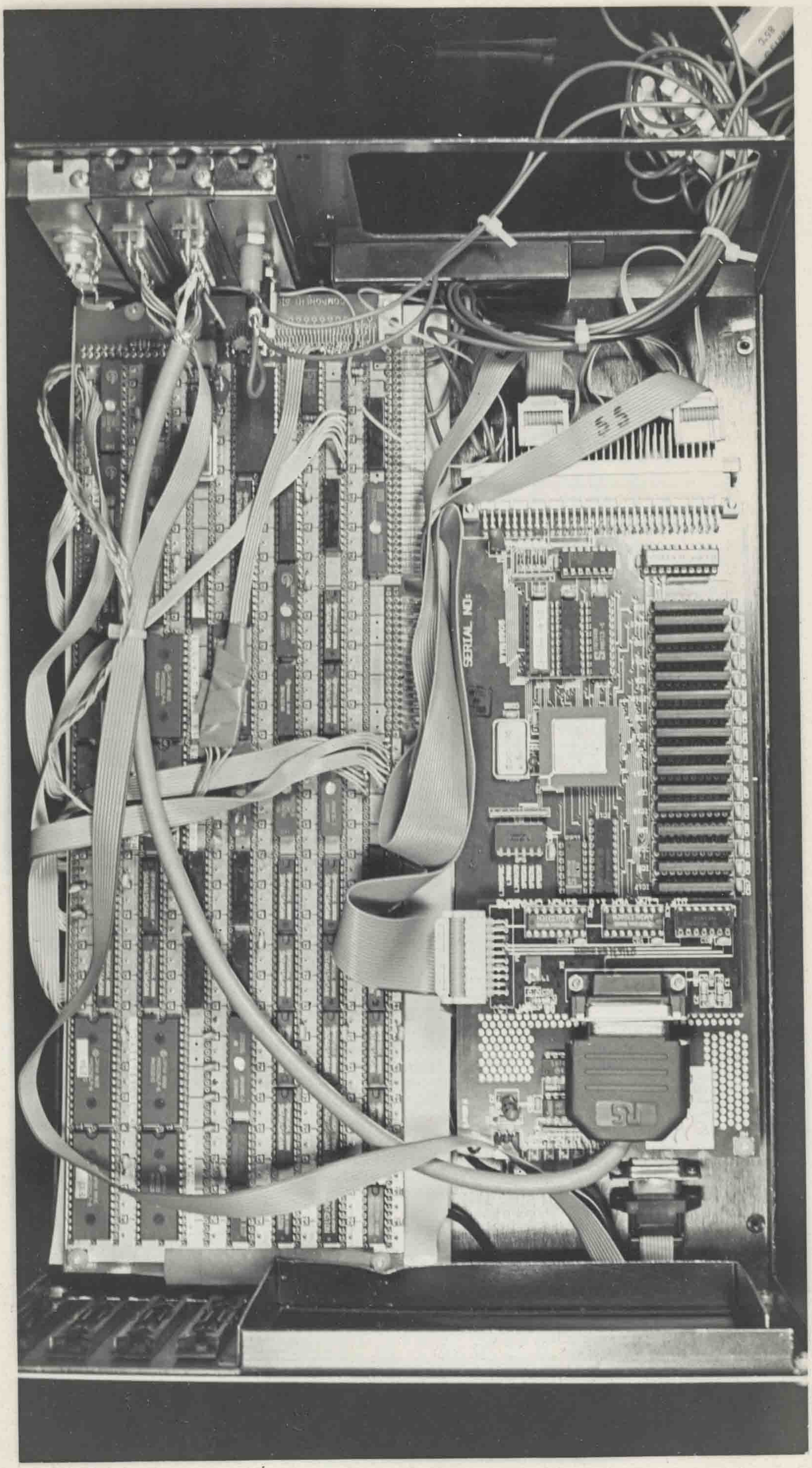


Plate 5.1 - The Optical Fibre Calibration System (OFCS) hardware

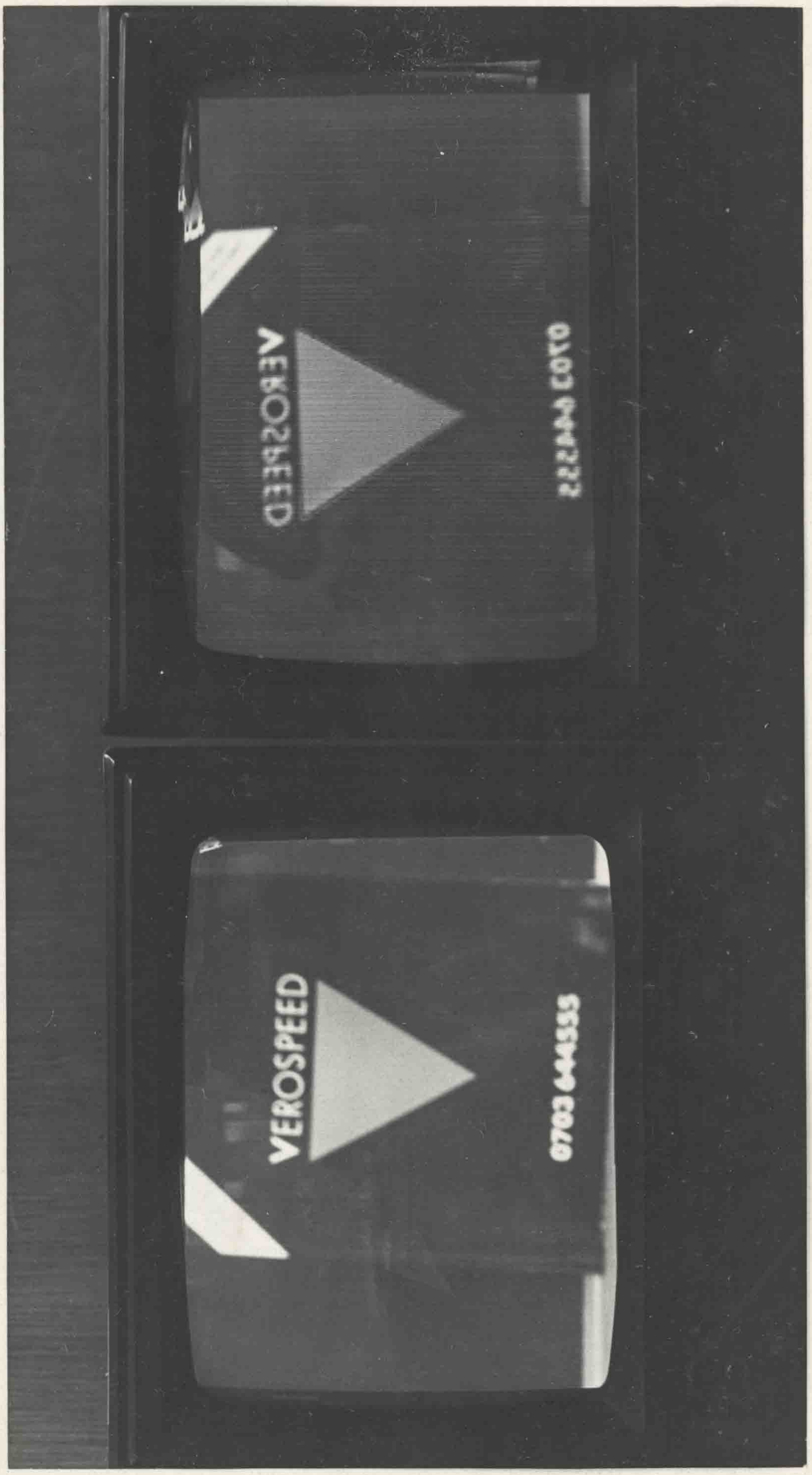


Plate 5.2 - A real - time mirror image produced by the OFCS

6. CALIBRATION BY TEST PATTERNS

It was seen previously that for high resolution optical fibre image transmission systems, the location of each fibre in the bundle for calibration purposes would be difficult if the fibres were of a small diameter (of the order of 10 µm) and impractical in terms of the length of time it would take to calibrate the fibre. It was therefore perceived that a new calibration method was required. This new method must be capable of calibrating the higher resolution fibre bundles to a reasonable degree of accuracy. This chapter details the new calibration theory, and its implementation. As the development of the calibration method proceeded, various problems were encountered. These problems are discussed and the solutions implemented are also detailed, producing three separate calibration methods, all based on the same principle. The experiments and developments described used the previously described hardware system as well as standard electronic apparatus to check the validity and accuracy of the results obtained. To fully test the results that were obtained, the best method will be used in conjunction with the specifically designed hardware equipment, as described in Chapter 5, and appropriate optical fibre bundles.

6.1 Principle Of Calibration Method

This principle of this calibration method differs from the previous one in that instead of seeking a relationship between the input and output points of each fibre, a mapping is sought for the pixels in the input and output images.

To achieve this, a series of test patterns were developed based on the location of the pixels in a matrix, and the addressing of the pixels in the matrix. All of the following discussions assume that the images concerned are of 256 pixel by 256 line resolution. If the image matrix is envisaged in a Cartesian coordinate system, each pixel within the image can be addressed by an x and y coordinate, the x coordinate being the pixel number along the line, and y the line number, as described by Figure 6.1. If digital numbers are used to represent the coordinates, then an 8 bit number is required to count the 256 possible different coordinate values, in the range 0 to 255, inclusive.

The principle of the calibration is to generate test patterns where certain pixels are illuminated according to their address in the pattern. The test pattern is then transmitted by the bundle of fibres, and the output of the fibre bundle is examined. The address of the pixels illuminated in the output image therefore have the corresponding input address pattern as part of their input address. The simplest test patterns illuminate areas corresponding to a unique bit pattern for the input address. Since the input address is made up of 16 bits (8 bits for y, 8 bits for x), 16 test patterns would be required to find the value for each bit in the input address. Each test pattern is therefore generated by illuminating the area corresponding to having that bit set to 1 in its address, and areas where that bit is 0 are set to black. Consideration of this principle shows that half of every test screen would be illuminated and half would not. The area illuminated varied in pattern, but always remained at half the total number of pixels. By examining the output of the fibre bundle for each test pattern, and noting the pixels that are illuminated in the output, the input address for each pixel in the output image can be sequentially built up one bit at a time. The resulting test screen patterns, for analysing the x coordinates are shown in Figures 6.2 (a) to (h). The test screens for the y coordinates are essentially of the same pattern, but rotated through 90°.

To perform the calibration, all the bits in the starting input addresses for all the pixels are set to 0. As each test pattern is transmitted, the current input address associated with the pixels illuminated in the output image is logically 'ORed' with the bit pattern corresponding to that test pattern. Since only one bit is set in the bit pattern that generated the test pattern, the input address for the pixels illuminated in the output are altered by having that bit set to 1 by the logical OR operation (a theoretical example of which is presented in Figure 6.3). The input addresses for each pixel in the output image is therefore built up one bit at a time.

6.2 Consideration of the Geometrical Optics

The calibration system depends on the transmission of images by the optical fibre bundle, and the formation of a video image from the transmitted test pattern. The previous calibration method relied on the direct injection of light from the pinhole into the fibre. It was envisaged that the test patterns would be considerably larger than the face of the optical fibre bundle. This indicated that the test patterns would have to be optically reduced in size to cover the input face of the bundle. At the output face, the image from the bundle would have to be focussed onto the sensing element

of the camera. Two lens systems would therefore be needed, one for each end of the bundle.

The selection of the lens arrangements would have to take into account the magnification factors at each end. Examination of the principle of the calibration and the nature of the test patterns shows that the input coordinates will range from 0 to 255, in both the x and y directions. This meant that the source image for reconstruction must occupy the entire video area in order to provide the correct pixel intensities for display (section 5.2.2.1). The image of the output end of the fibre bundle must therefore be magnified to occupy the video image area.

At the input end, all of the fibre bundle must be included in the calibration area to maximise the number of fibres used to transmit the image and thus preserve the resolution capability of the bundle. This issue is complicated by the fact that the typical fibre bundle is circular in cross-section and the test screen patterns are square. To include all of the input face of the bundle in the calibration area, the test patterns will have to reduced in size to a square whose sides are equal to or greater than, the diameter of the fibre bundle. If the image of the test pattern is larger than the face of the fibre bundle, the centres of the two may not be perfectly aligned. This does not matter, as long as the entire fibre bundle is covered by the test pattern, since this only displaces the number of fibres in the bundle which will be used to transmit different parts of the image. The effect of varying the above magnification factors will be seen in Chapter 7.

6.3 Description Of Apparatus

The apparatus used for the following experiments will be described with the reasons for the choice of each element. The experiments will be carried on two systems, one the specifically designed hardware system described in Chapter 5, the OFCS, and the other a standard digital image capture/display system, the Matrox PIP-1024B [8]. This was done to check that the operation of the OFCS was correct and did not affect the results obtained from the calibration. The change from the DT2853 frame store will be explained. For initial testing, a coherent optical fibre bundle will be used to compare the results obtained after calibration with the original output from the fibre bundle, and so gain some measure of the quality of the results. The coherent optical fibre bundle will also be used to set up the geometrical optics needed for the transmission of the test patterns since the focusing of the system will be easier to assess if a recognisable image is transmitted. The image analysis system is again

the Inmos B004 transputer board running Occam programs as described in section 5.3.2.

6.3.1 The Matrox PIP-1024B

The Matrox PIP-1024B is a frame store that is intended for use in IBM PCs or compatible machines. The PIP-1024 has four frame buffers, each of nominal resolution of 512 pixels by 512 lines. These frame buffers can be reconfigured to give a resolution of one frame buffer storage area of 1024 pixels by 1024 lines, and a display area of 512 pixels by 512 lines. The principles of its design are similar to the Data Translation DT2853, with the main differences stemming from the interface method to the host system.

The essential operation of the PIP-1024B is similar to that of the DT2853. However, the image data access mechanism is significantly different. There are two methods for accessing data from the PIP-1024. The first uses the input/output bus of the Opus PC V. A start address is first written to the PIP-1024, and the data from that address is then put onto the system's bus to be accepted by the processor. The start address is then automatically incremented to read the next pixel in the image. The whole image is presented as a contiguous storage area. It is therefore a simple matter to read the whole image by starting from the first pixel in the image, and then making 256 x 1024 accesses.

It is also possible to access the data from the PIP-1024 by a Direct Memory Access (DMA). This places a block of data, up to 64K, from the PIP-1024 into the processor's memory space. This process takes place via one of four DMA channels. The basic steps for performing a DMA access to the PIP frame store involves the following steps:

1. Set up the x,y coordinates for the base area to be accessed.
2. Set up the control registers to disable other interrupts.
3. Read the pixel data register on the frame store.
4. Enable the DMA and select read or write from the correct buffer pointer.
5. Initialise the DMA page register and controller.

Since the Inmos B004 transputer board communicated with the Opus PC V via the input/output bus, it was decided that the best method of getting the data from the PIP-1024 into the transputer's memory space was via the input/output interface to the PIP-1024, and not the DMA interface. This freed the 80286 processor from any communications task, as all the required handshaking was performed by the PIP-

1024 and the Inmos B004. A special interface controlling software algorithm was written in Occam. This software treated the PIP-1024 as a file. If the filename started with "P:<path><filename>", then data was either read from the PIP-1024 or written to it, depending on whether the file was opened as a read or write file. If the file name was of the form "Q:<path><filename>", then an image was captured into the PIP-1024 before allowing access to its data. During these accesses use was made of the auto-incrementing feature to access the entire frame without the need to specify every pixel address. All the file handling functions are part of TDS and operate under its control. The functions to control the PIP-1024 were formed into a library of routines under the name IMAGE.LIB. The transfer time for a whole frame between the PIP-1024 and the Inmos B004 is about 8 seconds. More detail about the operation of the PIP-1024 can be obtained from [*8*], and about IMAGE.LIB from [*12*]. It is for this reason, access via the input/output data bus of the Opus PC V, that the change from the DT2853 frame store was made. The DT2853 does not allow access in this way and interfacing to the Inmos B004 would involve a greater degree of complexity, having to read and write into the 80286 processor's memory area.

An important feature of the image acquisition systems needed for this application is the ability to display one frame buffer, while simultaneously capturing the output from the camera providing the necessary video timing for the display. This is relevant because the test images will be generated by software routines and displayed by the image processing system, while using this image processing system to capture the output from the fibre bundle. The concept is shown in Figure 6.4. The fields in each frame (see sections 4.2.5 and 5.1), must be captured at the correct moment to ensure that the orientation of the displayed image to the captured image is maintained. The manual for the PIP-1024 [*8*] did not specify whether this was possible or not. It was found to be possible only after a test program was written for this purpose.

The PIP-1024 was used in 512 pixel by 512 line mode. Inspection of the principle of calibration shows that 8 test screens would be required, for each of the 8 bit coordinates in a 256 square display. A display such as that of the PIP-1024 uses 9 bits for each x and y coordinate. It was decided to use the top 8 of the 9 bits available which meant that the test screen for the least significant bit was made up of double display lines being alternately illuminated as shown in Figure 6.5.

6.3.2 The optical fibre bundles

Two optical fibre bundles will be used for the calibration, one coherent and one incoherent. The incoherent bundle is the same used for the spot calibration method, the 3.5 mm diameter light guide, made up of approximately 4400 50 μm fibres.

The coherent optical fibre bundle was supplied by Vinten Scientific Systems Limited [*85*]. It is constructed from single fibres 10 - 12 μm in diameter, and are essentially single mode fibres. The construction of the fibres are of three layers: a glass core, a cladding, and another glass cover, similar to the 'W' profile index described in section 3.4.1.3 . The second glass layer is used to reduce losses from the cladding by reflecting this otherwise lost light back into the cladding, until it re-enters the core.

The diameter of the bundle is 4 mm and it is 1.3 m long. The specifications [*85*] show that the bundle contains 100,000 fibres, and is therefore capable of high resolution images, potentially exceeding those provided by the OFCS. The fibres are not arranged to a regular grid, and therefore produce some distortion of lines in the image, as can be seen from the results in Chapter 7. The entire length of the coherent optical fibre bundle is sheathed in a flexible plastic cover with the ends encapsulated in 8 mm long stainless steel ferrules as described by Figure 3.12. The end faces are polished to optical lens standards for improved uniformity of acceptance of light.

6.4 Test Screens On CRT

The test patterns are those shown in Figure 6.2 (a) to (h). The first method investigated for displaying those test screens was a Cathode Ray Tube (CRT) monitor. Examination of the principle of calibration reveals several criteria that must be satisfied for injecting the light from these patterns into the fibre to be calibrated. Principle among these is the accurate location of each test screen. To ensure accuracy of calibration, all the test screens must be located in the same place, otherwise alignment errors will occur between successive screens, since the calibration depends on the building up of the address for each pixel in the output image. Considering the test screen to locate the value for the least significant bit in the addresses, where the test screen is made up of pairs of lines or columns that are alternately illuminated if generated by the PIP-1024 frame store, a displacement of the image by one line or one column unit would cause a calibration error of 50%. This test screen is the most sensitive to such displacement errors.

6.4.1 Description Of Apparatus

The primary element in this calibration trial was the CRT monitor used. The optical arrangement, Figure 6.6, dictates that the source of the test screens must be as near as possible to the fibre to allow sufficient light to reach the fibre. This meant that the monitor had to be small enough to allow the entire image to be focused on the end of the fibre. A monitor whose display area was 12.5 cm in diameter was used for this purpose. The entire assembly was enclosed in a blacked out rectangular box to prevent the generation of errors from extraneous light sources, and to reduce the effect of reflections of the light sources contained within. A brief review of the theory of cathode ray tubes and display of images is presented to aid understanding of certain effects that will be noted later.

6.4.1.1 The Cathode Ray Tube Monitor

The principles of the cathode ray tube have been extensively documented [*18*], but are briefly reviewed here to explain relevant features. With reference to Figure 6.7, the image is generated by the transfer of energy from an electron beam that impedes on a phosphor screen. This causes the screen to emit light, the intensity of which is roughly proportional to the energy imparted to the screen [*18,86-88*]. The energy of the electron beam is controlled by an accelerating force provided by the Extra High Tension (E.H.T) anode at the screen. By varying the voltage of the E.H.T anode, the brightness can be varied. The scan control electromagnets provide the right horizontal sweep rate and vertical restart for each video field. The collimating anode imparts a small acceleration to the electrons as they leave the heated cathode, to direct them towards the focussing grid. The voltages and current drive capability for the entire system is usually obtained from a single power supply.

The contrast of the output image is determined by the range of amplitude of the video signal. The greater the range of amplitude, the greater the contrast, thus the brightest pixel will be much more differentiated from the darkest. It can be shown [*18,87*] that the contrast capability of any CRT is proportional to the square of the screen diameter and inversely proportional to the radius of curvature of the screen. It was therefore expected that the monitor chosen would not have a very high contrast capability, but that this would be offset by the ability to place the monitor nearer the fibre bundle to allow minimum attenuation of the light from the monitor before injection into the bundle of fibres. The brightness is dependent on the average level of the video signal, as this determines the electron flow from the cathode [*87,88*]. Consequently, a bright image of high contrast will produce an electron beam in the CRT of the monitor which is of average high velocity and whose velocity oscillates

rapidly, as its energy content must vary to produce the high contrast of the image. This can pose a serious problem to all but the best of power supplies for the CRT [87]. The result is that the display becomes distorted as the power supply cannot provide a constant voltage output for the large current changes that are required by bright pictures of high contrast. This aspect of CRT displays became of great significance as the development of the calibration routine proceeded.

6.4.2 Method

The steps to perform the calibration, using the coherent optical fibre bundle are as follows:

The camera end of the fibre was be focussed on the CCD element of the camera to ensure that sharp images are recorded by the frame capture system. The monitor display must be focussed onto the imaging end of the fibre to prevent errors that will be caused by blurred images. This is particularly important as blurred images are caused by multiples of the object being slightly displaced from each other at the image plane. When the image is focussed correctly, all the multiple images converge to form one sharp image [3]. As the calibration method is very dependent on the accurate illumination of the areas corresponding to the test patterns, the blurring of images would lead to errors in calibration.

The alignment of the monitor and the fibre also played an important role in the calibration set up. The two main factors for consideration are the different shapes of the monitor display area, which is rectangular, and the fibre face, which is round, and the relevant sizes of the two. The sizes can be varied by altering the magnification of the output end of the optical fibre bundle to arrive at the 'rectangle in a circle' arrangement or 'circle in a rectangle' arrangement shown in Figure 6.8 (a) to (c). It was observed that the 'rectangle in a circle' did not utilise all the fibres in the bundle and was subject to some loss of resolution as the individual fibres would occupy a larger area, and would use fewer fibres to illuminate the entire CCD area. However the calibration method dictates that the entire video display area must be covered by the bundle (section 6.2). It was not essential to align the centres of the circle and rectangle, as any misalignment of the fibre and monitor would only place more fibres in one half of the output image during the calibration and therefore the reconstruction. However, to maintain the uniformity of resolution of the fibre, and to aid calibration, the fibre and monitor were aligned according to their approximate centres. This was achieved by finding the entire area, in pixels, illuminated by the fibre, nominally 65536, for the OFCS, and then illuminating half the screen in both the horizontal and vertical directions. The alignment of the two was altered until half

the fibre area was illuminated. Various magnification factors were tested to investigate the effect of altering the pixel to single fibre image size. The use of a smaller image effectively increased the resolution of the optical fibre bundle, but the resulting image was obviously smaller, giving a smaller viewing area.

Having established the imaging system, the monitor must be set up to provide an adequate level of illumination with minimum distortion of the image. As the level of brightness and distortion varied between test screens (section 6.4.1.1), a compromise had to be settled upon. As each test screen contained the same size of illuminated area, it was expected that the contrast and brightness could be set at one level for all the test screens. However, in the test screens for the higher order bits, where the test screens had large areas of high brightness, the continual drain on the power supply storage and smoothing capacitors resulted in a very high distortion of the bright areas, if the contrast and brightness were too high. Reducing the contrast improved the display but did not sufficiently differentiate the bright from dark areas. Reducing the brightness allowed adequate differentiation but the low overall level of brightness meant that the absorption through the fibre bundle was too significant to allow enough information to be obtained from the resulting image.

The resulting compromise involved setting the contrast to a high level to allow good differentiation of the areas and then increasing the brightness to the highest level possible without visible distortion. The camera lens aperture also had to be adjusted to balance the depth of focus with the need to provide the largest possible aperture, which is needed to allow lower monitor brightness/contrast levels to be used.

The uniformity of the size of the image was also noted to be variable as the edges of the display were approached. Altering the overall size of the image sent to the monitor also affected the percentage of distortion. Reducing the size of the display and the image minimised this effect. The image was therefore reduced to 256 pixels wide and 256 lines high for the PIP-1024B. This was achieved by using the lowest 8 bits of the 9 bits required for the x and y pixel coordinates. The image was then offset by 128 pixels in the positive x direction and 128 lines in the positive y direction. This located the image in the centre of the display and is shown in Figure 6.9. The use of the lowest 8 bits meant that the final pixel address computation in the calibration illuminated alternate lines for the x coordinate and alternate pixels for the y coordinate. When alternate lines are illuminated in an interlaced display, the display data for one field is essentially absent. This is realised as a flickering displayed image, and further complicates the calibration procedure.

It was decided that for development purposes the data obtained for the calibration of the lowest significant bit would be ignored. This would result in an effective resolution of the output of the fibre to 7 bits for each coordinate, that is, a 128 pixel by 128 line display giving 16384 pixels. This is approximately one sixth of the resolution of the fibre bundle, but it was decided that the correction of this loss of resolution could be investigated after verification of the calibration theory. The experiment was carried out initially with the system of low effective resolution, to develop the calibration software, and to prove the theory of the calibration principle. It would also be nearer the resolution that could be obtained from the incoherent bundles available.

To perform the calibration from this point onwards, the sequence was executed under software control. The test screens were generated, displayed from one frame buffer on the PIP-1024 or the OFCS, and captured into the other buffer. The data was then moved into a two dimensional variable array in the program, where the relevant image processing was carried out to detect the pixels that were illuminated. The corresponding input addresses for these pixels were then modified according to the test screen being examined. On completion of this stage, the data was stored in a file for later use in reconstruction of images from the fibre.

6.4.3 Software Design

The hardware configuration and flow of data between the two transputer boards, Inmos B004s, and the PIP-1024B, is shown in Figure 6.10. This allows one board to run the TDS and the second to run the programs written and compiled on the first. Since every image needs an array of 256 Kb in size (262144 pixels each of 8 bits) for the PIP-1024B, it is necessary to have several times that much memory to manipulate the images for analysis. TDS occupies approximately half of the memory on one board, making it is essential to have the second board to avoid restrictions on the image processing algorithms. Since only the first board has a direct link to the host Opus PC V, the data for the file to be stored must be communicated back to the board running the TDS for filing to the hard disk. Later versions of the Inmos transputer boards have the capability for easier memory expansion in the form of TRAM modules [*75*]. The OFCS configuration is the same as that used before and shown in Figures 5.10 and 5.11.

The primary software problem is concerned with the detection of the illuminated pixels in the output image. Section 4.3.3 and [*52,53*] discussed the various methods available for feature extraction. The recurring problem was the choice of thresholds

to differentiate the illuminated pixels. The first investigation into choosing a threshold examined the histogram, generated as described by section 4.3.3.1, for each test pattern output from the fibre bundle. Samples of these histograms can be seen in Figures 6.14 (a) and (b) and Figures 6.15 (b), 6.16 (b) and 6.17 (b). Several peculiarities can be noted from these histograms. The first is that the test screens for the higher order bits produce histograms with two clearly differentiated peaks, representing the unlit and illuminated pixels as in Figure 6.14 (a). As the values of the lower order bits are investigated, the histograms approach a single peak profile with ripples superimposed over the range of pixel intensities found in the image as in Figure 6.14 (b). This indicated that the choice of threshold for the higher order bits would essentially be straightforward, becoming increasing complex for the lower order bits.

Several methods for choosing the threshold were investigated. It is obvious that the threshold to differentiate the illuminated and non-illuminated pixels lies somewhere in the region labelled the 'valley' as in Figures 6.14 (a) and (b). It became apparent that for the two lowest order bits in the input address, the fibre bundle's output did not produce a sufficiently clear image to allow a definite choice of the threshold. Therefore some image enhancement had to be implemented before the image could be analysed to obtain a threshold. The contrast in the image as well as the range of intensities would have to be improved to provide a greater differentiation between the illuminated and dark pixels.

The use of a high pass filter would improve the contrast in the image by highlighting the areas of rapid change in intensity as explained by Rosenfeld [*45*]. A scaling factor would be applied to all the pixels to increase the range of intensity. There are various methods that may be used for intensity scaling operations. Two of the most common are described by O'Gorman and Wu et al in [*89,90*], respectively. The principle of the high pass filter was introduced in section 4.3.6.1. The scaling factor was chosen according to the following derivation principles from [*45,49,89,90*].
Since the light output from the fibre would be low, as a low level was initially input, the range of intensities in the output would occupy a small section at the lower end of the range of intensities. If a_1 and a_2 are the minimum and maximum intensities in the output image, and b_a and b_z are the allowed minimum and maximum intensities, then

$$(b_z - b_a) > (a_2 - a_1)$$

A scaling factor, S, can be defined by the equation

$$S = (b_z - b_a) / (a_2 - a_1)$$

For any pixel intensity, p, in the image, the new value, p_i, to occupy the entire

available intensity range was calculated from the formula

$$p_i = Sp$$

If the minimum values in the two ranges are not zero, then the equation must be modified to

$$p_i = (Sp\,(p - a_1)) + b_a$$

Any erroneous values outside the possible range, 0 to 255 for an eight bit pixel value resolution, are set to 0 if lower and 255 if higher.

Increasing the contrast in the image by expanding the intensity range emphasised the background noise in the image. The source of this noise was explained in section 4.2.5. To minimise errors in pixel intensity from the offsets caused by this background noise, it had to be removed. Standard methods for extracting the required information from noisy images are discussed in [*45,47,91,92*]. A statistical investigation of the distribution of this noise was carried out. Twenty-five video frames were captured, with the lens aperture totally closed. The value of each pixel in the background was averaged over these frames. The mean, median and mode pixel intensity values of the average background frame were examined, and this revealed an approximately Gaussian distribution of low deviation. It can be shown that in such cases [*53*], the mode, or most commonly occurring value, represents the best guess of the value of the intensity of a pixel chosen at random from the image. This value was subtracted from the fibre bundle output image before applying the contrast enhancement method.

For the fibre bundle output when test screens for the lower order bits were transmitted, the distinction between the light and dark areas tended to be slightly blurred as can be seen in Figure 6.17 (a). This is in part due to the quality and sharpness of the original input image from the monitor, and partly due to small focussing errors at both the input and output ends of the fibre bundle, as well the finite resolution of the optical fibre bundle, which will be discussed in section 6.5.4. Examination of a cross-section of the image which intersects light and dark areas, shows that the change in intensity across a light/dark boundary was comparatively gradual, and this was seen as a blurred edge. Ideally, the boundary would be a step function as shown in Figure 6.14 (a). The high pass filter operator was applied to the fibre bundle's output of test screens to improve the definition of the edge.

The results of the choice of threshold by the above method showed that errors were present, partly due to the non-uniform nature of the test pattern source, and partly due to small variations in the transmission characteristics of the optical fibre bundle.

These errors were a direct result of using a single threshold value for the whole image. It was decided to find a threshold for each pixel in the image. The calculation of this threshold was relatively straightforward. A totally illuminated image, by an image on the monitor, of the output face of the fibre bundle was acquired. A totally black image of the monitor screen was also acquired. The threshold was taken to be the midpoint between the two resulting intensities at each pixel location. If the resulting pixel intensity when the optical fibre bundle was transmitting a test pattern was above this threshold, then the pixel was deemed to be receiving light from the test pattern, and would have its corresponding input coordinate modified accordingly.

The simplified overall operation of the software to perform the calibration is expressed by the flow diagram in Figure 6.11, and the main software routines are listed in Appendix D. The image enhancement techniques provided an output that was sufficiently improved to allow more accurate location of the upper six bits of each coordinate of the input address for each pixel.

6.4.4 Discussion

From the bit orientated nature of the calibration routine, any error in deciding the value of a bit in the input address for any output point can result in large errors being generated. Consider the example below:

If the x coordinate, x_i, of a particular input point that corresponds to the address (x_o,y_o), of a pixel in the output display is, in bit form

1 0 0 0 1 0 0 0
Bit 7 Bit 0

If an error was made in the value of bit 7, the erroneous value of x_i would be

0 0 0 0 1 0 0 0
Bit 7 Bit 0

The decimal value of x_i would be 8, instead of 136, thus mapping the output from the input point (x_i,y_i) into the wrong half of the output display.

The source of this error could be the wrong choice of the threshold value by one unit of intensity, which may indicate that the pixel in the output image at (x_o,y_o) was not illuminated when it should be. This is obviously a great error, and the effect is decreased for the lower order bits. An error in bit number one of eight, numbered as above, would cause an error of location in the displayed image by 2 pixels. Fortunately, it was easier to choose the correct threshold for the higher order bits, because of the better image definition in the test screens. The method of choosing

the threshold also ensures that the best possible choice of threshold is made for each pixel in the output image.

The other major source of error was distortion of the high contrast images required for the test screen. The effect is the same as above, if the wrong decision is made about the classification of any pixel as being illuminated or not. Unless the input image is uniformly illuminated, and the level of illumination does not vary between test screens, errors will be generated. Errors in the least significant bit will shift the mapping of the input points by one pixel. This error is comparatively small, with respect to the resolution of the display, and is even smaller when compared to the resolving power of the incoherent optical fibre bundle that would be used for the tests. Thus, the calibration method produces a mapping which is highly sensitive to errors in the most significant bit of the coordinate addresses, but essentially immune to the value of the least significant bit. Plate 6.1 shows two identical monitors, displaying the same test pattern. The left-hand monitor shows considerable distortion of the test pattern, caused by turning the brightness and contrast levels up, to allow adequate differentiation of the black and white areas. The right hand monitor shows no visible distortion, but the brightness level is much lower.

For these reasons, it was recognised that a better source of test patterns was required to locate the higher order bits. The use of a CRT monitor would produce an unacceptable degree of error when it was most important, for the higher order bits, and would therefore have to be replaced. The results of this calibration method trial with the coherent optical fibre bundle can be seen in Chapter 7. It was decided not to pursue the calibration of the incoherent bundle with this apparatus in view of the high degree of error that was observed. The results from the PIP-1024B and the OFCS were identical, showing an independence of result to the resolution of the image capture/display system used.

6.5 Test Screens on Slides

In order to overcome the inherent problems of using a CRT monitor to display the test patterns, it was decided that a fixed, non-luminescent display should be used to provide the test images. This would take the form of a transparent slide onto which the test patterns had been photographed. This would free the calibration system from any errors caused by variable test screen size and intensity, but present new problems of physical location of the test patterns and accurate photography to print the test screen on the slide.

The method of choosing the threshold was deemed satisfactory, and would therefore be used for the next experiment. The bulk of the software would remain the same as the principles involved were identical to the previous trials.

6.5.1 Description Of Apparatus

The only new element in this calibration trial was the test screen pattern generator. All the other components remained the same as in the previous trial (section 6.3) and will not, therefore, be described here. The CRT monitor was replaced by photographic slides and the method of producing and locating these slides will be described.

6.5.1.1 The Laser Printer and Postscript

A laser printer is a device that uses a laser beam to create an electrostatic charge which is then used to attract fine particles of a toner [*83*]. The electrostatic charge is created on a drum type mechanism, in the image of the pattern to be printed. This drum is then used to attract the particles that will be used to provide the colour of the printed material. Paper is then passed over this drum, which is heated, to facilitate a permanent transfer of the toner to the paper. Using this method, printing resolutions of 300 dots per inch have been achieved in the Apple LaserWriter® laser printer [*83*].

This printer is driven by a Postscript [*84*] controller. This is essentially a software language dedicated to text and graphics printing. The combination of the laser printer and postscript were considered to be a convenient and accurate method of producing a paper copy of the test screen patterns. The software described the size of the test pattern and the bars were filled in as required for each test screen. The listing for the software is provided in Appendix D. The accuracies achievable in the printing process were limited by the resolution of the printer. However, an accuracy of 300 dots per inch or a possible error of $\pm$ 0.0033 inches (approximately $\pm$ 0.01 mm) over a length of approximately 100 mm was considered adequate in this application.

6.5.1.2 The Test Slides and Holders

Since the source of the test screens was not self illuminating, a light source had to be used to produce the image. The characteristics of this light source must include a sufficiently diffused light that does not highlight any particular area of the test slide, but produces an even illumination. It must also be sufficiently bright, to differentiate the image from its background for the reasons explained previously. The light source

must also be powered from a direct current (d.c.) source. An alternating current (a.c.) source would provide variable lighting, at a frequency of 50 Hz, if mains powered. Since the camera captures images at a rate of 25 frames per second, it is possible to capture images that are only partially illuminated, as the camera is d.c. powered and not synchronised to the mains frequency. The chosen light source was a d.c. powered 25 watt halogen light bulb with a reflector, placed behind a diffusion filter to provide a uniform spread of light. The diffusion filter took the form of a convex lens placed near the light bulb, within the focal length of the lens and two diffusion screens between the lens and test slides. Placing the light bulb within the focal length of the lens would produce a larger, virtual image of the light bulb [*3*], thus spreading the light over a larger area to cover the entire area of the test slides.

The first step in producing the test screens on the slides was the accurate drawing of the test screens for photographing. The original images were printed onto a high quality, smooth surfaced paper to minimise distortions, to a size of 92 mm square. A border was drawn around the test screens to mark the limits of the image and for use as a registration mark in the location of the image. These images were then accurately reduced, photographically, to a size of 40 mm square and printed onto photographic slides. Slide film produces a positive image of the subject, and does not require printing onto photographic paper. The end result is an image that duplicates the test screen pattern required, with black regions for the dark bands and is transparent for the light bands. Illumination of the test slide from behind, by a uniform and diffuse light source, would provide the test pattern images for the calibration. The slide film was then clasped between two glass plates 54 mm square, for protection and rigidity. The glass plates were held together by a square plastic frame, as shown in Figure 6.12.

The next step was to devise a method for accurately locating every test slide in the same position. The elements to do this are also shown in Figure 6.12. Every test slide was inserted into a subframe. The position of the image in the subframe was variable in the negative or positive direction for both axes. This was achieved by allowing a 2 mm gap around the plastic frame of the image. The plastic frame would be held in its final position by four grub screws placed at centre of each side of the plastic slide frame (see Figure 6.12).

Each subframe had been produced to a reasonably high dimensional tolerance of $\pm$ 0.05 mm, and the surfaces had been hand polished to fit with the minimum play in the subframe holder. The subframe holder, Figure 6.12, was a grooved U-shaped

holder to allow the subframes to slide downward into place. The final location of the subframe could be adjusted by means of two screws in the base of the holder. The vertical position could be altered as well as the horizontal tilt of the subframe. The subframe holder was mounted onto a rod that could be inserted into a holder specially designed to be attached to an optical bench. The use of an optical bench allowed the alignment of the axes of the elements of the calibration system as well as providing a rigid base for the entire system. As the dimensions of the image and display area have been defined as being square, it is possible to use the same test slide for calibration of both the x and y coordinates, by rotating the slides through 90°. Therefore only 8 test slides are required to perform the calibration to 16 bit accuracy. The entire set of test slides in their subframes is shown in Plate 6.2, and a subframe is shown in the holder in Plate 6.3.

6.5.2 Method

The basic method of calibration remains unchanged from that described in section 6.4.2. The major difference is the removal of the need to set up the monitor. This was replaced by a system to locate the slides in the same position within the subframe. The basic principle was to use the edges of the border placed around the test screen image for the location. A magnified section of vertical and a horizontal side, and their corner of intersection were chosen as the registration marks to provide the location. The coordinates of three equally spaced points on each line were found. The position of the test slide was varied until the lines were respectively vertical and horizontal, with respect to the camera, and in the same arbitrarily chosen position as all the other test slides. By implication, the corner of intersection would also be in the same place. The concept is shown in Figure 6.13. The choice of the pixels to use as the coordinates for the line is complicated by the fact that the lines are more than one pixel thick. The centre point of the width of the line was chosen as the most representative point of the line. The method of finding the centre of the line is described in section 6.5.3.

To improve the accuracy of location of the registration lines, the test slide was placed near the camera, thus providing greater magnification of the image for a higher resolution of the lines in question. However the magnification could not be of such magnitude that the three samples of coordinates of points on each line could not be used to determine whether the line was truly vertical or horizontal. A sufficiently long section of the lines had to be visible to make this judgement. An arbitrary value of 25 % of the length of both lines was chosen as being long enough for an accurate measurement. After each test slide was located in the desired

position, the grub screws were used to lock it into place, and the final position re-checked for misalignment during this procedure.

After each test slide was located, the calibration could be performed. The sequence of operation is unchanged except for the allowance of a pause to change the test slides.

6.5.3 Software Design

The first issue of the software design was to find the coordinates of the points used for the registration marks. The histogram of intensities of a cross-section of the border lines (Figure 6.13) shows a peak of intensity distribution that falls away the to the background intensity value. It was decided that the width of the peak was expressed by the distance between the shoulder values indicated in Figure 6.13, as indicated by Hertz [*93*]. The centre of the width of the line was taken to be the midpoint between the shoulder values. To find the shoulders of the profile, a series of tests were carried out on the profiles of the border lines on several test slides. The histogram of intensities were stored together with the gradients at every intensity in the histogram. The shoulders were manually chosen for each histogram, and the gradients at the shoulders were noted. This gave a statistical set of values for the gradient at the shoulders. Half of these values were positive and half were negative, indicating the gradients at pairs of shoulders. The most commonly occurring pair of values, the mode, was chosen as a representative value of the gradients at the shoulders, for reasons discussed by Sahoo et al [*53*].

All subsequent histograms were analysed to reveal the shoulders by comparison to the expected value of gradient. The centres of the lines were then calculated from this information. The three points chosen to give the orientation of each border were separated from each other by 100 pixels for the horizontal border, and 100 lines for the vertical border. To allow for the error in finding the centre of the borders, a variance of 2 units (lines or pixels) was allowed in the coordinates when assessing whether the border was vertical or horizontal. The examination of the alignment of each slide in the subframe was done manually, using the measurements obtained by the image analysis system. This was considered adequate since it was expected that the alignment would be a one time operation, and the adjustments to the position would have to be done manually, regardless of how the measurement data was analysed.

With the test slides placed in the same position in the subframe/holder combination, the calibration was carried out. The software routines used were the same as those used previously, with the removal of the test screen display routines, and the insertion of a 'wait for key' routine that was used to indicate the correct insertion of the test slides.

6.5.4 Discussion

The reasons for using the test slides over the CRT monitor were vindicated. The distortion and variable illumination effects were greatly reduced. However a few new features were observed.

Observation of the images reconstructed using the data from this calibration method and apparatus, showed that new sources of error were introduced. These errors showed up as a regular grid pattern on the reconstructed image (see Chapter 7). The conclusion was reached that these errors were the result of inaccurate location of the test slides, which manifested itself as the erroneous calibration of the pixels at the black/white borders. Consideration of the lowest order pair of bits in the input address coordinates, showed that for the reasons outlined in section 6.4.4, the system was insensitive to location errors at this level, and visual examination of the raw output image for the least significant bit confirmed that the fibre was incapable of resolving the pattern on the test screen. Figure 6.17 (a) shows the transmission of the test pattern for bit 1 (numbered 0 to 7). This was considered to be the limit of the resolution of the optical fibre bundle.

The specifications for the coherent fibre bundle [*85*] state that the maximum resolution is 36 line pairs/mm. The test pattern for the least significant bit contains 128 line pairs. When this pattern covers the entire fibre area, a diameter of 4 mm, this results in an image of 32 line pairs/mm. This is very close to the specified limit of resolution of the optical fibre bundle. The nominal fibre to pixel ratio for the coherent bundle/OFCS system is 100,000 to 65536, or approximately 1.5 : 1. In practice, this ratio is lower since not all of the fibres' output fall on the camera sensing array (because of the square in a circle arrangement shown in Figure 6.8 (a)). Comparison of the number of fibres in the square area (16 mm^2) used for image transmission, to the entire fibre bundle area (4 $mm^2 \times \pi$) shows that approximately 75 % of the fibres are used for the image transmission, giving an approximate effect fibre to pixel ratio of 1.15 : 1. Since the fibres are not aligned to a regular grid pattern, the output from some fibres fall between pixels, further lowering the effective resolution of the system. The overall result was that the test pattern for the lowest

order bit was not adequately resolved for any meaningful data to be obtained. The data from this test screen could therefore be ignored, to give a calibrated resolution of 128 square units.

It was realised that much more accurate machining of the test slide holders was required to achieve consistently good results. Even if a good set of slides were manufactured, the wear of insertion and removal would probably give rise to errors as the calibration was performed repeatedly. A more robust and accurate system of generating the test patterns would have to be developed.

6.6 Test Screens on Plasma Display

In order to overcome the difficulties of both the CRT test screen generator and the slide test screens, it was realised that a system which combined the advantages of both, and the disadvantages of neither, would be required. Such a device would have to have the accurate location of the CRT and the distortion free image of the slides. There are several electronic displays that meet these criteria, among them Liquid Crystal Displays (LCD), arrays of Light Emitting Diodes (LED), and plasma panel displays [*86-88*]. The LED and plasma devices are self illuminating, LCD devices are usually backlit, and all use electronic methods to locate objects in the display areas.

6.6.1 Description of Apparatus

The chosen display device for this stage of the experiment was a plasma panel. LCD displays tend to have a higher level of flickering and, unless they are backlit, do not have a very high light output. Even when they are backlit, the light output is usually relatively low, and therefore would not lend itself to use in this application. Arrays of LEDs are available but not in the density and resolution necessary for accurate calibration [*88*].

The resolution of the chosen plasma panel was 640 by 480 pixels, and was more than adequate for the display of test screens 256 pixels square. Plasma panel displays depend on the emission of light that occurs when excited atoms , produced by electron impact in the plasma, radiate their energy of excitation. The ionization of the gas is achieved by the application of a potential difference to the gas at low pressure. The potential difference required depends on the parameter *pd*, where *p* is the gas pressure and *d* is the separation of the cathode and anode, and is typically of

the order of 130 volts. The colour of the light is dependent on the gas mixture used, the majority of plasma panels having a characteristically orange glow due to the radiation produced in the $2P^5\ 3P^1$ to $2P^5\ 3S^1$ transition in neon.

The drive circuits for such a panel consists of a multiplexed bus system that selects or deselects the anodes for each element. When the anode is energised the surrounding gas is ionised. The current to the anode is cycled at around 50 KHz, to a peak voltage of 100V, start and stop pulses of about 150 volts also being required. Each voltage pulse following a start pulse is accompanied by light emission. The actual hardware for the plasma display formed part of the display of a Toshiba T5200 portable computer [*93*]. Access to the display panel drivers had to be achieved through programs run on that machine. The software to do that is described below in section 6.6.3.

The plasma display has the advantage of precise location of the display, a distortion free display, and uniform light output. The disadvantages are a non-synchronised on-off display and comparatively low light outputs. These problems could be overcome with various software procedures. The low light level of the plasma display is evident in Plate 6.4, which shows the panel displaying a test pattern.

6.6.2 Method

Again, the calibration method is essentially unchanged from the previous methods. The test screens were sequentially displayed on the plasma panel, captured into the PIP-1024 frame store or the OFCS, and then processed and analysed to extract the required information for the calibration. It was essential to maintain the physical alignment of the plasma display and the other elements in the optical system. Since the display had to be attached to the base unit of the computer, this posed some difficulty. This was resolved by making careful measurements of the size of the plasma display to align the centre of the display with the optical axis of the system. To avoid the errors that could rise by oblique viewing of the screen, it had to be perpendicular with that axis. A set-square was use to make sure that the orthogonality of the system was correct, and then the plasma display panel was clamped into place.

6.6.3 Software Design

The basic problem of the calibration still existed, that is, extracting the image of the test screen from its background and differentiating the light and dark areas of the

image. One new complication was introduced - the unsynchronised flickering of the test screen. While this is not visible to the eye when looking directly at the test screen, it becomes obvious when viewed through another camera system, cycling at a different frequency. However, this feature is overcome by capturing several frames of the display, and then taking an average value of these frames.

The test screens were generated by a program, written in C, ***screens.c*** [Appendix D], and executed on the Toshiba T5200. This program makes use of the graphics functions provided by the Microsoft C package [*94*]. The display area was 256 pixels square and centred in the display area of the plasma panel.

The average technique for capturing the displayed image works for several reasons. Examination of the theory of beat frequencies [*39*] shows that for two slightly different frequencies, f_1 and f_2, where f_1 is greater than f_2, the beat frequency, f_b, is given the equation

$$f_b = f_1 - f_2$$

For the two frequencies concerned, 75 Hz and 25 Hz, the beat frequency is 50 Hz. This theory is based on a pure sinusoidal oscillation, but gives some indication to the interaction of the two systems. This indicated that 50 times a second the oscillations of the two systems varied from being in phase to completely out of phase and back into phase.

Since the light output from any point in the test image is independent of any other point in the display, the total image could be made up by illuminating every pixel in the image in turn. The total output would then be the sum of the output from each pixel. The CCD element of the camera effectively integrates [*41*], that is sums, the light output from the display over a period of 40 ms, or three cycles of the plasma display. Thus any video frame of the display will contain the sum of at least two whole display cycles, and two sections of cycles at either end of the capture time window of 40 ms. Thus if the single video frame was averaged over two display cycles, and ignoring the third disjointed cycle, a possible error of approximately 33% would be realised. Since the display data was completely stationary, it could be assumed that the entire display was actually summed three times in every video frame captured by the camera, and that all the data would be present in every video frame. The analysis of more complex images, for human perception, acquired by a similar technique, is undertaken by Chitrasert and Rao in [*95*].

Experimentation showed that this was nearly the case. Small video synchronization glitches on capturing from the camera would sometimes cause small areas at the start or end of the display image to be missed during a capture operation. The solution was to take an average of the image over a longer time window of more than one video frame. Because the transfer of data to the transputer system from the image acquisitions was not instantaneous, a continual set of video frames could not be captured. A random sample of 25 video frames of the plasma display was therefore captured to reduce the possibility of errors due the glitching effect on the frame synchronization on capturing. This proved satisfactory in practice, providing a good picture of the display to extract the information required for the calibration.

Although the final image provided a clear picture of the plasma panel display, the overall light levels were low, which increased the difficulty of separating the dark and light areas in the image by software techniques. The same scaling and contrast enhancement methods that were applied before (section 6.4.3) were implemented again in an effort to improve the contrast and brightness of the captured image. Because of the improved accuracy and definition of the original picture, these methods yielded a much improved image, without significantly altering the physical location of the pixels and the overall spread of intensity in the image. However, comparison between the results of images that had been enhanced, and the originals, and then both subjected to a threshold operation showed that there was no significant difference to the end result. It was therefore decided that the experiment would use the original image in the interest of introducing as few variables as possible.

The investigation of the choice of threshold was repeated to find out whether a better method was available. The calculation was first carried out by finding the histogram of intensity spread in the image, and was based on a technique suggested by Kapur et al [*96*]. This revealed the typical profile of two peaks for the background and the illuminated fibres in the bundle, with a 'valley' of intensities between the two peaks, shown in Figure 6.14 (a). The 'valley' indicates those pixels on the edges between the light and dark areas, where the intensities taper from the light to dark values. The threshold was taken to be the point in the valley that was half way between the two peaks. The peaks were located by performing first and second order differentiations on the histogram. The first differentiation indicated the points of zero or near zero, slope in the profile. The second differentiation showed whether these points were minima or maxima. This method showed that the profile was not a smooth distribution of intensities, especially for the lower order test bits, where

focussing errors were of most consequence. This meant that uneven points in the histogram could be indicated as peaks in the intensity distribution. These was labelled 'false peaks' for the purposes of the experiment (see Figure 6.14 (b)).

The validity of these peaks was checked by examining the separation between them. If they were closer than the expected separation, then the higher peak was chosen as the best representative value. Repeated testing showed that the best choice of peaks were separated by an intensity of about 50 units of intensity. Having chosen a threshold, the output of the fibre was modified according to this threshold value and the resulting image displayed. This was then visually compared with the original fibre output to determine whether a good threshold had been chosen for that test screen. If the choice was good, the calibration was allowed to proceed, and if not then the sequence for that test screen was repeated. From this visual feedback, it was decided that a more quantitative method of assessing the worth of a chosen threshold was needed.

By the nature of the test screen patterns, half of the total fibre area would be illuminated by every test screen. A good choice of threshold would therefore place half of the total fibre area, in pixels, above and half below the threshold value. To implement this calculation, the whole fibre bundle's output area had to be known. This was found by illuminating the entire calibration area on the plasma display panel. This image was then transmitted by the fibre, and the output analysed. The total illuminated area was calculated in pixels, and half this value was chosen as the target for the following calibration.

To find this new threshold value, the output from the fibre bundle was captured as described above. The histogram of intensity values was compiled, and then a summation of pixels at each intensity was conducted from the maximum intensity downwards. This was done until the total number of pixels above the threshold just exceeded the target value. The number of pixels rarely exactly matched the target value. The threshold was chosen such that the number of pixels was nearest the target value, either just under or just over.

The threshold value found by this method closely matched the value found by the two peaks method, especially so for the test patterns for the higher order bits of the coordinates. The two peaks method becomes less accurate as the calibration proceeds to the lower order bits. The primary problem is the limited resolution of the fibre bundle, causing a blurring of the test screens as the image is transmitted

through the bundle. Examination of the histogram of intensities for such an output shows that the peaks are less well defined. For the least significant bit the histogram shows a profile of one peak of near Gaussian distribution of intensities (Figure 6.17 (b)). In this instance, the area threshold method provides the best indication of the threshold to be used. The calibration method was tried with the peak method choice of threshold for the higher order bits and the area method for the lower order bits. The results still showed some error, partly because of the inaccuracy of the peak method, and small misalignments with the test screens and the fibre bundle causing errors in illuminating exactly half of the bundle.

The threshold method which give the most consistent result was that described in section 6.4.3: the choice of a threshold for each pixel in the image by finding its maximum possible intensity when transmitting a light image, and its minimum intensity, the threshold being the midpoint of these two values. A further investigation of the principle, applied to more complex images, is described by [*97*]. The more simple method described in section 6.4.3 was therefore used in the final calibration method.

6.6.4 Discussion

The use of the plasma display provided the accuracy and consistency of test screen definition that was expected. The disadvantages were the low light intensity output and the non-synchronised display. Both of these factors were overcome, to different levels of success, by software compensation techniques. The following discussion is concerned with the calibration of the high resolution coherent optical fibre bundle.

The combination of the relatively limited resolution of the fibre bundle and camera system did not allow the limit of accuracy of calibration possible from the use of the test patterns to be achieved. The data from the lowest order test screen was sufficiently distorted by the fibre to make it necessary to disregard the results obtained for that test pattern. However, as the results (chapter 7) show, it is possible to realise a creditable reconstructed image from the reduced resolution of this system. The use of a coherent fibre bundle for the experiment allowed the results to be viewed in direct comparison with the true fibre bundle output, and aided in the detection of sources of error and the implementation of the necessary image processing algorithms.

Detailed discussion and analysis of the results for each source of the test patterns will be undertaken in Chapter 7, but several factors that must be taken into account

Detailed discussion and analysis of the results for each source of the test patterns will be undertaken in Chapter 7, but several factors that must be taken into account in the design of the calibration theory and system need to be outlined. Of primary importance is the resolution of the imaging system. The absolute limit of resolution is determined by the fibre bundle used to transmit the image (section 3.5.2.1). The various methods of producing and displaying the test patterns will determine the limit of resolution of the calibration. In using the plasma display panel, it became obvious that the resolution limit of the bundle of fibres was exceeded. It has been explained (section 6.4.4) how the wrong choice of threshold for an illuminated pixel in the output image can lead to large errors in the relocation of data in the transmitted image. The use of the plasma display panel reduces the variation in intensity in the original test pattern and therefore minimises this source of error. Enclosing the calibration system in an optically sealed environment further reduces the errors that can occur by reflection of the test images and extraneous light sources.

The same software routines were applied to the calibration of the incoherent fibre bundle. It was expected that the limited resolution of this bundle would play a significant part in the effectiveness of the calibration and therefore the quality of the result. One other factor to be considered in the calibration of the incoherent bundle is the focusing of the test patterns onto the input face of the fibre. There is obviously no direct comparison that can be made between the input and output images to decide when the input image is in focus. The solution used was to focus the image onto the input end of the coherent bundle and then replace it with the incoherent bundle in the same position. The different diameters of the bundles also had to be taken into account when focusing the image of the test patterns onto the input end. The image for the coherent bundle was reduced to occupy an area that covered a concentric circle of 3.5 mm on the input face of the coherent bundle, the diameter of the incoherent bundle. The result of inaccurate focusing of the test pattern onto the input face of the fibre bundle can be seen in Figures 6.16 (a) and (b), and compared with the histogram for the focused image in Figure 6.15 (a) and (b).

Using the same calculation as in sections 3.5.1.1 and 6.5.4, shows that the 3.5 mm bundle contains 4400 fibre, 75 % (3300) of which will be used for the transmission of images over a square area. This yields an effective resolution of 57 units square compared to over 250 units in each direction for the coherent bundle. This would indicate a much lower resolution of the test screens and therefore substantially reduced resolution after calibration.

6.7 Calibration of Incoherent Optical Bundle

The two primary issues to be considered in implementing the calibration by test patterns were the difficulty in focusing the test images onto the input face of the bundle, and the low resolution capability of the incoherent optical fibre bundle. The effect of incorrect focusing of the test patterns was seen in Figures 6.16 (a) and (b), where the histogram plot for the image shows the difficulty of choosing a correct threshold for the image. The wrong choice of threshold for any pixel in the image will give rise to an error in the reconstructed image, the displacement of the intensity in the image being dependent on the significance of the address bit at which the error was generated, as described in section 6.4.4.

The first issue was overcome by using the calibration data from the spot calibration method for the incoherent bundle to reconstruct the output in order to visually determine when the input image was in focus. The results of this operation can be seen in Chapter 7. This method also allowed the resolution limit of the bundle to be clearly seen, with allowances made for the errors generated from the spot calibration method. This was seen to be bit the test pattern for bit number 3, in the numbering system 0 to 7, 0 being the least significant bit. This showed that the bundle could be calibrated to a 5 bit resolution by this method. This is equivalent to 32 units of resolution square, which is lower than the theoretical limit of 57 units as predicted above.

It was decided that the test patterns that gave the best results with the coherent fibre bundle would be the most suitable for use with the incoherent bundle. Examination of the results for the calibration of the coherent bundle of fibres (Chapter 7), showed that all the methods used, for various reasons, were subject to some degree of error. It was decided that the best overall performance was obtained from the photographic slides and independent illumination method. This method was therefore used to calibrate the incoherent optical fibre bundle, as described in section 6.5. The results for this calibration trial will be reviewed in the next chapter.

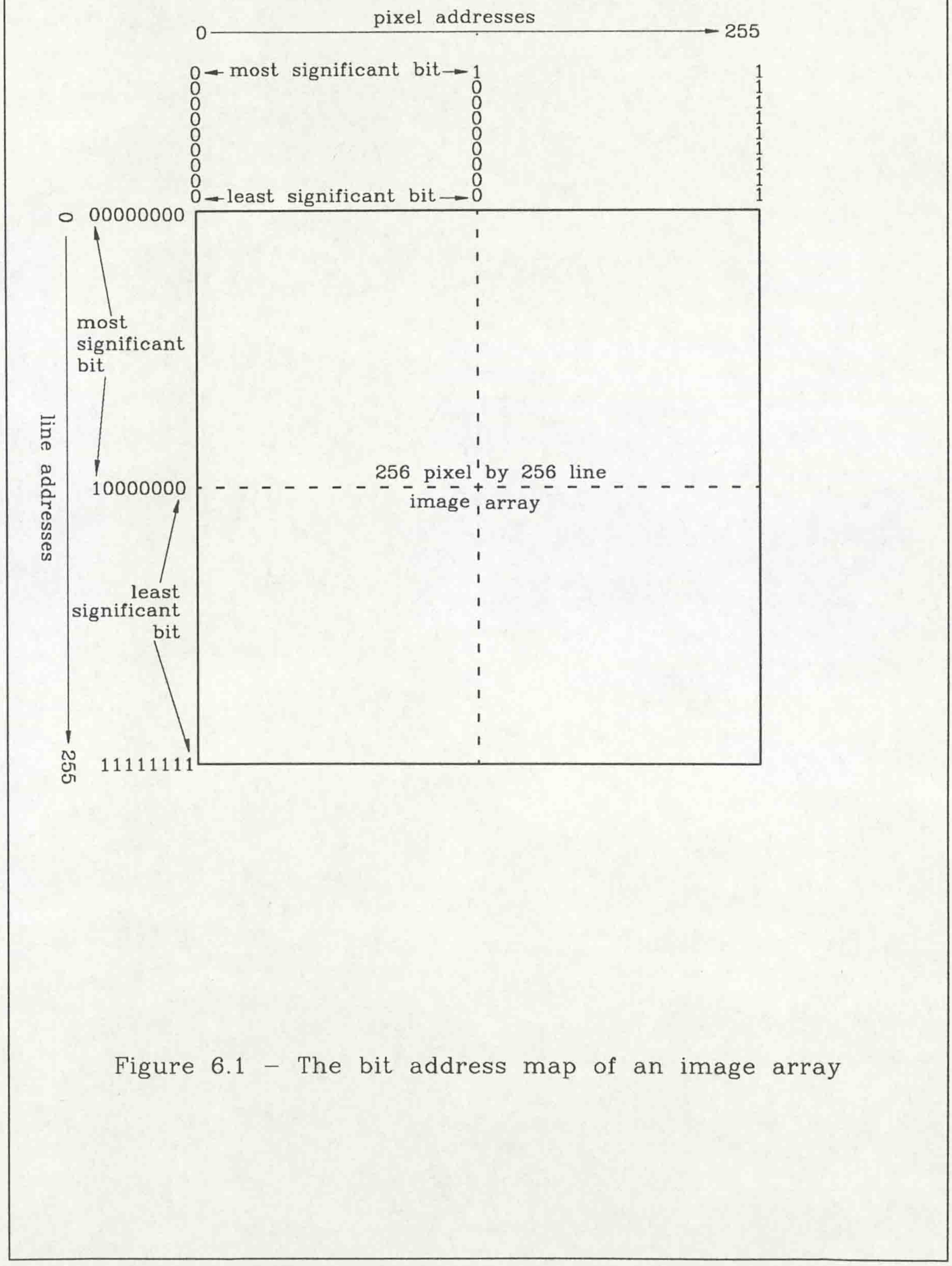


Figure 6.1 – The bit address map of an image array

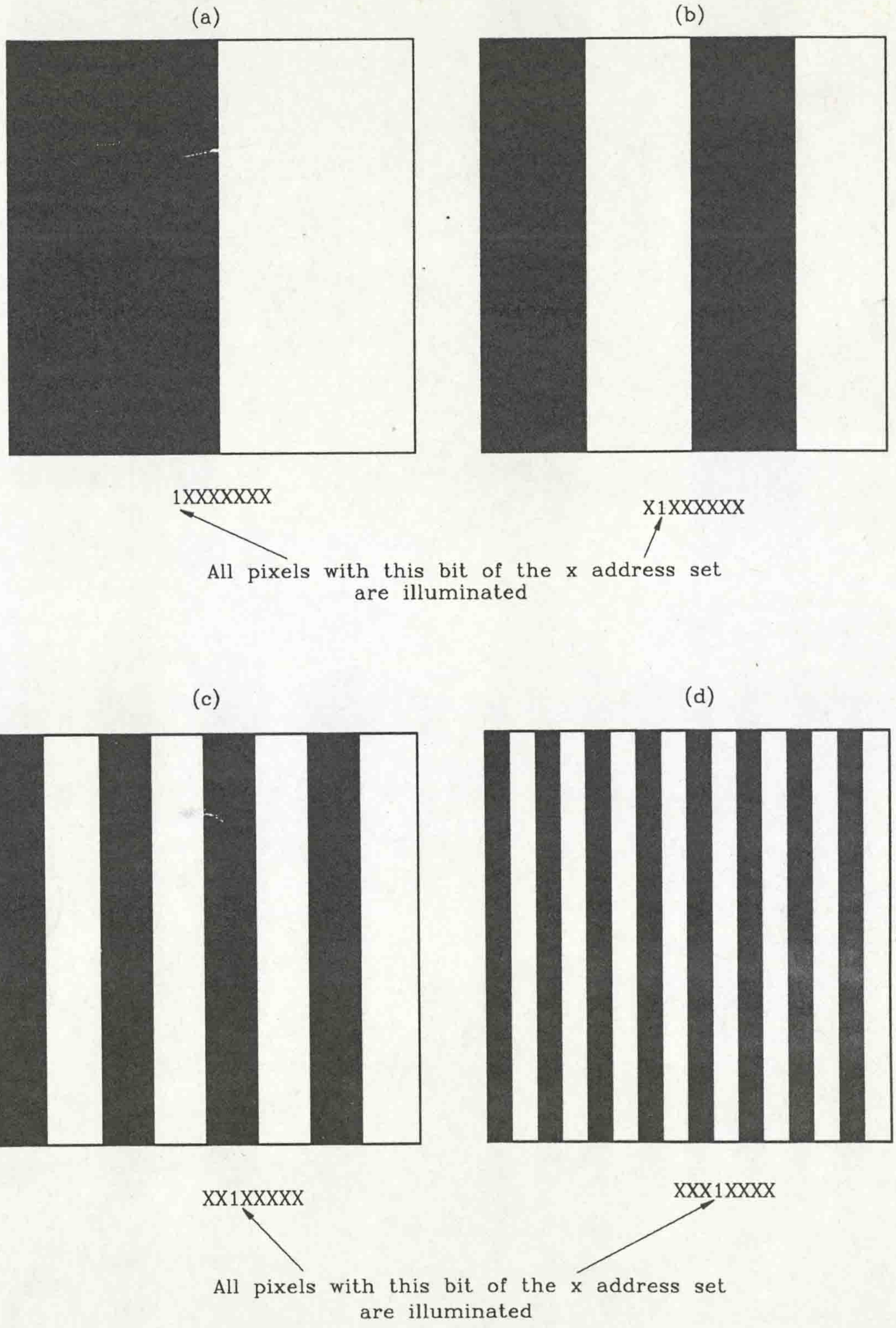


Figure 6.2 (a) – (d) – the test screen patterns for the upper 4 bits of the x address

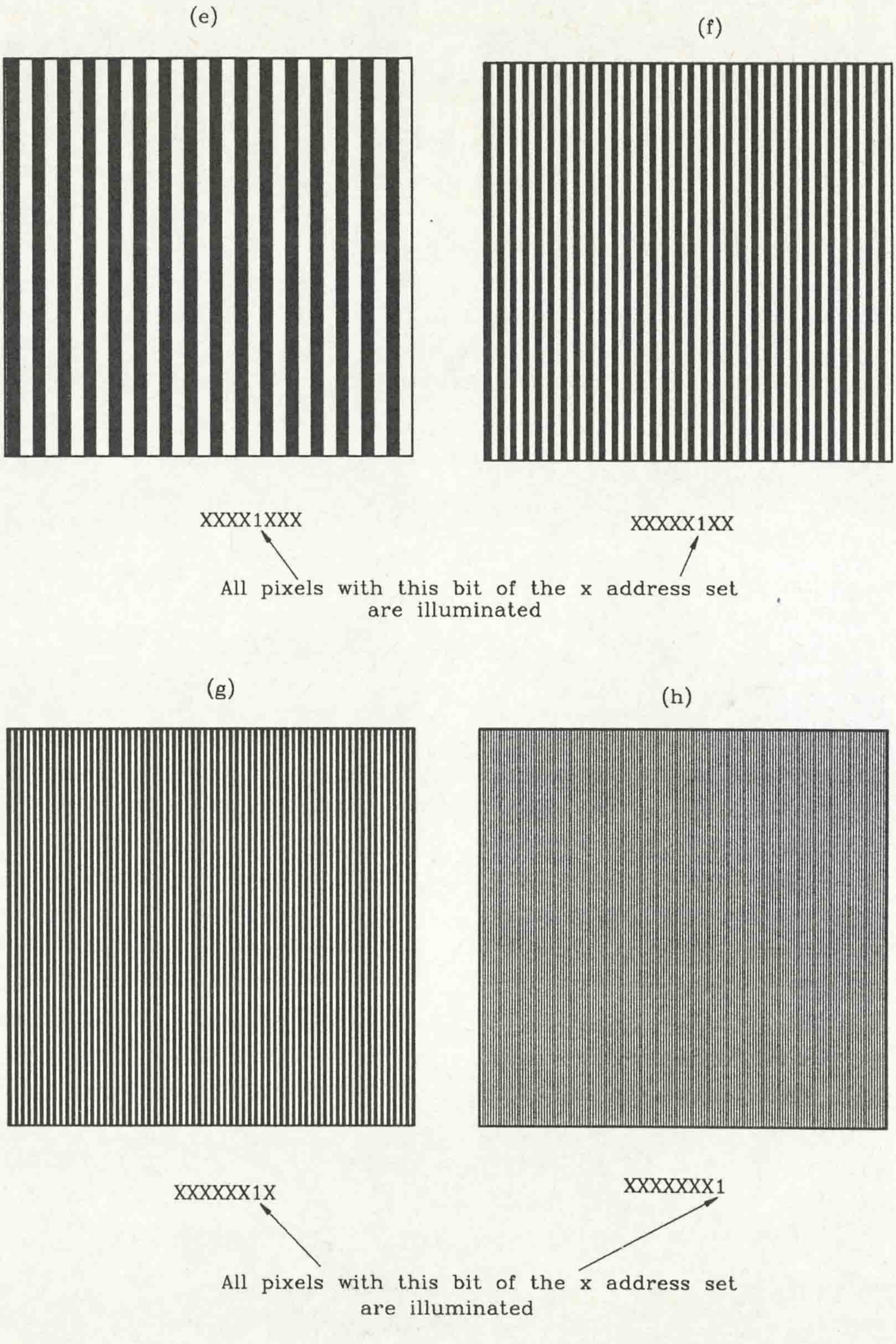


Figure 6.2 (e) – (h) – the test screen patterns for the lower 4 bits of the x address

Output coordinate (x or y) for which
input coordinate is sought : 11010011

Input coordinate initially set to : 00000000

Test Screen coordinate pattern	Was output coordinate illuminated?	Result of Logical OR operation between test screen pattern and input coordinate
10000000	YES	10000000
01000000	YES	11000000
00100000	NO	11000000
00010000	NO	11000000
00001000	YES	11001000
00000100	YES	11001100
00000010	NO	11001100
00000001	YES	11001101

Sample result : Output coordinate 11010011 is mapped to
Input coordinate 11001101

Figure 6.3 – An example of the formation of an input coordinate

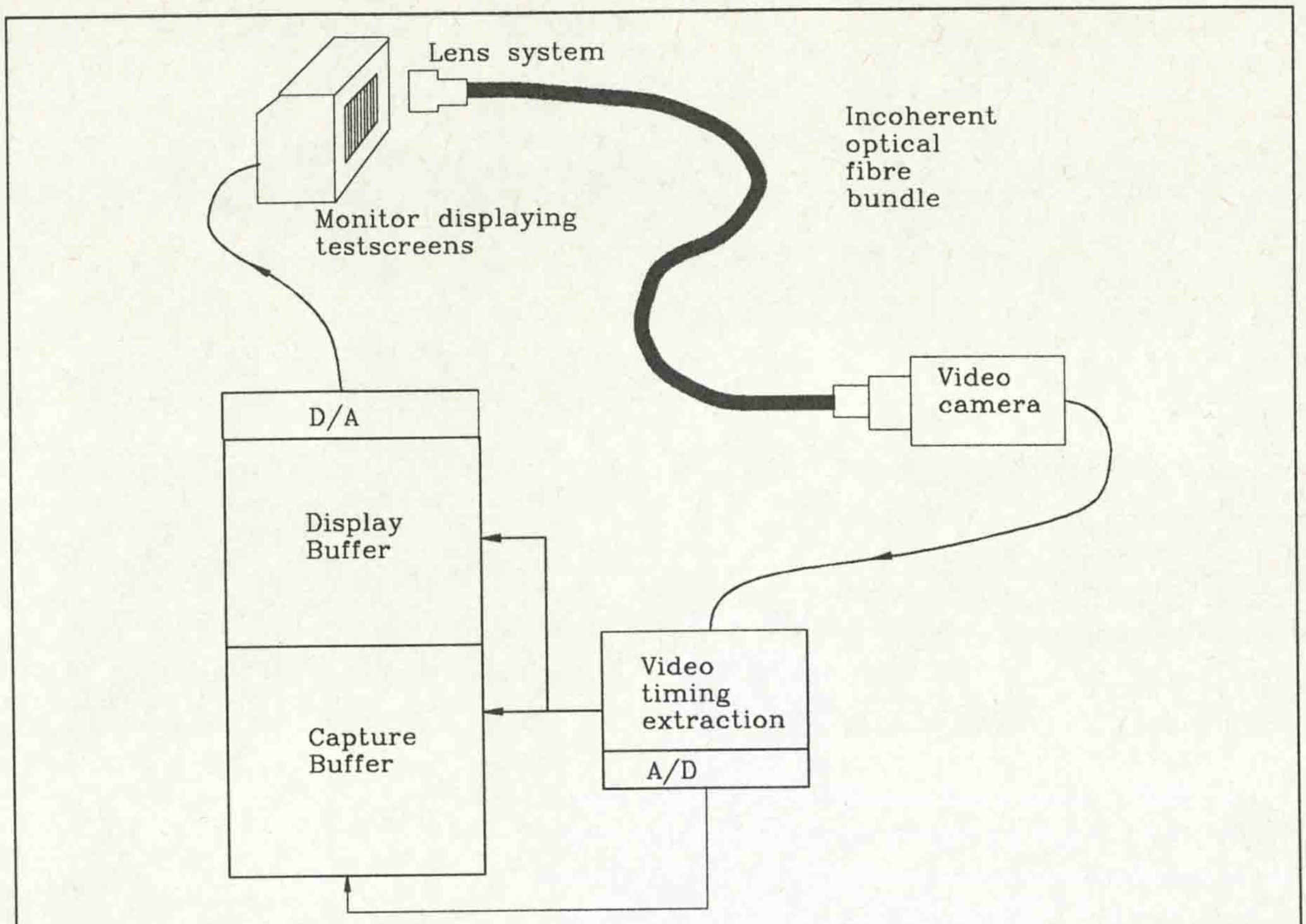


Figure 6.4 – The simultaneous display and capture of video data

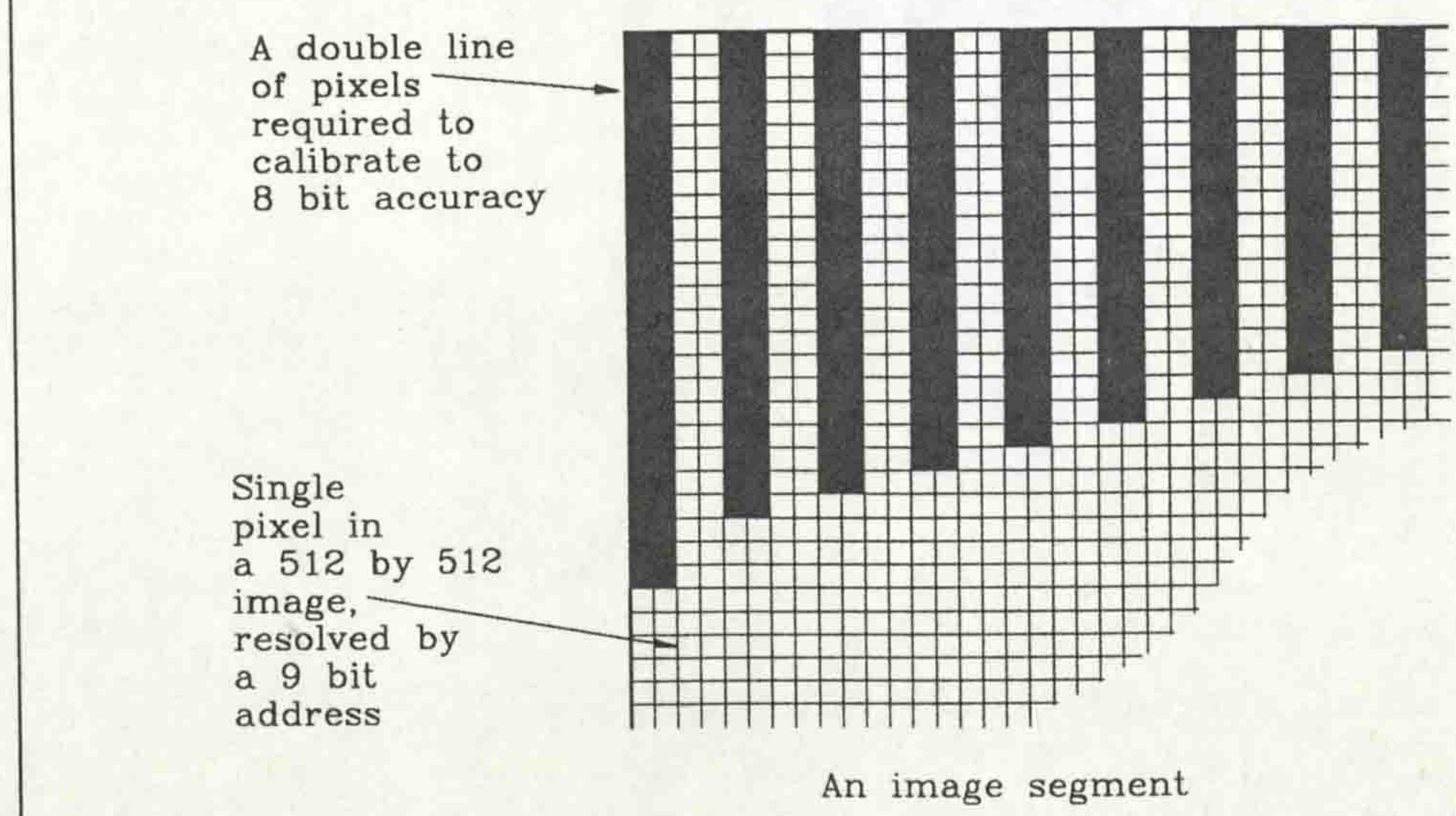


Figure 6.5 – Use of a 512 by 512 array to calibrate to 8 bits

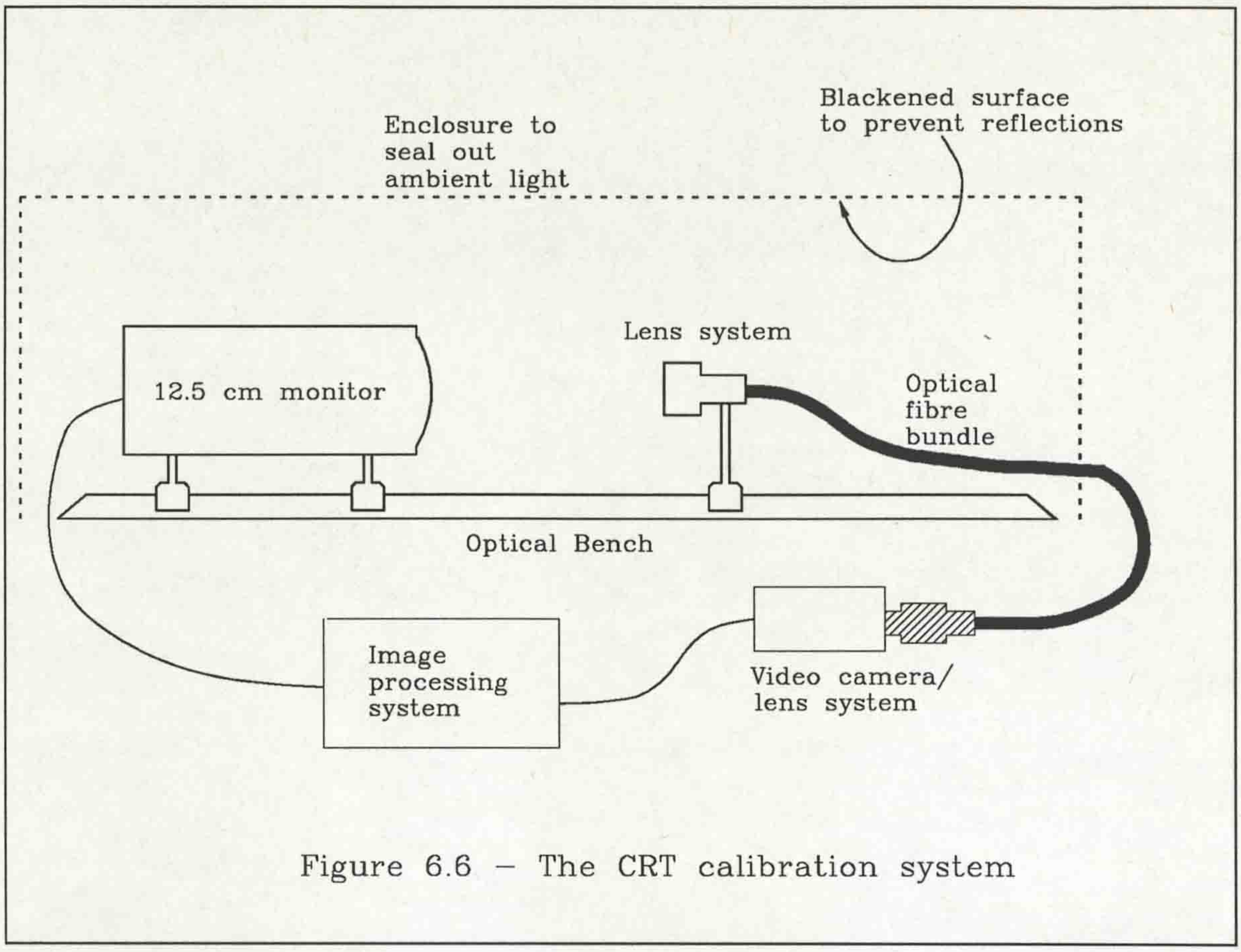


Figure 6.6 – The CRT calibration system

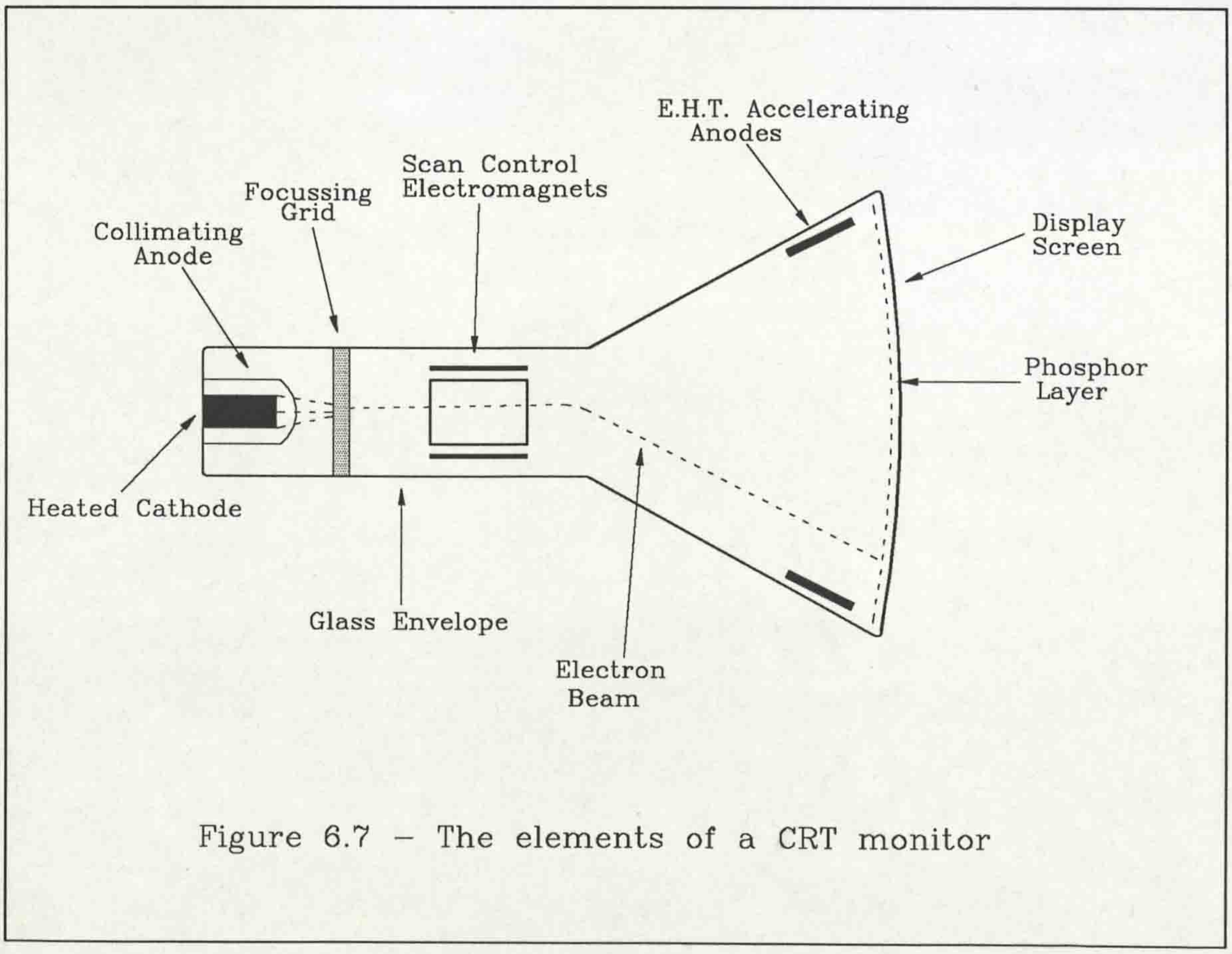


Figure 6.7 – The elements of a CRT monitor

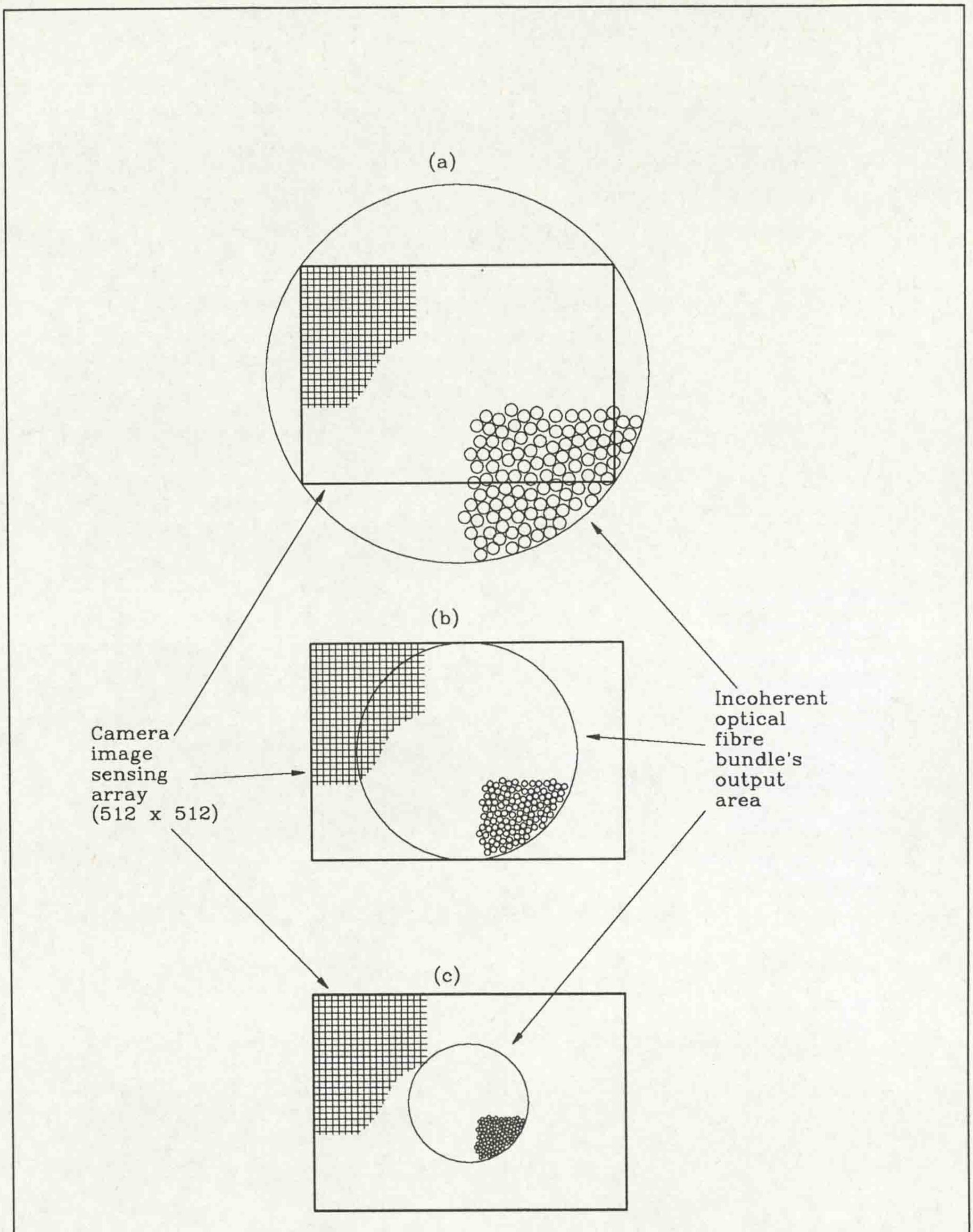


Figure 6.8 (a) – (c) – The effect of various magnifications on the bundle size/ sensor size ratio

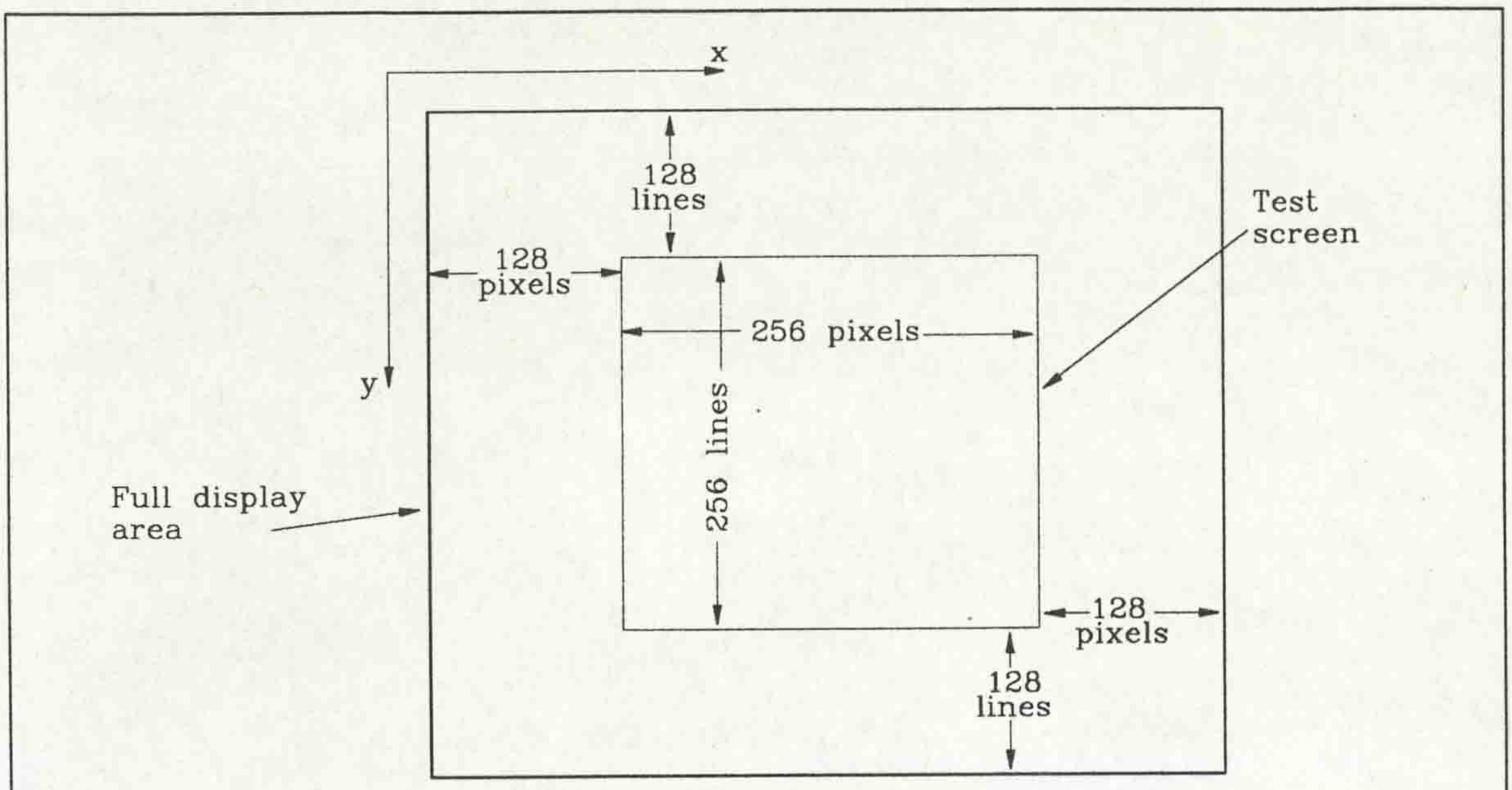


Figure 6.9 The location of the test screens within the display area of the PIP-1024

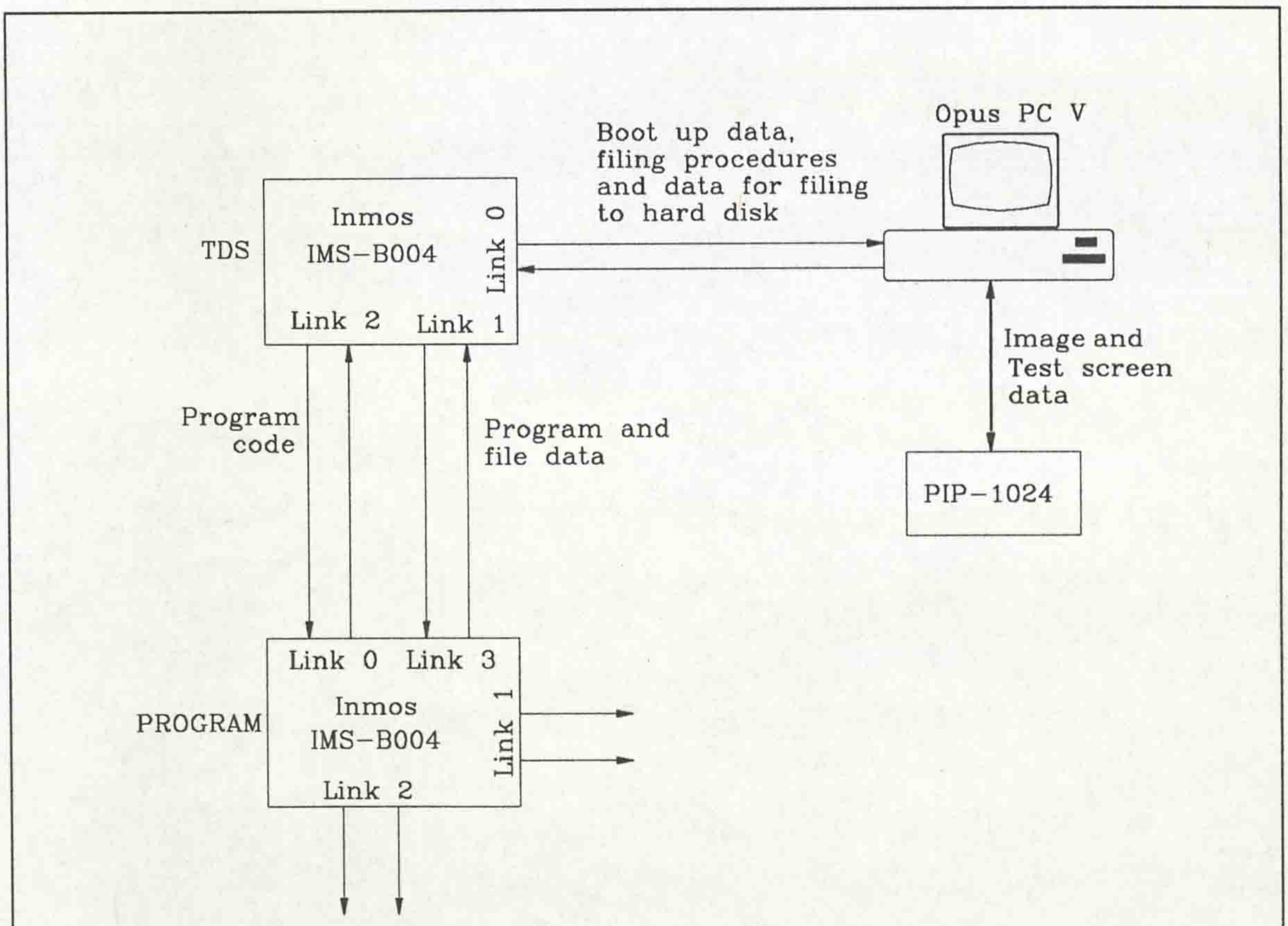


Figure 6.10 - Software data flow for the PIP-1024 system

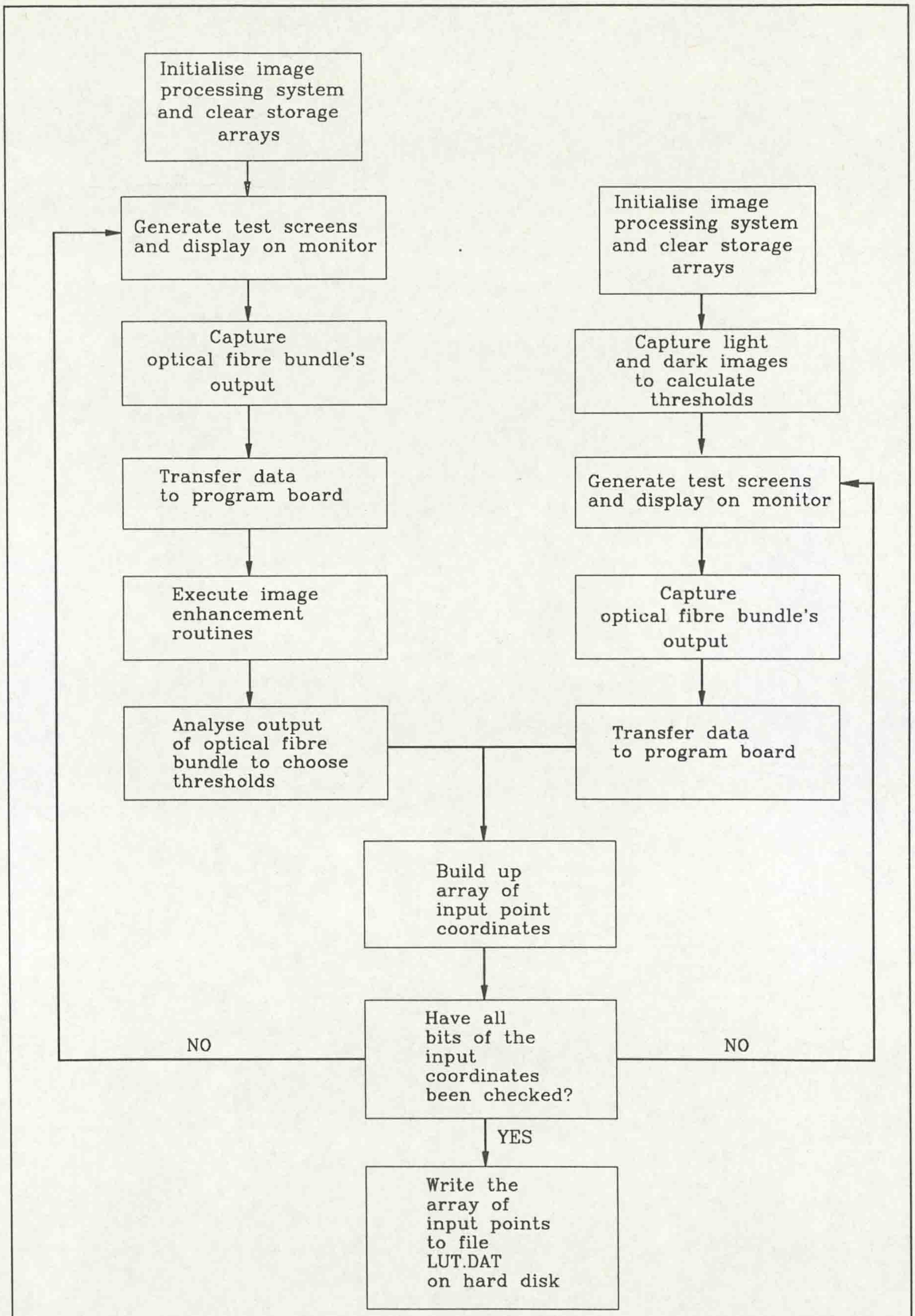


Figure 6.11 – The structure of the calibration by test screens

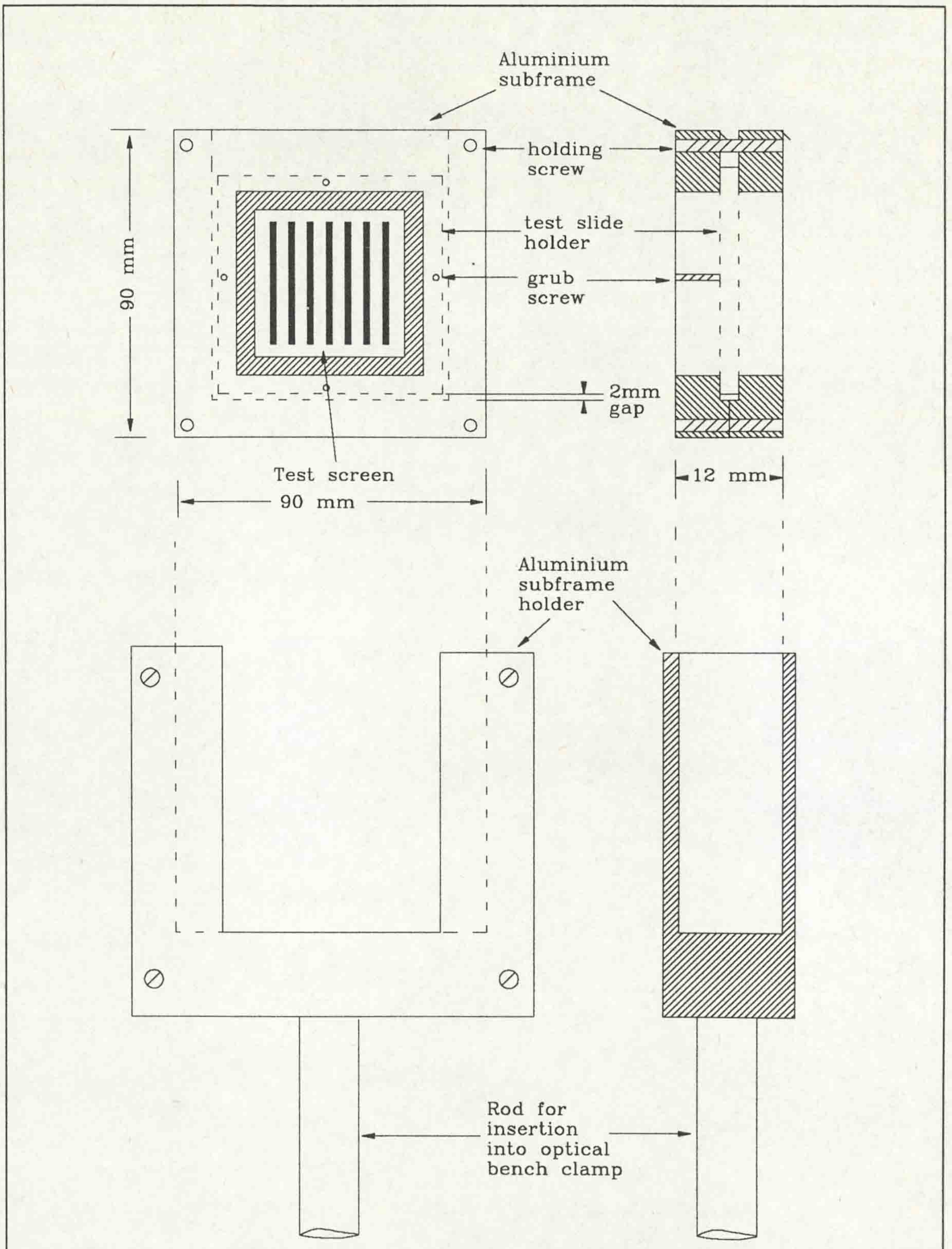


Figure – 6.12 The test slide, the subframe and holder

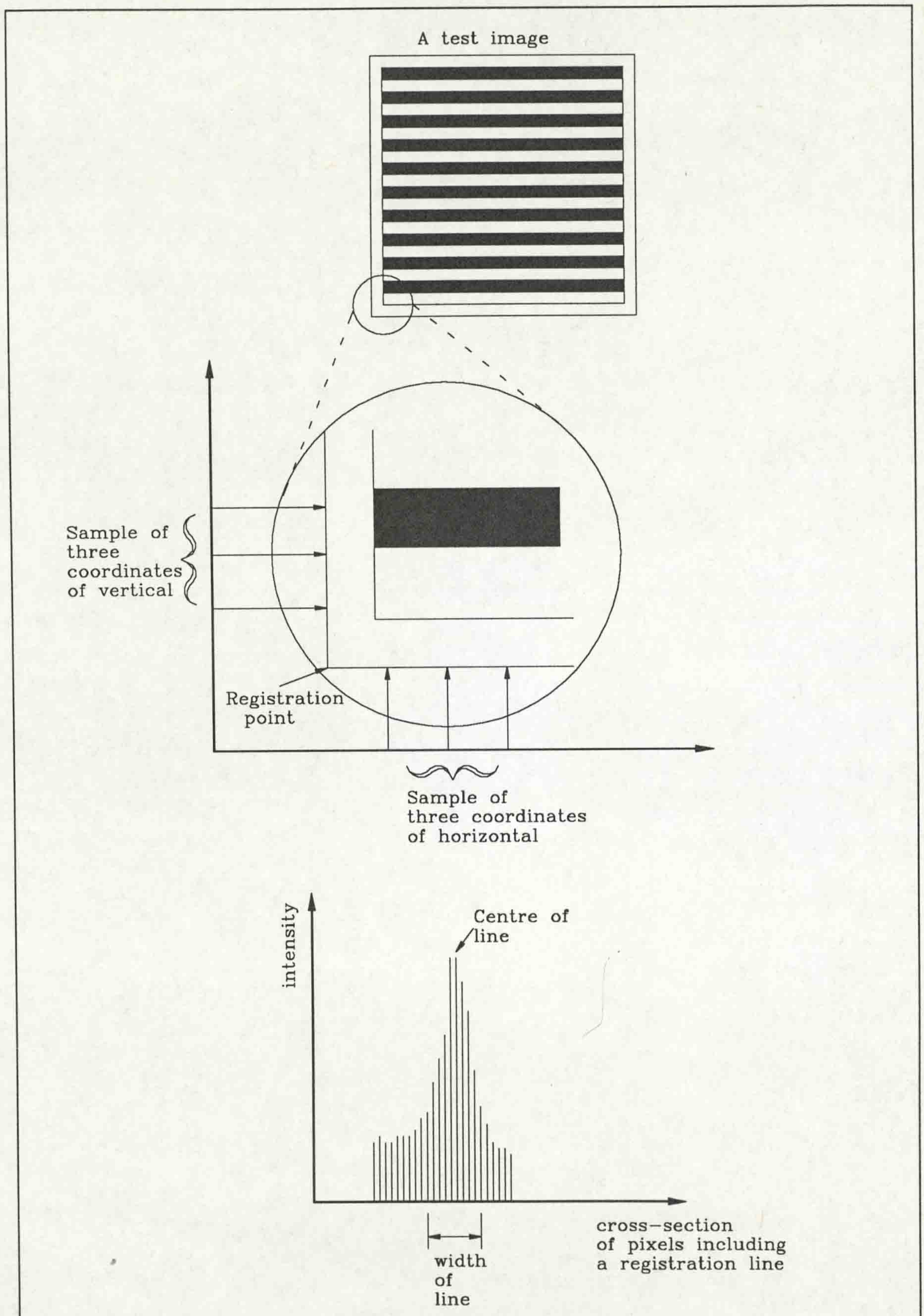


Figure 6.13 – The location of test slides by registration marks

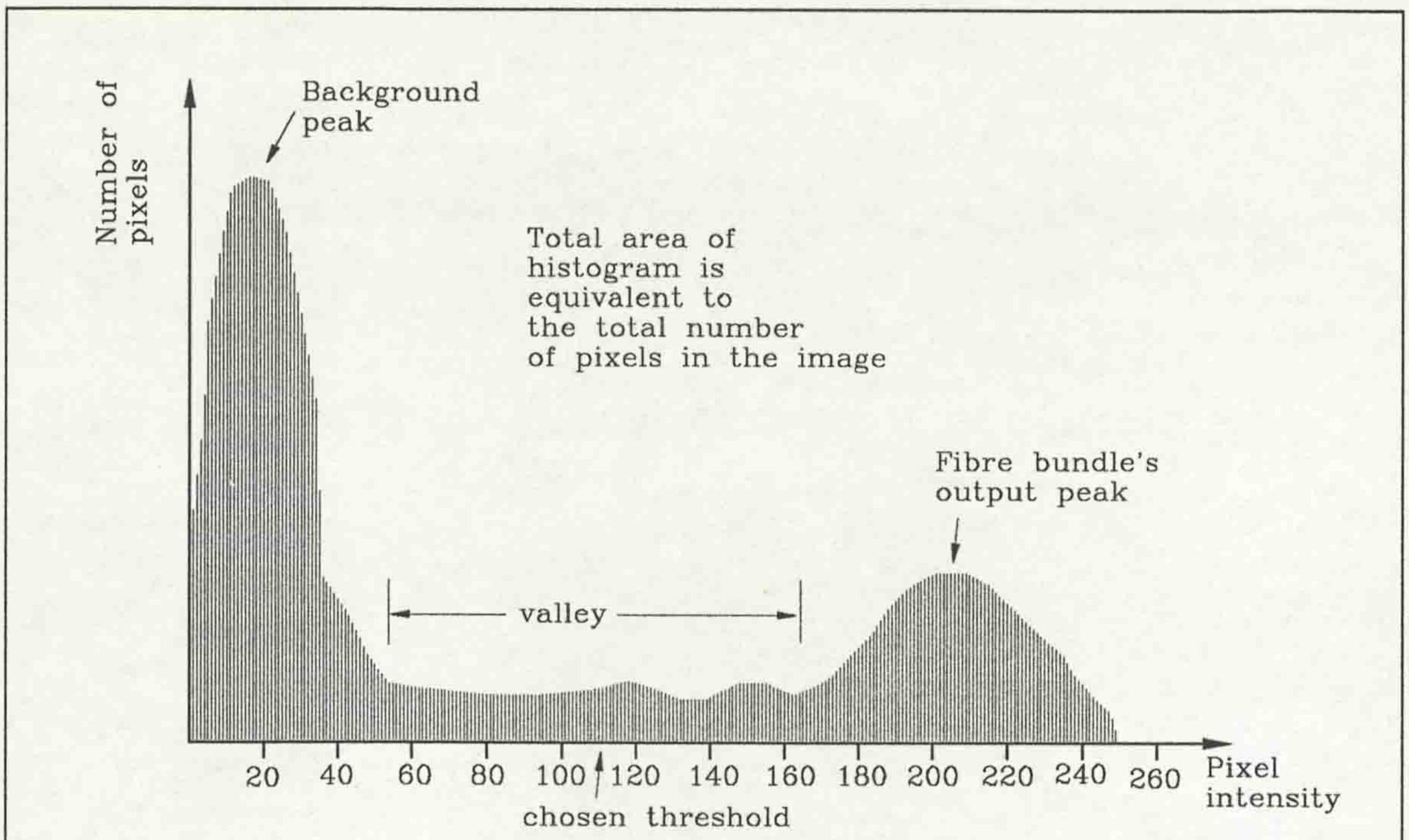


Figure 6.14 (a) – Typical intensity distribution for a test screen output from an optical fibre bundle

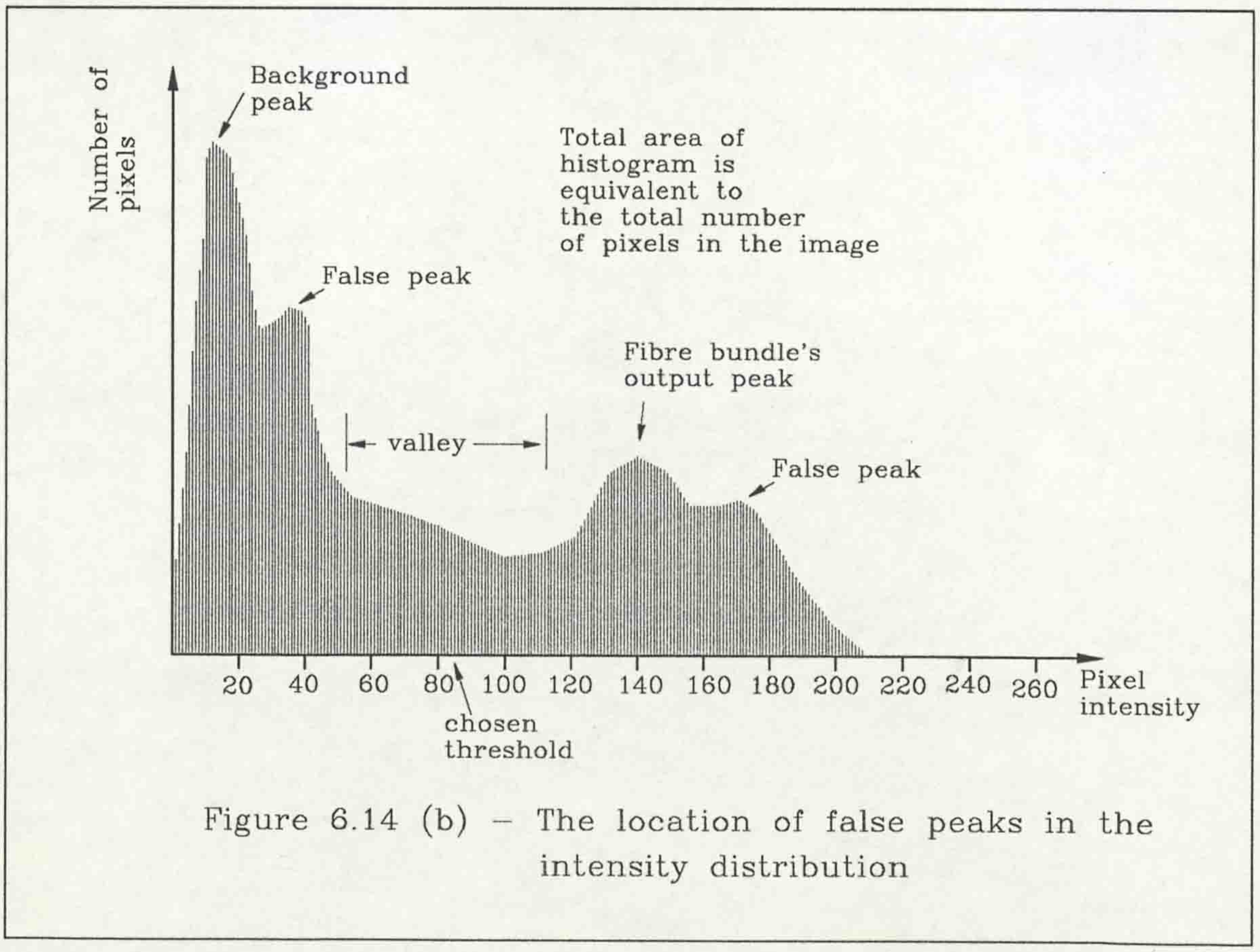


Figure 6.14 (b) – The location of false peaks in the intensity distribution

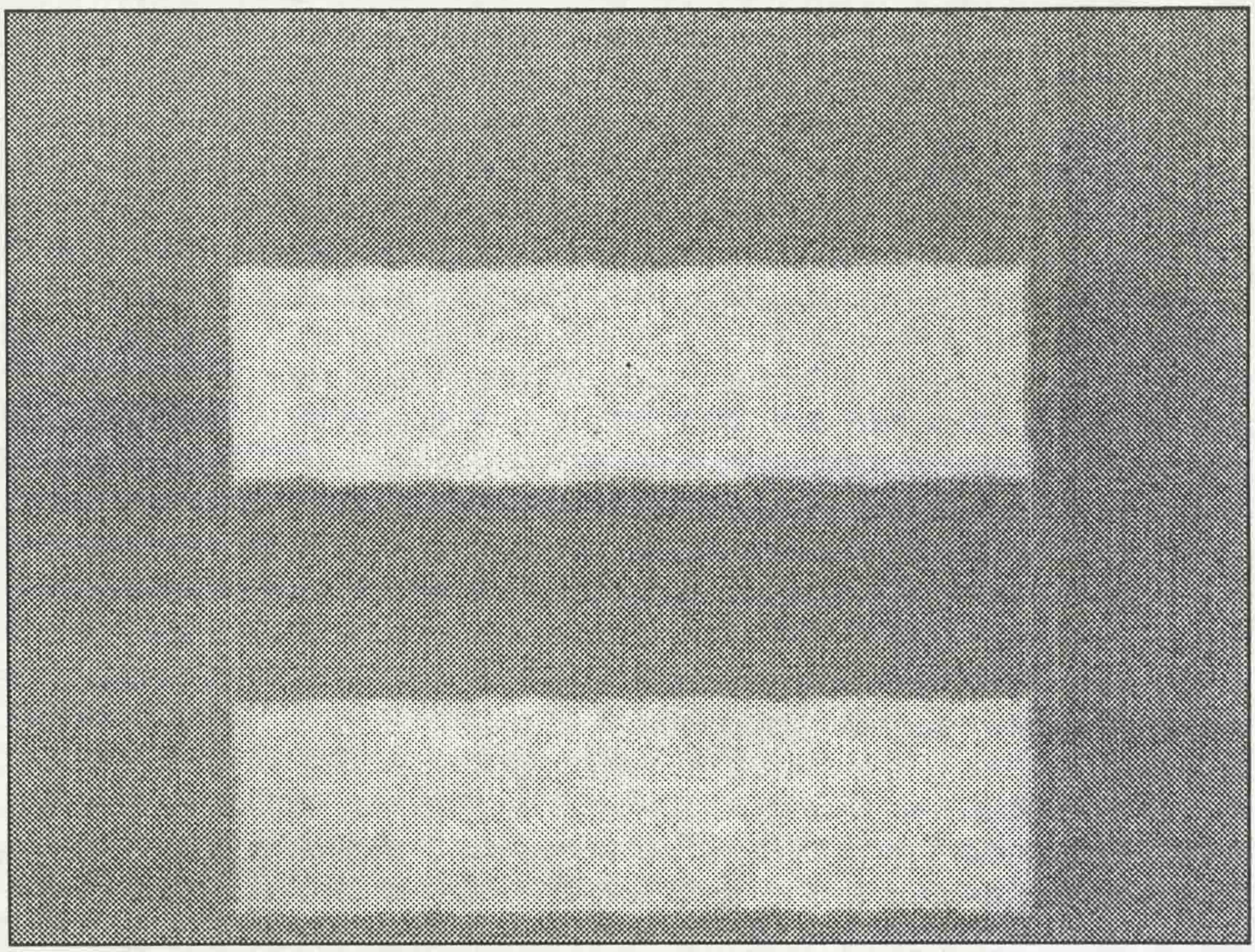


Figure 6.15 (a) - The image of a well focussed high order test pattern

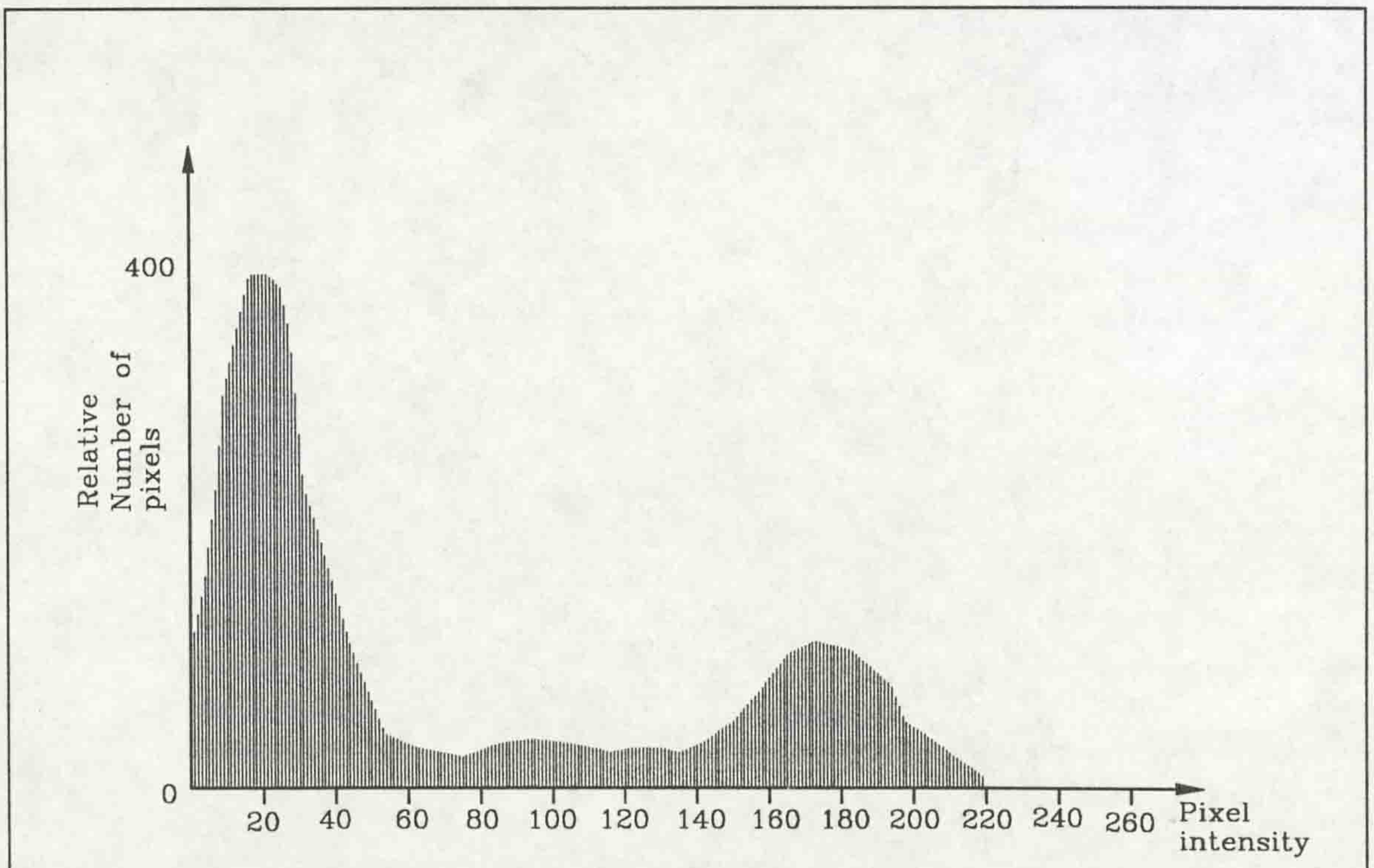


Figure 6.15 (b) – The histogram plot of intensities of the image in Figure 6.15 (a)

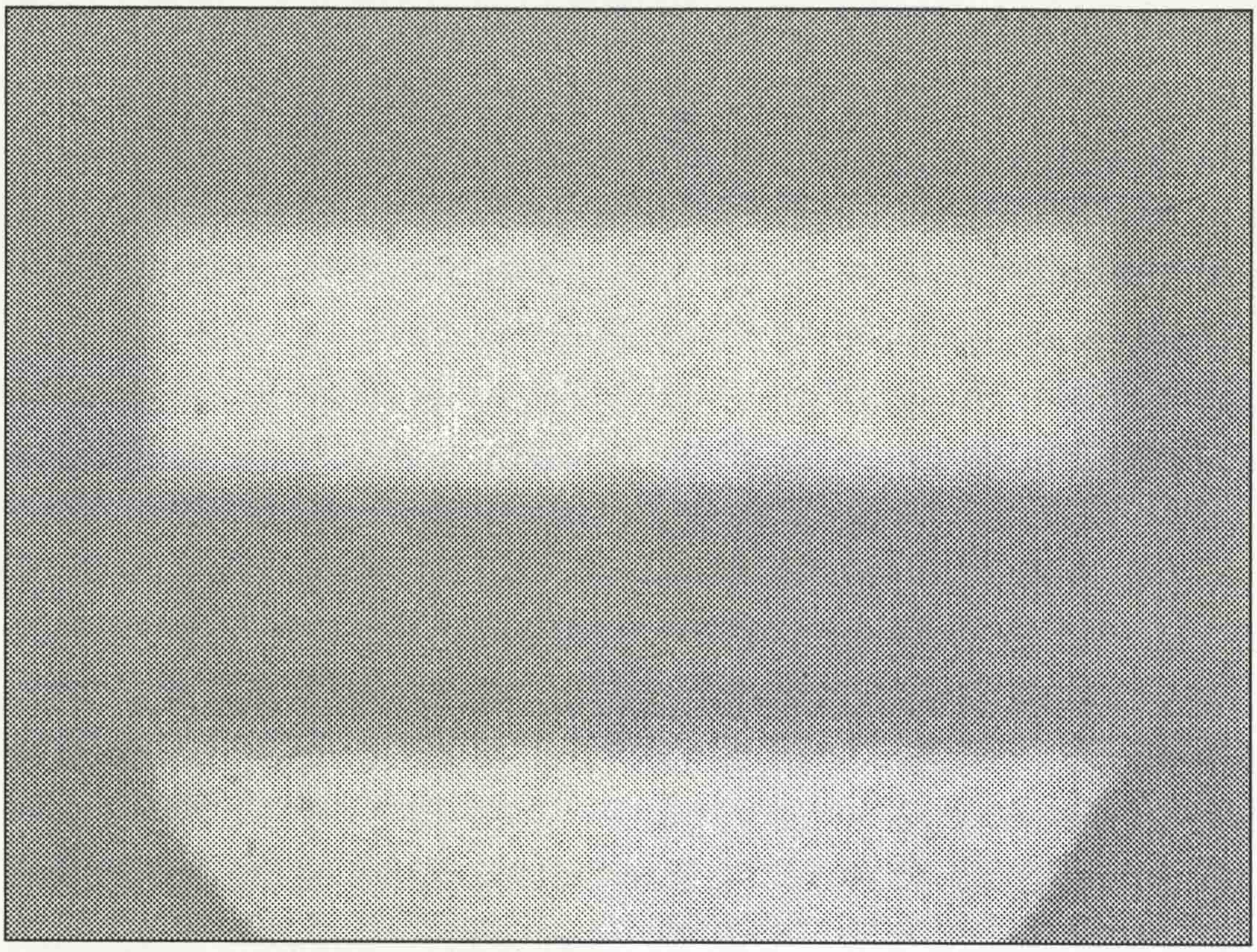


Figure 6.16 (a) - The image of a poorly focussed high order test pattern

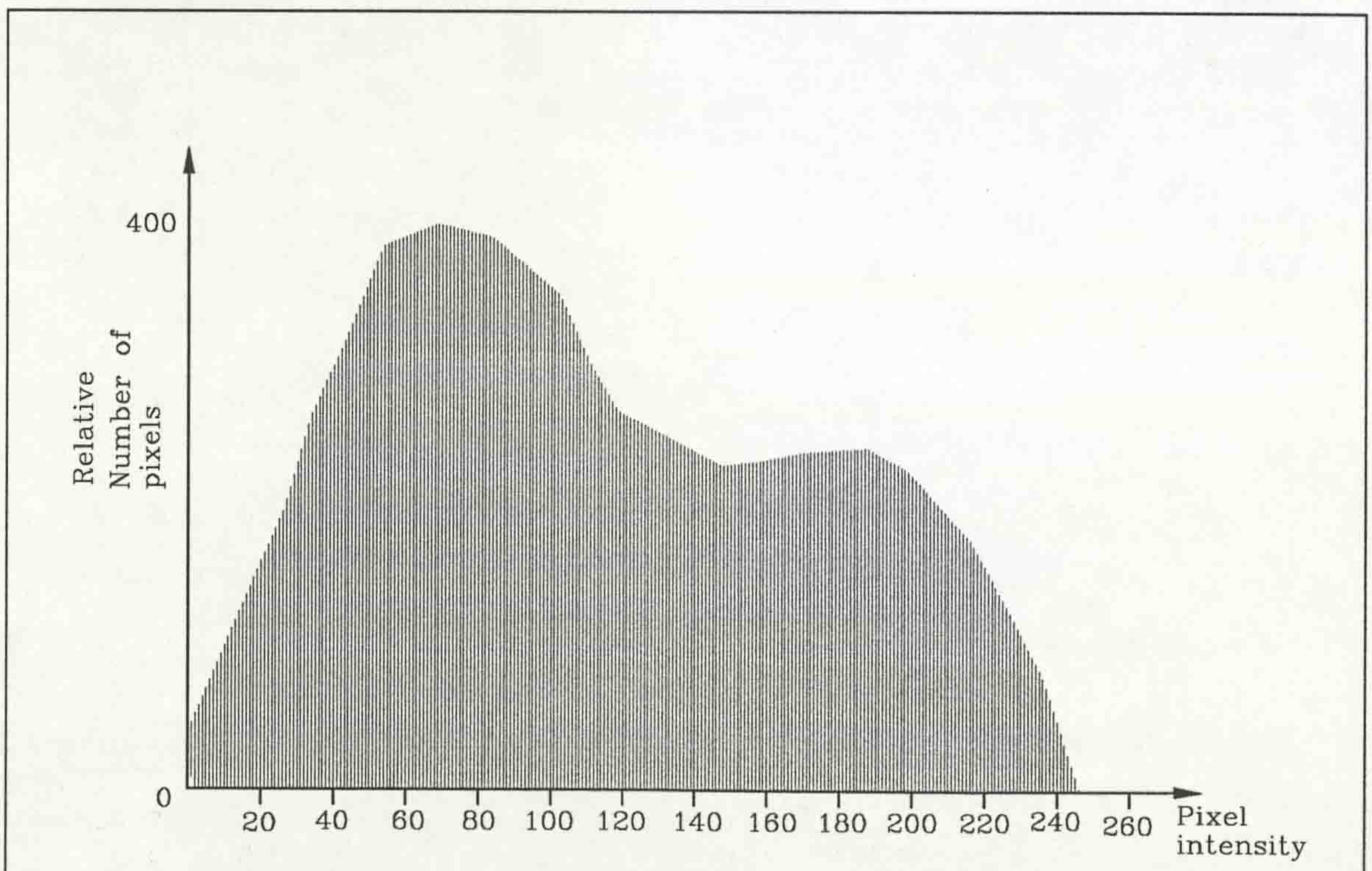


Figure 6.16 (b) – The histogram plot of intensities in the unfocussed image in Figure 6.16 (a)

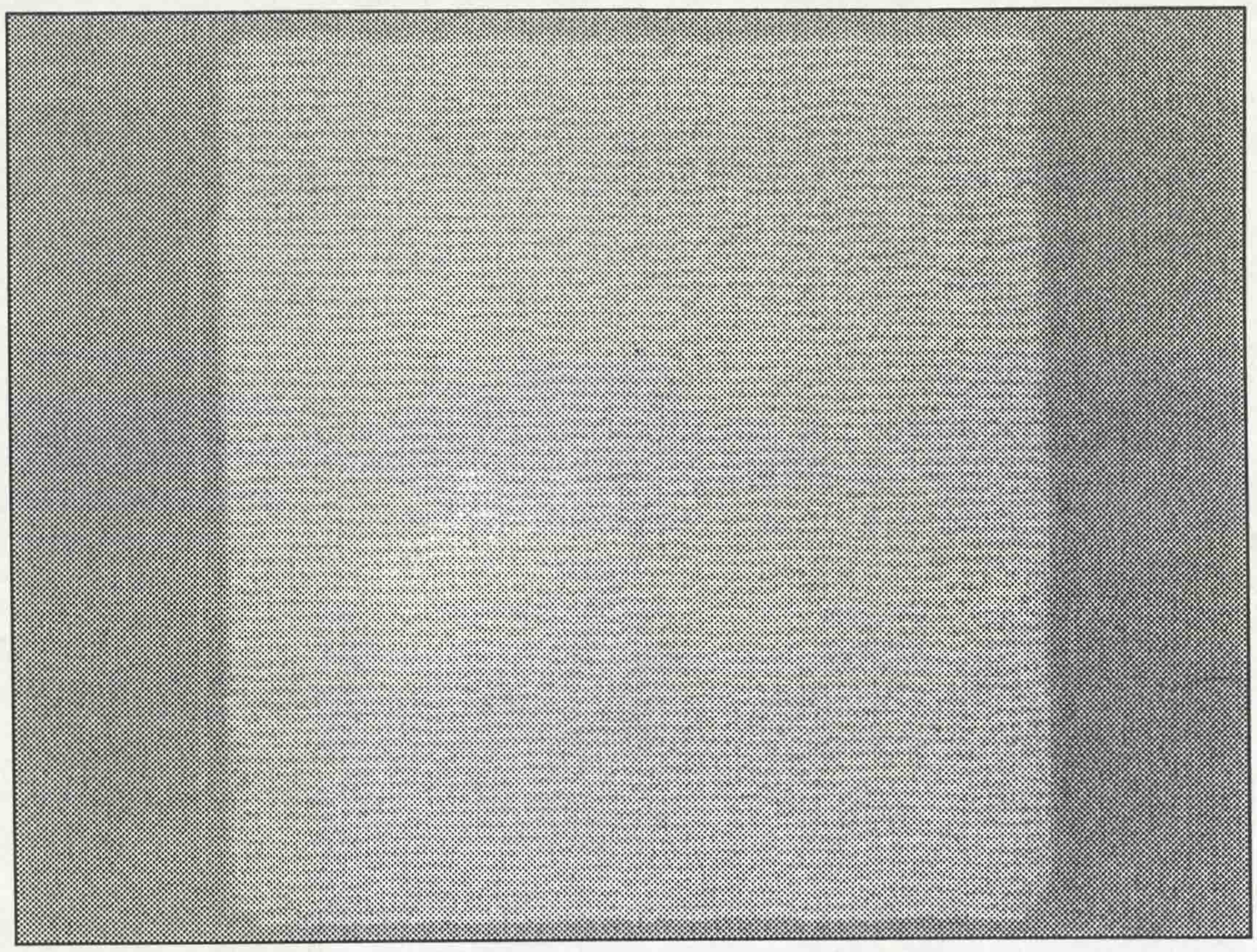


Figure 6.17 (a) - The image of a low order test pattern

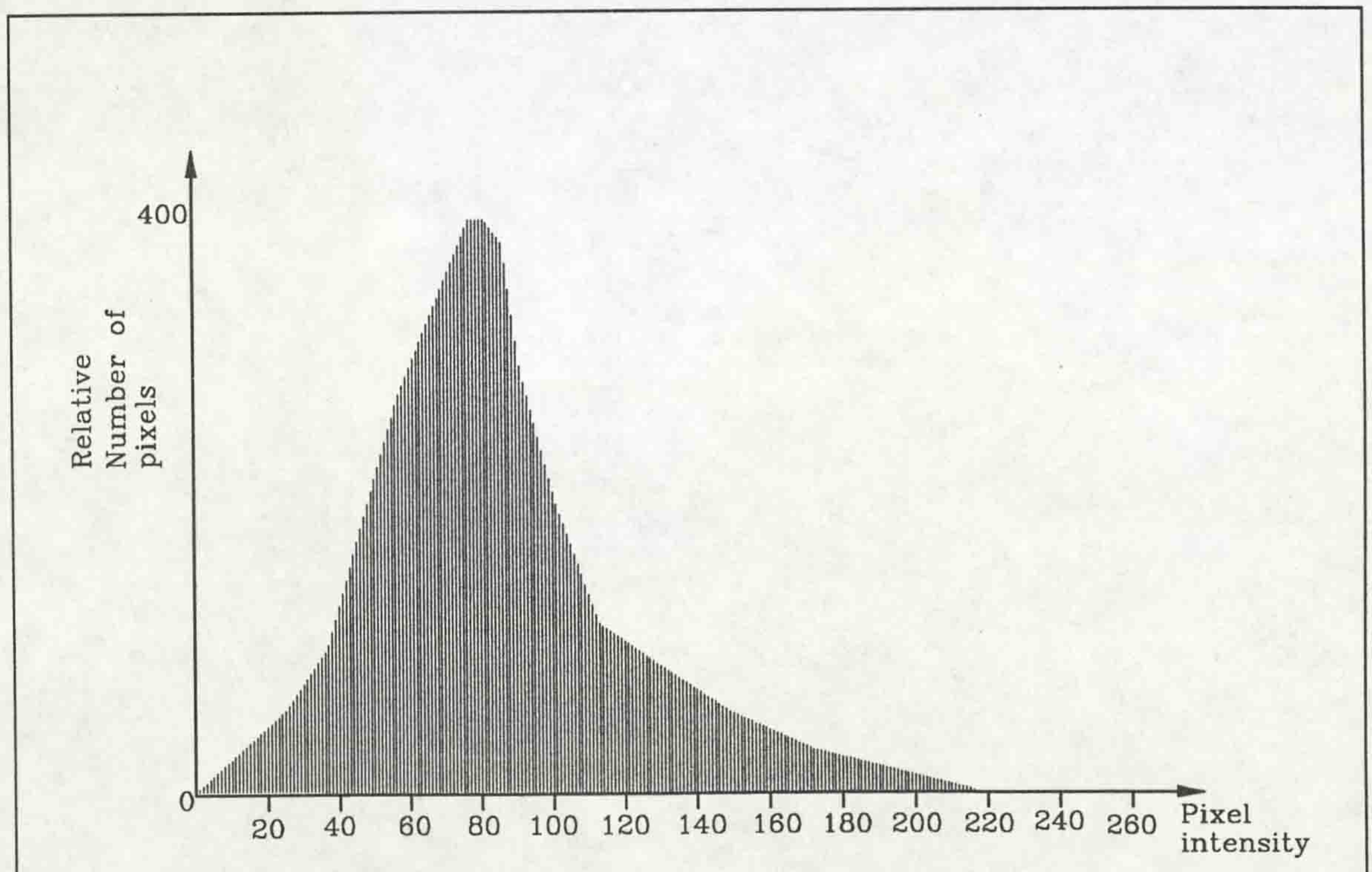


Figure 6.17 (b) — The histogram plot of intensities of the image in Figure 6.17 (a)

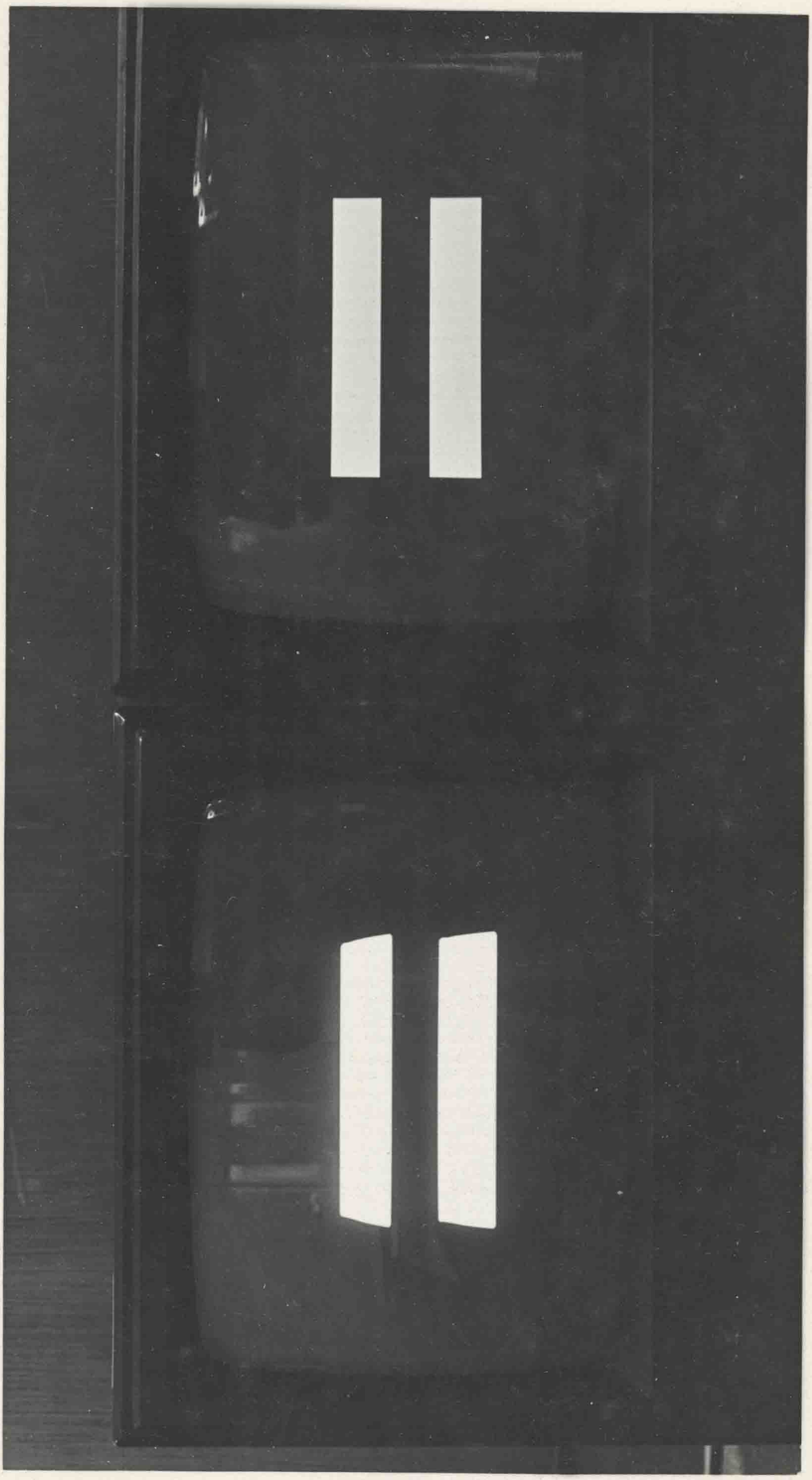


Plate 6.1 - Two identical monitors, the one on the left showing distortion at high brightness/contrast levels

CALIBRATION BY TEST PATTERNS

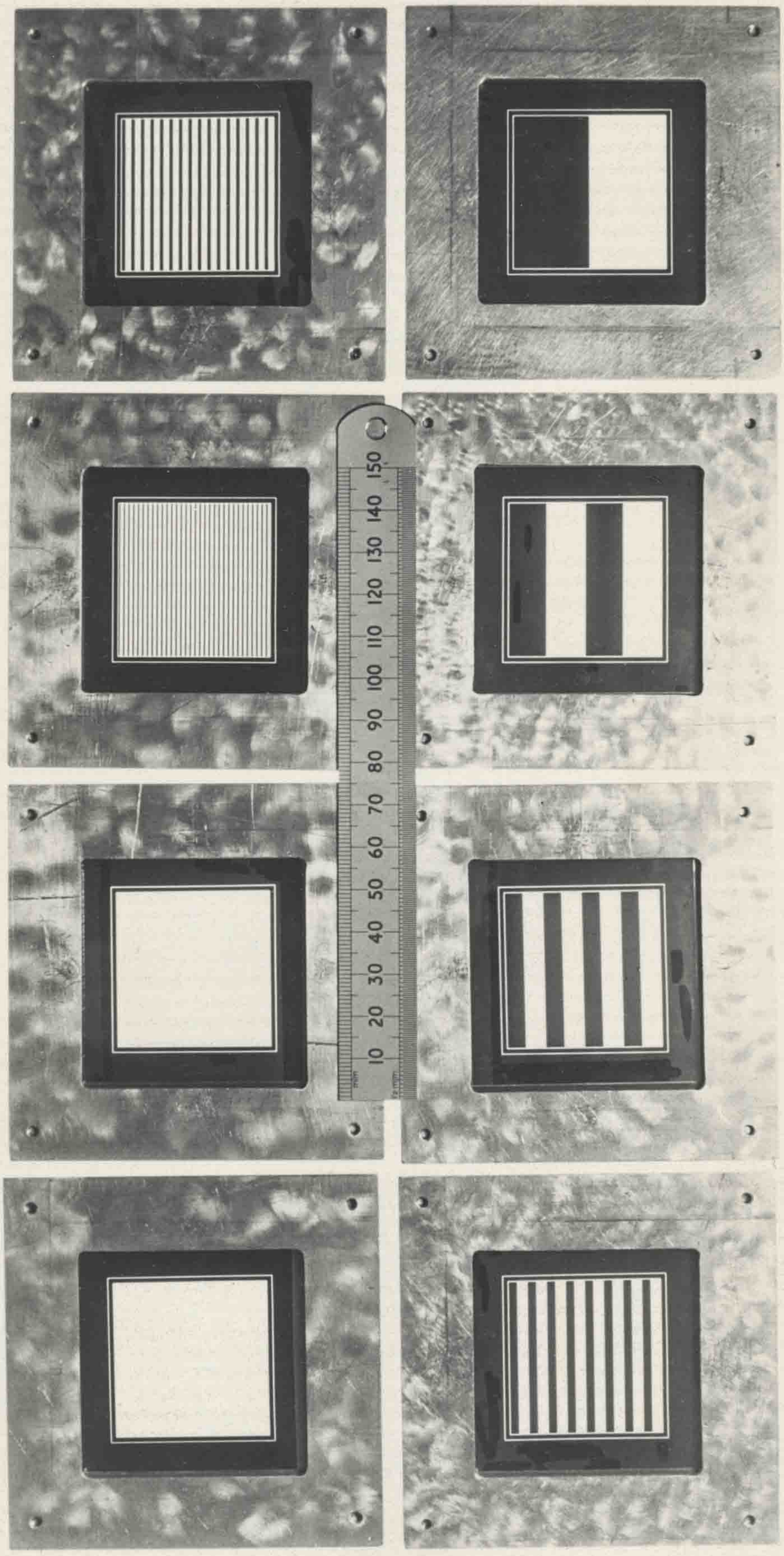


Plate 6.2 - The complete set of photographic test slides in their subframes

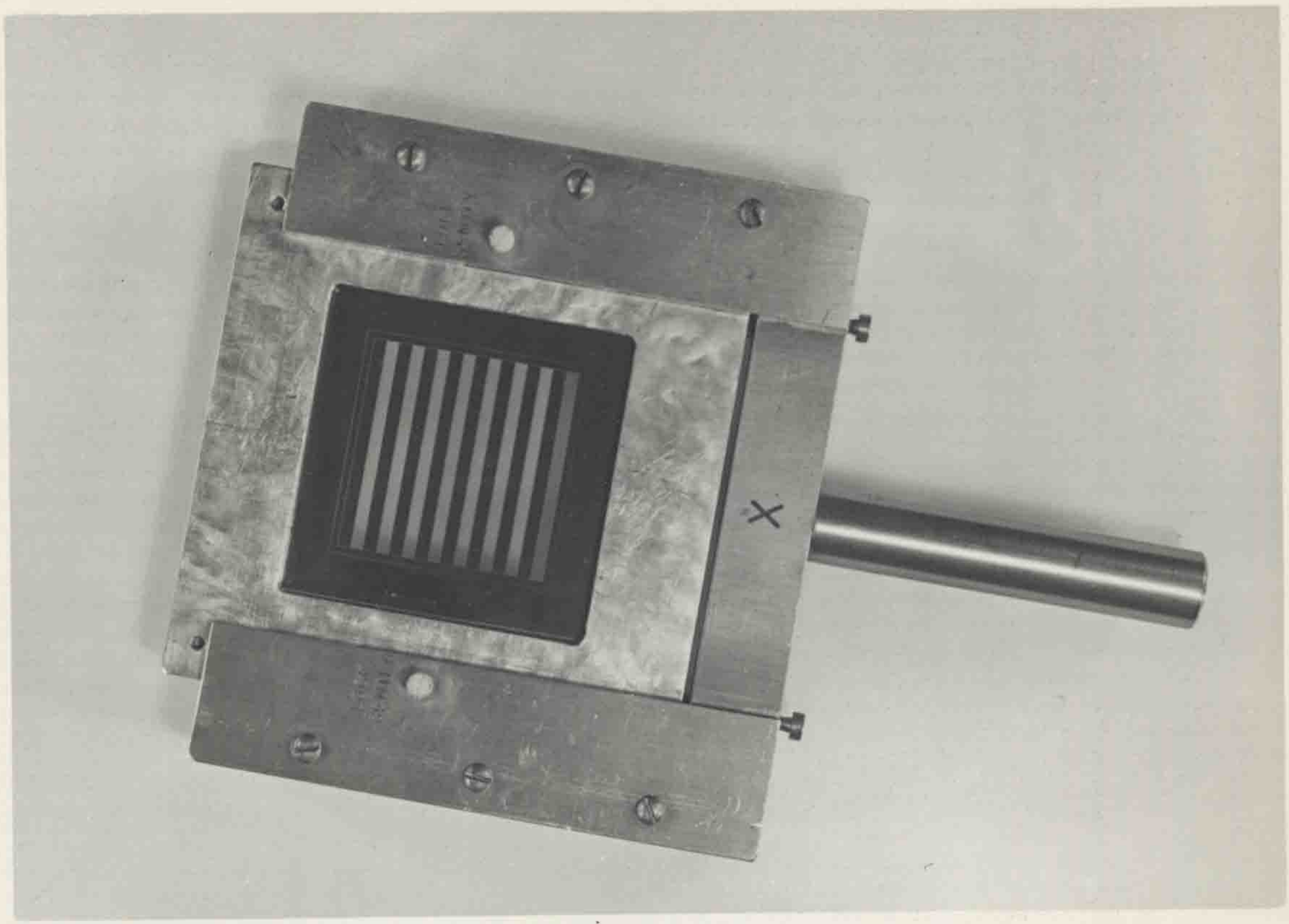


Plate 6.3 - A Photographic Test Pattern in the Holder

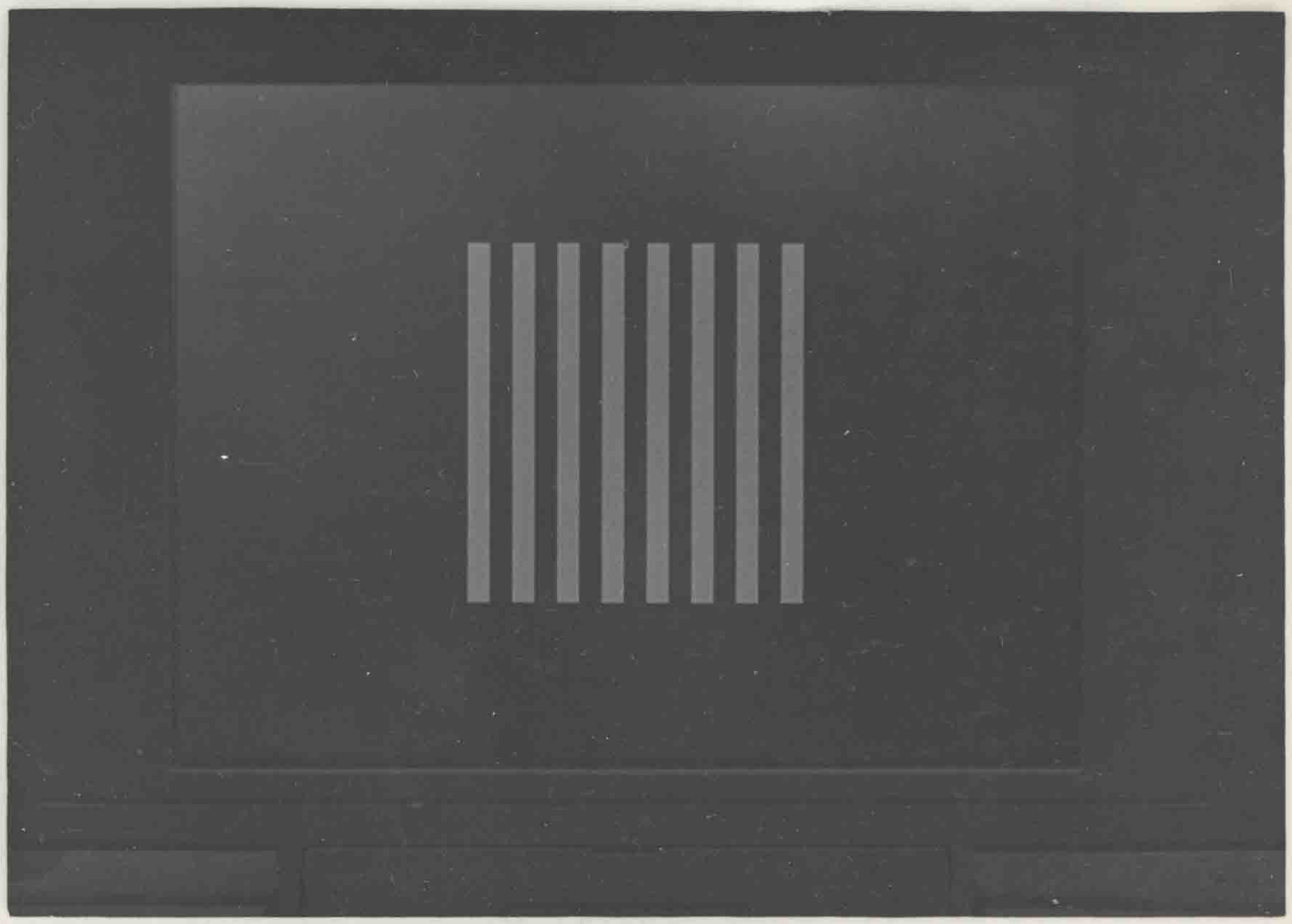


Plate 6.4 - A Test Pattern on the Plasma Display Panel

7. RESULTS AND DISCUSSION

This chapter presents the results for all the experiments carried out as described in the previous chapters. An explanation of the features of each calibration system is put forward with each result. The major aspects of the results have been discussed in the relevant chapters since it was expedient to justifying the development of the next step of the project. Attention will be drawn to the major features referred to previously, as well as smaller factors which should be considered should the work described in this thesis be further developed, and/or calibrated incoherent optical fibre bundles used for endoscopy.

The non-photographic images printed in this document have been produced by the Apple LaserWriter printer [*83*] from a postscript file [*84*]. This postscript file is produced from a translation of the intensities in the image into a grey level which is printed at each pixel position in the final output image. The laser printer produces a grey scale output by varying the density of dots used to create the image. The density of dots can vary from 0 dots per inch (for full white) to 300 dots per inch for full black [*83*]. In theory 300 shades of grey can be printed, but in reality approximately 32 noticeably different levels can be perceived by the eye. Therefore a small loss of detail was encountered when compared to the original video image. This loss was not deemed to have significantly affected the points that have been and will be, made. Algie [*98*], develops a theory for the relationship between tonal levels and the perceived quality of printed images. A photographic sample of a result is presented for comparison. Every result has been presented with the accompanying input image for a direct appraisal of the capabilities of each calibration method.

The images used to assess the accuracy and limits of each calibration system varied in complexity from a relatively simple low definition, high contrast scene (black text on a white background), to simple shape/object recognition (common tools used in the experiment) to complex and detailed scenes such as a face (relatively low contrast), and an outdoor view (bright with a large range of intensities). This range of images will reveal the strengths and weaknesses of image transmission through optical fibre bundles by non-modulated light as well as highlighting the salient characteristics of each calibration method.

7.1 Results for DT2853/Opus PC V Spot Calibration

The primary issue in determining the location of fibres in the output image is their distinction from the background. Section 4.3.3.2 discussed various thresholding methods for doing so, and the method chosen was presented. Image 7.1 (a) shows the original image output from the fibre bundle when a 10 µm pinhole was used for the light source. It clearly shows the illumination of several individual fibres, as well as the background noise that was present from the camera. A single output point had to be chosen from this image. The chosen threshold had to exclude he background noise, and preferably, the more dimly illuminated fibres.

Image 7.1 (b) shows the output after a threshold operation, with the threshold set to an intensity value of 75. The background was set to the black level, and the three brightest fibres' output can clearly be seen. The image processing algorithms must then chose the centre of the brightest area as the correct output point for the input position. This can be seen as the spot at upper left of the image. With the DT2853/Opus PC V system, a search was conducted over the image area to find the illuminated fibre outputs and then to decide which was the brightest (section 4.3.3.3). This method would have to differentiate between the individual areas before attempting to find the brightest spot.

A similar image is obtained from the OFCS/transputer system. The choice of the correct output point is determined as described in section 5.6.1. The matrix search area would be centred on every pixel that was illuminated in the thresholded image, a total of 18 pixels in Image 7.1 (b), to find the brightest area, and subsequently choose the centre of the brightest matrix area.

The background noise that is visible in Image 7.1 (a) must be noted as it will play an important part in the test pattern calibration methods. If the light levels in the test pattern images are low, then this noise will represent a significant part of the image to be analysed, and produces a distortion of the intensities in the image, and could alter the decision as to whether the pixel intensity should be above or below the threshold value. As can be seen from Image 7.1 (b), the chosen threshold provides the required result.

It will be seen in the following results that it is possible to calibrate a low resolution optical fibre bundle for the transmission of low complexity images.

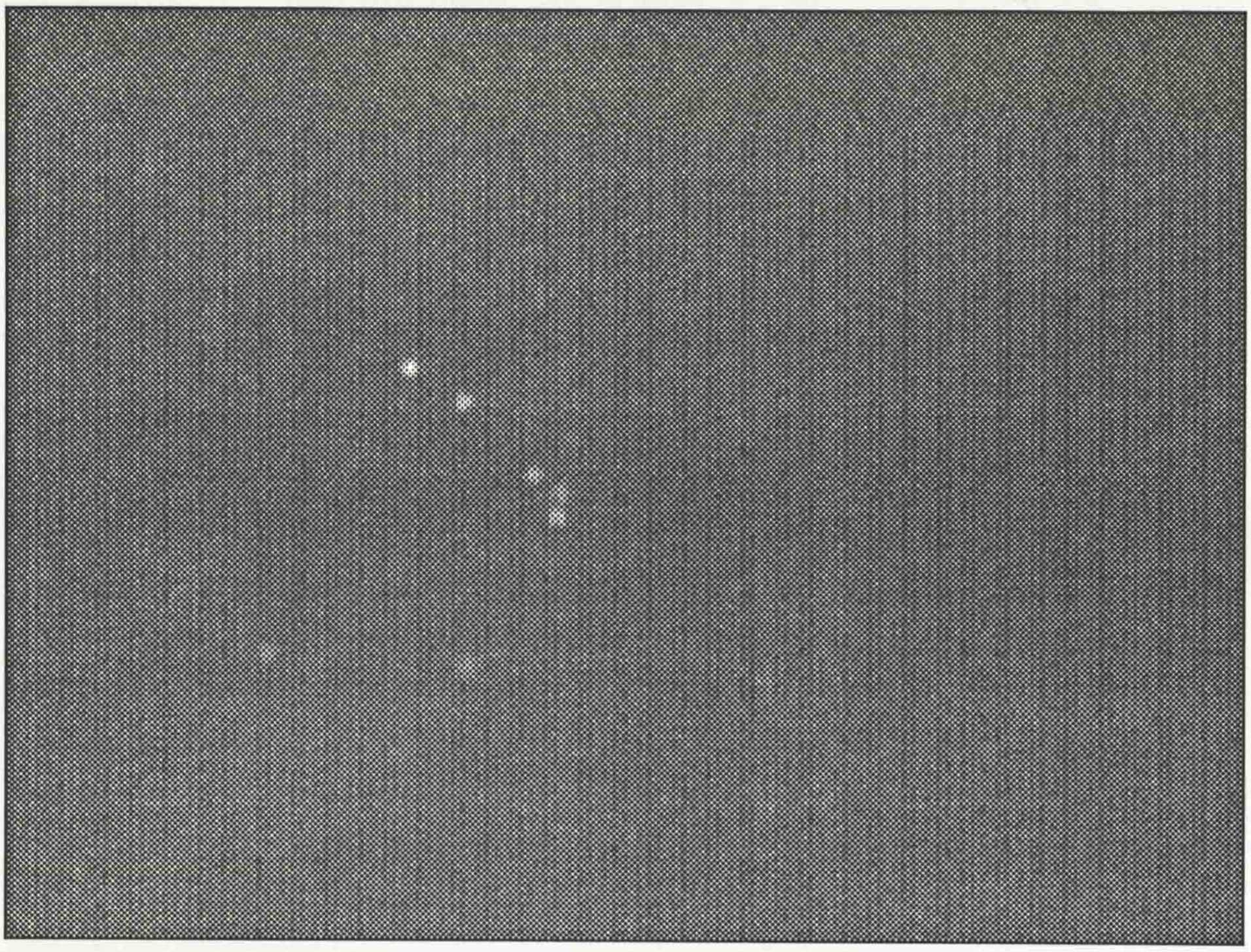


Image 7.1 (a) - The illumination of multiple fibres by a 10 mm pinhole

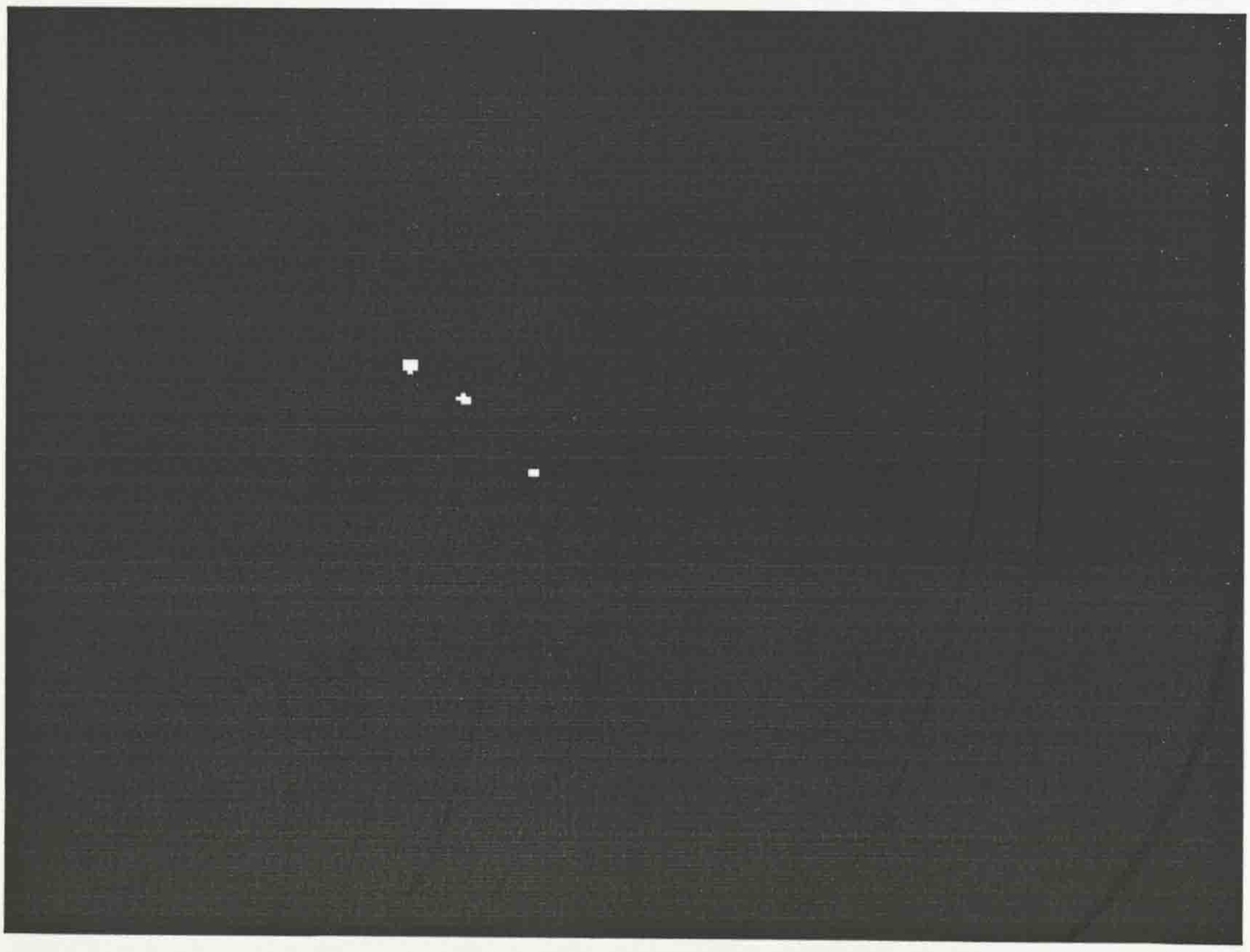


Image 7.1 (b) - The above image after applying a threshold

After the fibre was calibrated, it was used to transmit the image of some text. The output from the fibre bundle can be seen in Image 7.2 (a). It shows a section of the 2200 single fibres transmitting the intensities that make up the image of the text. The output is totally incoherent, but shows the approximate triangular packing arrangement of the fibres in the bundle, as well as gaps in areas where the fibres are not as regularly arranged.

This image is the source for the reconstructed image shown in Image 7.2 (b). This result was obtained using the 2000 points obtained directly from the calibration. The original incoherent image was then reconstructed from the LUT generated by the mechanism described by section 4.3.6.2. The text being transmitted can be seen in the reconstructed image in Image 7.2 (b) from the original output as shown in Image 7.2 (a). The reconstructed image clearly shows the characters below:

ST CARD
R GREG

The original characters were black on a white background. The white background appears grey in the reconstructed image, partly due to the absorption of light by the bundle, the small lens aperture used for better focusing of the input image, and the compression of intensities by the laser printer. The black text contains some grey areas, which were caused by errors in the calibration data. This feature is also a result of the difficulty in focusing the input image onto the face of the optical fibre bundle, as an assessment of the accuracy of this focus could not be carried out until the image was reconstructed. This meant that the edges of the text were not sharply defined, and the blurring appears as the background intensity spreading over the text. The image shows that the calibration and reconstruction procedure was basically correct, and could certainly be used for the transmission of simple images of small intensity range.

Image 7.3 (a) shows the incoherent bundle transmitting another image of text. The reconstruction data is the same as for Images 7.2 (a) and (b). The text in the reconstructed image in Image 7.3 (b) is clearer than that in Image 7.2 (b). This was achieved by improving the focus at the input end of the bundle. The characters transmitted were

A
Test
Card

The text in Image 7.3 (b) is very distinct, and shows the importance of accurate focusing arrangements. The edges of the text are not straight. This arises from the low resolution of the bundle and the spread function applied to the LUT. The low resolution affects the intensities at the black/white text/background edge because the output from the fibres that fall along these edges will be neither black nor white, but have an intensity value that is between the two depending on the average amount of light falling on the fibre face (section 3.2.2) and the acceptance cone of the fibres. The fibres in the bundle are not arranged to a regular grid as can be seen in Images 7.2 (a) and 7.3 (a). This also distorts the edges to produce the jagged borders seen in Images 7.2 (b) and 7.3 (b).

The calibration of a 3.5 mm (4400 fibres) bundle was also carried out. For this experiment, the magnification factor at the output end of the bundle of fibres, into the camera was reduced to allow all the fibres to be viewed. This meant that the size of the individual fibres was reduced as can be seen in Image 7.4 (a). The image of the end of the fibre is not central in the viewing area because of a small inaccuracy in the machining of the components to attach the bundle of fibres to the lens, and the lens to the camera. The definition of the image transmitted by such a bundle will be greater than the 2.5 mm bundle, because of the greater number of fibres used for transmitting the image.

After calibration, the bundle was used to transmit the same image as for the test shown in Images 7.3 (a) and (b). The input size of the image was slightly reduced to test the theoretically improved resolution of the bundle. The smaller input image would require better definition of the edges and intensities in the image if it was to remain recognisable. The reconstructed image can be seen in Image 7.4 (b). The text is still clearly defined. The quality of a reconstructed image is affected by the following factors:

The accuracy of the calibration - errors in the LUT will cause a displacement of the intensities in the image, corrupting the output.

The resolution of the bundle - the complexity of images that can be obtained from the bundle is determined by the number of fibres in the bundle and the density. The more fibres that are used to define an image, the finer the detail that will be perceived.

The focusing of the image at the input face of the bundle - unfocused images emphasise the low resolution of the bundle by increasing the blurring of edges, making the output more difficult to decipher.

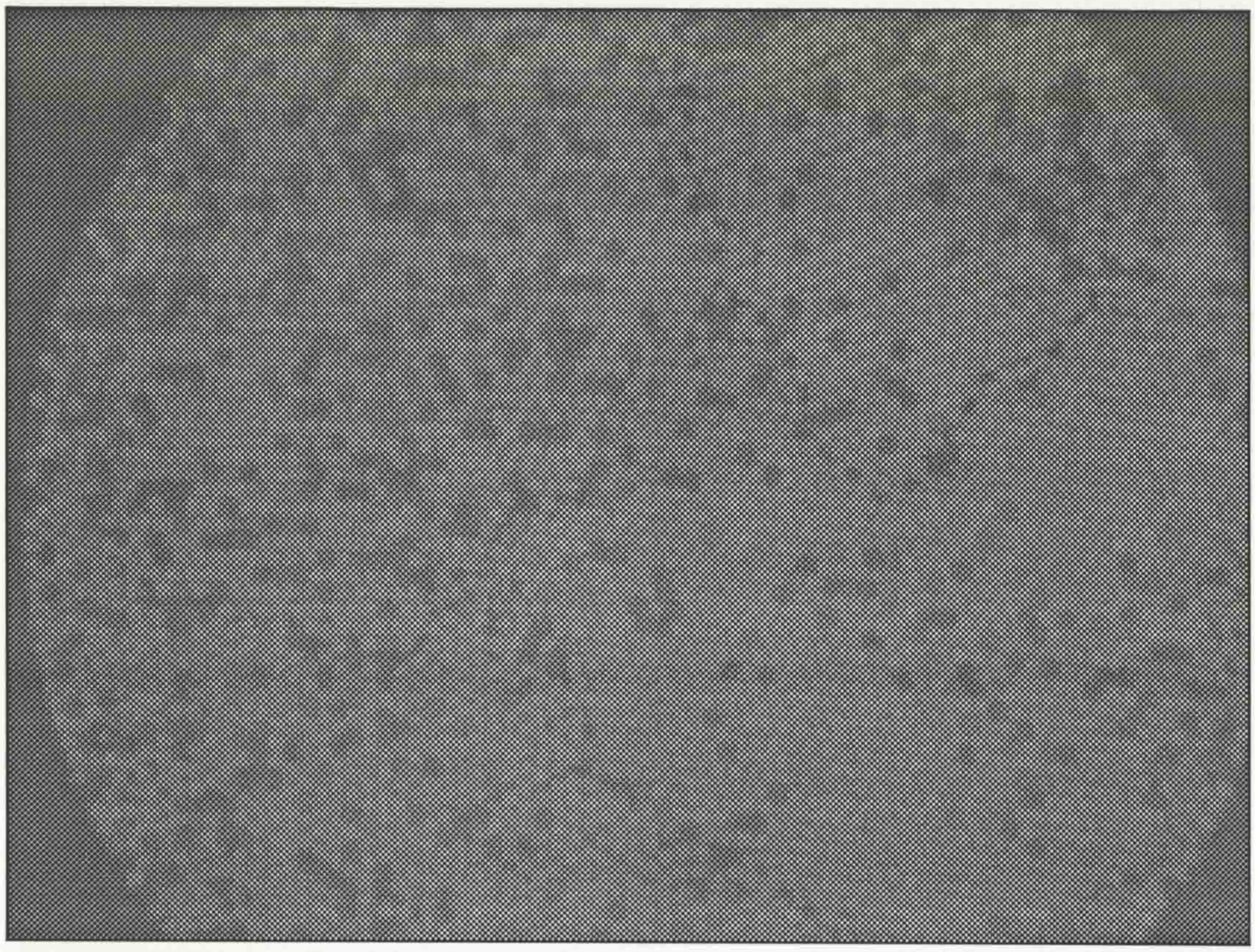


Image 7.2 (a) - The input to the reconstruction process for an LUT generated by the smoothing process described in section 4.3.6.2

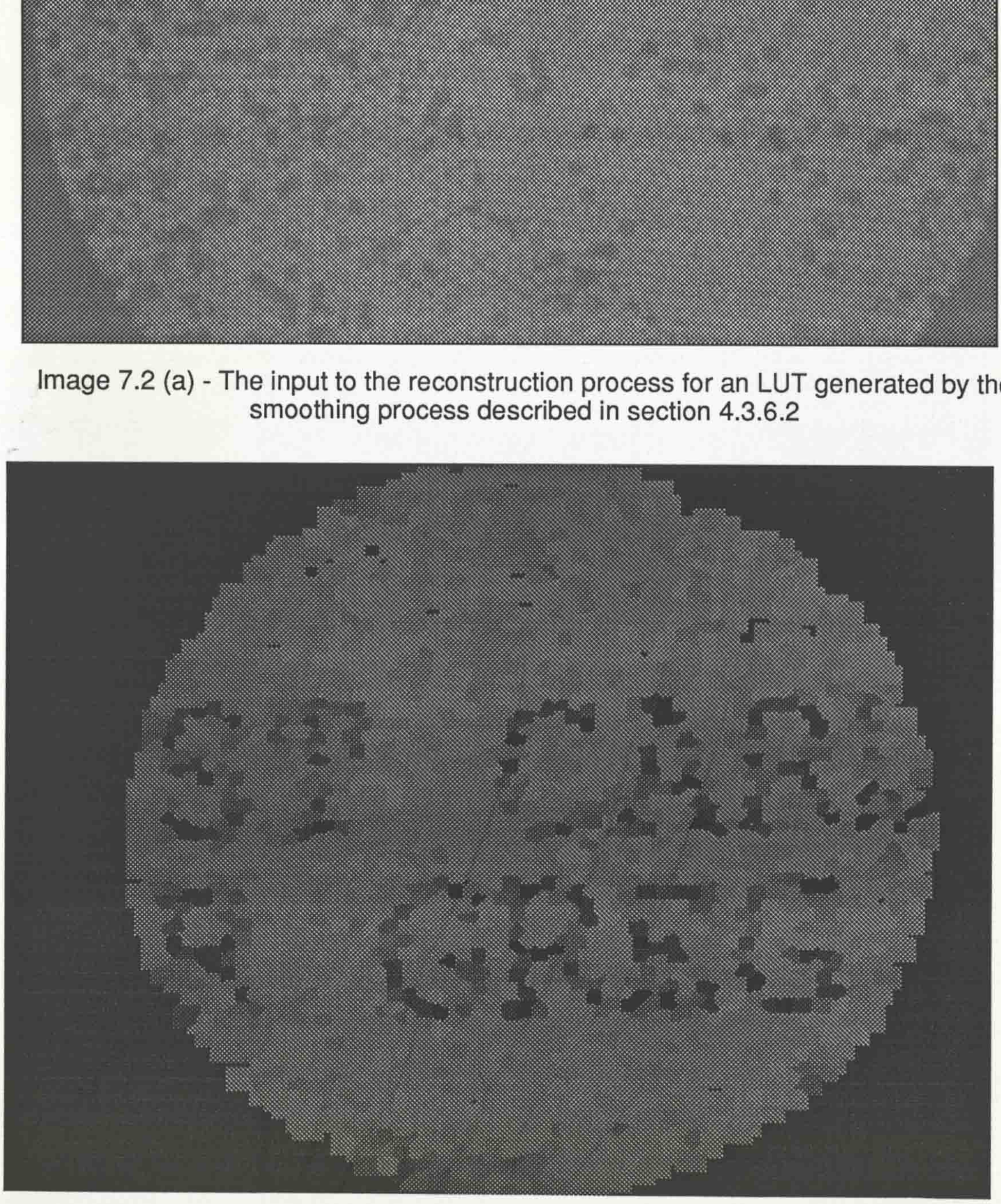


Image 7.2 (b) - The reconstructed output using the intensity spreading algorithm shown in Figure 4.13

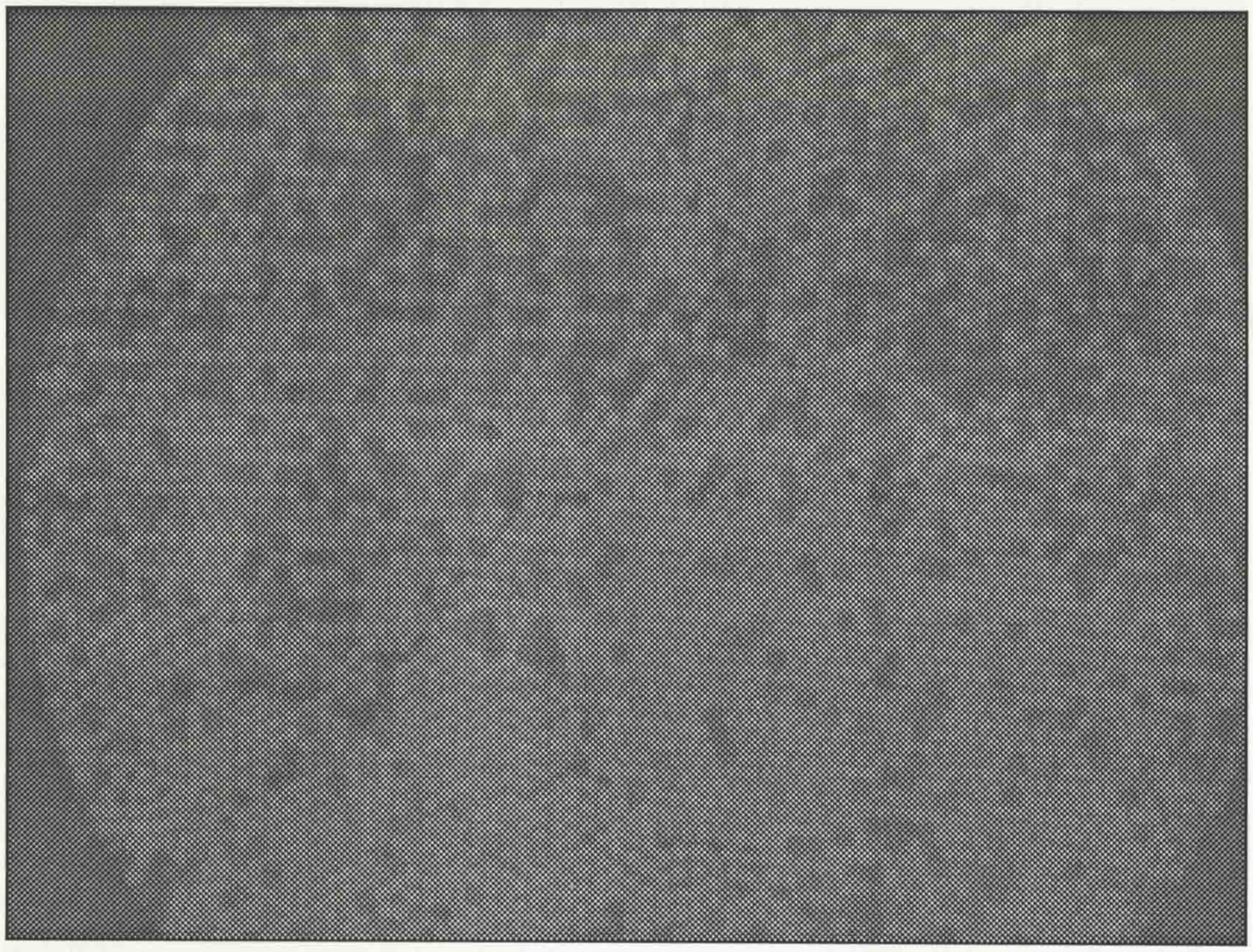


Image 7.3 (a) - The input image of large text

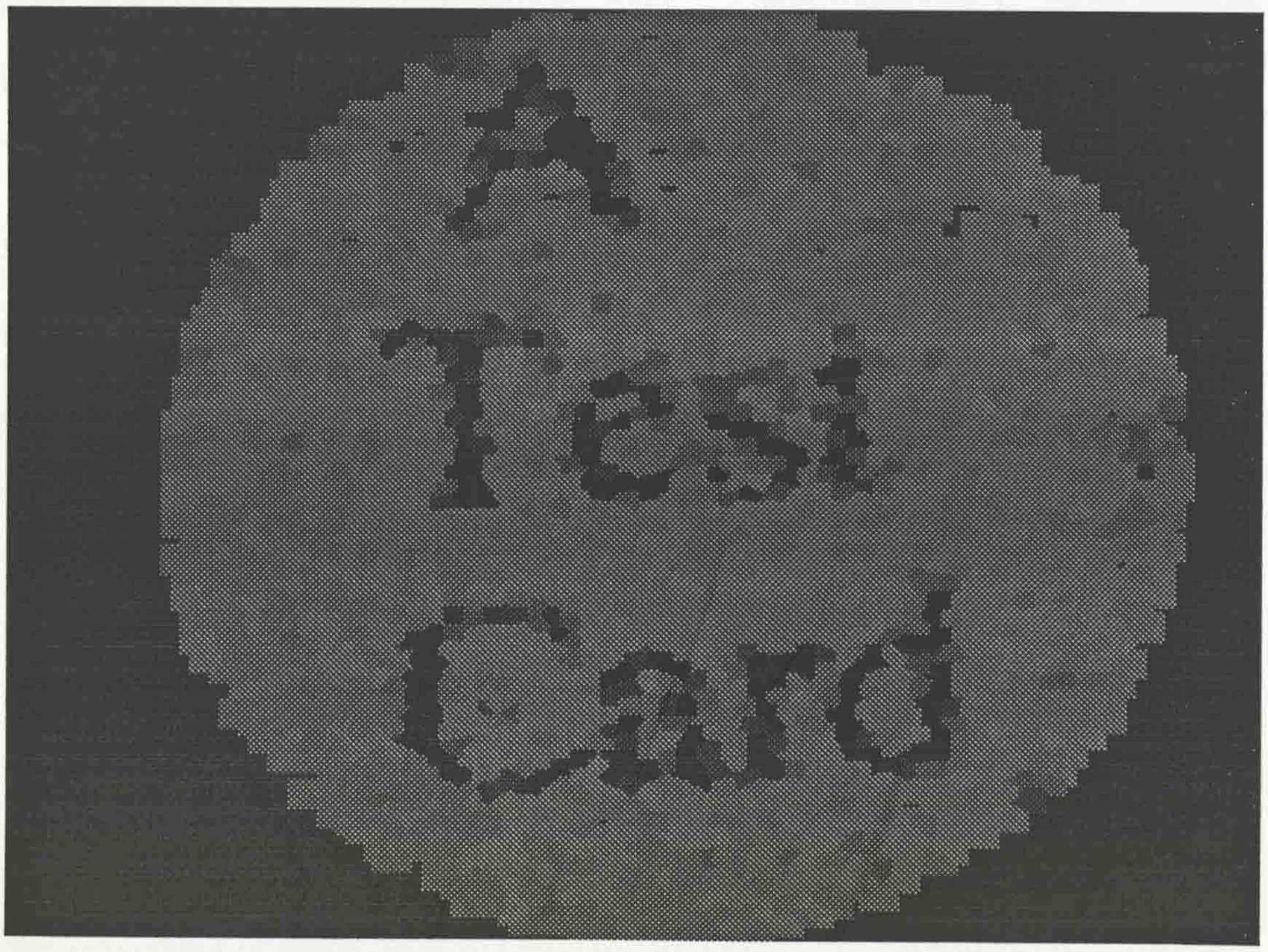


Image 7.3 (b) - The reconstructed image showing the clearly defined text

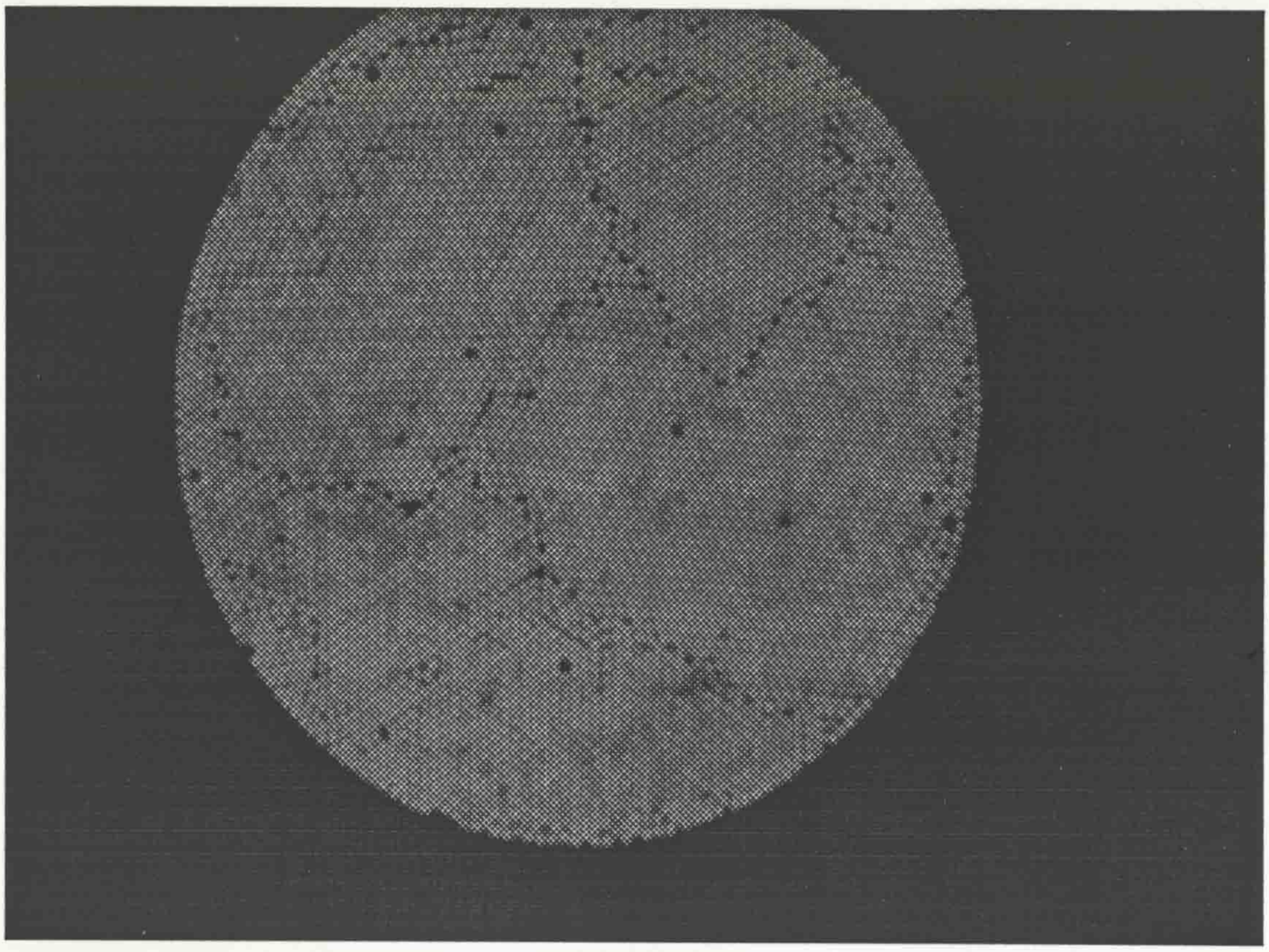


Image 7.4 (a) - The input image of text at a reduced magnification of the bundle's output

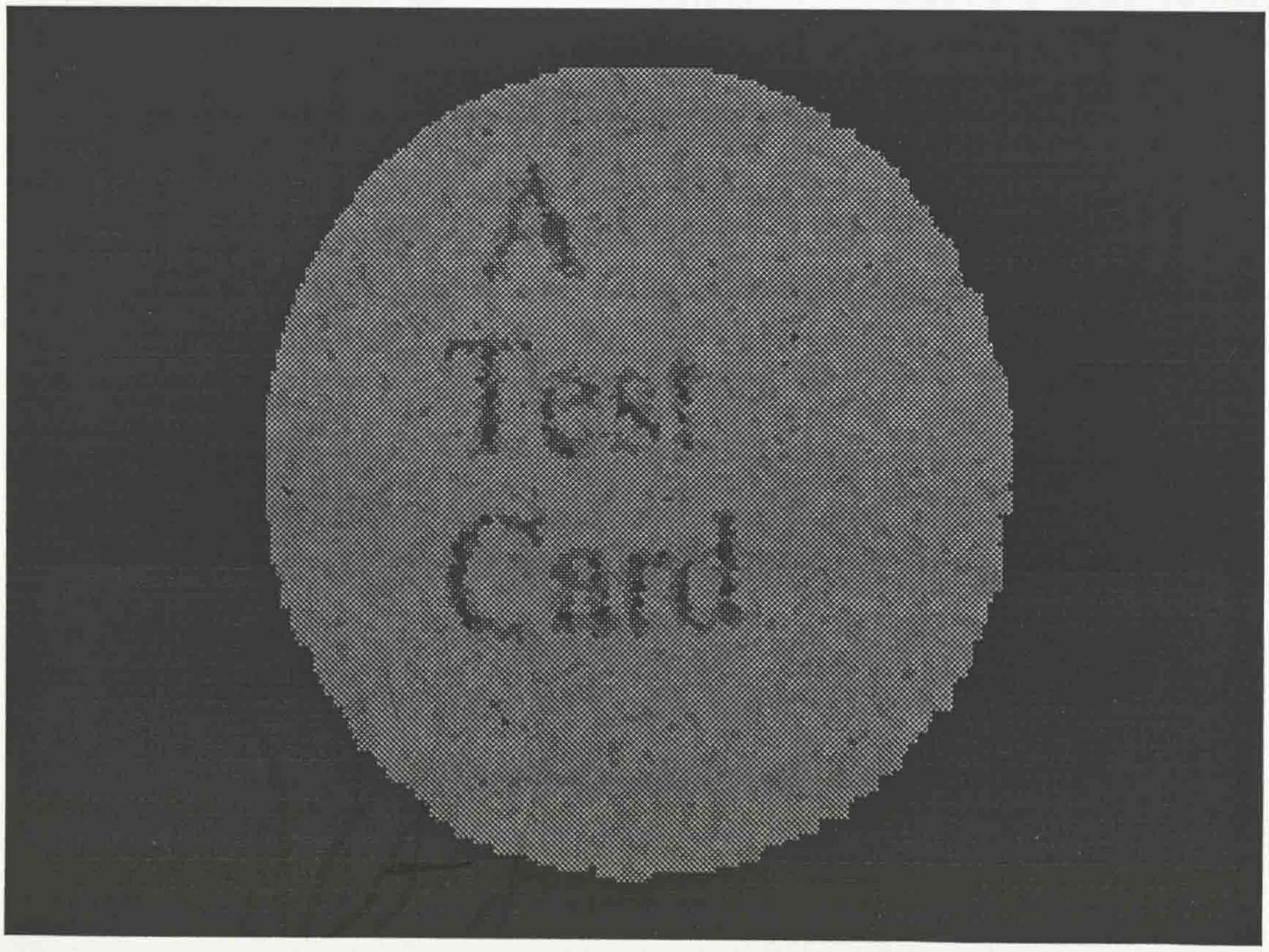


Image 7.4 (b) - The reconstructed image showing improved resolution of the text

7.2 Results for OFCS/Transputer Spot Calibration

The following results show the quality of images that were obtained from this trial. The presentation shows the input image and below that the reconstructed output. The output is obtained in real-time from the OFCS. The spreading function applied to the LUT for the DT2853/Opus PC V was implemented as described by section 5.2.2.1. The duplication of outputs for different input points was achieved by reducing the step size between input sample points from 40 µm to 20 µm. For a bundle of fibres 3.5 mm in diameter, this would produce a range of input points from 0 to 175 units in both the x and y directions. The camera and OFCS system used standard video rates and sampling frequencies to produce an image of aspect ratio 4 : 3. If the original LUT, with a square aspect ratio, was used, the resulting reconstructed image would be horizontally narrowed. To compensate for this effect, the x coordinate values could be scaled by a factor of 1.33, to restore the original aspect ratio, or the y coordinates reduced by 25%. If the x coordinates were scaled upwards, using integer values to perform this operation would result in gaps appearing in the resulting x coordinate values. Consider the example below:

	Original x coordinate	Scaling factor	Resulting x coordinate
X_1	100	1.33	133
X_2	101	1.33	134
X_3	102	1.33	136

The three consecutive coordinates X_1, X_2, X_3 are scaled to give the non-consecutive values 133, 134, and 136 respectively. Thus pixel intensities will not be written to output locations whose x coordinates are 135, and will appear as a gap in the reconstructed image. For this reason, it was decided to reduce the values of the y coordinates by 25%. This would, conversely result in some y coordinate values being duplicated by the same reasoning as above. This was deemed to be more acceptable than scaling the x coordinates.

Thus the x values would occupy a range of 0 to 175 pixels and the y values a range of 0 to 131 pixels. This would place the reconstructed image in the top left of the output display area. For cosmetic reasons, it was decided that a centrally located output would be preferable. To achieve this, the centre of the reconstructed image, at location (65,85) would have to be translated to the display centre location (128,128). The surrounding coordinates were also subject to an identical translation to maintain the spatial relationship between the pixels in the reconstructed image. The results of these operations can be seen in Images 7.5 (b), 7.6 (b), 7.7 (b), and 7.8 (b).

Images 7.5 to 7.8 show the incoherent bundle transmitting images of increasing complexity, in order to test the capabilities of the bundle and calibration method. The upper images again show the original incoherent input, and the lower show the reconstructed output from a LUT generated as described by section 5.6.

Images 7.5 (a) and (b) show the outputs when transmitting a relatively simple image of some text. This image is nominally made up of two intensities, the black text and white background. Due to non-uniformity of the transmission of light by the bundle and the small lens apertures used for improved focusing, the image appears as dark grey text on a light grey background. Image 7.5 (b) shows the accurate reconstruction of the input image to give a good definition of the transmitted image. The black/white borders of the image are again distorted by the relatively low resolution of the optical fibre bundle. The original image was a larger version of the text below:

A
SAMPLE
OF
TEXT

The gaps within the image represent areas where single fibres were not found during calibration. The focusing of the image onto the input face of the fibre was made considerably easier by the real-time reconstruction of the incoherent image, allowing changes to be easily assessed.

Images 7.6 (a) and (b) show the outputs when the bundle was transmitting a more complex image of a digital mouse next to a pair of pliers. The outline of each item is clearly visible with the shadows created by the lighting. The low resolution of the bundle is shown by the inability to resolve the text that was present on the bottom right corner of the mouse. The reconstructed image does show that the bundle is capable of resolving large objects in a scene, which certainly could not be directly inferred from the incoherent image in Image 7.6 (a).

Images 7.7 (a) and (b) show the outputs for an detailed image of wide intensity variation. The reconstructed image shows a face where the general features can be perceived, but the finer characteristics are unrecognisable. The original image shows the intensity variation in the object being observed.

Images 7.8 (a) and (b) show the fibre bundle transmitting a street scene. The image is made up of the front half of a car, parked next to a kerb, on a single line, with a

post box on the pavement. The light coloured top of the post box, darker middle, and black base are clearly visible. The detail in the dark coloured car is not clear, and can just be recognised as such.

The images showed that the calibration and reconstruction theory was correct, and that the OFCS function correctly. The quality of the images was limited by the quality of the optical fibre bundle and its low resolution. For simple images such as large text, the bundle functioned adequately, but for more complex images the perceived result degraded with increasing complexity of image.

Images 7.9 (a) and (b) show the inputs and outputs to the OFCS using a more modified LUT. The coordinates contained in the LUT were first scaled, as described in section 7.2, and then the resulting gaps were filled with values from adjacent non-zero locations to give a more complete image. Image 7.9 (b) shows the output from this LUT and shows that the output now occupies most of the display area, but the text output was further blurred by the spreading LUT function. The result is obviously degraded when compared to Image 7.5 (b). This method of improving the image was therefore not pursued further.

Plate 7.1 shows the real-time reconstruction of the output of an incoherent bundle when transmitting a simple image of some text.

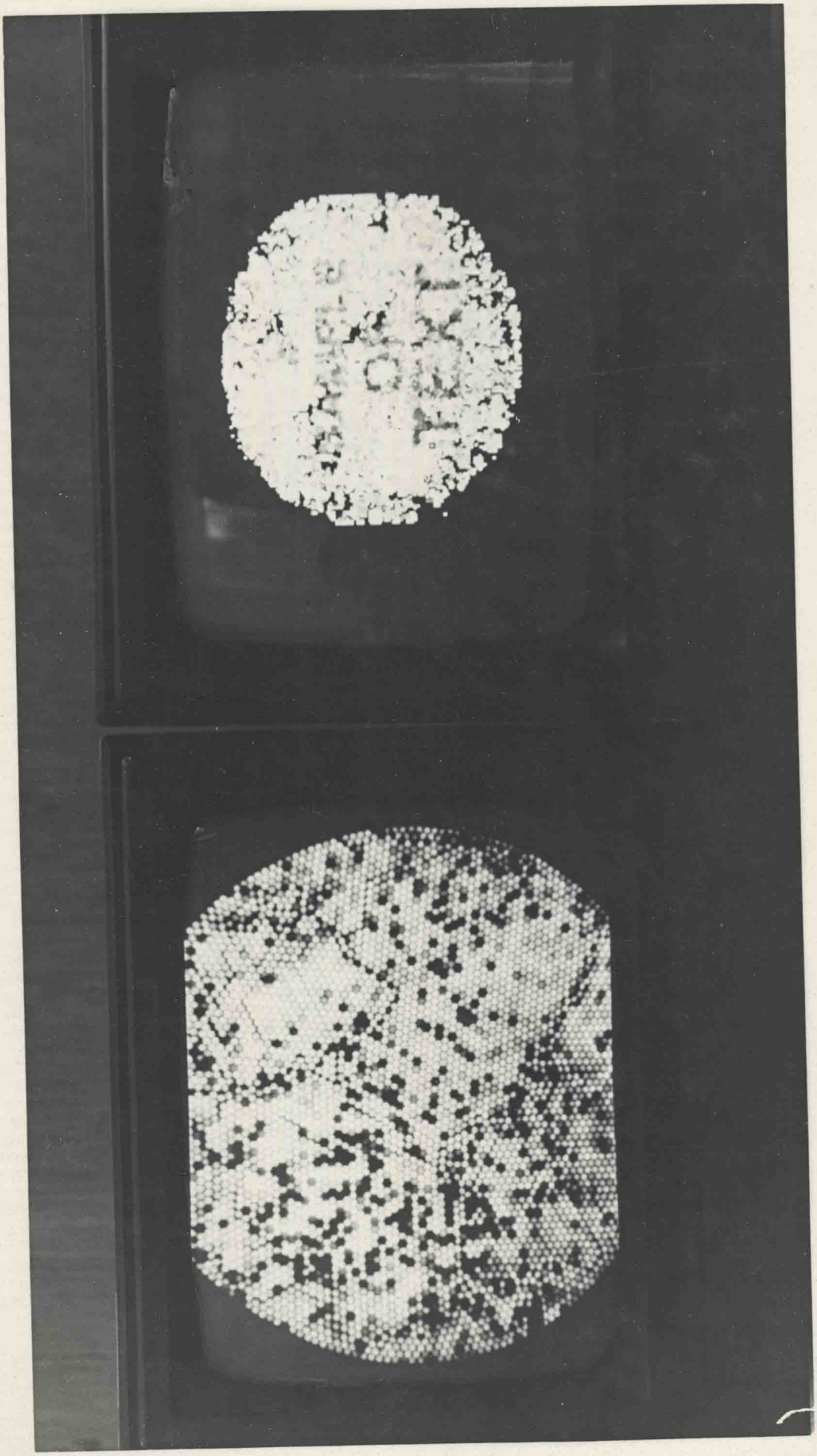


Plate 7.1 - Real-time reconstruction of the output of an incoherent bundle by the C

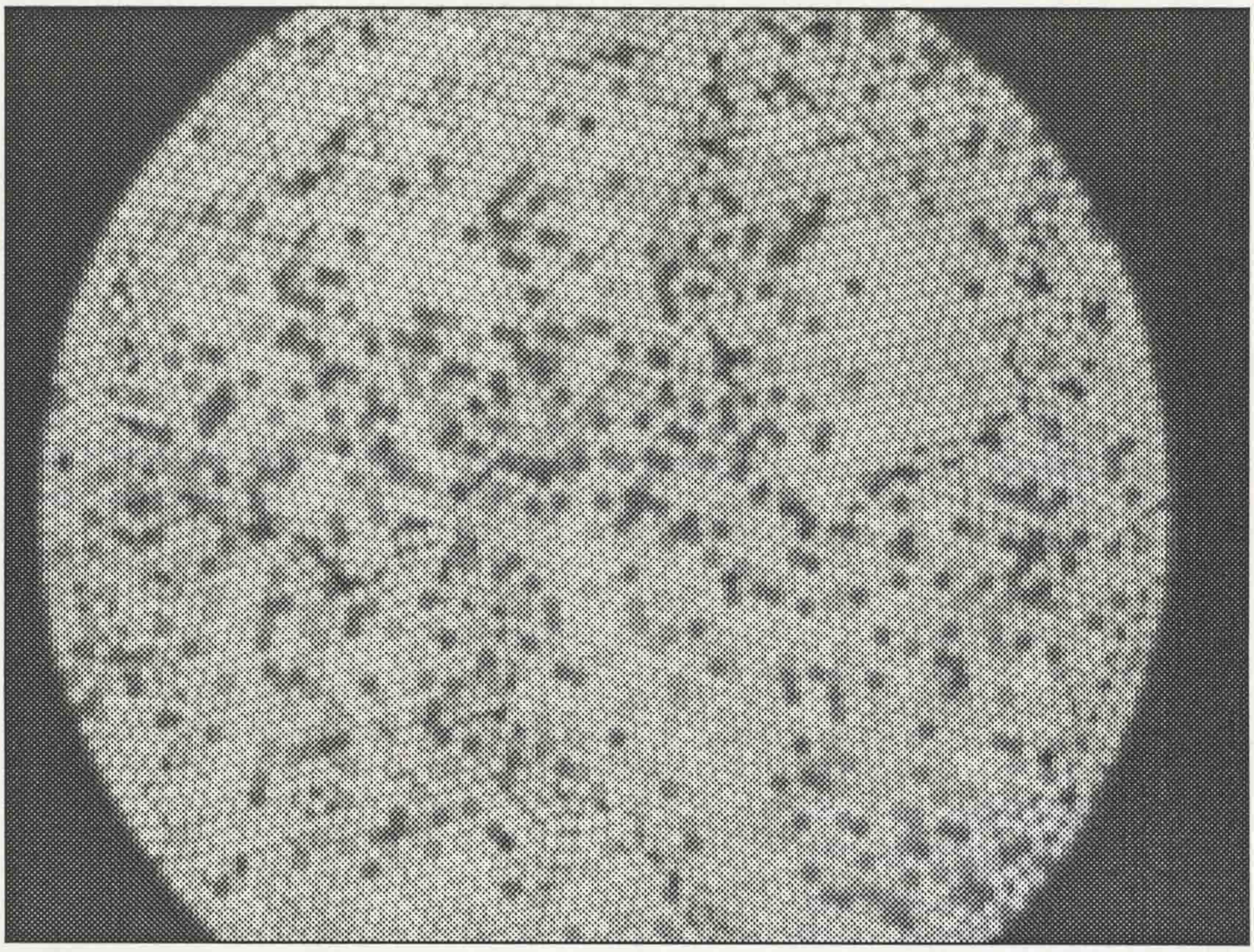


Image 7.5 (a) - The output from an incoherent bundle showing the random nature of the output

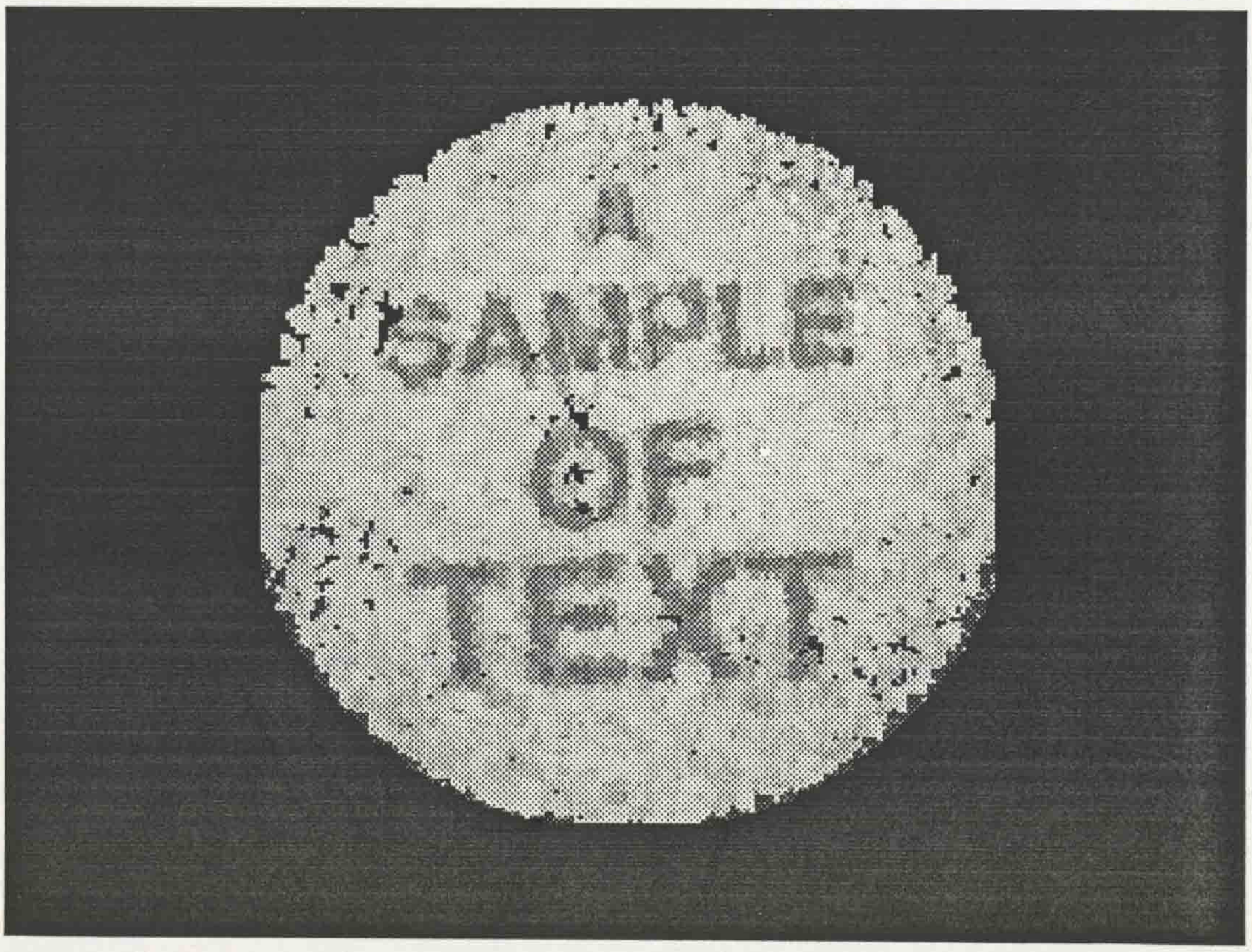


Image 7.5 (b) - The reconstructed image showing the transmitted image above

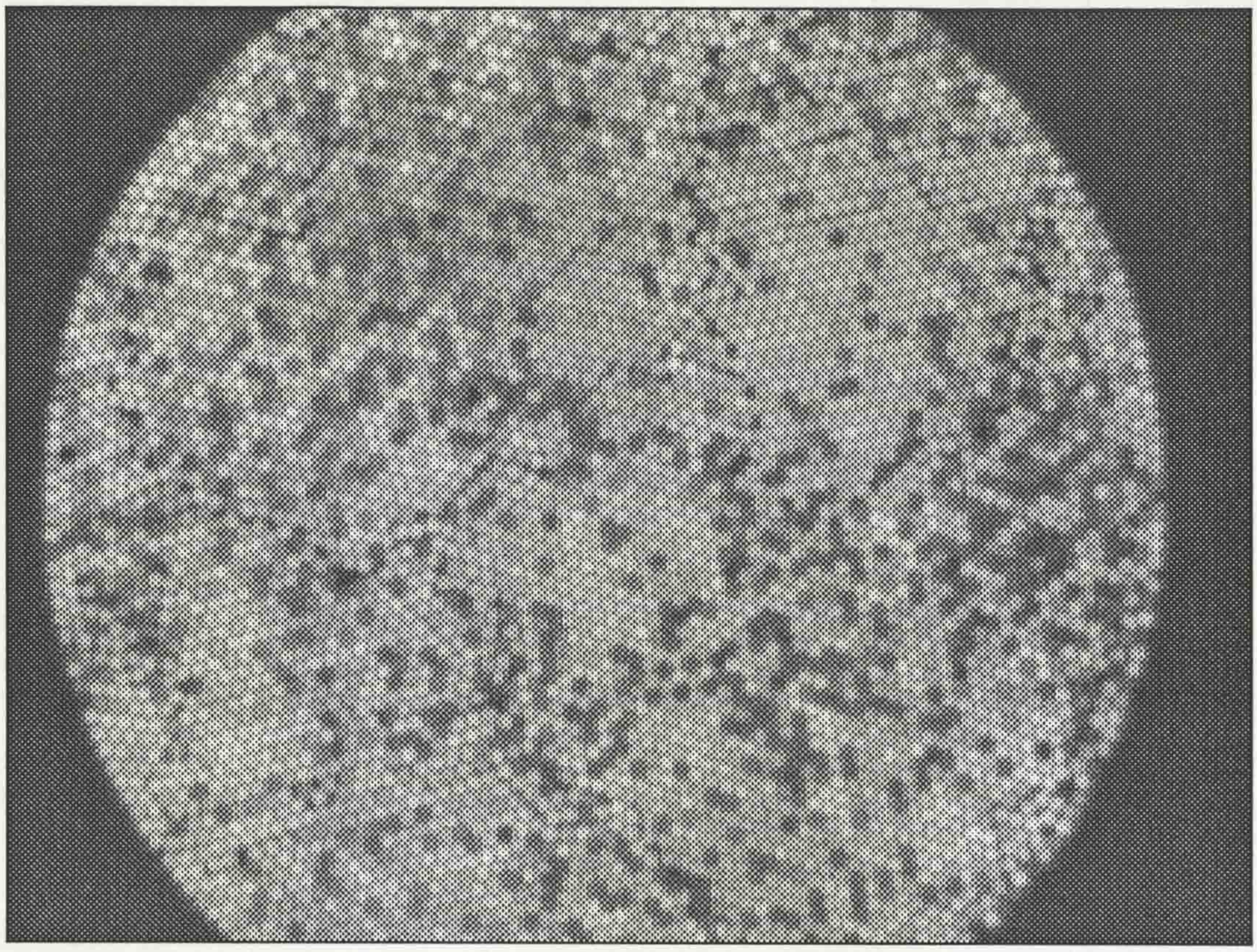


Image 7.6 (a) - The incoherent image of wider intensity range

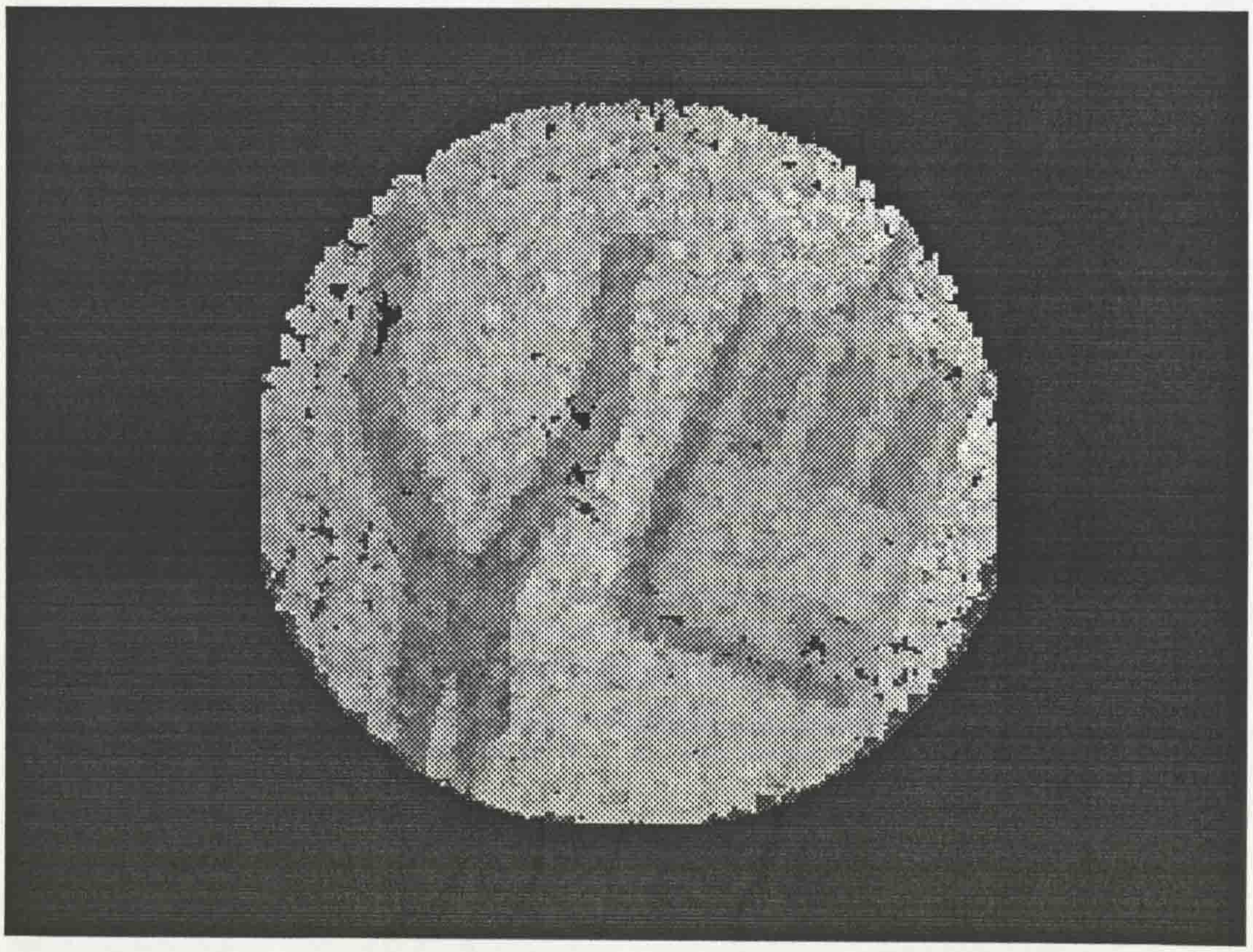


Image 7.6 (b) - The reconstruction of the image of greater complexity

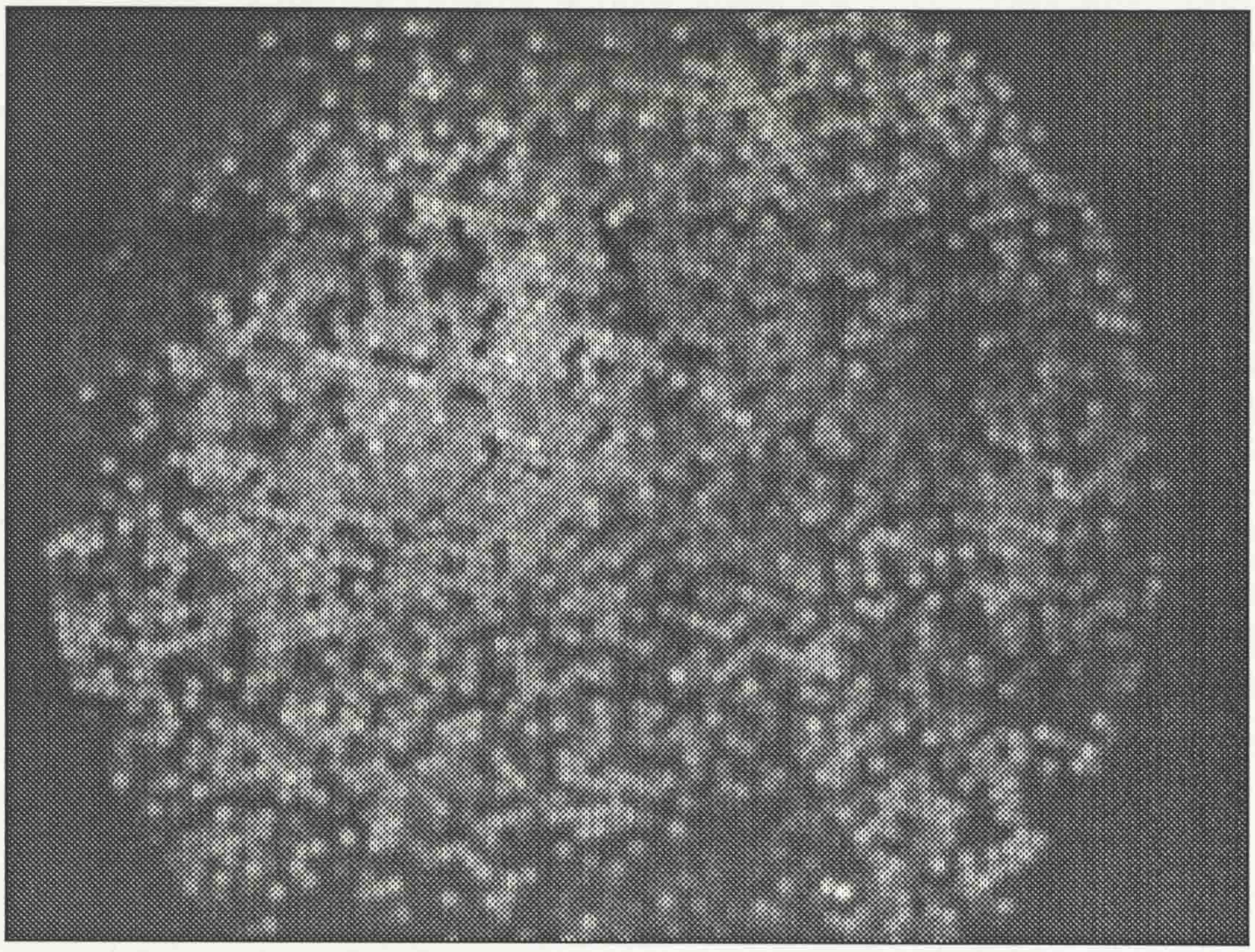


Image 7.7 (a) - The incoherent image showing large intensity variations

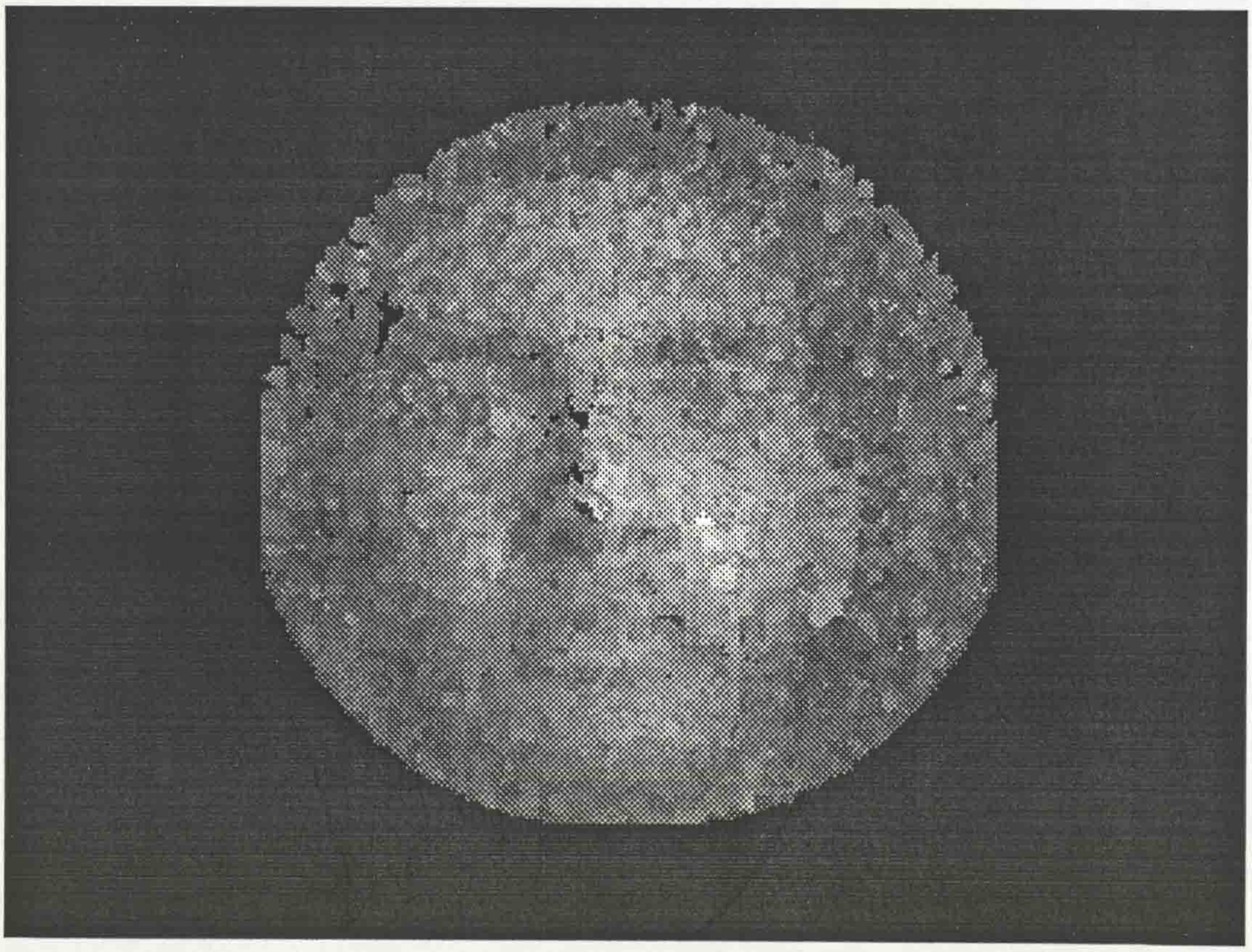


Image 7.7 (b) - The reconstructed image showing the resolution capability of the bundle

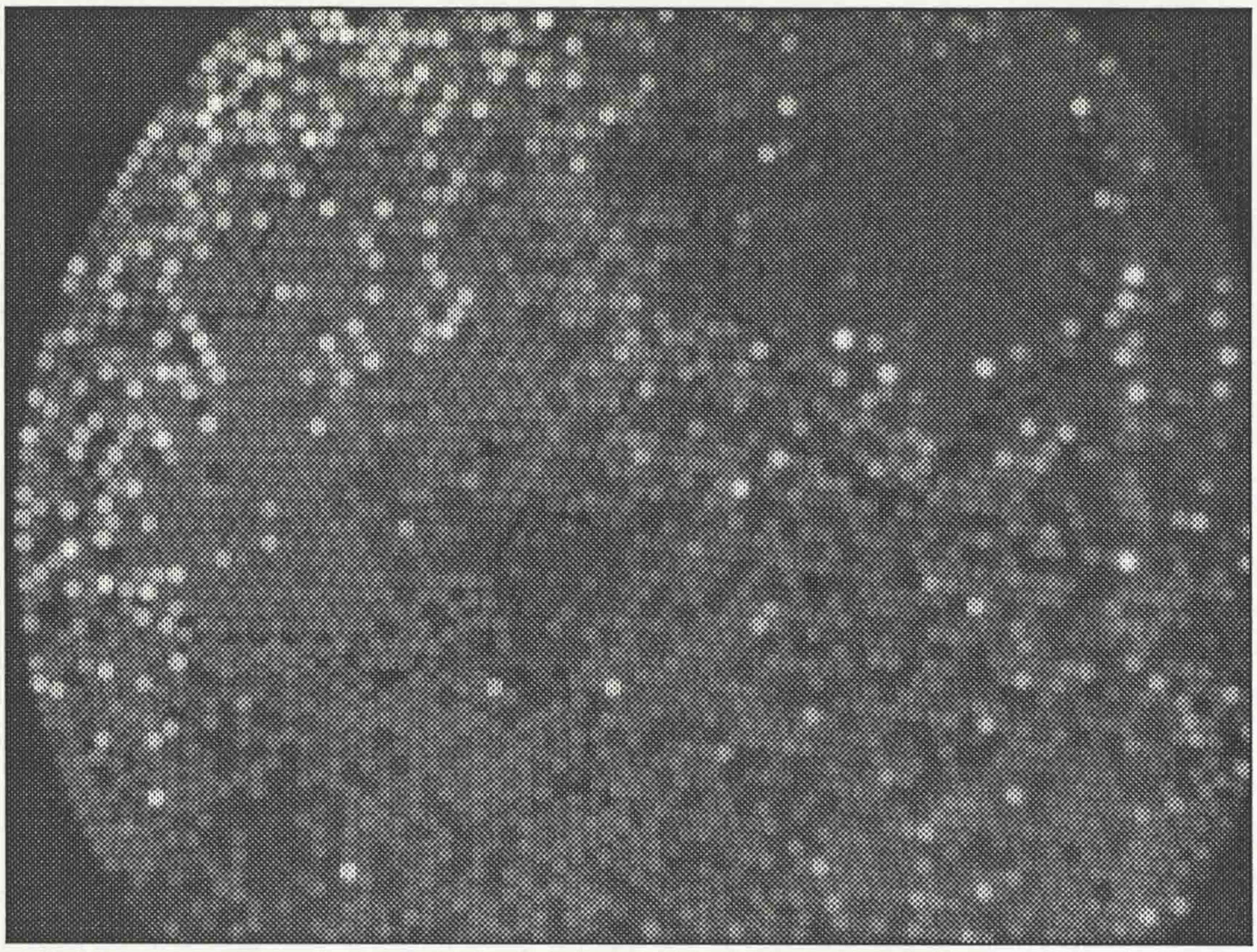


Image 7.8 (a) - An incoherent image

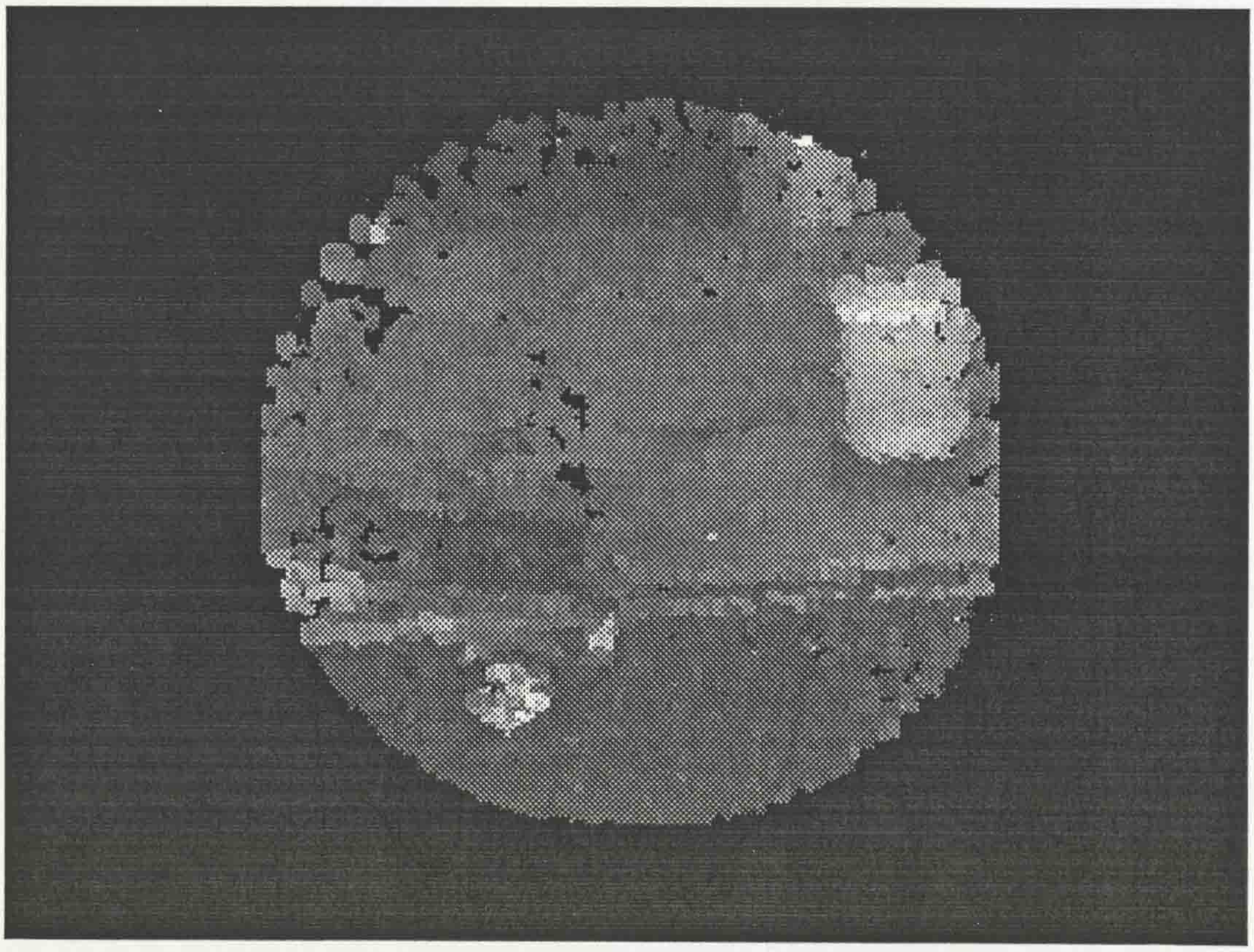


Image 7.8 - The reconstructed output showing a street scene.

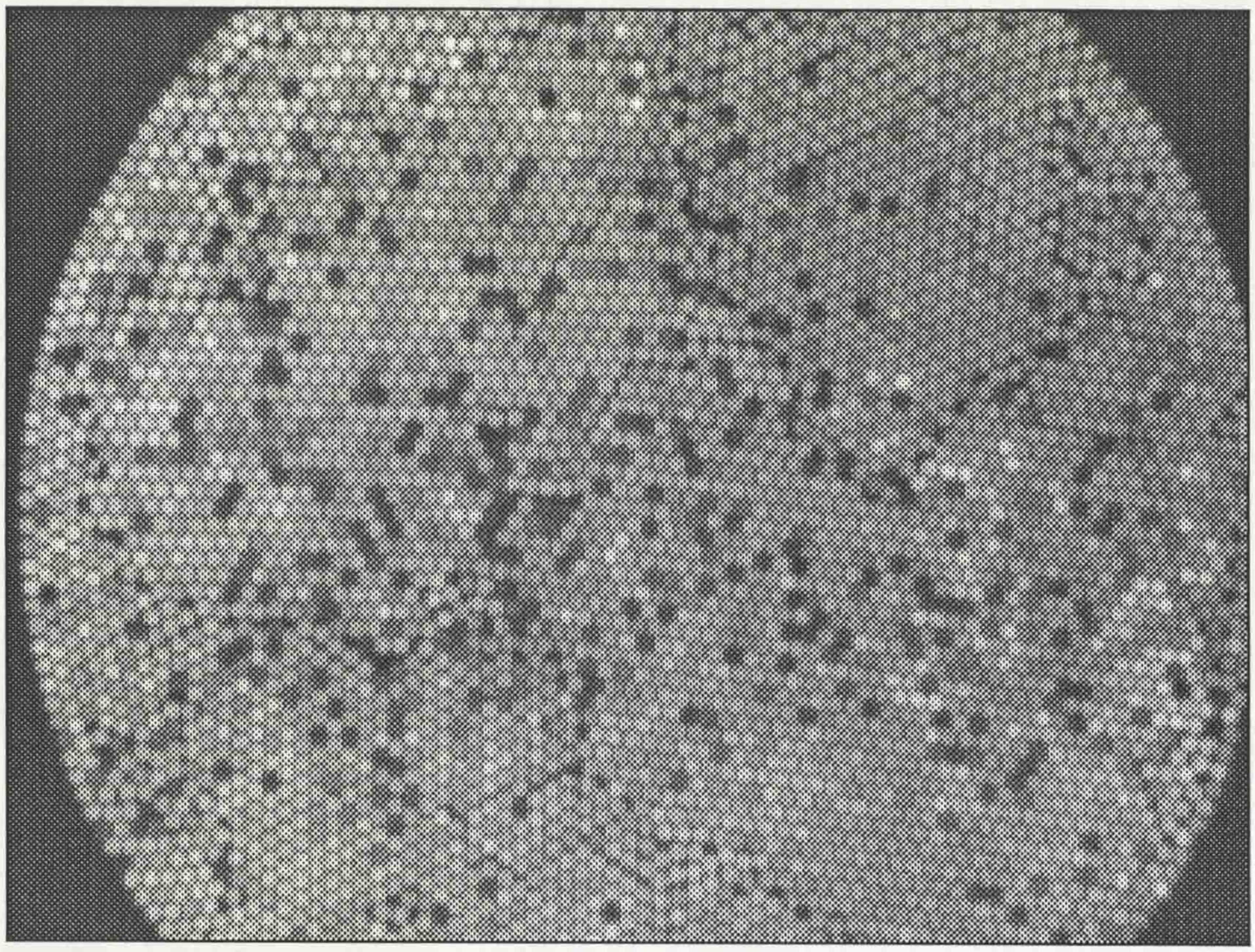


Image 7.9 (a) - An incoherent image

Image 7.9 (b) - The reconstructed output by a LUT scaled and smoothed

7.3 Results for OFCS/Transputer Test Pattern Calibration

The results from the test pattern calibration trials are dependent on three main factors: the accurate choice of threshold for each test screen, the accuracy of location, definition and lighting of the test patterns, and the fibre bundle's resolution capability.

The results for this section are divided into five sections:

1. The results from the threshold operation,
2. The calibration results for the test screens displayed on a cathode ray tube monitor,
3. The results for the test slide patterns calibration,
4. The results for the plasma panel display calibration, and
5. The results for the calibration of an incoherent bundle by test slides.

Images 7.10 to 7.12 show the output from the coherent bundle when transmitting three different test screens. The threshold method used for each input image was described by section 6.4.3, where an individual threshold for each pixel in the image was calculated. This was achieved by finding the maximum brightness of the pixel, that is when it was transmitting the light part of test screen, and the minimum intensity when it was transmitting the dark part of a test screen. The threshold was then chosen to be the midpoint between these two intensities.

Images 7.10 (b) and 7.11 (b) show very accurate thresholding of the input images. The edges of the images are well defined, following the contours of the original image. This indicated that the choice of threshold could not be improved. This is due to the fact that the threshold method is independent of non-uniform illumination of the test patterns, as long as the variation did not change between successive test patterns, and is also independent of non-uniform light acceptance/transmittance of the bundle of fibres, as a threshold is chosen for each fibre to take into account the above characteristics. The particular test pattern shown in the images was obtained from the test slides.

In Image 7.12, the thresholding method shows some signs of error. The test screen being transmitted is for bit 1 (the least but one significant bit). The problem in chosing an accurate threshold for such an image is complicated by the approach of the resolution capability of the fibre bundle by the highly detailed image. Small focusing errors distort the edges, which increase in number as the significance of the input bit being tested for decreases (Figures 6.2 (a) to (h)). This will result in errors

in the lower order bits, which cause a relatively small displacement of intensities in the reconstructed image, and will appear as a blurring of the reconstructed image.

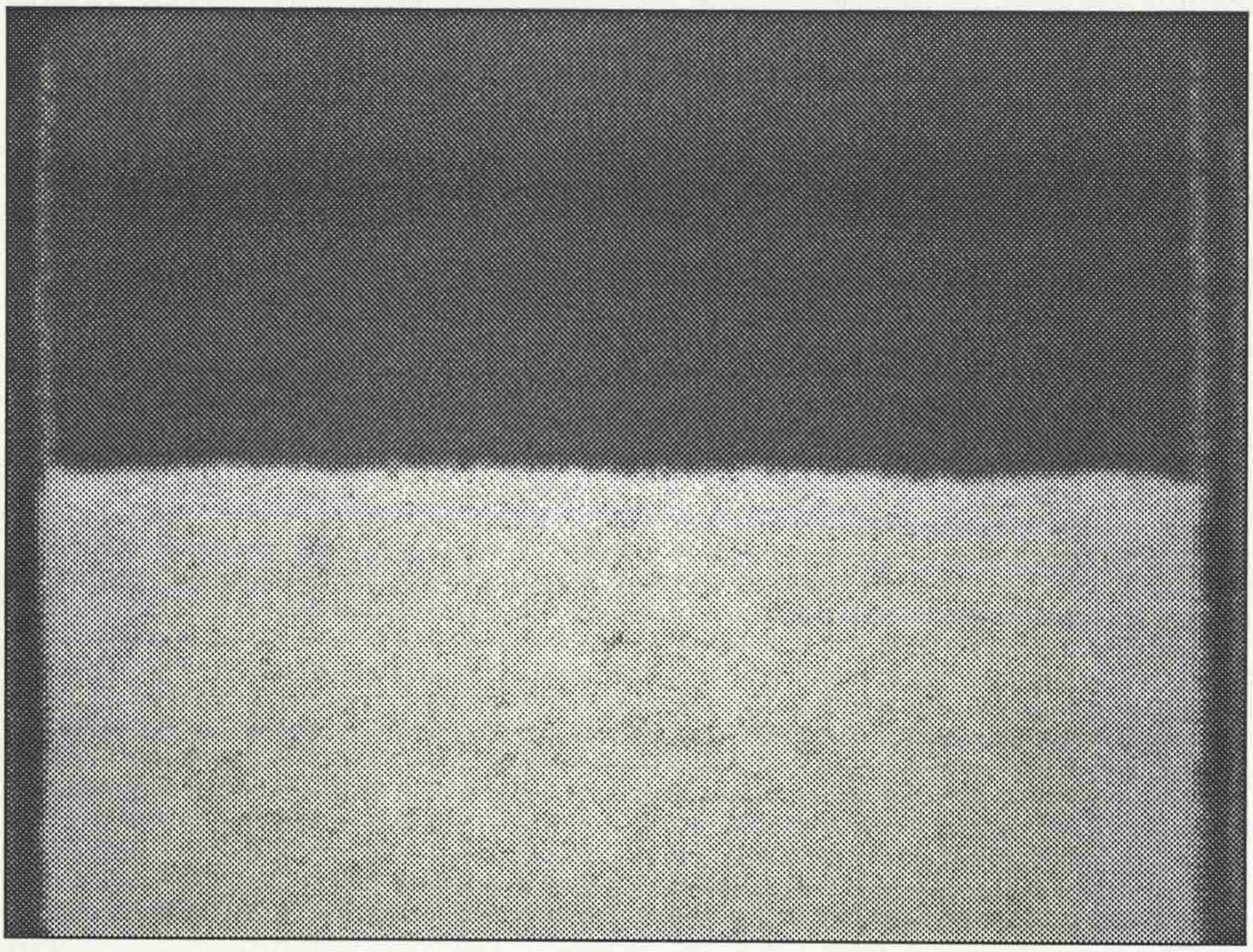


Image 7.10 (a) - The output from the coherent bundle when transmitting a test pattern

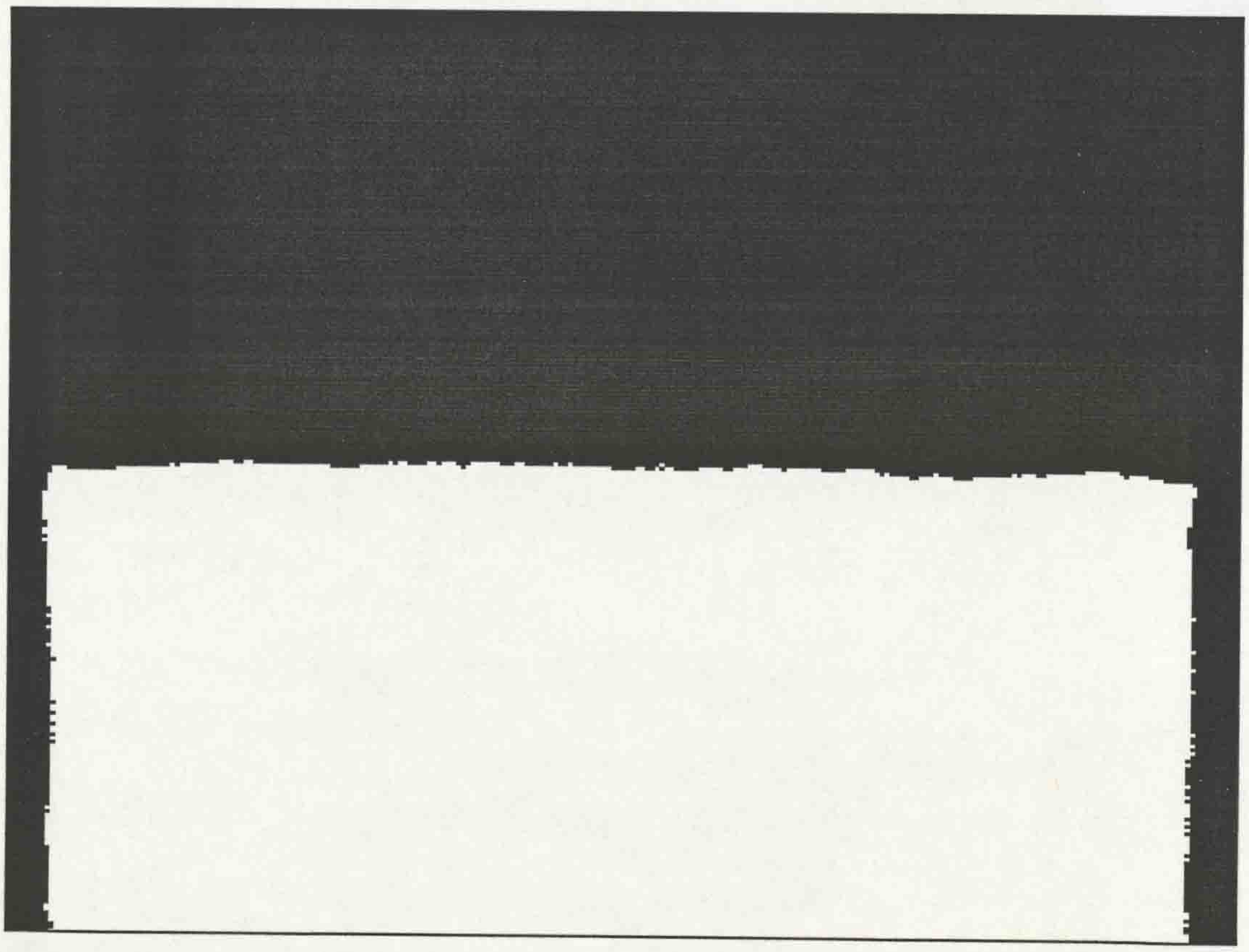


Image 7.10 (b) - The thresholded output showing accurate threshold choice

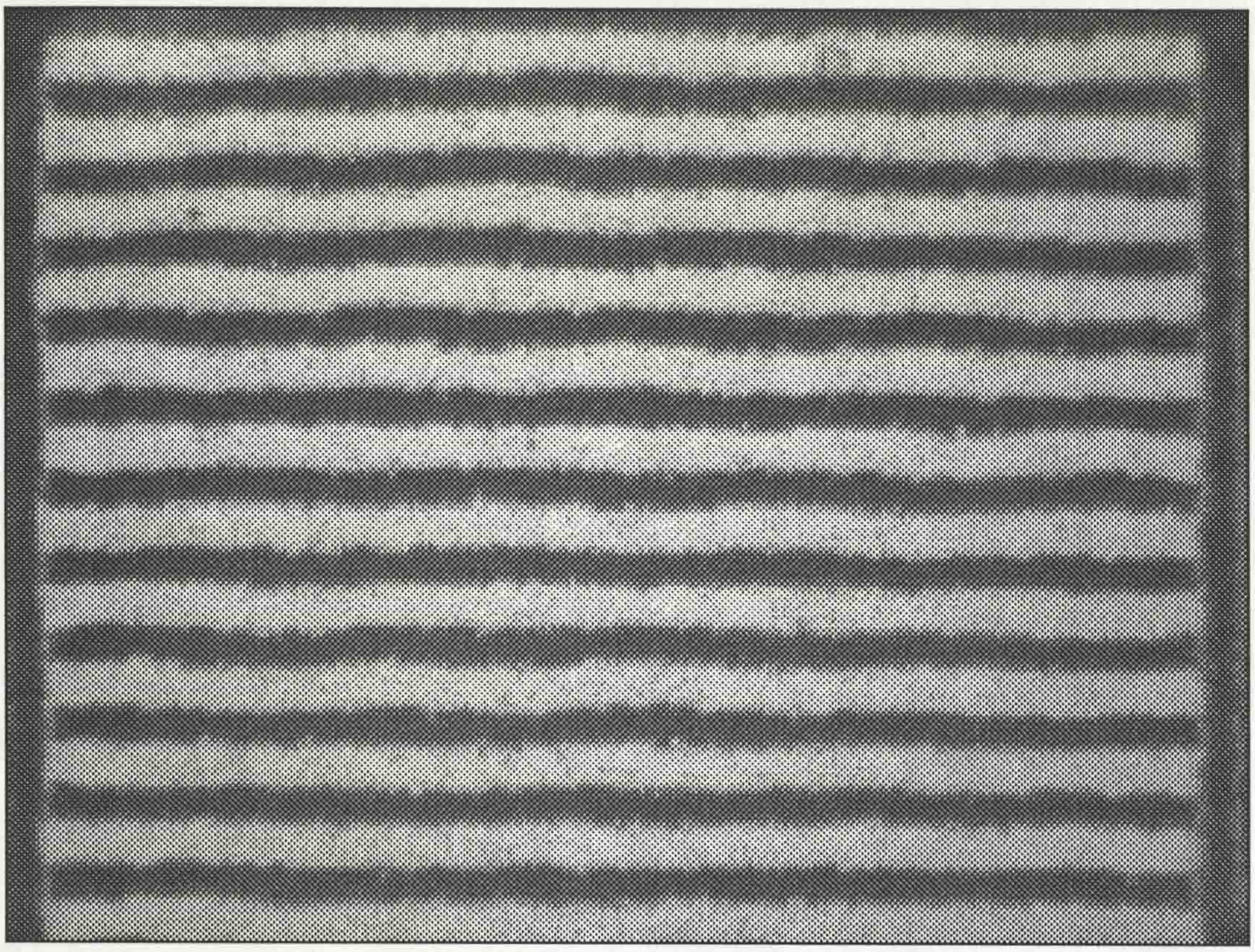


Image 7.11 (a) - The output from the coherent bundle when transmitting the test pattern for bit 4 (Numbered 0 to 7)

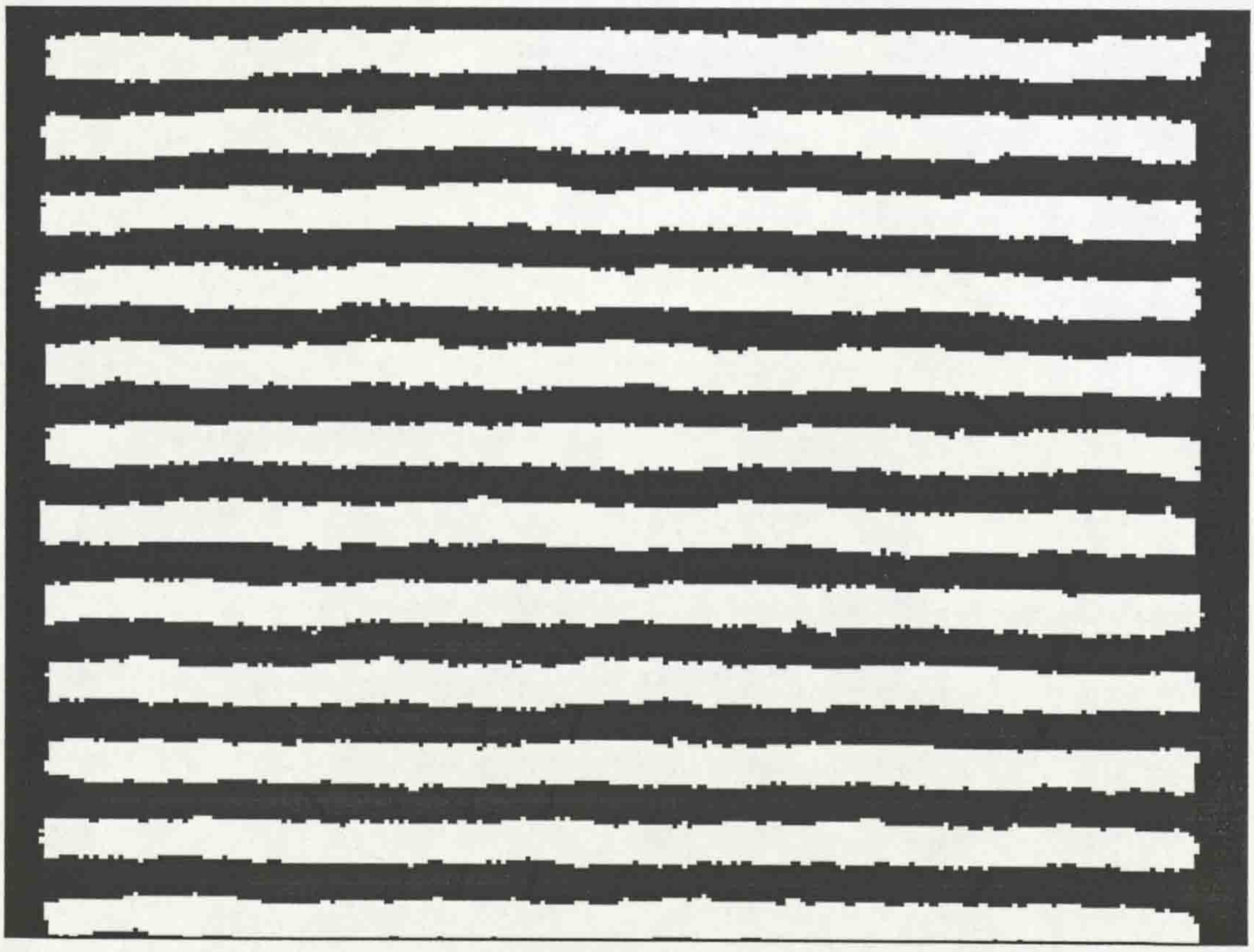


Image 7.11 (a) - The thresholded output showing the accuracy achieved

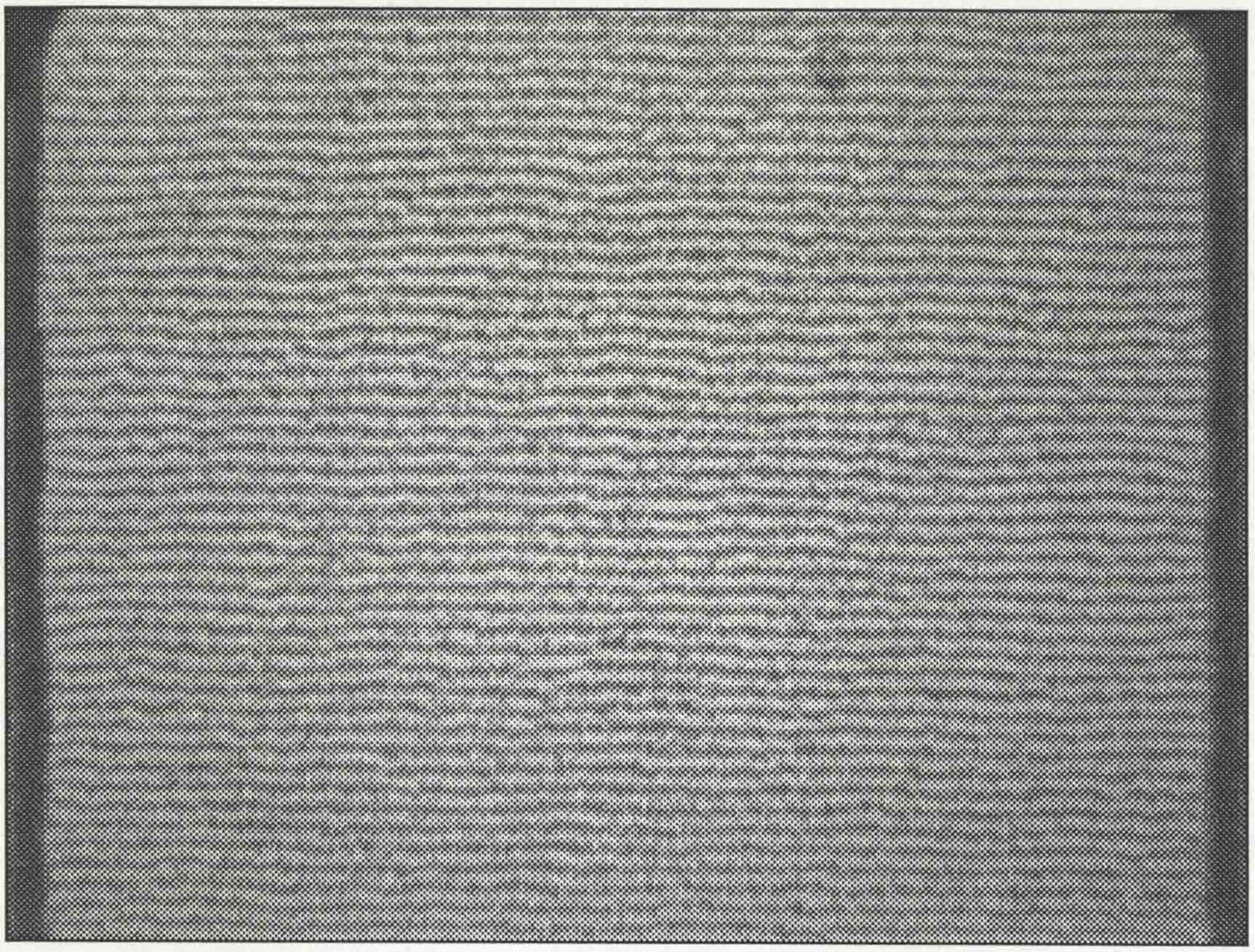


Image 7.12 (a) - The output from the coherent bundle when transmitting the test pattern for bit 1 (Numbered 0 to 7)

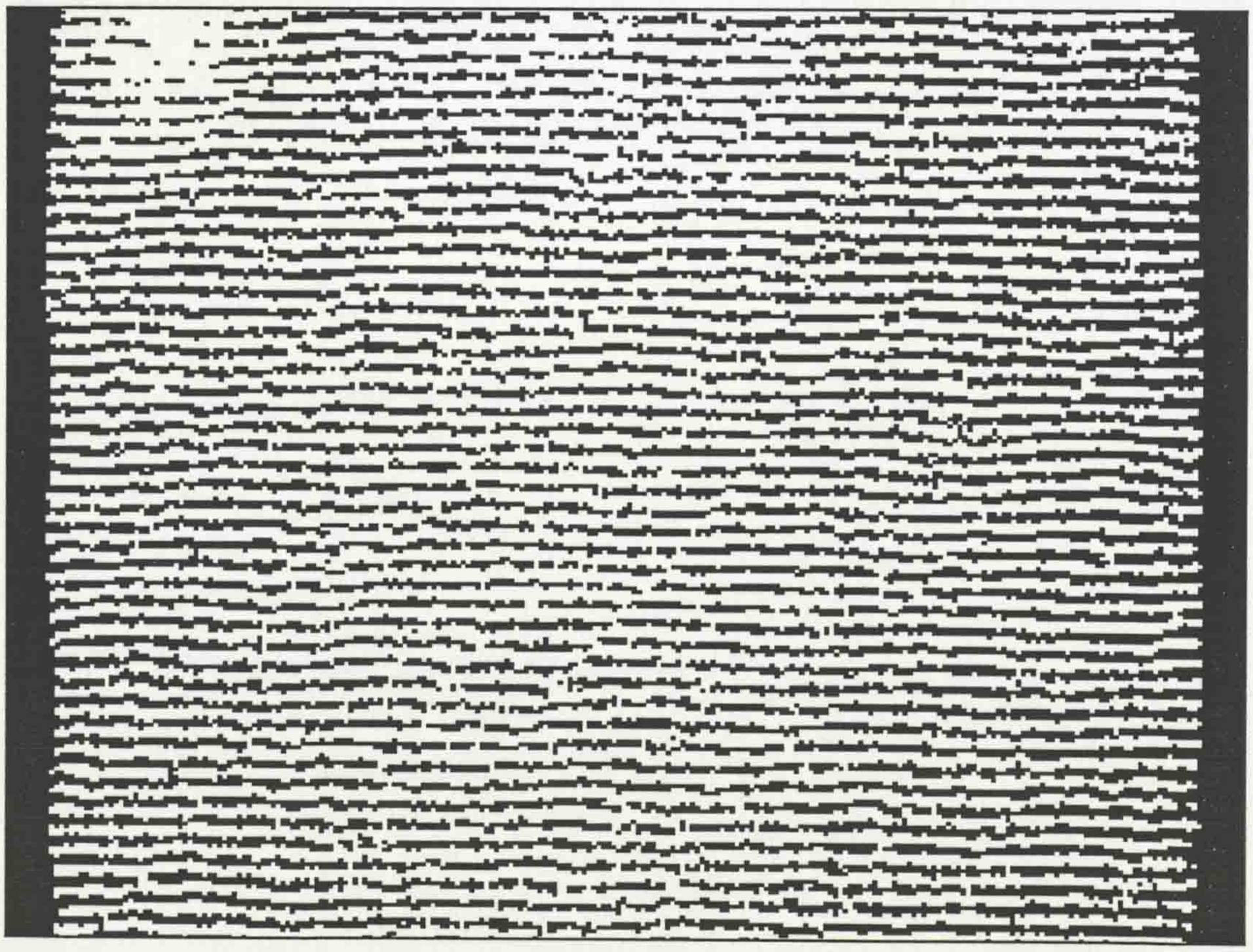


Image 7.12 (b) - The thresholded output showing the loss of accuracy with low order bits

7.3.1 High Resolution Optical Fibre Bundle (Coherent)

The first trials were conducted with the coherent optical fibre bundle. This was primarily due to the fact that this would allow direct comparison with a coherent image to assess the accuracy and worth of any calibration method as different methods were implemented. Images 7.13 to 7.19 all apply to the calibration of this bundle. In some tests, to simulate an incoherent bundle the coherent optical fibre bundle was rotated through an angle of 90° about its longitudinal axis. If the theory of the calibration method was correct, the image would be returned to a vertical position.

7.3.1.1 Calibration by CRT monitor

For assessment of the result of this trial, two input images were used: the simple image of black text on a white background, and three objects on a white background. The three objects were a digital mouse and pair of pliers as before, and a round brass tube. The tube would be used to assess the reconstruction of round objects.

Examining Images 7.13 (a) and (b) shows that a significant degradation of image quality was introduced. The majority of the distortion appears to have occurred as a result of errors in the y coordinates in the LUT. The distortion can clearly be seen as black vertical lines. On initialisation, before calibration, the entire set of values in the LUT is set to zero. Any output point not assigned an input will therefore remain at zero. During reconstruction, intensities will be remapped from line and/or column zero by these unaltered LUT entries. The black lines in Images 7.13 (b) and 7.14 (b) are therefore from the first column of the image which is black since it does not transmit any of the image in this instance.

The main part of Image 7.13 (b) also shows some displaced intensities. This occurred when, during calibration, a pixel was wrongly chosen to be above or below the threshold, and its input coordinate modified accordingly. The effect of such errors was introduced in section 6.4.4. The distortions appear to have occurred mainly in the x coordinates, which is consistent with the distortion patterns of the monitor, which tended to occur primarily in the horizontal direction. The distortion is regularly spaced, as would be expected if any one bit of the input address was altered. The image of the text is recognisable as such, and is readable.

Image 7.14 (b) shows the reconstructed image of the objects. The more complex image, with the greater variation in intensities shows up the errors in the y coordinates. A regular grid pattern can be seen, apparently superimposed over

the image. The grid pattern corresponds to the black/white borders of the test screens, where the wrong decision was made as to the relationship of the intensity, at that location, to the threshold. This error arises because the intensity function at the borders is not the sharp step function it should be, but a gradual change of intensity, albeit over a small distance. This gradually changing intensity function at the borders alters the intensity output of the fibres transmitting the feature, moving them towards the threshold value, and over or under that value. If the right decision was made in the main areas of the test pattern, the majority of the pixels would be correctly remapped. As the number of borders in the test pattern increases with decreasing significance of the input address bit being tested for, the number of errors generated will increase.

The problem associated with the borders is further complicated by the nature of light transmission of single optical fibres. The total light output of each fibre is dependent on the average amount of light falling on the input end (section 3.2). The fibres that transmit the intensities at the border will be receiving a partly illuminated image and the output intensity will lie somewhere between the bright and dark levels, and may be below or above the threshold when they should not be. This would occur at every border in the test pattern. As the number of borders increases, the displacement caused by the errors at the borders decreases. The errors are still noticeable, and appear as a spreading of intensities in the reconstructed image, effectively reducing the resolution of the bundle.

The distortion caused by the spread of intensities at the borders in the test patterns and the distortion of the test screens themselves, by the CRT monitor, were the reasons for changing to a different test pattern generator.

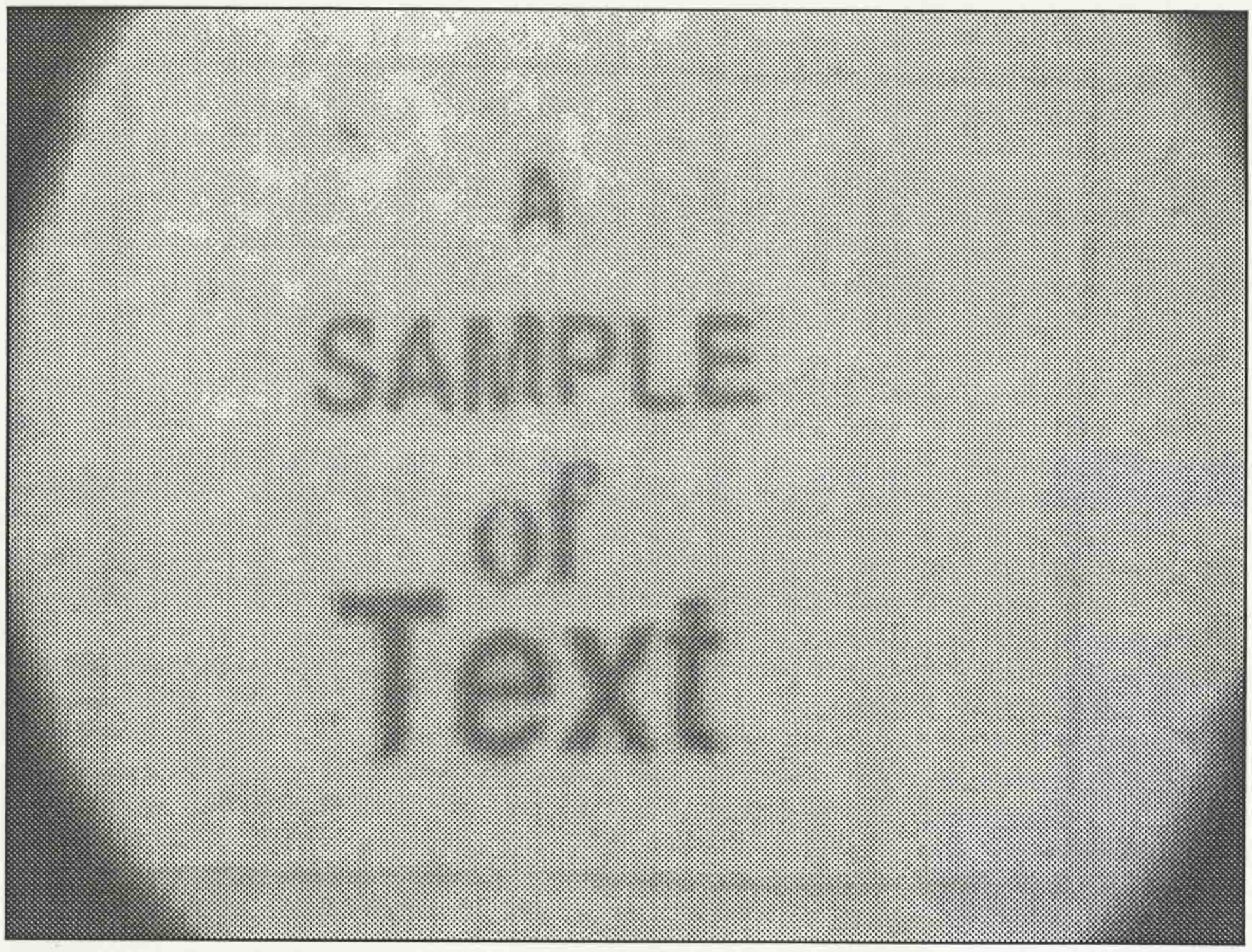


Image 7.13 (a) - The output from the coherent bundle when transmitting a simple image

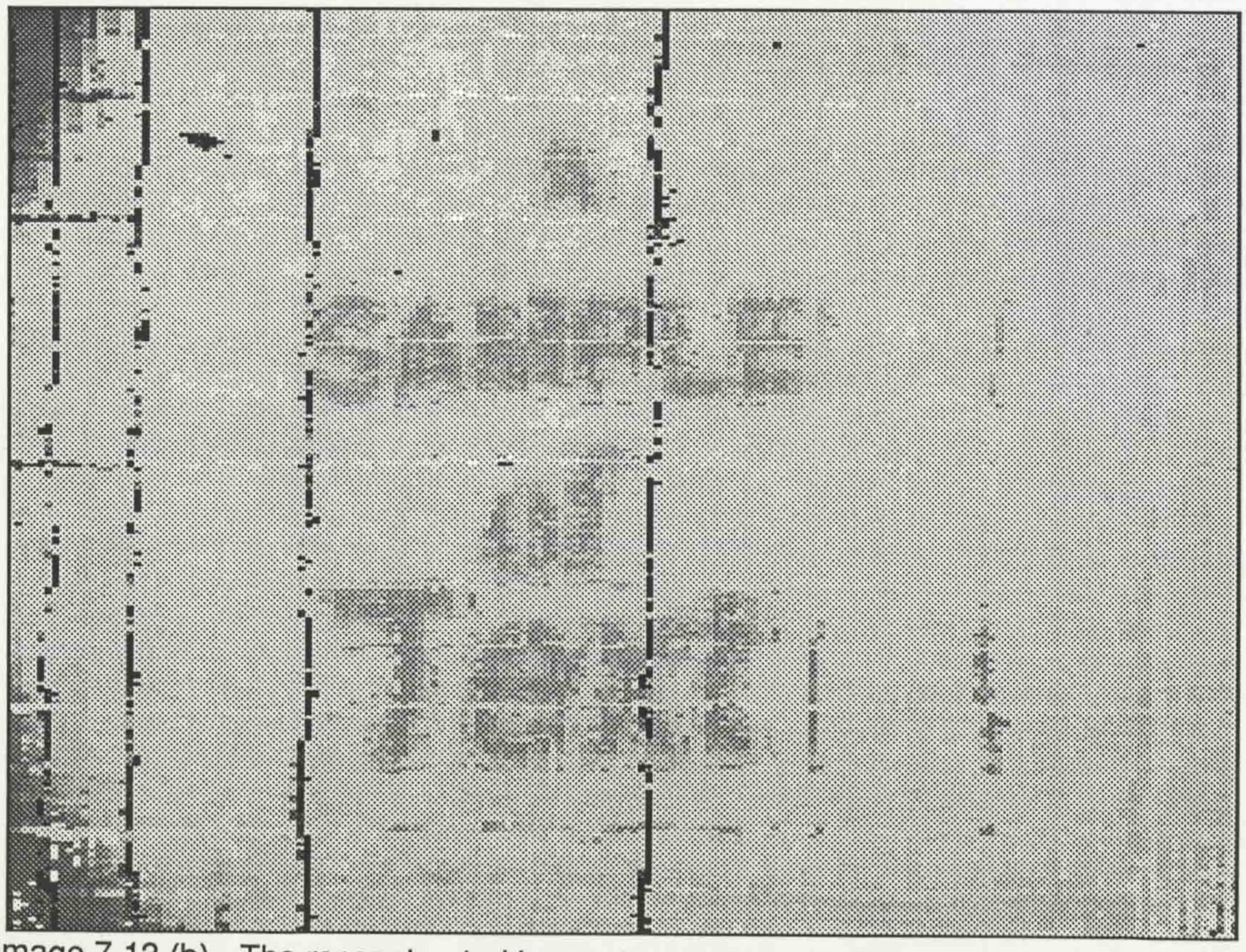


Image 7.13 (b) - The reconstructed image from a LUT obtained by CRT calibration

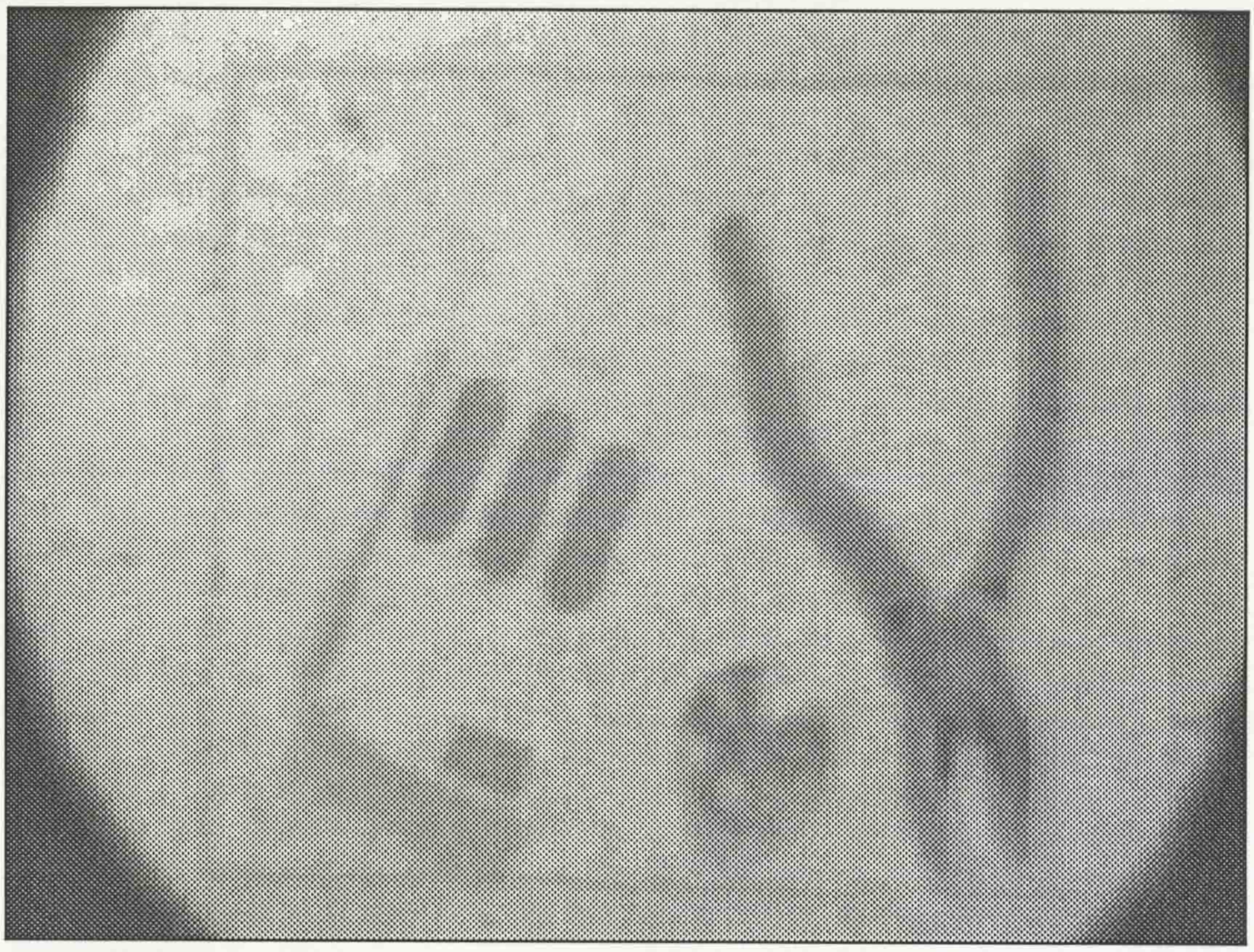


Image 7.14 (a) - The coherent bundle's output for a more complex image

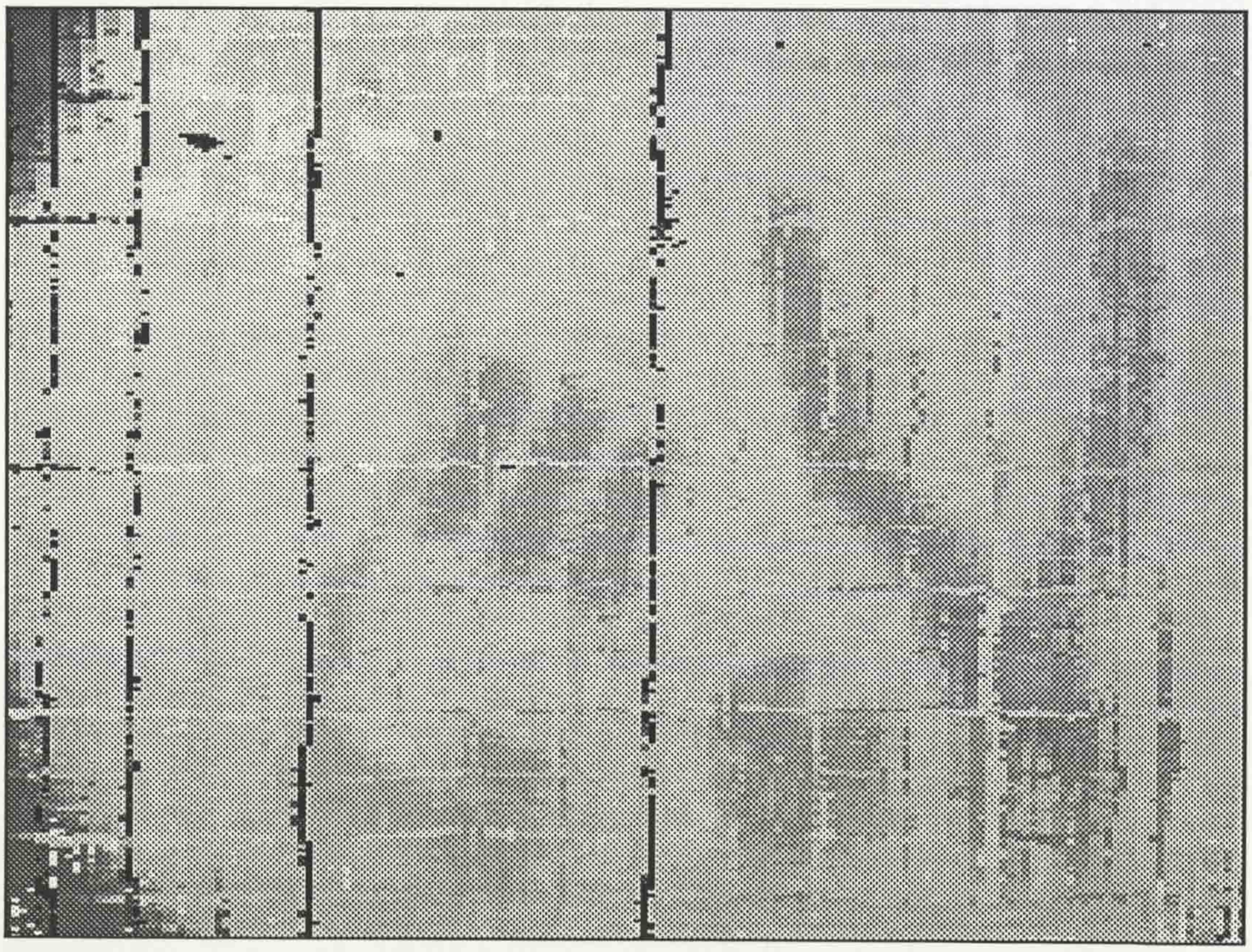


Image 7.14 (b) - The reconstructed image (LUT from CRT calibration) showing errors generated by test screen distortion

7.3.1.2 Calibration by Photographic Test Slides

The two primary reasons for using the photographic slides for calibration by test patterns were a consistent image size and uniform light distribution over the image, making the threshold operation more accurate. These expected improvements are realised, as can be seen in Images 7.15 to 7.17. The borders between the black and white areas of the test pattern should be sharper than those on the monitor, since the edge was defined by a physical change in intensity, rather than the change of phosphorescence as in the CRT monitor [*18*].

Errors are still present in the reconstructed images (Images 7.15 (b), 7.16 (b), and 7.17 (b)). These fall into two categories: errors caused by misalignment in the set of photographic slides, and errors caused by the wrong choice of threshold. It was seen, in Images 7.10 to 7.12, that the thresholding method was as accurate as could be expected, but was subject to some error at the black/white borders. The borders present the same problem as explained in section 7.3.1.1, where the fibres transmitting the image of the borders carry an intensity that falls on the wrong side of the threshold, causing an error in the input address allocation. The improved edge definition of the test slides reduces this error, but neutralises the improvement by increasing the errors caused by location tolerances.

The main location errors can be seen as the two horizontal black lines in Images 7.15 (b) and 7.16 (b). Images 7.16 (b) and 7.17 (b) show the effect of the regular grid pattern of errors caused by misallocation at the lower bit levels. The errors cause displacement of the intensities within the image. Although the image is still recognisable, the direct comparison with the coherent output of the fibre shows significant image quality losses. The wider the range of intensities in the image, the greater the effect of the errors, since the displacement of intensities is then subject to a greater probability of that intensity being incorrectly displayed in an area of much different intensity. The effect is very noticeable in Image 7.17 (b).

The quality of images from this calibration was deemed to contain too many location errors, and the loss of picture quality compared to the original, was too great. Most of the errors were due to the location of the photographic slides, and if this could be improved, then the errors could be reduced. The reconstruction does show however, that the images were correctly rotated back to the vertical, while maintaining the spatial relationship between the elements of the images. The effect of the lower order bit errors is to reduce the perceived resolution of the bundle of fibres.

The reduced resolution is a result of the misallocation of bit values, similar to those described in section 6.4.4. Consider the example below for an input address x_i:

1 1 1 0 1 0 0 1

Bit 7 Bit 0

If an error was made in the value of bit 0, the erroneous value of x_i would be

1 1 1 0 1 0 0 0

Bit 7 Bit 0

The decimal value of x_i would be 232 instead of 233. This would cause the intensity at the input pixel location for this output point to be displaced by one unit of horizontal position. If all the surrounding pixels at the input position were correctly remapped, this erroneously located pixel would appear as a smearing of the intensities associated with the object being displayed, effectively reducing the resolution of the reconstruction procedure. The errors can be clearly seen in the results, because of the coherent nature of the bundle. If the bundle was incoherent, the errors would be randomly distributed over the image.

The reconstructed images, where the original was rotated through 90°, show an increase in size and the loss of part of the image. The effect is a result of the rotating function of the LUT, and the combination of a square calibration area and the 4 : 3 (x : y) aspect ratio of the display. If the original image was rotated back to an upright position as dictated by an unmodified LUT, the x coordinates would become the y coordinates and vice versa. This would cause the image to be compressed into a 3 : 4 aspect ratio. To correct this effect, this values in the LUT were scaled to negate the effect.

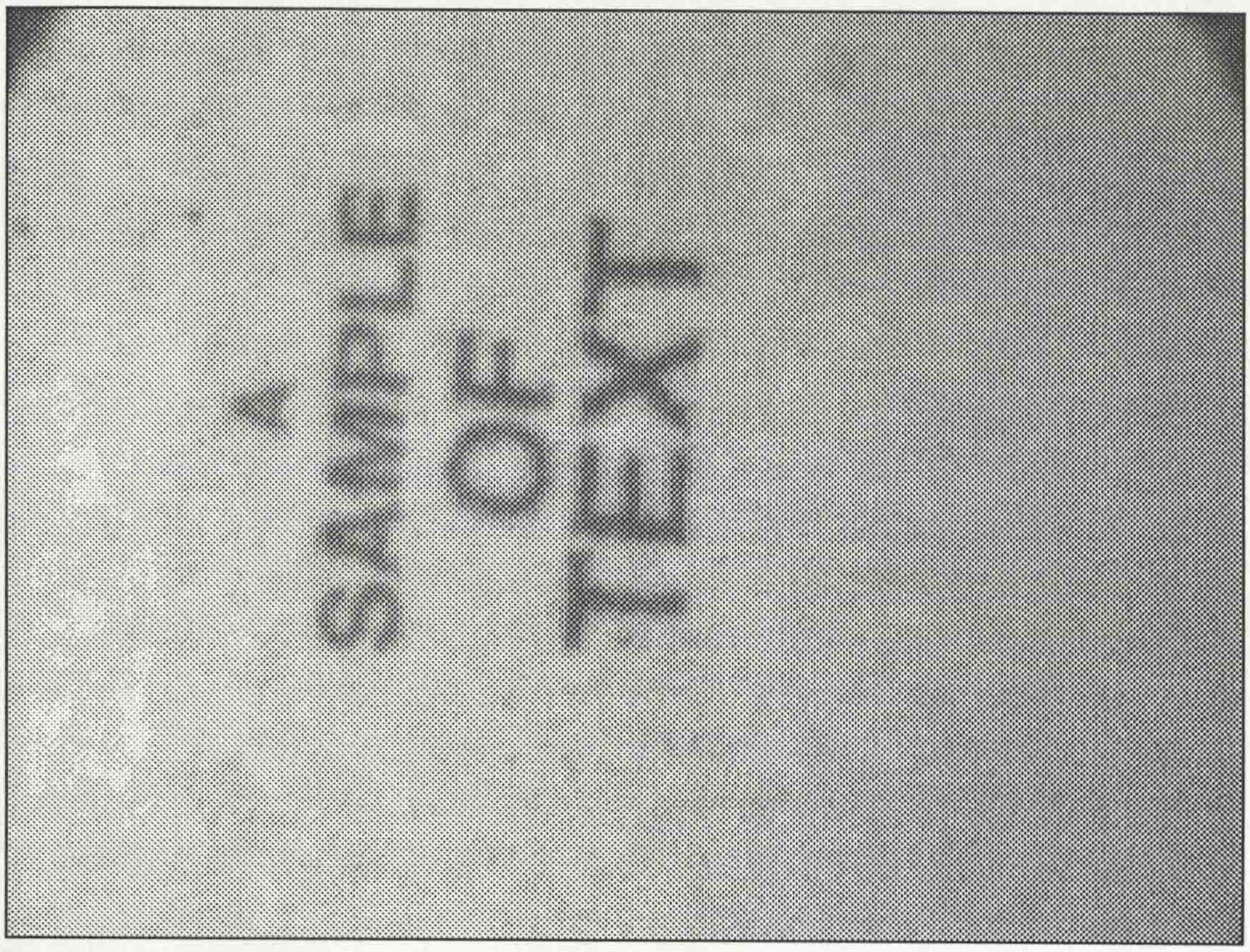


Image 7.15 (a) - The output from the coherent bundle

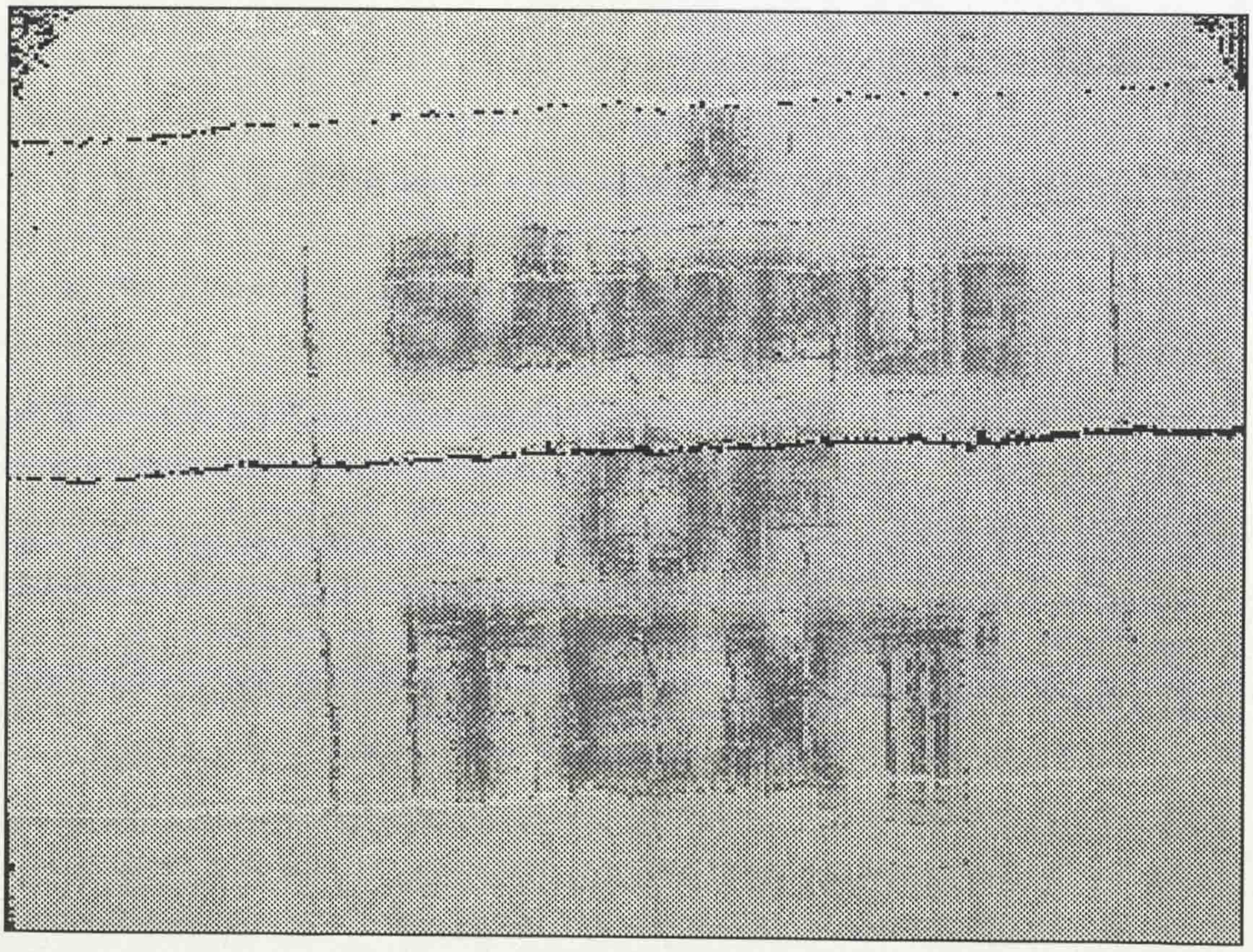


Image 7.15 (b) - The reconstructed output showing the correction of the 90° rotation of the input image (a pseudo incoherent model)

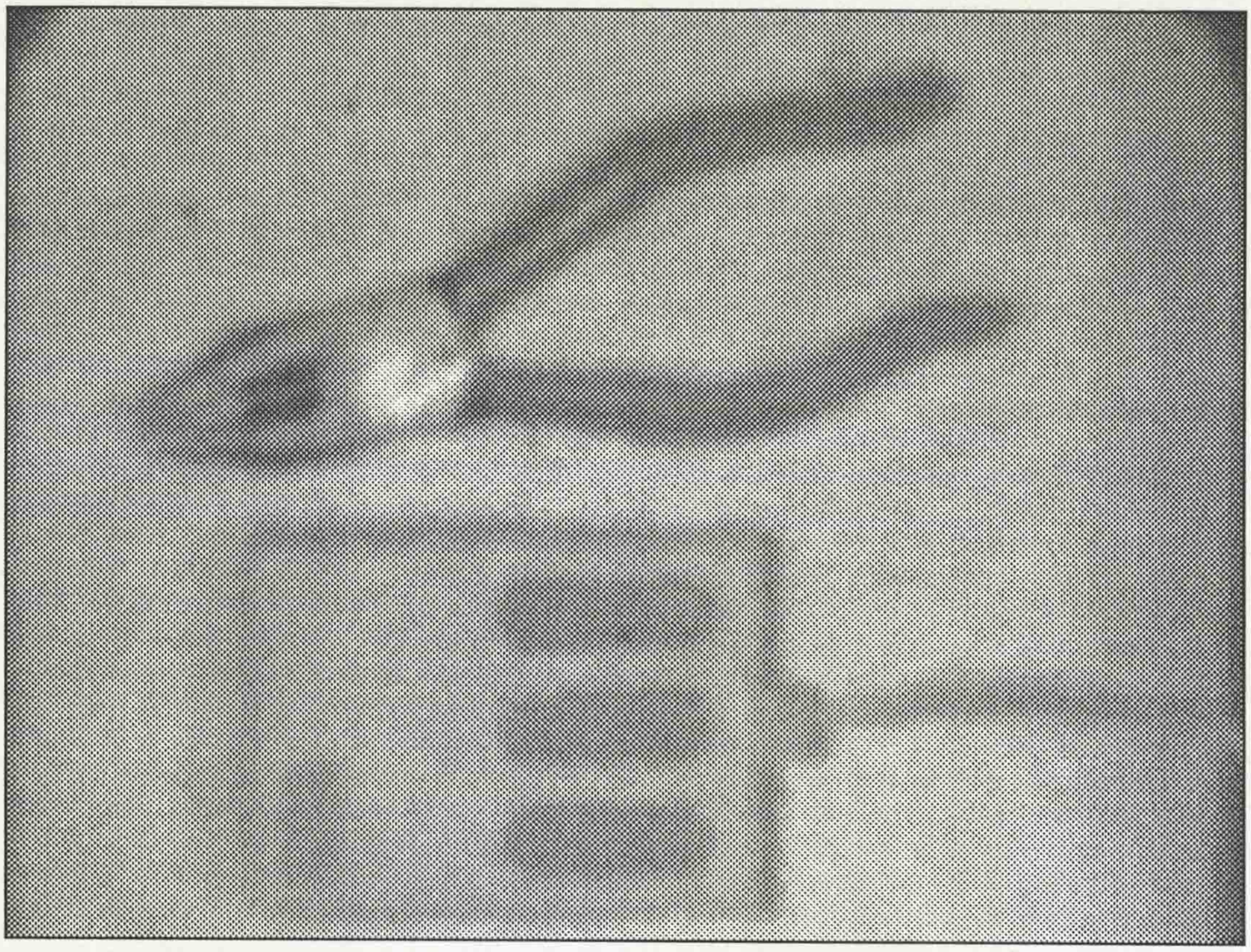


Image 7.16 (a) - The output of the coherent bundle when transmitting a complex image

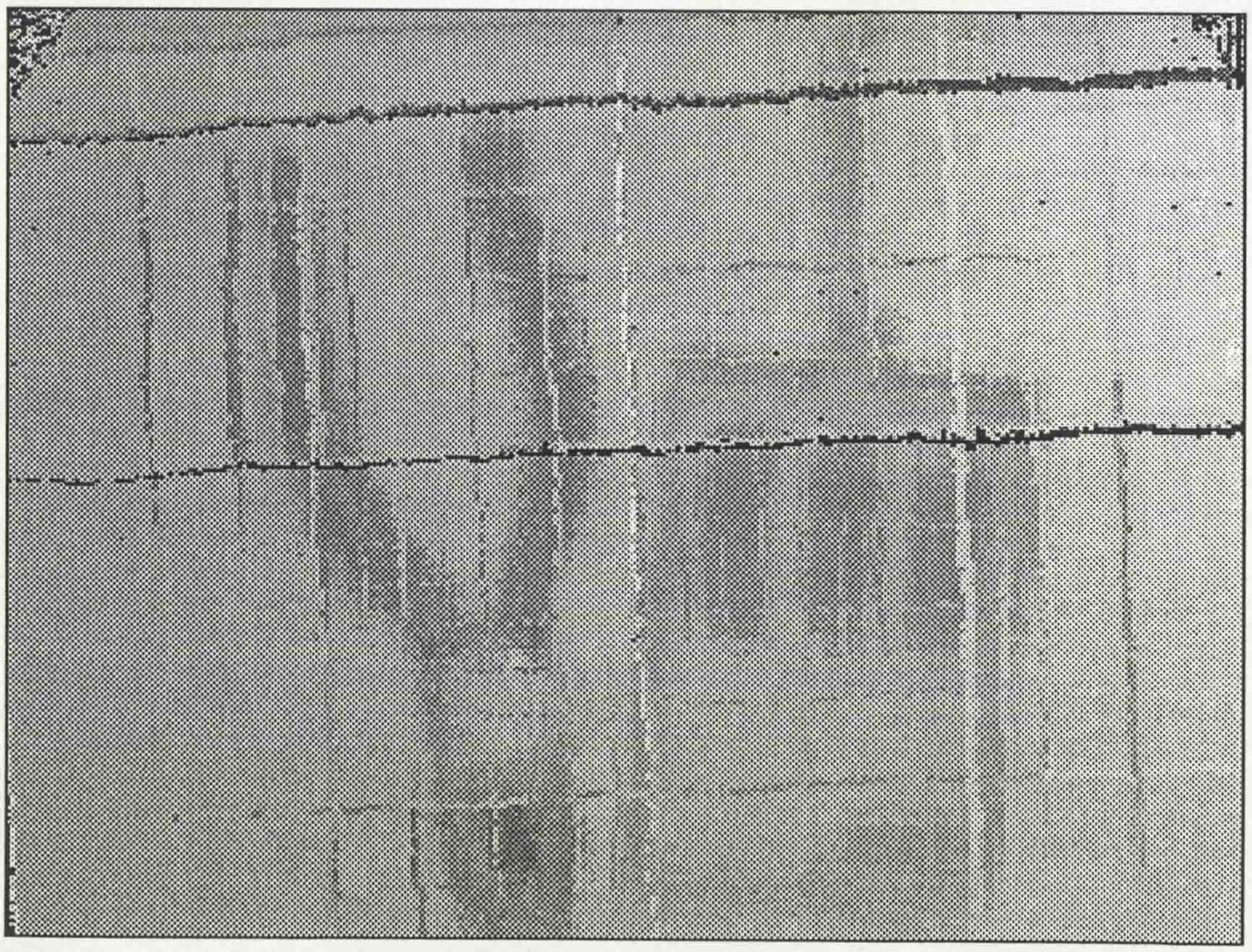


Image 7.16 (b) - The reconstructed output clearly showing the errors generated

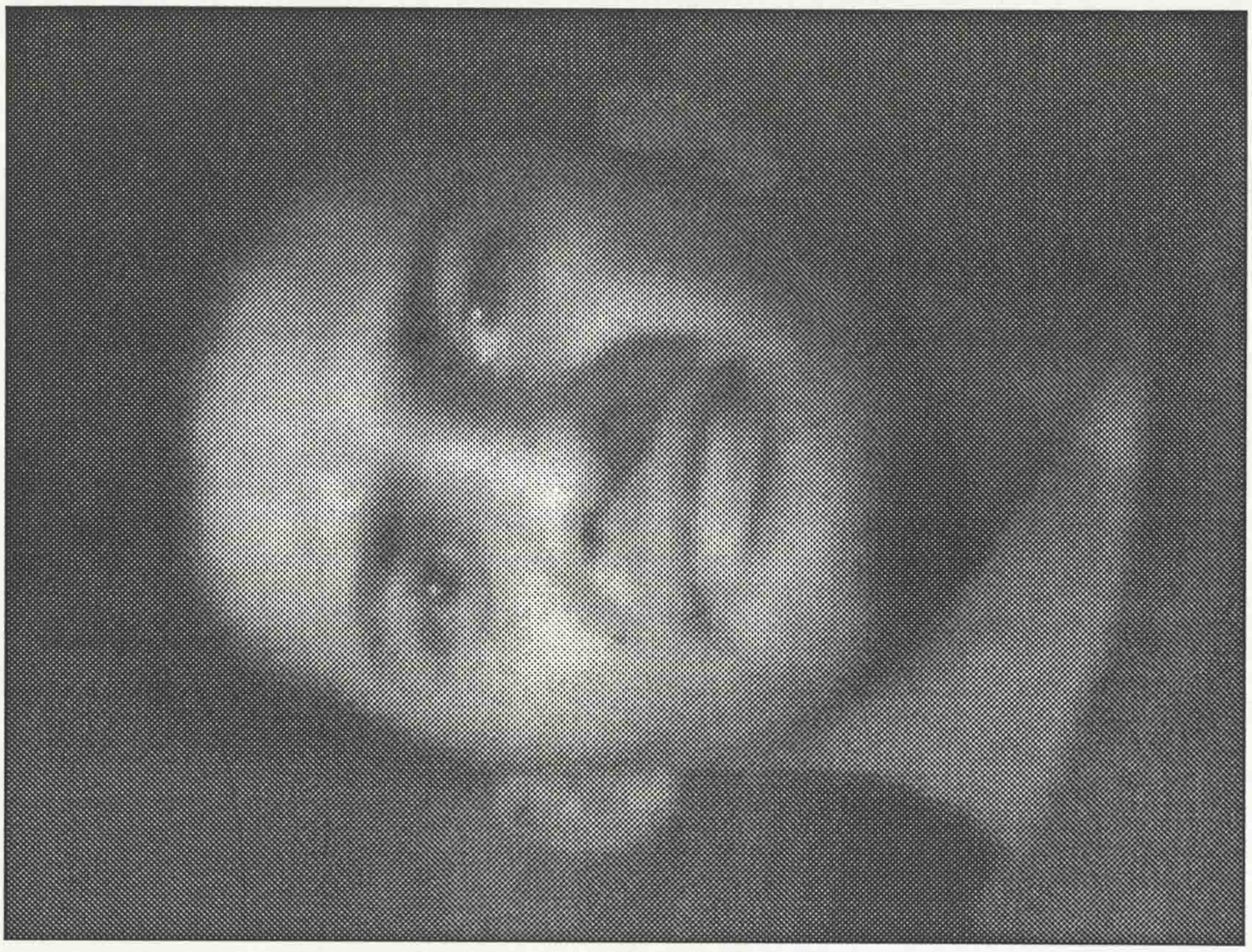


Image 7.17 (a) - The output from the coherent bundle transmitting a complex image of large intensity variation

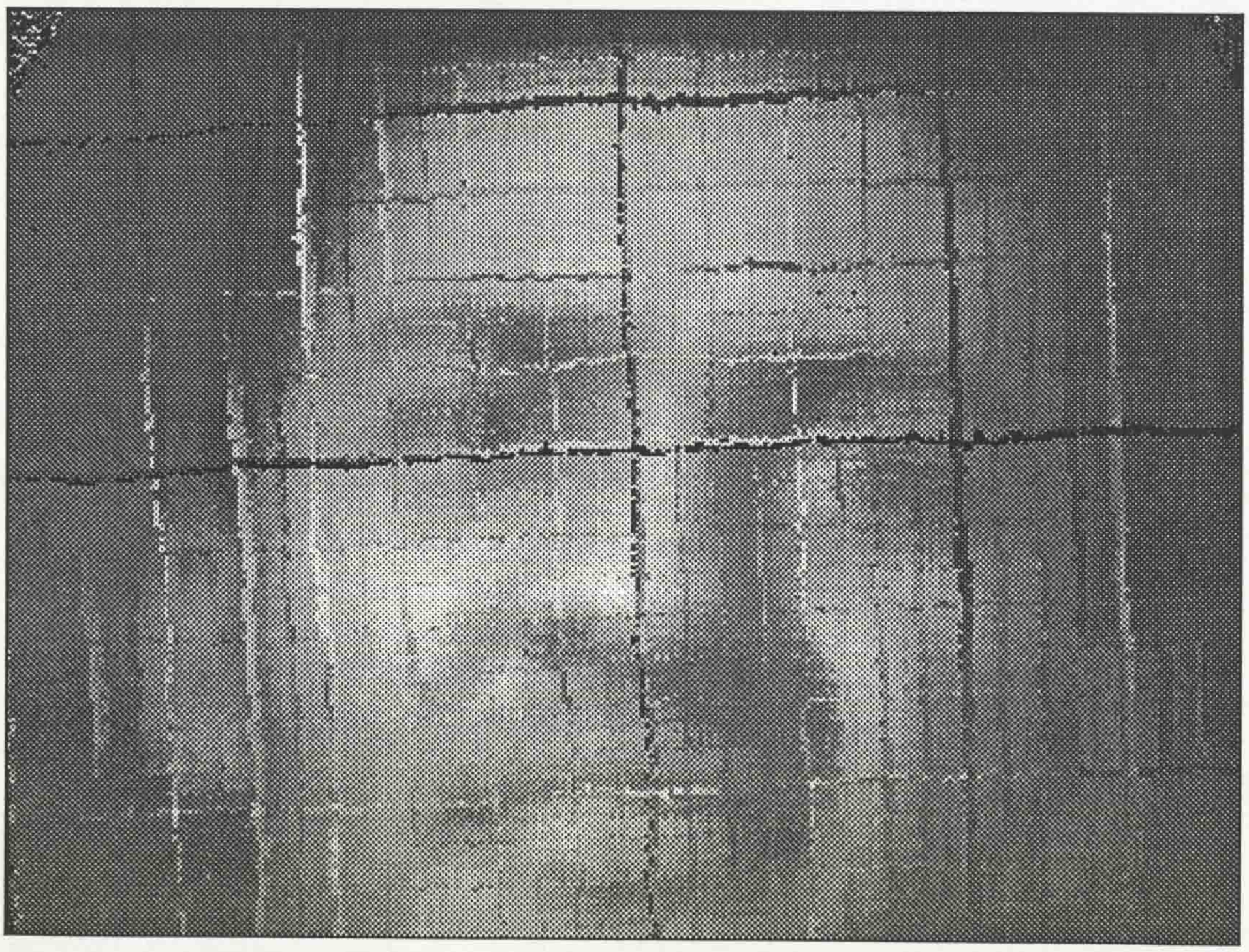


Image 7.17 (b) - The reconstructed output, showing correct rotation

7.3.1.3 Calibration by Plasma display

The calibration by plasma panel test pattern should in theory provide the ideal solution to the previous problems that arose. It would provide an accurate, uniformly illuminated test pattern for the calibration. In practice, this proved to be so, but another problem arose. The light output from the plasma panel was comparatively low, requiring the lens apertures at the input and output end of the fibre bundle to be opened to large aperture values [*38*].

Lens theory [*3,38*] shows that the larger the aperture, the more shallow the depth of focus of the lens. Under such conditions, the black/white borders of the test patterns (dark grey/orange for the plasma panel), are not well defined. As a result, the calibration is again susceptible to same error source as described by section 7.3.1.1 . The accuracy of placement of the test patterns, and their uniformity, improved the overall calibration mechanism, but allocation of the lower order bits in the input addresses, where the test patterns contain many edges, are susceptible to a greater percentage of error. The flickering plasma panel display also provided a source of error. The capture of multiple images, followed by an averaging function (section 6.6.5) would cause small variations in image intensity between successive capture and average operations. Thus the images used to calculate the thresholds could contain slightly different intensities to those contained in the test pattern images. This would lead to error in the calculation of the thresholds for the images.

These theories are reinforced by the results shown in Images 7.18 to 7.20. The input images were again rotated through 90° to check that the calibration result would correct the rotation. This can be seen to be so in the reconstructed Images 7.18 (b), 7.19 (b), and 7.20 (b). The source images were text, tools and a face. The reconstructed image of the text, Image 7.18 (b), shows that the text was more accurately located than with the previous two methods. However, the lower order bit errors cause the image to be spread over its immediate display area, effectively blurring the output image. The text is still visible, although the characters are less well defined.

The image of the tools, taken at a low light level, is reconstructed to give a passable representation of the original. The grid pattern of distortion is again present, due to the factors explained previously. The reconstruction of the face shows a recognisable image, again blurred by the effects of the LUT obtained from the plasma panel calibration.

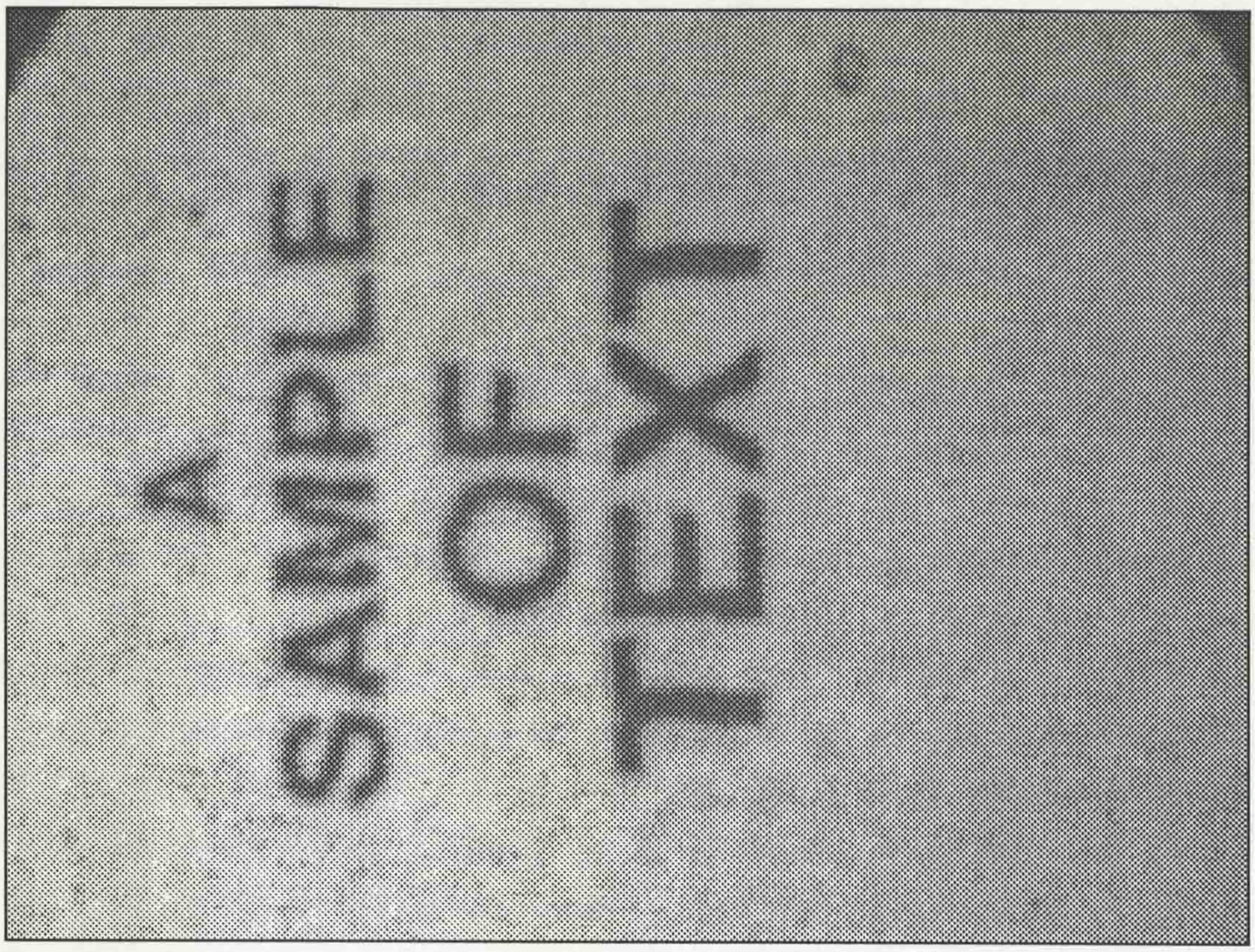


Image 7.18 (a) - The output of the coherent bundle

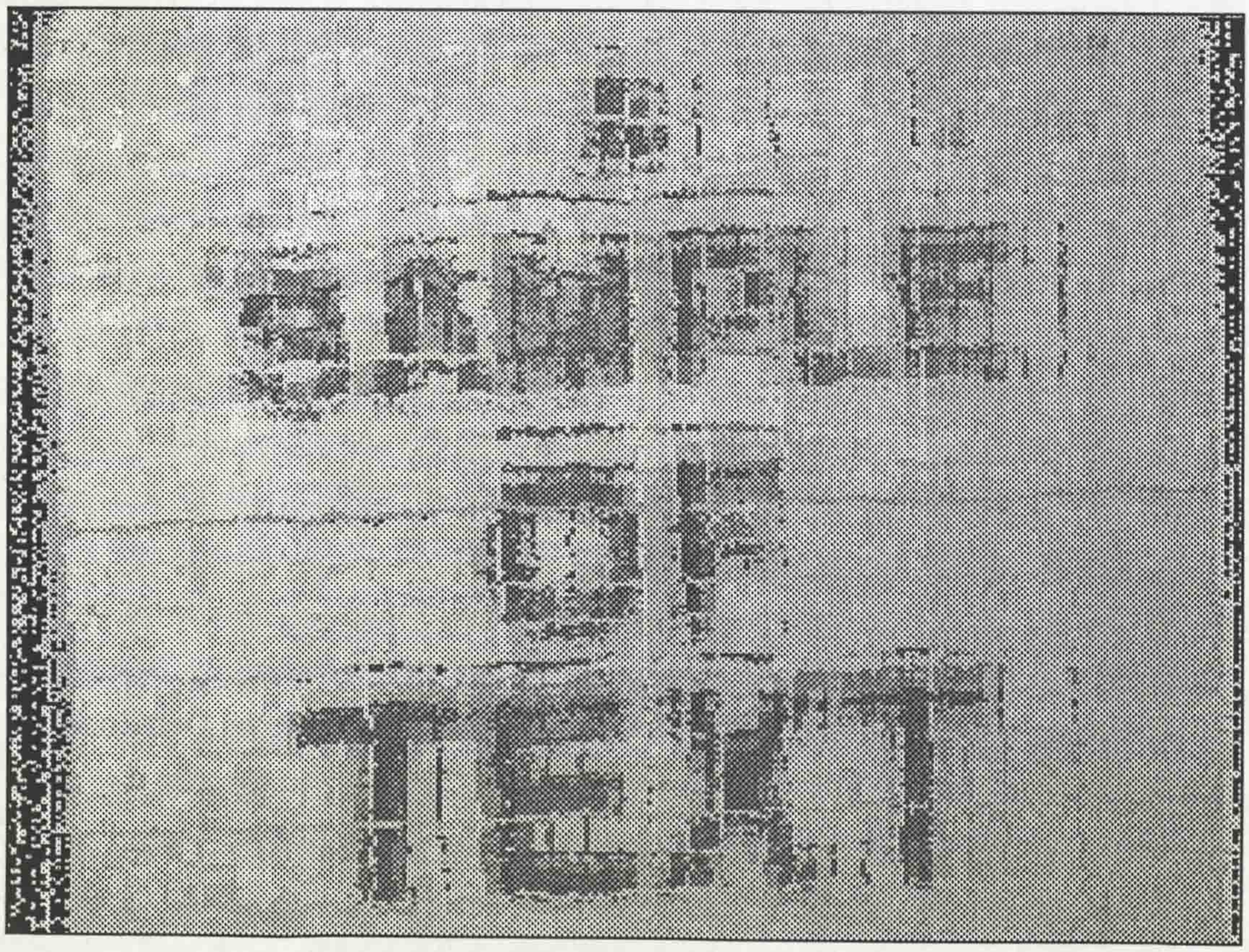


Image 7.18 (b) - The output reconstructed by a LUT obtained by plasma panel calibration

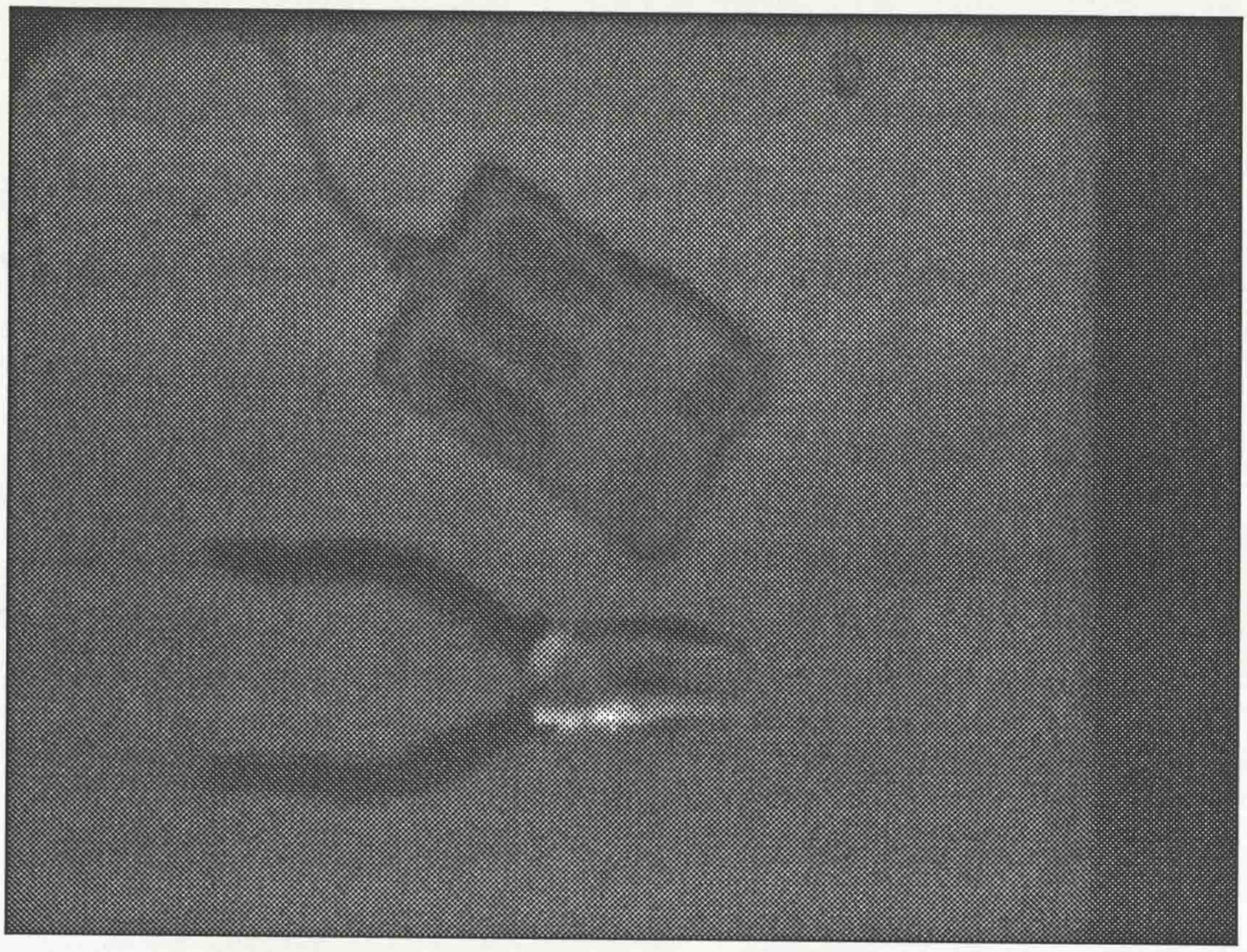


Image 7.19 (a) - The output of the coherent bundle

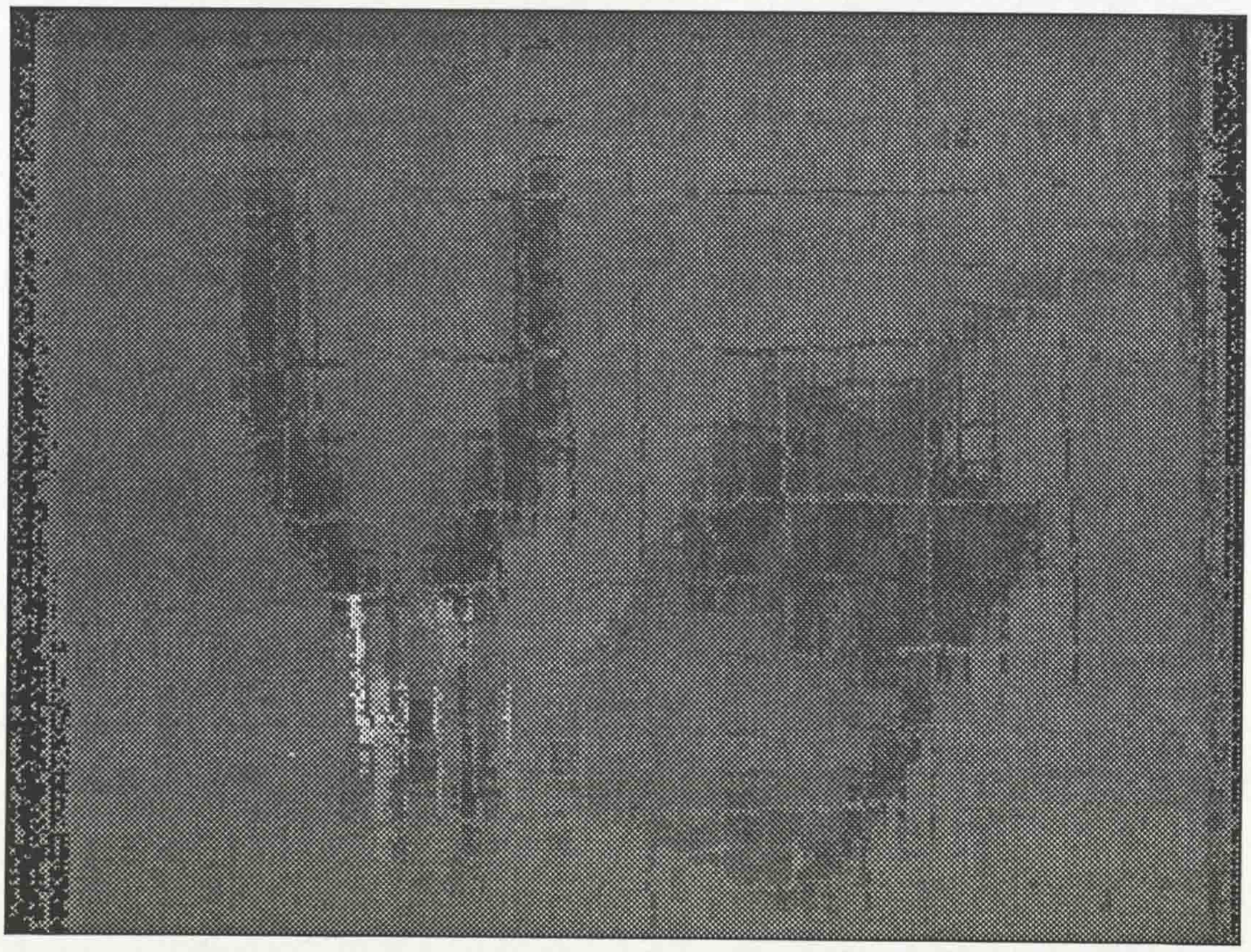


Image 7.19 (b) - The reconstructed output

Image 7.20 (a) - The output of the coherent bundle

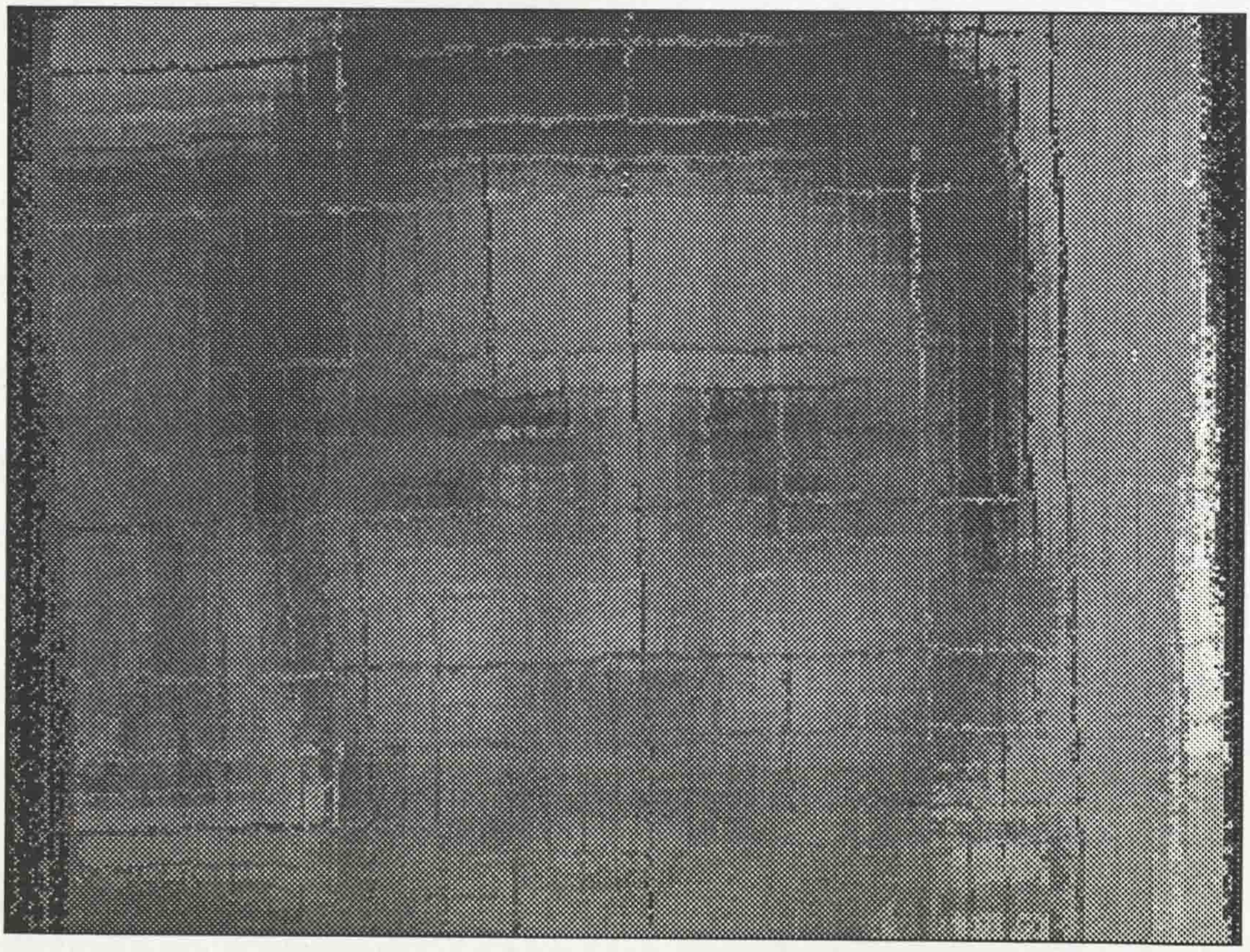


Image 7.20 - The reconstructed output

7.3.2 Low resolution optical fibre bundle (Incoherent)

Examination of the previous calibrations by test patterns shows that all the sources of images tried were subject to various sources of error. All the errors produce the same effect - the displacement of intensities in the reconstructed image. The bit oriented nature of the test pattern calibration method allows a regular error pattern to be seen in the reconstructed image when a coherent bundle of fibres was used for the calibration trials. For an incoherent bundle, these errors would not appear as a regular grid, but as general random distortion of the image.

The resolution of the bundle used for the calibration is of paramount importance to the ability of the test pattern method to perform the calibration. The bundle must be able to adequately resolve the test patterns before any results could be taken. To this end, a primary set of results were taken where the test patterns were transmitted by an incoherent bundle, and the output reconstructed by a LUT obtained for the optical fibre bundle by the spot calibration method. Images 7.21 to 7.23 show the results, input and output images, for this test. Allowing for the small percentage of errors in the data in the spot calibration LUT, good results are shown for the test pattern for bit 5, Image 7.21 (b), although the black/white borders are slightly blurred. The source of the test patterns was the photographic slides. These were chosen because of the high light output and their good edge definition. It was decided that the misalignment of the test slides would be of little consequence compared to the low resolution capability of the bundle.

For the output for the test patterns for bits 2 and 1, Images 7.22 (b) and 7.23 (b), the resolution of the test patterns can be seen to decrease rapidly as the number of black/white borders increases. The test pattern for the lowest order bit, bit 0, the fibre bundle was totally incapable of resolving the image. It was not expected that, with the 3.5 mm bundle containing approximately 4400 fibres, calibration by the test pattern method would be possible. A combination of the errors that were inherent in the calibration procedure, the apparatus used, and the low resolution of the bundle would interact to produce results that would contain a large, if not unacceptable of errors.

Images 7.24 (a) and (b) show the input and output results respectively, that were obtained when using the incoherent bundle, after calibration by test pattern, to transmit the test pattern for bit 7 of the y address (Figure 6.2 (a) rotated through 90° clockwise). The output shows a near incoherent image, justifying the premise that a high resolution incoherent fibre bundle is needed for the calibration by test patterns.

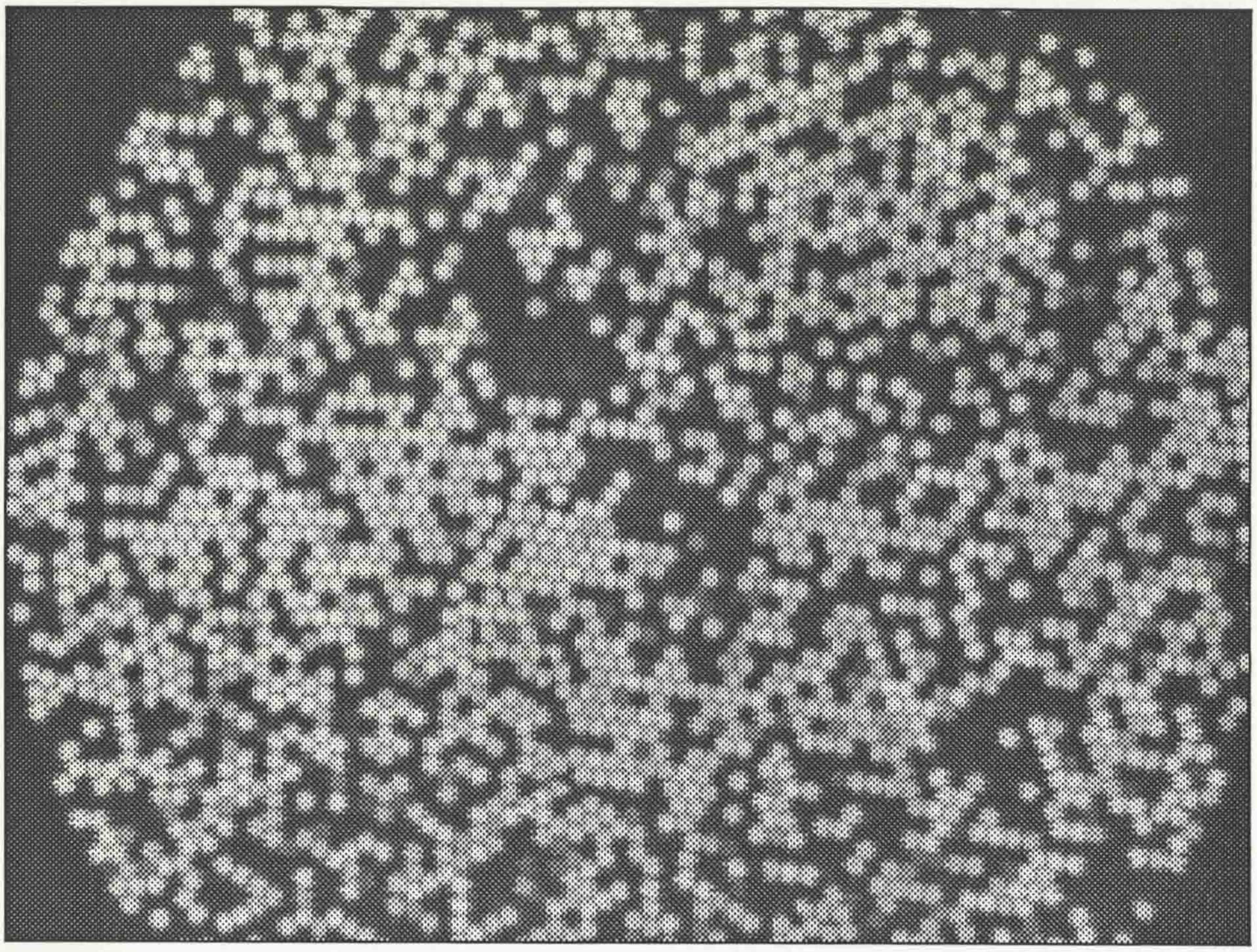


Image 7.21 (a) - The output of an incoherent bundle when transmitting a test pattern

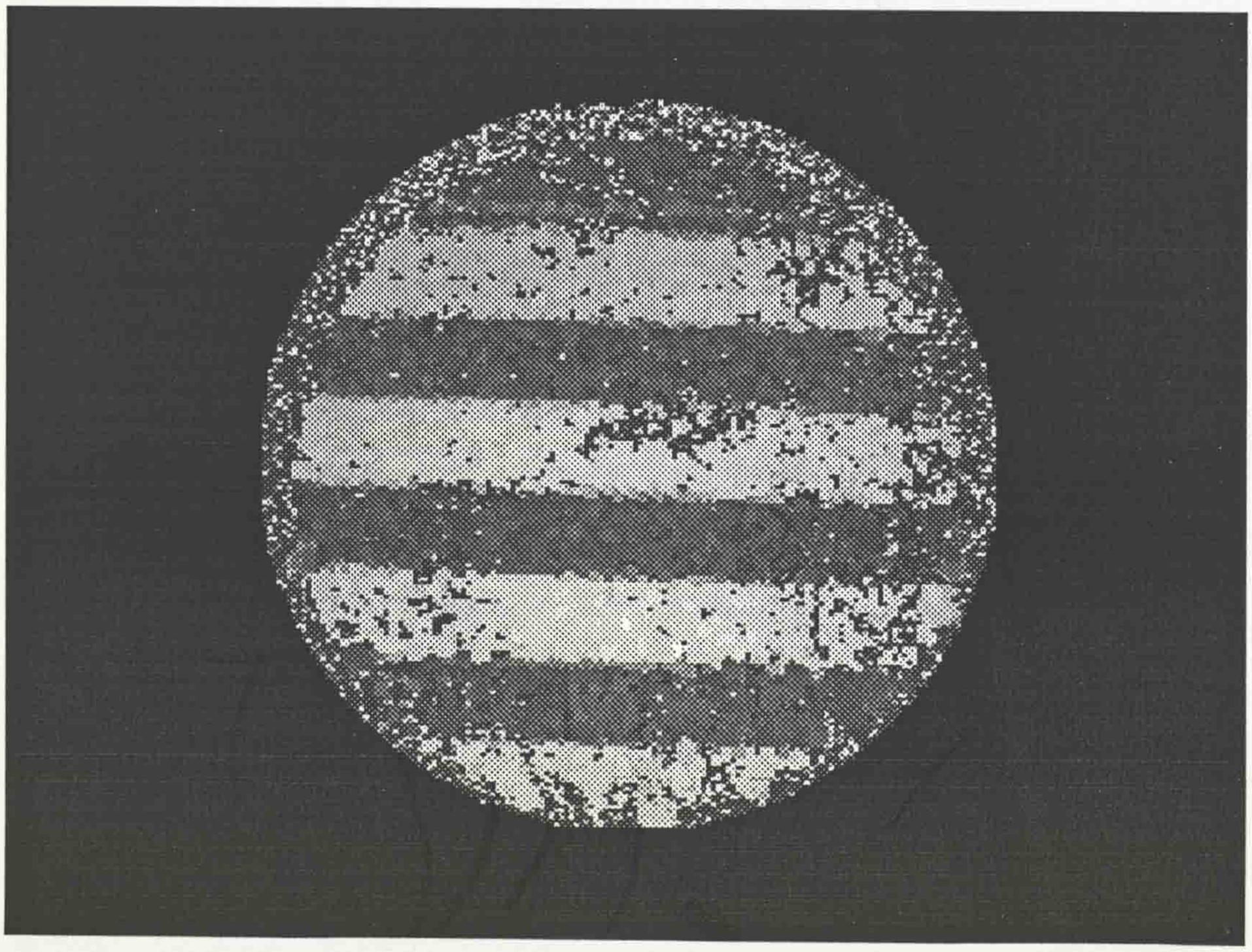


Image 7.21 (b) - The output as above reconstructed through a LUT obtained by spot calibration

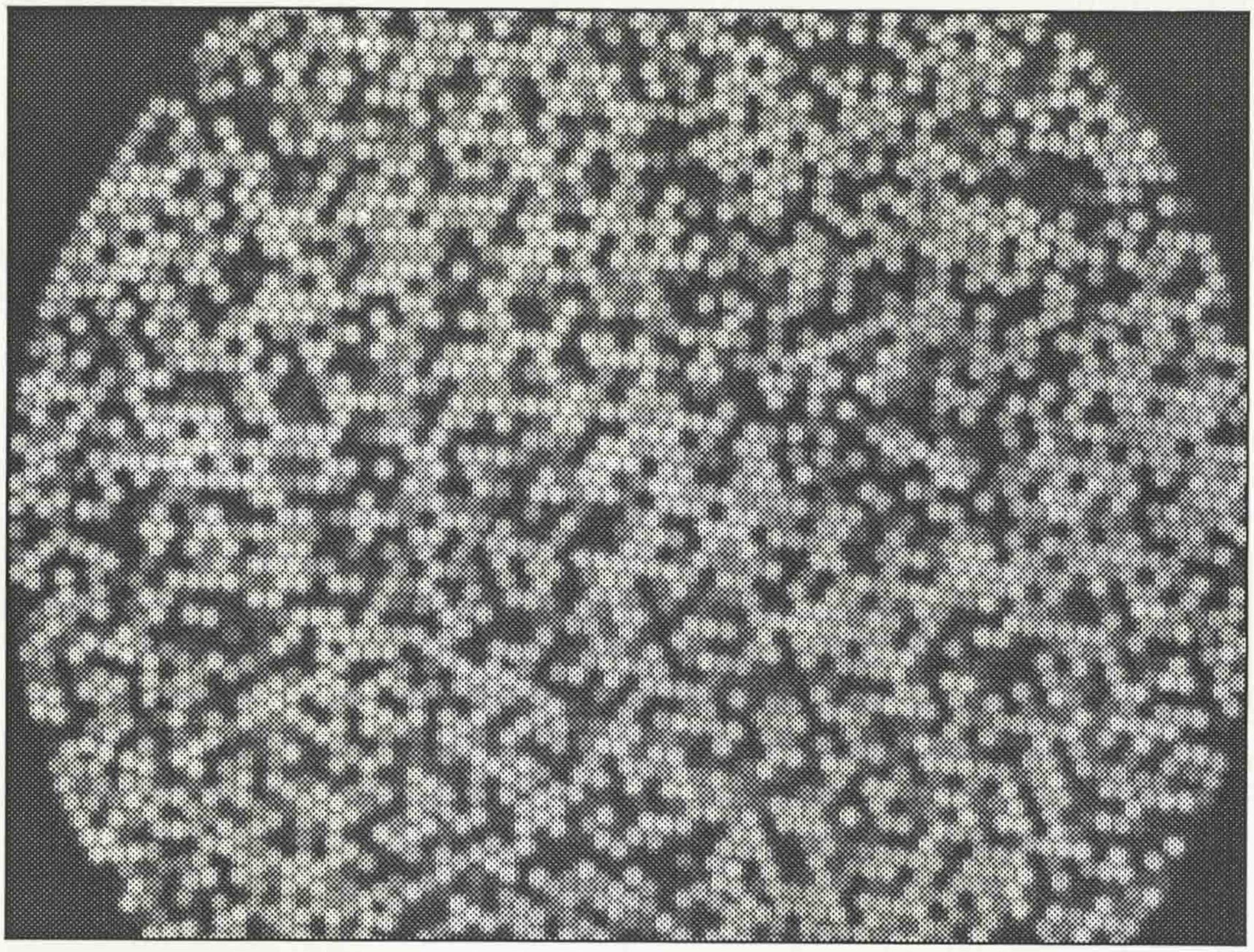


Image 7.22 (a) - The output of the incoherent bundle when transmitting a low order test pattern

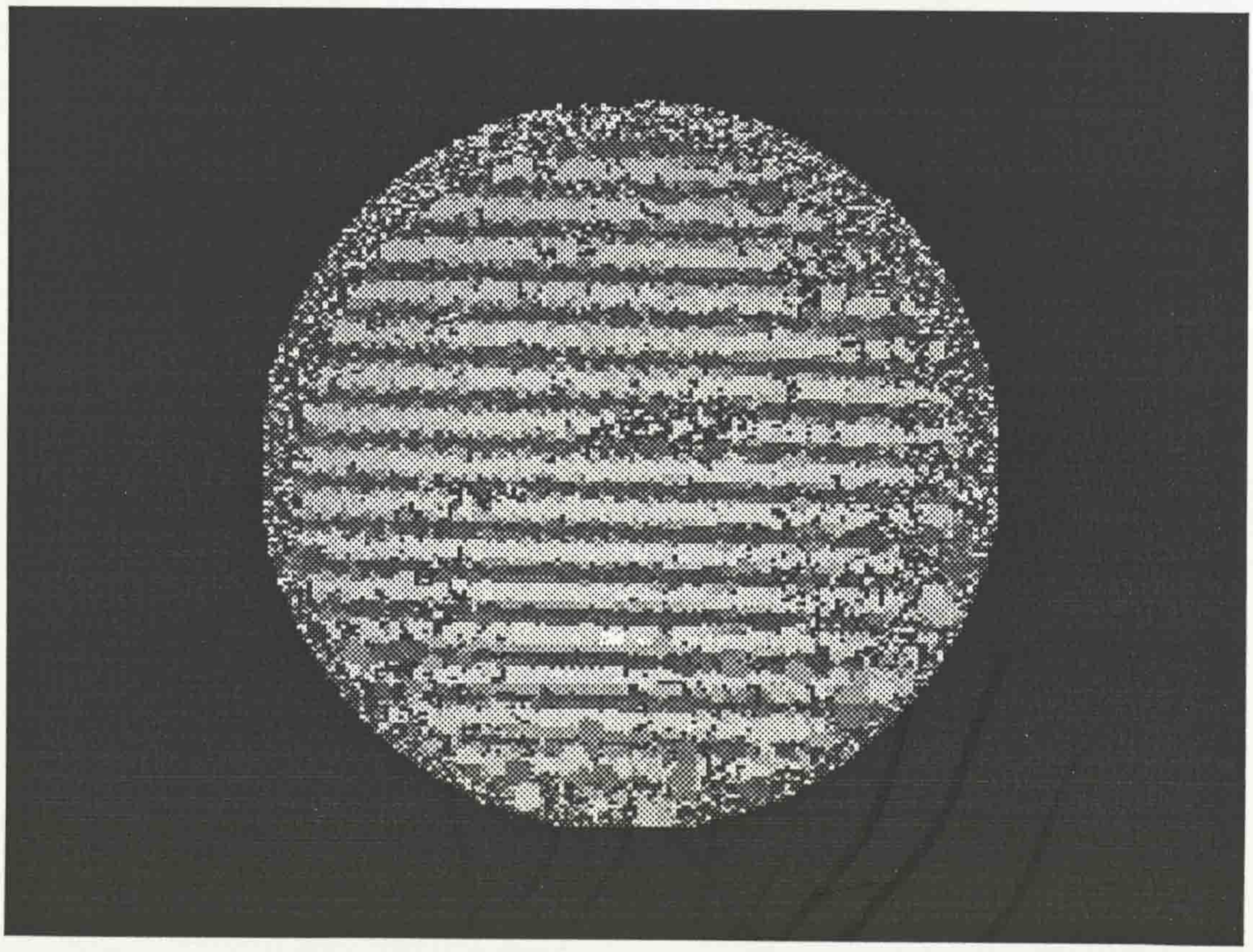


Image 7.22 (b) - The above output reconstructed through a spot calibration LUT

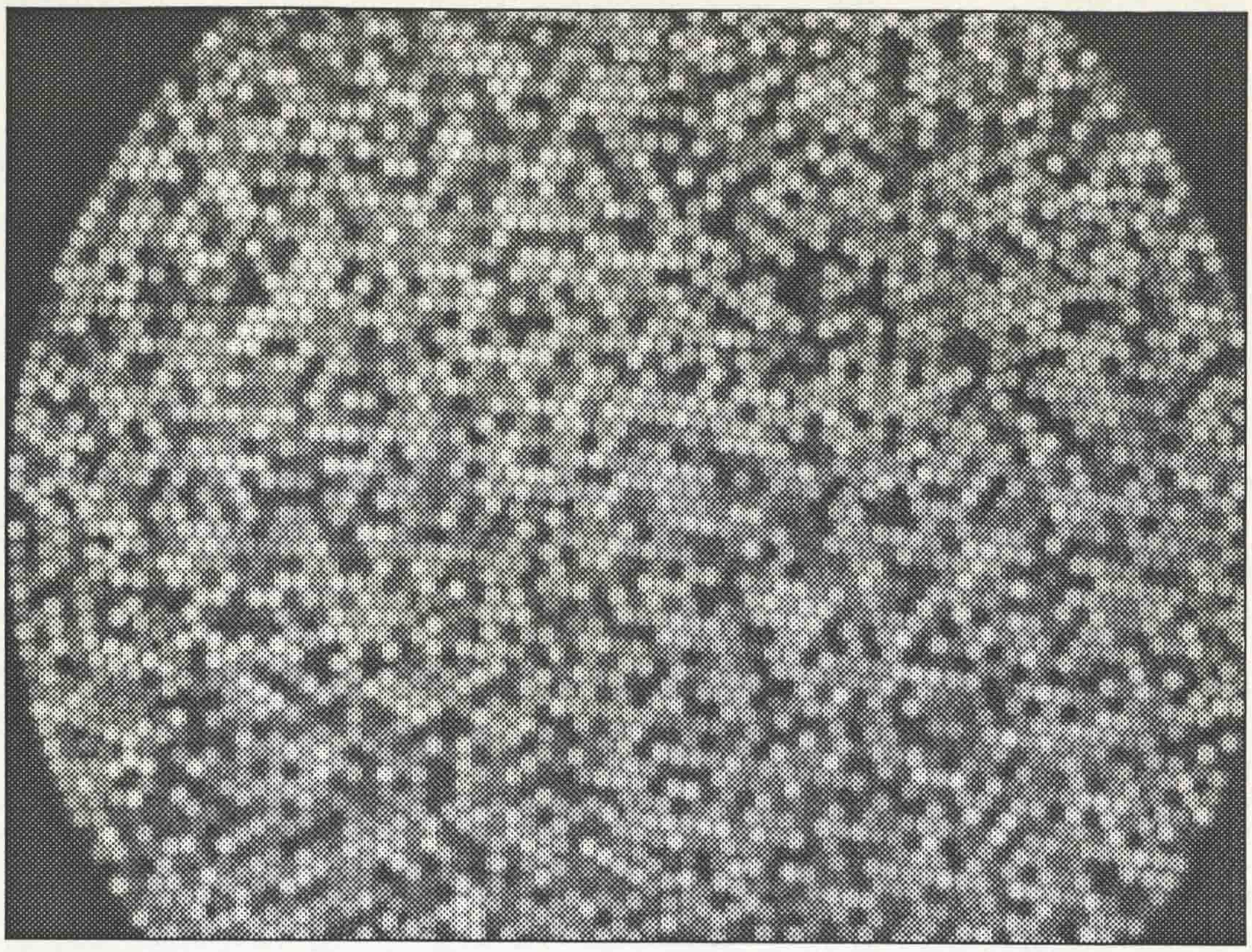


Image 7.23 (a) - The output of the incoherent bundle when transmitting the test pattern for bit 1 (Numbered 0 to 7)

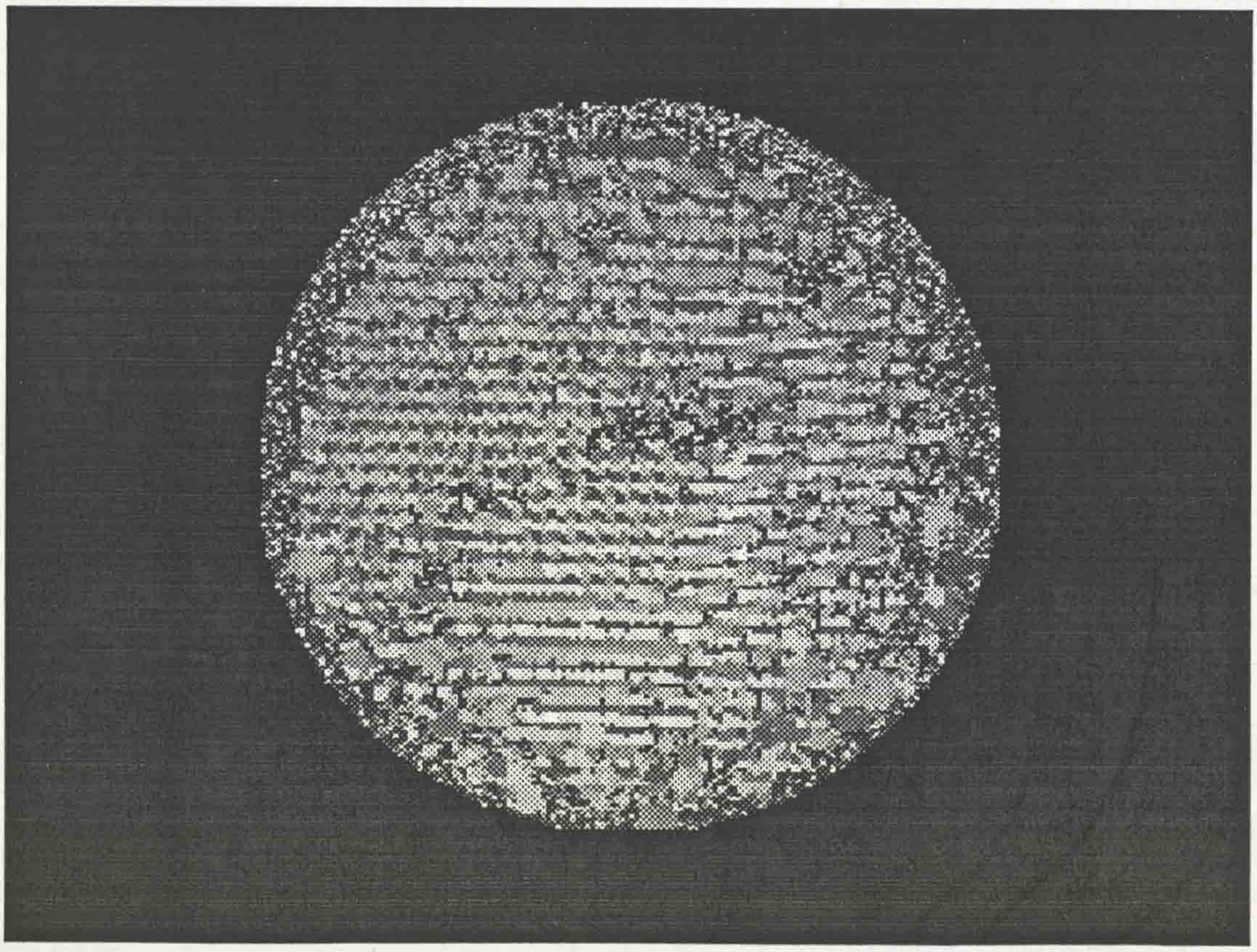


Image 7.23 (b) - The reconstructed output of the above image.

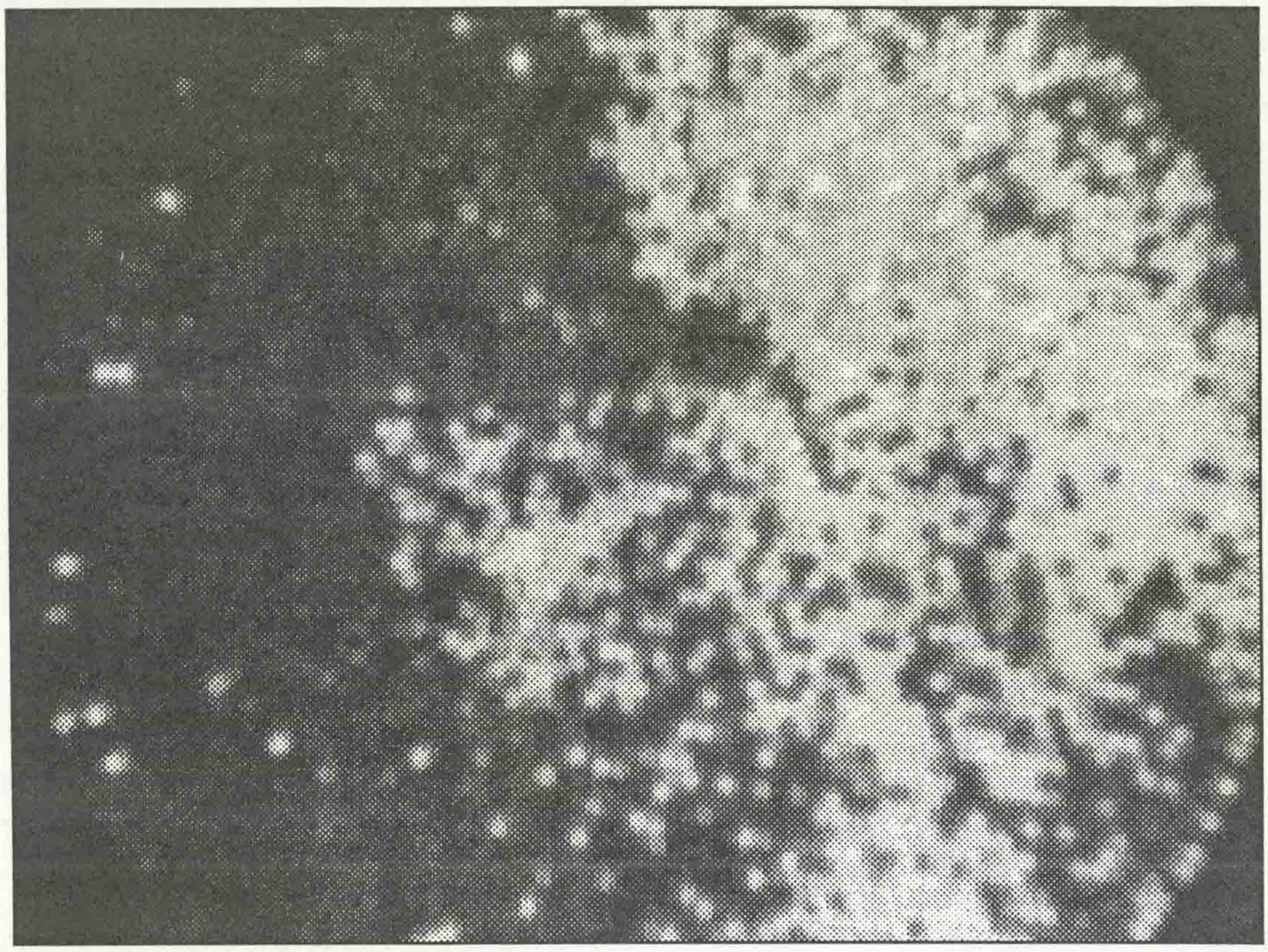


Image 7.24 (a) - The output of the incoherent bundle when transmitting the test pattern for bit 7

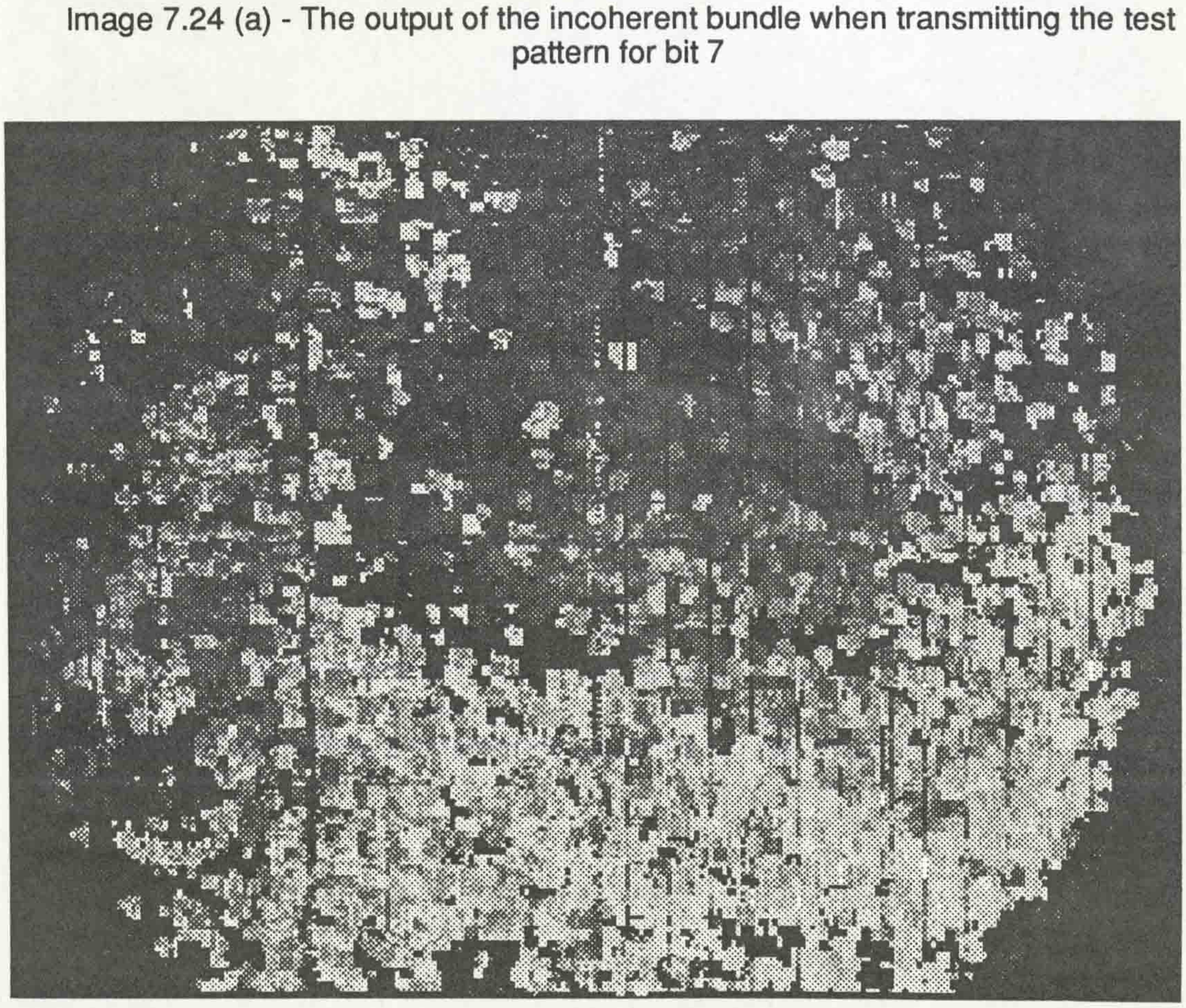


Image 7.24 (b) - The above output reconstructed through a LUT obtained by test pattern calibration

7.4 Discussion

The results presented in this chapter show that is possible to calibrate an incoherent optical fibre bundle for the transmission of images by non-modulated light. The two methods implemented have different strengths and weaknesses. The spot calibration method is very accurate, but takes a longer time, the length of which is directly proportional to the number of fibres in the bundle, and is therefore suitable for low resolution optical fibre bundles. The test pattern calibration is capable of calibrating high resolution bundles, but its use for low resolution bundles is limited. The time taken for a test pattern calibration is independent of the resolution of the bundle. It is however, susceptible to a greater degree of error, based on the location, illumination and display accuracy of the test patterns, and the processing operations to determine which pixels are illuminated by the test patterns.

The issue of the adequacy of resolution of the bundles will of course be dependent on the intended application. The necessary resolutions for various applications are examined by Ahuja et al [*99*]. There are, at present, several methods available for the improvement of the perceived image from a low resolution or noisy original [*100 - 102*]. The methods presented have been well documented, but they were not implemented in this thesis for two primary reasons. The first is that the major effort was expended on developing the calibration procedures to the highest degree of accuracy possible, and seeking to improve the results by such means. The second is the target of pursuing real-time reconstruction of the optical fibre bundles' output to provide a video image. The use of the complex image restoration/enhancement algorithms available would require much more sophisticated hardware systems to achieve real-time images. Plates 7.2 and 7.3 show an incoherent and coherent bundle transmitting the same image, for a direct comparison of their outputs.

The quality of images from the calibrated low resolution bundle show that it is particularly suited to low complexity image transmission, such as text. The necessary requirements for the transmission of text by low resolution systems is examined by Stringa [103]. The quality of images obtained by the test pattern calibration is limited by the particular errors inherent in each source for the test patterns, as well as the limited resolution of the bundles, particularly the incoherent bundle. Although the coherent bundle was of a higher elemental resolution than the OFCS (100,000 versus 65536 picture elements), the perceived resolution is less for the optical fibre bundle. The reasons for this loss of resolution is due to the irregularity of the arrangement of the fibres. This feature is discussed in more detail in [*29*], and by Silverman et al in [*104*].

The major source of error for the test pattern calibration was inaccurate definition of the test patterns. For the CRT source, this was a result of the changing image sizes and intensities between test screens, although the overall location of the test patterns was accurate. The photographic test slide offered good illumination and image definition, but were subject to location errors between test patterns. The plasma display offered the best solution, potentially, but was hampered by the low overall light output, and the unsynchronised cycling of the image intensities. The low light output required large lens aperture to provide an adequate level of differentiation between the light and dark areas of the test patterns. This resulted in a loss of depth of focus, blurring the image slightly, making the detection of the edges in the test pattern inaccurate. The final choice was a compromised setting for the aperture and focusing, and using image processing techniques to improve the contrast range of the acquired image.

Examining the results for the different test pattern calibrations shows that the photographic slides gave the best results, with less high bit error than the CRT method, and better low bit accuracy than the plasma display. The OFCS performed as it was designed to, but was subject to reliability malfunctions, primarily due to the method of wiring used. The clock generation circuitry (section 5.3.4) used for the video data system was subject to fluctuating cycle times as the voltage supply output varied, with varying current demands. This is due to the changing signal threshold levels for the devices used as the source input voltage to each device varied. This effect manifested itself as a slight displacement of some pixels, as other systems, driven by the same power supply altered their current demands. This problem could be easily solved by the use of a more stable power supply and/or isolating the power supply for the OFCS from other systems.

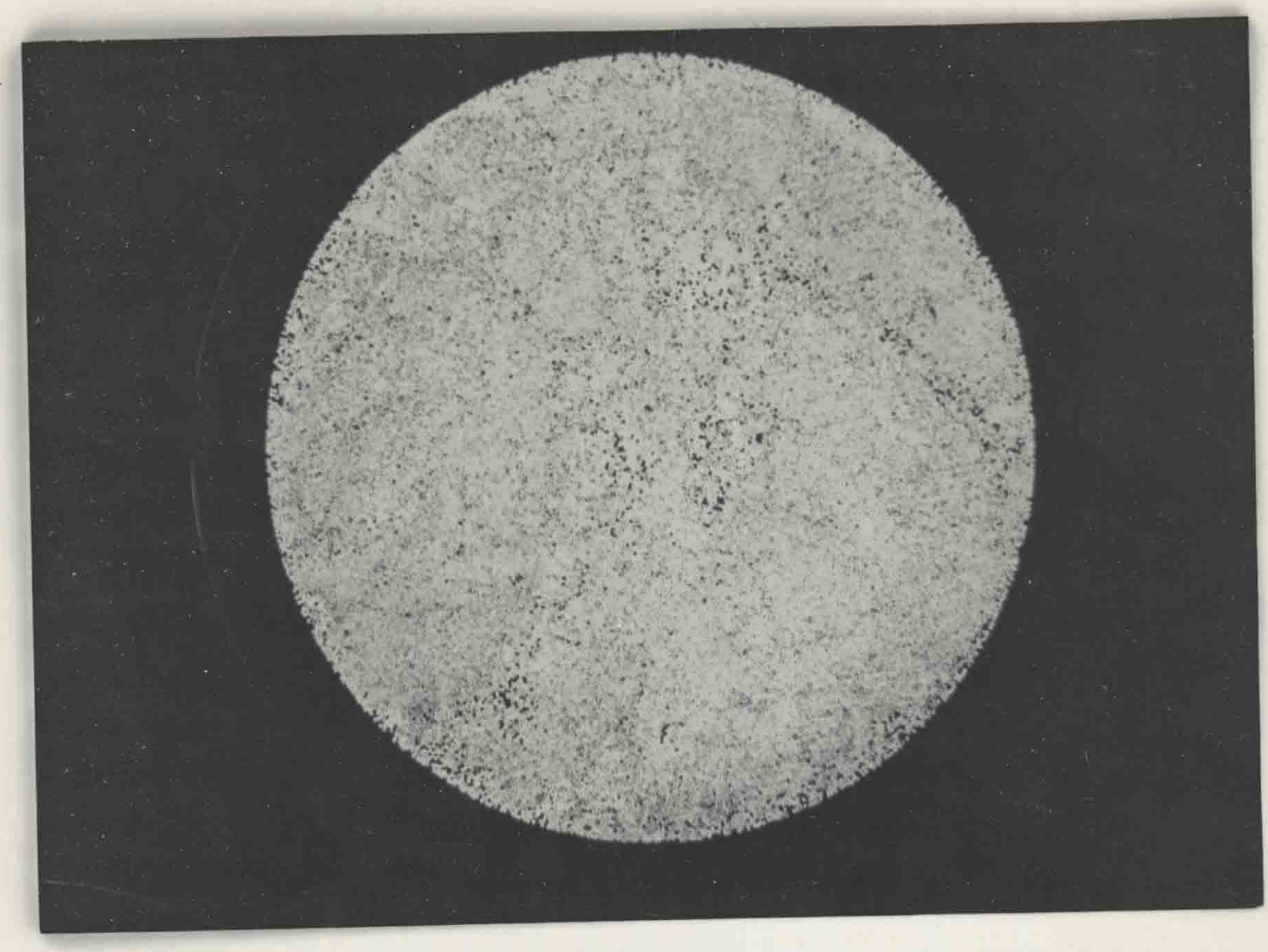


Plate 7.2 - The Output of an Incoherent Bundle when Transmitting Text

Plate 7.3 - The Output of a Coherent Bundle when Transmitting Text

8. FURTHER WORK AND CONCLUSIONS

This final chapter describes possible applications for the use of calibrated incoherent optical fibre bundles, and outlines further work that may be carried out to exploit the calibration methods described in this thesis. It also draws conclusions from the work and results presented, with a view to summarising the achievements of this project and outlining areas for further work.

8.1 Applications

To date, optical fibres have found widespread use in the field of digital telecommunication, and to a lesser extent, endoscopy, using bundles of coherent fibres to carry the image in non-modulated light form, and incoherent bundles for illumination of the scene to be viewed. The applications of calibrated incoherent bundles are seen to be as an inexpensive replacement for coherent bundles, while providing the availability of long lengths of optical fibre bundles for image transmission.

Endoscopes find widespread use in the medical and veterinary fields for internal inspection of patients. In the medical field, the endoscopes are made up of coherent optical fibre bundles 1 to 1.5 m long. In the veterinary field, the bundles are approximately 1 m longer, for internal inspection of large animals, such as horses. These optical fibre bundles are usually of very high resolution, of the order of 250,000 10 μm single mode fibres. Such resolutions are necessary to provide an accurate and clear image of the scene. However, with the advent of the CCD, electrical endoscopes are finding increasing use. In this type of endoscope, a small CCD is used as the imaging source at the input end of the endoscope.

For industrial applications, endoscopes find use in the remote inspection and testing field. The environments tend to be more hostile than in the medical field, with high levels of electromagnetic radiation, possibly heat and nuclear radiation. In such environments, an electrical endoscope would be susceptible to a greater degree of noise than an optically based endoscope. In addition in areas of relatively high nuclear radiation, the fibres are subject to a greater degree of transmission losses [*28*], and after prolonged exposure, suffer permanent damage, rendering the bundle useless for image transmission. It such applications, the replacement of coherent

bundles would prove expensive. With the OFCS, the calibrated incoherent bundle can be easily and more cheaply replaced with another, and the new calibration data set for the replacement.

The possibility of providing very long lengths of calibrated incoherent bundles for use as endoscopes, would allow the visual inspection and monitoring of industrial processes and scenes that were not possible before. Possible applications for endoscopes are presented in [*19,28,105*]. In some monitoring applications, such as explosives handling, the presence of an electrical signal near the subject would be undesirable. For such applications, the use of a long length of calibrated incoherent optical fibre bundle would enable the source image to be transmitted to a remote point before being converted to an electrical signal by video camera. The hardware design allows video images of the subject to be displayed, allowing continuous monitoring of the scene through the incoherent bundle. Thus, the possible areas of application for calibrated incoherent optical fibres are seen to be as replacements for coherent bundles, areas where long lengths of bundles are required for image transmission, and areas where the optical fibre bundles are damaged by the environment in which they are used,, making the use of coherent bundles very expensive.

8.2 Further Work

The calibration methods developed in this thesis have two disadvantages: the spot calibration method provides its best results with low resolution bundles, and takes a relatively long time to be executed, and the test pattern method works with high resolution fibre bundles and takes a relatively short time, but is subject to a high degree of error. If the intended use of the calibrated bundle requires a high resolution, then the test patterns must be used. If a low resolution is adequate, then the spot calibration method could be used.

Further work on this project should centre on improving the accuracy of the test pattern calibration method, as for most applications a high resolution would be desirable. The photographic test slides provide the best basis for the calibration, and the work should involve ensuring better location of the slides. The optical equipment used in these trials was not of the utmost quality, and better lens and focusing systems could be used.

The hardware reconstruction system function correctly, but for improved reliability and lower electromagnetic interference, a printed circuit board design should ideally be used. To reduce the calibration time, and to expand the possible applications of the hardware system, the data communication to and from the host transputer should be carried out on the data and address lines of the transputer, and not the links. It is also suggested that the resolution of the video system be improved to 512 pixels by 512 lines, to make maximum use of the high resolution optical fibre bundles.

Currently, the availability of high resolution, incoherent optical fibre bundles is limited, being made only to order, at a relatively high price, destroying the cost advantage of incoherent bundles. It is envisaged, that with a completely developed and marketable calibration system, optical fibre bundle manufacturers would take up production of high resolution incoherent optical fibre bundles, in long lengths.

8.3 Conclusions

It has been shown that the calibration of incoherent optical fibre bundles for the transmission of images by non-modulated light transmission is possible. Two methods of calibration were presented, both achieving results to different levels of success. The reconstruction of the output of the fibre bundles was achieved in real-time, subject to a delay time of one field period of 20 ms, by a specifically designed hardware system. This hardware system was also used for the calibration system trials, in prototype form.

It was realised, through experimentation, that the resolution of the optical fibre bundle played a very important role in the quality of the images that could be obtained from the calibration system. The spot calibration method provided accurate results with low resolution optical fibre bundles, where the single fibres were 5 times larger in diameter than the high resolution bundles. The calibration time for the spot method is directly proportional to the number of fibres in the bundle. The time taken for the calibration by single fibre illumination was reduced from approximately 14 hours for a 2.5 mm bundle, using the Data Translation DT2853 frame store, to 6 hours for a 3.5 mm diameter bundle using the OFCS. The 2.5 mm bundle contained approximately 2200 fibres, and the 3.5 mm bundle 4400. The dedicated hardware system is therefore more suitable, as expected, to the calibration procedures requirements.

The calibration by test patterns is more suitable to high resolution optical fibre bundles. An incoherent bundle of the necessary resolution was not available for the calibration trials, and the tests were carried out on a coherent bundle of 100,000 single mode fibres. This allowed the principle to be developed, and potential difficulties overcome, to some extent. The major difficulty of the test pattern method is the accurate and consistent reproduction of the required set of test patterns. The necessary software for image processing and analysis to produce the results have been completely developed. The analysis of the results for the test pattern calibration shows that the photographic slides produces the overall best results. The CRT method of producing the test slides was subject to too much intensity and pattern size variation. The plasma display method provided images that were too dimly illuminated for accurate threshold selection at the low order bit test patterns, and was therefore subject to small errors in location of the intensities, effectively blurring the reconstructed images, and reducing the perceived resolution of the optical fibre bundle.

In conclusion, the test pattern calibration method is capable of calibrating optical fibre bundles of the resolution necessary for high quality images, in an acceptable time. The source of the test patterns needs to be improved to realise the full potential of the system.

REFERENCES

[1] Kerkof, F.; Werner, W.
Television: An Introduction to the Physical and Technical Principles
Phillips Technical Library, 1952, pages 1 - 6, 78 - 80

[2] King, G.J.
A Beginner's Guide to Colour Television
Newnes-Butterworth, pages 1 - 60

[3] Amos, S.W.; Birkinshaw, D.C.
Television Engineering: Principles and Practice, Volume 1
Iliffe Books Ltd., 1963, pages 66 - 125

[4] Zworykin, E.E.; Morton, G.A.
Television: The Electronics of Image Transmission
John Wiley and Sons Inc., 1940, pages 91 - 122

[5] Chambers, S.P.
TIPS: A Transputer Based Real-Time Image Processing System
Ph.D. Thesis
University of Liverpool, December 1990

[6] PIPE: Parallel Image Processing Engine
Akebia Ltd., 1989

[7] Data Translation DT2853 User Manual
Data Translation, Inc
Document UM-06548-A-2905, 1987, pages 5-1 - 5-30, 6-1 - 6-4, A-1 - A-3

[8] PIP-1024B Hardware Manual, Revision 3
Matrox Electronic Systems
September 1986

[9] Rosenfeld, A.; Kak, A.C.
Digital Picture Processing, Second Edition, Volume 1
Academic Press, 1982, pages 4 - 8

[10] Wang, D.C.C.; Vagnucci, A.H. and Li, C.C.
Digital Image Enhancement: A survey
Computer Vision, Graphics and Image Processing
Volume 24, Number 3, December 1983, pages 363 - 381

[11] Illingworth, J.; Kittler, J.
A Survey of the Hough Transform
Computer Vision, Graphics and Image Processing
Volume 44, 1988, pages 87 - 116

[12] White, R.M.
Automated Industrial Measurement and Inspection Using Image Processing
Ph.D. Thesis
University of Liverpool, February 1991

[13] Alia, G.; Barsi, F. and Martinelli, E.
Angular Spline: A New Approach to the Interpolation Problem in Computer Graphics
Computer Vision, Graphics and Image Processing
Volume 39, Number 1, July 1987, pages 56 - 72

[14] UK Patent Application GB 2 124 054 A
Classification H4F CA S27L S27T1 S53D S61 S83B S89S9
Application Number 8317907, Filed on 1 July 1983
N.V. Philips, The Netherlands

[15] UK Patent Application GB 2 092 859 A
Classification H4F D12X D27L D27T1 D79 D83C DX
Application Number 8203417, Filed on 5 February 1982
American Optical Corporation, USA

[16] UK Patent Application GB 2 128 839 A
Classification H4F D12X D18K D27L D30K D61 D79 D83B D83C DX
Application Number 8229495, Filed on 15 October 1982
Dainichi-Nippon Cables Ltd, Japan

[17] UK Patent Application GB 2 082 012 A
Classification H4F D12M D13A D26G D27L D27S D27T1 D79 D83B D83C DX
Application Number 8118952, Filed on 19 June 1981
Light Optics Ltd., Berks. England

[18] Von Ardenne, M.
Cathode Ray Tubes
Isaac Pitman and Sons Ltd., 1939

[19] Wolf, H.F.
Handbook of Fiber Optics : Theory and Applications
Granada Publishing Limited, 1979, pages 429 - 464

[20] Advani, M.P.; Georghinades, C.N.
Jointly Optimal Receivers for Optical Pulse-position Modulation Channels
IEEE Transactions on Communications
Volume 38, Number 2, February 1990, pages 179 - 186

[21] Fibre Optic Light Guides, Document No. ML 4145
Pilkington Electro-optic Materials Ltd., 1988

[22] Midwinter, J.E.
Optical Fibers for Transmission
John Wiley and Sons, 1979 pages 1 - 12, 17 - 21

[23] Okoshi, T.
Optical Fibers
Academic Press, 1982 pages 1 - 15, 37 - 42

[24] Gowar, J.
Optical Communication Systems
Prentice-Hall International Series in Optoelectronics, 1984 pages 1 - 25.

[25] Kao, K.C.
Optical Fibre
Peter Peregrinus Ltd, 1988

[26] Senior, J.M.
Optical Fiber Communications : Principle and Practice
Prentice-Hall International Series in Optoelectronics, 1985, pages 11 - 50.

[27] Personick, S.D.
Fiber Optics : Technology and Applications
Plenum Press, 1985

[28] Hill, D.A.
Fibre Optics
Business Books, 1977

[29] Wolf, H.F.
Handbook of Fiber Optics : Theory and Applications
Granada Publishing Limited, 1979 pages 50 - 74

[30] Fibre Optic Light Guides, Document No. ML 4145
Pilkington Electro-optic Materials Ltd., 1988

[31] Kreyszig, E.
Advanced Engineering Mathematics, 6th Edition
John Wiley and Sons, 1988, pages 581 - 718

[32] Opus PC V AT Compatible User's Guide
Opus Technology, 1987

[33] IBM DOS Ver. 3.00
Technical Reference, Document Number 6322677
IBM Corporation, 1984

[34] Technical Reference: Personal Computer AT
IBM Personal Computer Hardware Reference Library
International Business Machines Corporation, 1984

[35] Norton, P.
Programmer's Guide to the IBM PC
Microsoft Press, 1985

[36] Oriel Stepper MIKE Drives: Electrical Specifications
Oriel Scientific Ltd, 1987

[37] Oriel Two Axis Stepper Motor Driver Model 20010
Instruction Manual, Document M-20010
Oriel Scientific Ltd, 1987

[38] Langford, M.
Advanced Photography, Fifth Edition
Focal Press, 1969, pages 15 - 36

[39] Nelkon, M.; Parker, P.
Advanced Level Physics, Fourth Edition
Heinemann Educational Books, 1977

[40] Streetman, B.G.
Solid State Electronic Devices, Second Edition
Prentice-Hall International Inc., 1980, pages 355 - 361

[41] CCD Sensors, Systems and Development Technology
CCD Imaging Databook
Fairchild Weston, 1989, pages 5, 231 - 245, 267 - 269

[42] Pulnix TM-460 Data Sheet
Pulnix America Inc., 1986

[43] Kernighan, B.W.; Ritchie, D.M.
The C Programming Language
Prentice-Hall, Inc, 1978

[44] Wyatt, A.L.
Using Assembly Language
Que Corporation, 1988

[45] Rosenfeld, A.; Kak, A.C.
Digital Picture Processing, Second Edition, Volume 1
Academic Press, 1982 pages 209 - 427

[46] Chellappa, R.; Sawchuk, A.A.
Digital Image Processing and Analysis
Volume 1: Digital Image Processing
IEEE Computer Society Press, 1985

[47] Chellappa, R.; Sawchuk, A.A.
Digital Image Processing and Analysis
Volume 2: Digital Image Analysis
IEEE Computer Society Press, 1985

[48] Maragos, P.; Ziff, R.D.
Threshold Superposition in Morphological Image Analysis Systems
IEEE Transactions on Pattern Analysis and Machine Intelligence
Volume 12, Number 5, May 1990, pages 498 - 504

[49] Chochia, P.A.
Image Enhancement using Sliding Histograms
Computer Vision, Graphics and Image Processing
Volume 44, Number 2, November 1988, pages 211 - 229

[50] Kautsky, J.; Nichols, N. and Jupp, D.L.
Smoothed Histogram Modification for Image Processing
Computer Vision, Graphics and Image Processing
Volume 26, Number 3, June 1984, pages 271 - 291

[51] McCollum, A.J.; Bowman, C.C.; Daniels, P.A. and Batchelor, B.G.
A Histogram Modification Unit for Real-Time Image Enhancement
Computer Vision, Graphics and Image Processing
Volume 42, pages 387 - 398, 1988

[52] Kittler, J.; Illingworth, J. and Foglein, J.
Threshold Selection based on a Simple Image Statistic
Computer Vision, Graphics and Image Processing
Volume 30, Number 2, May 1985, pages 125 - 147

[53] Sahoo, P.K.; Soltani, S.; Wong, A.K.C. and Chen, Y.C.
A Survey of Thresholding Techniques
Computer Vision, Graphics and Image Processing
Volume 41, Number 2, February 1988, pages 233 - 260

[54] Nakamura, A. and Kunio, A
Digital Circles
Computer Vision, Graphics and Image Processing
Volume 26, Number 2, May 1984, pages 242 - 255

[55] Amir, I.
Algorithm for Finding the Center of Circular Fiducials
Computer Vision, Graphics and Image Processing
Volume 49, Number 3, March 1990, pages 398 - 406

[56] Press, W.H.; Flannery, B.P.; Teukolsy, S.A. and Vetterling, W.T.
Numerical Recipes in C: The Art of Scientific Computing
Cambridge University Press, 1988

[57] Liu, Z.; Caelli, T.
A Sequential Adaptive Recursive Filter for Image Restoration
Computer Vision, Graphics and Image Processing
Volume 44, 1988, pages 332 - 349

[58] Stremler, F.G.
Introduction to Communication Systems, Second Edition
Addison-Wesley Publishing Company, 1982, pages 71 - 149

[59] Winograd, S.
On Computing the Discrete Fourier Transform
Mathematics of Computation
Volume 32, Number 32, January 1978, pages 175 - 199

[60] Auslander, L.; Ephraim, F.; Winograd, S.
New Algorithms for the Multidimensional Discrete Fourier Transform
IEEE Transactions on Acoustics, Speech and Signal Processing
Volume ASSP-31, Number 2, April 1983, pages 388 - 403

[61] Lee, S.U.; Chung, S.Y.
A Comparative Study of Several Global Thresholding Techniques for Segmentation
Computer Vision, Graphics and Image Processing
Volume 52, Number 2, November 1990, pages 171 - 190

[62] Tang, G.Y.; Lien, B.
Region Filling with the use of the Green Theorem
Computer Vision, Graphics and Image Processing
Volume 42, pages 297 - 305, 1988

[63] Basu, A.; Brown, C.M.
Algorithms and Hardware for Efficient Image Smoothing
Computer Vision, Graphics and Image Processing
Volume 40, Number 2, November 1987, pages 131 - 146

[64] Stremler, F.G.
Introduction to Communication Systems, Second Edition
Addison-Wesley Publishing Company, 1982, pages 657 - 660

[65] Winkel, D.; Prosser, F., The Art of Digital Design: An Introduction to Top Down Design
Prentice-Hall, 1980

[66] CUPL User's Manual (Rev 3.0)
Assisted Technology, CAD Systems INC
1989, pages 2-1 - 2-4, C-1 - C-29

[67] Programmable Logic Handbook, Document No. AI-RRD-50M-11/86-0
Advanced Micro Devices, 1986

[68] ZL30A Device Support (Rev 30A35)
Stag Electronic Designs Ltd, 1989

[69] Cypress Semiconductors CMOS Databook
Cypress Semiconductor Corporation, January 1988

[70] Altera Databook
Altera Corporation, 1987, pages 2-38 - 2-46

[71] Horowitz, P.; Hill, W.
The Art of Electronics
Cambridge University Press, 1980, pages 331 - 334

[72] The Transputer Databook, Second Edition
Inmos Ltd
Inmos Databook Series, 1989

[73] Pountain, D. ; May, D.
A Tutorial Introduction to OCCAM Programmming
(Including Language Definition)
Inmos Ltd., 1988

[74] The Transputer Applications Notebook: Architecture and Software
Inmos Ltd
Inmos Databook Series, 1989

[75] The Transputer Applications Notebook: Systems and Performance
Inmos Ltd
Inmos Databook Series, 1989

[76] The TDS User's Reference
Inmos Ltd
Inmos Databook Series, 1988

[77] LM1881 Data Sheet, Document No. DS001875/10M17
National Semiconductors Corporation, 1987

[78] Farnell Electronic Components
Sales Catalogue
October 1990 - March 1991, page 645

[79] Hitachi Data Sheets: Hitachi High Speed & Low Power A/D D/A Converters
Catalogue No. CS-E494
Hitachi Ltd., 1987,, pages 9 - 15

[80] Application Notes: HA19209, HA19210, HA19211, HA19212, Rev 1.0
Hitachi Ltd., 1987

[81] Hitachi IC Memory Data Book
Hitachi Ltd., 1987, pages 143 - 150

[82] The Inmos Graphics Databook, Second Edition
Inmos Ltd
Inmos Databook Series, 1989

[83] Apple LaserWriter and LaserWriter Plus User's Guide
Apple Computer Inc, 1986, pages 120 - 123

[84] PostScript Language Reference Manual
Adobe Systems Incorporated
Addison-Wesley Publishing Company Inc., 1985

[85] Vinten Fibre Optic Image Bundles, UHR and 3G Type
Technical Specifications
Vinten Scientific Systems Ltd, 1984

[86] Sherr. S.
Electronic Displays
John Wiley and Sons, 1979

[87] Kuehn, R.L.
Display Systems Engineering
McGraw-Hill Book Company, 1968

[88] O'Gorman, L.
A Note for Histogram Equalization for Optimal Intensity Range Utilization
Computer Vision, Graphics and Image Processing
Volume 41, Number 2, February 1988, pages 229 - 232

[89] Wu, L.; Zhaohui, X.
Scaling Theorems for Zero-crossings
IEEE Transactions on Pattern Analysis and Machine Intelligence
Volume 12, Number 1, January 1990, pages 46 - 54

[90] Bumbaca, F.; Smith, K.C.
A Practical Approach to Image Restoration for Computer Vision
Computer Vision, Graphics and Image Processing
Volume 42, Number 2, May 1988, pages 220 - 233

[91] Hunt, D.J.; Nolte,L.W; Reibman, A.R.
Hough Transform and Signal Detection Theory Performance for Images with Additive Noise
Computer Vision, Graphics and Image Processing
Volume 52, Number 3, December 1990, pages 386 - 401

[92] Hertz, L.; Schafer, R.W.
Multilevel Thresholding Using Edge Matching
Computer Vision, Graphics and Image Processing
Volume 44, Number 3, December 1988, pages 279 - 295

[93] T5200 Portable Computer Reference Manual
Toshiba Corporation, 1988

[94] Microsoft C 5.1 Optimizing Compiler
Document Number 410840017-500-R04-0887
Run-Time Library Reference, 1989, pages 48 - 55

[95] Chitrasert, B; Rao, K.R.
Human Visual Weighted Progressive Image Transmission
IEEE Transactions on Communications
Volume 38, Number 7, July 1990, pages 1040 - 1044

[96] Kapur, J.N.; Sahoo, P.K. and Wong, A.K.C.
A New Method for Grey-level Picture Thresholding using the Entropy of the Histogram
Computer Vision, Graphics and Image Processing
Volume 29, Number 3, March 1985, pages 273 - 285

[97] Abutaleb, A.S.
Automatically Thresholding of Grey-level Pictures Using Two Dimensional Entropy
Computer Vision, Graphics and Image Processing
Volume 47, Number 1, July 1989, pages 22 - 32

[98] Algie, S.H.
Resolution and Tonal Continuity in Bilevel Printed Picture Quality
Computer Vision, Graphics and Image Processing
Volume 24, Number 3, December 1983, pages 329 - 346

[99] Ahuja,N.; Tuceryan, M.
Extraction of Early Perceptual Structure in Dot Patterns: Integrating Region, Boundary and Component Gestalt
Computer Vision, Graphics and Image Processing
Volume 48, 1989, pages 304 - 356

[100] Andrews, H.C.
Monochrome Digital Image Enhancement
Applied Optics, The Optical Society of America
Volume 15, February 1976, pages 495 - 503

[101] Narayanan, K.A.; Rosenfeld, A.
Image Smoothing by Local Use of Global Information
IEEE Transactions on Systems, Man and Cybernetics
Volume SMC-11, Number 12, December 1981, pages 826 - 831

[102] Goshtaby, A; Chung, S.Y.; Barsky, B.A.
B-Spline Curves and Surfaces Viewed as Digital Filters
Computer Vision, Graphics and Image Processing
Volume 52, Number 2, November 1990, pages 264 - 275

[103] Stringa, L.
A New Set of Constraint-free Character Recognition Grammars
IEEE Transactions on Pattern Analysis and Machine Intelligence
Volume 12, Number 3, December 1990, pages 1210 - 1217

[104] Silverman, B.W.; Jennison, C.; Stander, J.; Brown, T.C.
The Specification for Edge Penalties for Regular and Irregular Pixel Images
IEEE Transactions on Pattern Analysis and Machine Intelligence
Volume 12, Number 10, October 1990, pages 1017 - 1024

[105] Hull, B.; John, V.
Non-Destructive Testing
Macmillan Education Ltd., 1988, pages 126 - 128

APPENDICES

APPENDIX A

SPOT CALIBRATION SOFTWARE FOR DT2853

DT2853 Spot Calibration

Calib_2.c :

```
   /*************************************************************************/
   /* NAME : Calib.C                                                        */
   /* To calibrate an optical fibre bundle                                  */
   /* This requires the fibre's parameters to be entered at run time        */
   /*************************************************************************/

#include "\lc\stdio.h"
#include "\lc\math.h"

unsigned _stack = 60000;

#define up    1          /* Definitions looking into fibre */
#define right 4          /* from pinhole                   */
#define down  3
#define left  2

main()
{
   /* Oriel controlling functions */

   extern void com_set(),go(),move(),enable(),disable();
   extern int chord();
   extern int *strg_cha();

   /* Frame Store functions   */

   extern int allocate(),free_mem();
   extern void mov_buf();
   extern char readpixl();
   extern void camera(),capture();

   /* Files  */

   FILE *dfp,*fopen();
   char datafile[50];


   /* Variable declarations   */

   unsigned int error_code;
   unsigned int segment;

   /* For Motor control*/

   double radius;
   float diameter;
   unsigned int steps,next,travel,bottom,forward,back,count,offset;
   int *go_val_cha;
   unsigned int go_value,one,two,three,four;
   unsigned int x_in,y_in,mid_x_in;

   /* For searching frame  */

    unsigned int i,j;
   int threshold;
   unsigned int x,y,line_start,column_start,first_line;
   unsigned char pixel_value;
   unsigned int pixels_on_line[100][100],lines[50],number_of_lines;
   unsigned int number_of_pixels_on_line[50],check_on_y,last_line,new_line;
   unsigned int sum,average,number_of_pixels,count_of_pixels_on_line;

   /* For Time   */

   unsigned char *clock;
   unsigned char start_time_min,start_time_sec,stop_time_min,stop_time_sec;
   unsigned char start_time_hour,stop_time_hour;
   unsigned int hours,minutes,seconds;

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

   /* Allocate 256Kb of system memory    */

segment = allocate(256);

error_code = system("echo off");
error_code = system("cls");

printf("\n\n\tEnter Storage [path] : file for data : ");
scanf("%40s",datafile);
dfp = fopen(datafile,"w");
   if (dfp == NULL)
   {
      printf("\x07\x07\tERROR opening FILE : %s\n\n",datafile);
      exit(1);
```

```
    }
printf("\n\n\tEnter diameter of fibre (mm) : ");
scanf("%f",&diameter);

printf("\n\n\tPixel brightness threshold : ");
scanf("%d",&threshold);


count = 24;                          /*  No of motor steps*/
radius = (diameter*1000.0)/2.0;
bottom = ((radius*2)/count)+1;
radius = ((bottom/2)*count);
mid_x_in = bottom/2;

    /*********************************/
    /*  Calibration starts HERE !    */
    /*********************************/

    com_set();
    enable('A');
    enable('B');

    camera(0,0,0,0); /*  Display    */

    go_value = (bottom/2);  /*  Work out starting position for Y  */
    go_val_cha = strg_cha(go_value,count);
    one = *(go_val_cha+3);
    two = *(go_val_cha+2);
    three = *(go_val_cha+1);
    four = *go_val_cha;
    go('B','-',one,two,three,four);    /* Motor to start position */

    offset = (chord(1,radius,count)/2);
    go_val_cha = strg_cha(offset,count);
    one = *(go_val_cha+3);
    two = *(go_val_cha+2);
    three = *(go_val_cha+1);
    four = *go_val_cha;
    go('A','-',one,two,three,four);        /* Motor to start position    */

    x_in = ((bottom/2)- offset);
    y_in = 0;

    printf("\n\n Starting parameters :");
    printf("\n\t\tradius\t\t: %f",radius);
    printf("\n\t\tvertical steps\t: %d",bottom);
    printf("\n\t\tx_in\t\t: %d",x_in);
    printf("\n\t\ty_in\t\t: %d",y_in);

    /*  Start stopclock   */

    getclk(clock);
    start_time_hour = *(clock+4);
    start_time_min = *(clock+5);
    start_time_sec = *(clock+6);

    printf("\n\t\ttime\t\t: %d:%d:%d",start_time_hour,start_time_min,start_time_sec);

    for (steps = 1; steps <= bottom; steps++)
    {
        disable('B');

        next = steps+1;
        forward = chord(steps,radius,count);
        back = chord(next,radius,count)/2;

        for (travel = 0; travel <= forward; travel++)
        {
            move(left,0,0,2,4);
            x_in++;

            /*      SEARCH current frame     */

            capture(0,0,0,0);
            mov_buf(segment,0);

            column_start = 50;
            line_start   = 0;

            for (y = line_start; y <= 474; y+=2)
            {
                for(x = column_start;  x <= 426; x+=2)
                {
                    pixel_value = readpixl(x,y,segment);
                    if (pixel_value > threshold)
                    {
                        if (y > 0)
                            first_line = y-1;

                        number_of_lines = 0;
                        number_of_pixels = 0;
                        sum = 0;
                        check_on_y = first_line;
                        last_line = first_line-1;

    for (y = first_line; y <= 474; y++)
    {
        new_line = y;
        count_of_pixels_on_line = 0;

        for (x = 50; x <= 426; x++)
        {
            pixel_value = readpixl(x,y,segment);

            if (pixel_value > threshold)
            {
                if (new_line > last_line)
                {
                    number_of_lines++;
                    last_line = new_line;
                }
                check_on_y = y;

                count_of_pixels_on_line++;
                number_of_pixels++;
                pixels_on_line[number_of_lines][count_of_pixels_on_line] = x;
                lines[number_of_lines] = y;
                sum += pixel_value;
            }
```

```
                    }

                    if (y != check_on_y)
                        break;

                    number_of_pixels_on_line[number_of_lines] = count_of_pixels_on_line;
                }

                if (sum >= 200)
                {
                    average = sum/number_of_pixels;

                    fprintf(dfp,"\n%d %d %d",x_in,y_in,number_of_lines);
                    for ( i = 1; i <= number_of_lines; i++)
                    {
                        fprintf(dfp,"\n%d %d ",lines[i],number_of_pixels_on_line[i]);

                        for ( j = 1; j <= number_of_pixels_on_line[i]; j++)
                            fprintf(dfp,"%d ",pixels_on_line[i][j]);
                    }
                    fprintf(dfp,"\n%d %d",number_of_pixels,average);
                }
            }
        }
                    /* Alternate starting point for every other line  */
                    if (column_start == 50)
                        column_start = 52;
                    else
                        column_start = 50;
                }
            }

            go('A','+',0,0,0,0);
            go_val_cha = strg_cha(back,count);
            one = *(go_val_cha+3);
            two = *(go_val_cha+2);
            three = *(go_val_cha+1);
            four = *go_val_cha;
            go('A','-',one,two,three,four);

            x_in = mid_x_in - back;     /* Reset x_in to start of this new chord  */

            enable('B');
            move(down,0,0,2,4);
            y_in++;
        }

        fclose(dfp);

        /*  Park motors   */

        enable('A');
        enable('B');

        go('A','+',0,0,0,0);
        go('B','+',0,0,0,0);

        disable('A');
        disable('B');

        /*  Release memory allocated to frame buffer */

        free_mem(segment);

        /*  Calculate elapsed time  */

        getclk(clock);
        stop_time_hour = *(clock+4);
        stop_time_min = *(clock+5);
        stop_time_sec = *(clock+6);

        printf("\n\t\t\tStopped at:

    %d:%d:%d",stop_time_hour,stop_time_min,stop_time_sec);

        if (stop_time_hour < start_time_hour) /*  New Day    */
            hours = start_time_hour - stop_time_hour;
        else
            hours = stop_time_hour - start_time_hour;

        if (stop_time_min < start_time_min)       /* New hour   */
            minutes = ((60 - start_time_min) + stop_time_min);
        else
            minutes = stop_time_min - start_time_min;

        if (stop_time_sec < start_time_sec)       /* New minute */
            seconds = ((60 - start_time_sec) + stop_time_sec);
        else
            seconds = stop_time_sec - start_time_sec;

        printf("\n\n\tElaspsed Time : %d:%d:%d\n",hours,minutes,seconds);

    }
```

Routines.c :

```
/*************************************************************************/
/*  This program contains controlling functions for the oriel steppers   */
/*************************************************************************/
/*  Go(axis,sign,step3,step2,step1,step0)                                */
/*  axis A -- 65 B -- 66;                                                */
/*************************************************************************/

go(axis,sign,stp3,stp2,stp1,stp0)
char axis,sign;
int stp3,stp2,stp1,stp0;

{
    extern sendchar(),com_set();
    extern char recvchar();
    char character,carriage;

    int step3,step2,step1,step0;
    char a[11],b[11];
    int i,k,motor;
    char asc[10];           /* Ascii numbers */
    k = 48;
    for (i = 0; i <= 9; i++)
    {
        asc[i] = k;
        k++;
    }

    motor = 0;

    step3 = asc[stp3];
    step2 = asc[stp2];
    step1 = asc[stp1];
    step0 = asc[stp0];

/*  Initialise instructions          */

    b[0] = 'B';
    b[1] = 'H';
    b[2] = 'G';
    b[3] = sign;
    b[4] = step3;
    b[5] = step2;
    b[6] = step1;
    b[7] = step0;
    b[8] = ',';
    b[9] = 'S';

    a[0] = 'A';
    a[1] = 'H';
    a[2] = 'G';
    a[3] = sign;
    a[4] = step3;
    a[5] = step2;
    a[6] = step1;
    a[7] = step0;
    a[8] = ',';
    a[9] = 'S';

    com_set();

/*  Send instruction */

    if ( axis == 65 )
    {
        for ( i = 0; i <= 9; i++ )
        {
            character = a[i];
            sendchar(character);
        }
        while (motor != 'a')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */

    }

    else if ( axis == 66 )
    {
        for ( i = 0; i <= 9; i++ )
        {
            character = b[i];
            sendchar(character);
        }
        while (motor != 'b')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */
    }

return(motor);
}

/*************************************************************************/
/*  Move(dir,step3,step2,step1,step0)                                    */
/*  dir : 1 -- up; 2 -- down; 3 -- right; 4 -- left                      */
/*************************************************************************/

move(direct,stp3,stp2,stp1,stp0)
int direct;
int stp3,stp2,stp1,stp0;
{
    extern sendchar(),com_set(),disable();
    extern char recvchar();

    char character;
    char motor,carriage;

    int step3,step2,step1,step0;

    char up[10],down[10],left[10],right[10];
    int i,k;

    char asc[10];     /* Ascii numbers */
```

```
    motor = 0;
    k = 48;
    for (i = 0; i <= 9; i++)
    {
        asc[i] = k;
        k++;
    }

    step3 = asc[stp3];
    step2 = asc[stp2];
    step1 = asc[stp1];
    step0 = asc[stp0];

/*  Initialise instructions          */

    up[0] = 'B';
    up[1] = 'H';
    up[2] = 'T';
    up[3] = step3;
    up[4] = step2;
    up[5] = step1;
    up[6] = step0;
    up[7] = ',';
    up[8] = '-';
    up[9] = 'S';

    down[0] = 'B';
    down[1] = 'H';
    down[2] = 'T';
    down[3] = step3;
    down[4] = step2;
    down[5] = step1;
    down[6] = step0;
    down[7] = ',';
    down[8] = '+';
    down[9] = 'S';

    left[0] = 'A';
    left[1] = 'H';
    left[2] = 'T';
    left[3] = step3;
    left[4] = step2;
    left[5] = step1;
    left[6] = step0;
    left[7] = ',';
    left[8] = '-';
    left[9] = 'S';

    right[0] = 'A';
    right[1] = 'H';
    right[2] = 'T';
    right[3] = step3;
    right[4] = step2;
    right[5] = step1;
    right[6] = step0;
    right[7] = ',';
    right[8] = '+';
    right[9] = 'S';

    com_set();

/*  Send instruction */

    switch (direct)
    {
    case 1 :
        for ( i = 0; i <= 9; i++ )
        {
            character = up[i];
            sendchar(character);
        }
        while (motor != 'b')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */
        break;

    case 2 :
        for ( i = 0; i <= 9; i++ )
        {
            character = right[i];
            sendchar(character);
        }
        while (motor != 'a')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */
        break;

    case 3 :
        for ( i = 0; i <= 9; i++ )
        {
            character = down[i];
            sendchar(character);
        }
        while (motor != 'b')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */
        break;

    case 4 :
        for ( i = 0; i <= 9; i++ )
        {
            character = left[i];
            sendchar(character);
        }
        while (motor != 'a')
            motor = recvchar();
        carriage = recvchar();      /* Read carriage return */
        break;

    default :
        break;
    }

return(0);
}

/*****************************************************************************
*   Parks pinhole at 0,0                                    *
```

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

park()
{
extern com_set(),sendchar();
extern char recvchar();
char carriage,motor;

com_set();
enable('A');
enable('B');

go('A','+',0,0,0,0);

       while (motor != 'a')
          motor = recvchar();
       carriage = recvchar();      /* Read carriage return */

go('B','+',0,0,0,0);

       while (motor != 'b')
          motor = recvchar();
       carriage = recvchar();      /* Read carriage return */

disable('A');
disable('B');
}

/**************************************************************************
*   Home to    0,1250                                                    *
**************************************************************************/

home()
{
extern com_set(),sendchar();
char cr,dummy;

cr = 0;

com_set();


go('A','+',0,0,0,0);
    while ( cr == 0 )
    {
       cr = recvchar();
    }
dummy = recvchar();  /* Read carriage return */

go('B','-',0,6,7,5);
    while ( cr == 0 )
    {
       cr = recvchar();
    }

dummy = recvchar();  /* Read carrige return  */

disable('A');
disable('B');
}

/**************************************************************************/
/*  enable(axis)                                                          */
/**************************************************************************/

enable(axis)
char axis;
{
    extern sendchar(),com_set();

    char character;
    char ins[2];
    int i;


    ins[0] = axis;
    ins[1] = 'E';

    com_set();

/*  Send instruction */

    for ( i = 0; i <= 1; i++ )
    {
       character = ins[i];
       sendchar(character);
    }
}

/**************************************************************************/
/*  disable(axis)                                                         */
/**************************************************************************/

disable(axis)
char axis;
{
    extern sendchar(),com_set();

    char character;
    char ins[2];
    int i;


    ins[0] = axis;
    ins[1] = 'D';

    com_set();

/*  Send instruction */

    for ( i = 0; i <= 1; i++ )
    {
       character = ins[i];
       sendchar(character);
    }
}
/**************************************************************************/
/*  This routine calculates the points required to track a circle      */
```

```
/*  chord(i);   -   i = vertical steps from top        */
/*************************************************************************/

#include "\lc\math.h"
int chord(i,radius,count)
int i,count;
double radius;
{
    double x,y;
    int halfway,limit,steps;
    double square;

    halfway = radius/count;

    if ( i >= halfway )
        y = ( (i*count) - radius );
    else
        y = (radius - (i*count));

        square = ( (radius*radius) - (y*y) );
        x = sqrt(square);
        limit = x;
        steps = limit/count;
        steps += 1;
    return(2*steps);
}
/*************************************************************************/

int *strg_cha(number_string,count)
int number_string;
int count;
{
    int number;
    int thou,hun,ten,unit;
    int remain1,remain2,remain3;
    int cha[4];
    int *pcha;

number = number_string*count;
thou = number/1000;
cha[3] = thou;
remain1 = (number - (1000*thou));

hun = remain1/100;
cha[2] = hun;
remain2 = (remain1 - (100*hun));

ten = remain2/10;
cha[1] = ten;
remain3 = (remain2 - (10*ten));

unit = remain3;
cha[0] = unit;

pcha = cha;

return(pcha);
}
```

Coords.c :

```
/*********************************************/
/*  This reads in the calibration data       */
/* and calculates the x,y input positions    */
/*********************************************/

unsigned _stack = 50000;

#include "\lc\stdio.h"
int far x_in[5000];
int far y_in[5000];
int far x_out[5000];
int far y_out[5000];

main()
{

int i,x[5000],y[5000];
int end;
float fltx,flty;
int error_code;
char confirm_in,confirm_out;

int y_in_min,y_in_max,x_in_min,x_in_max;
int x_out_range,y_out_range,x_in_range,y_in_range;
int x_out_min,x_out_max,y_out_min,y_out_max;
float x_step,y_step,mid_x,mid_y;

char readfile_screen[50],writefile_screen[50];
char *readfile,*writefile;
FILE *fopen(),*read_fp,*write_fp;

    /****  Title page****/

error_code = system("cls");
printf("\n\t*****************************************");
printf("\n\t* To calculate the pixel input locations *\n");
printf("\t* from the calibration data              *\n");
printf("\t*****************************************\n\n");

    /*  Open input file  */

printf("\n\tRead from : \\DATA\\OUTPUT1.DAT ? (y/n)");

if ( (confirm_in = getche()) != 'y')
{
    printf("\n\n\tEnter New FILE : ");
    scanf("%40s",readfile_screen);

    read_fp = fopen(readfile_screen,"r");
    if ( read_fp == NULL )
    {
        printf("\x07\x07\tError opening file : %s",readfile_screen);
        exit(1);
    }
}

else
{
    readfile = "\\data\\output1.dat";
    read_fp = fopen(readfile,"r");
    if ( read_fp == NULL )
    {
        printf("\x07\x07\tError opening file : OUTPUT1.DAT\n");
        exit(1);
    }
}

i = 1;
while ( (fscanf(read_fp,"%d",&x_in[i])) != EOF )
{
    fscanf(read_fp,"%d %d %d",&y_in[i],&x_out[i],&y_out[i]);
    end = i;
    i++;
}
fclose(read_fp);

printf("\n\tNumber of entries :  %d\n",end);

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

    y_in_min = 250;
    y_in_max = 0;

    x_in_min = 250;
    x_in_max = 0;

    for ( i=1; i<=end; i++ )
    {
        if (y_in[i] < y_in_min)
            y_in_min = y_in[i];
        if (y_in[i] > y_in_max)
            y_in_max = y_in[i];
        if (x_in[i] < x_in_min)
            x_in_min = x_in[i];
        if (x_in[i] > x_in_max)
            x_in_max = x_in[i];
    }
    printf("\n\tVertical input range   : %d to %d\n",y_in_min,y_in_max);
    printf("\tHorizontal input range : %d to %d\n",x_in_min,x_in_max);

printf("\n\tEnter  VERTICAL  ouput range (min max) : ");
scanf("%d %d",&y_out_min,&y_out_max);

printf("\n\tEnter HORIZONTAL ouput range (min max) : ");
scanf("%d %d",&x_out_min,&x_out_max);

    x_in_range = x_in_max - x_in_min;
    y_in_range = y_in_max - y_in_min;

    x_out_range = x_out_max - x_out_min;
    y_out_range = y_out_max - y_out_min;
```

```
    x_step = x_out_range/x_in_range;
    y_step = y_out_range/y_in_range;
    mid_x  = (x_in_max - x_in_min)/2;
    mid_y  = (y_in_max - y_in_min)/2;

printf("\n\tWrite to : \\Data\\Calib.dat ? (y/n) ");
if ((confirm_out = getche()) != 'y')
{
    printf("\n\n\tEnter New Output FILE : ");
    scanf("%40s",writefile_screen);

    write_fp = fopen(writefile_screen,"w");
    if ( write_fp == NULL )
    {
        printf("\x07\x07\tError opening file : %s",writefile_screen);
        exit(1);
    }
}

else
{
    writefile = "\\data\\calib.dat";
    write_fp = fopen(writefile,"w");
    if ( write_fp == NULL )
    {
        printf("\x07\x07\tError opening file : \\DATA\\CALIB.DAT\n");
        exit(1);
    }
}
printf("\n\tForming LUT data ");

    for (i = 0; i < end; i++)
    {
        fltx = (255.5+(((float)x_in[i]-mid_x)*x_step));
        x[i] = fltx;

        flty = (255.5+(((float)y_in[i]-mid_y)*y_step));
        y[i] = flty;

        fprintf(write_fp,"%3d %3d %3d %3d\n",x[i],y[i],x_out[i],y_out[i]);
    }
fclose(write_fp);
}
```

Support.c :

```
/*      To find the centre of a circlular area       */

int *centre(ypoints,xpoints,linepixs,lines)
int ypoints[75];
int xpoints[75][75];
int lines,linepixs[75];
{
    int i;
    int x_first,x_last;

    int y_max,x_max;
    int y_min,x_min;

    int *pcentre,centre[2];

    y_min = ypoints[1];
    y_max = ypoints[lines];

    x_min = 512;
    x_max = 0;

/*   All array locations start at 1    */

    for ( i = 1; i <= lines; i++)
    {
        x_first = xpoints[i][1];
        x_last = xpoints[i][linepixs[i]];

        if (x_min > x_first)
            x_min = x_first;

        if (x_max < x_last)
            x_max = x_last;

    }

    centre[0] = ((x_max + x_min)/2);
    centre[1] = ((y_max + y_min)/2);

pcentre = centre;

return(pcentre); /*   Return pointer to centre array   */
}

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

/* To decide whether spot is uniform */

sort(y_points,x_points,linepixs,lines)
int y_points[75],x_points[75][75],linepixs[75];
int lines;
{
int i,j,good;
int diff_x,diff_y;

good = 1;
for (j = 1; j <= lines; j++)
{
    if (linepixs[j] > 1)
    {
        for ( i = 2; i <= linepixs[j]; i++)
        {
            diff_x = (x_points[j][i] - x_points[j][i-1]);
            if (diff_x > 1)
                good = 0;
        }
    }
}

if (lines > 1)
{
    for (j = 2; j <= lines; j++)
    {
        diff_y = (y_points[j] - y_points[j-1]);
        if (diff_y > 1)
            good = 0;
    }
}
    return(good);
}
```

The following control access to the Data Translation DT2853.

DT2853.asm :

```
;-------------------------------------------------------------------------
;   This program contains functions which initialise the control registers
;   and Look Up Tables of the DT2853 frame store.It also controls the
;   operation of DT2853 via functions CAMERA, CAPTURE, DISPLAY.
;   DATA transfer is also facilitated between the frame store and programs.
;   It is to be called from a C program which supplies the values for
;   initialisation and control.
;-------------------------------------------------------------------------
;
    TITLE   Board initialisation and data access
    SUBTTL  29/1/88
    NAME dt2853
;
    INCLUDE DOS.MAC
    PUBLIC   read_pix,wrt_pix,camera,capture,show_buf,ilut,olut
;
;NOTE: The base address for control registers has been CHANGED to 310h
;   For details of registers see DATA TRANSLATION manual pp 5-5 to 5-30
;
;   Define register addresses:
;
    base_add equ 310h
    incsr1       equ base_add ; Video Input Control/Status 1
    incsr2       equ base_add + 2 ; Video Input Control/Status 2
    outcsr       equ base_add + 4 ; Video Output Control/Status
    cursor       equ base_add + 6 ; Cursor control
    index        equ base_add + 8 ; Contains the LUT index
    inlut        equ base_add + 0aH   ; Contains LUT entry
    redgrn       equ base_add + 0cH   ; Contains the red & green LUT
    blue     equ base_add + 0eH   ; Contains the blue LUT entry
;
;-------------------------------------------------------------------------
;
;      DATA items now declared and descriptor tables set up.
;   For details of descriptor table see IBM tech. ref. pp 5-149,5-150.
;
    DSEG
;
;   The following is the descriptor table for reading from the frame store
;
        dummy        DQ  0

        gdt_loc      DQ  0

        src_seg_limit    DW  0ffffH
        src_base_add DB  ?,?,?; DT base address = A00000H
;                        To be filled by pointer
        src_dat_ri   DB  93H
        src_dat_res  DW  00H

        tgt_seg_limit    DW  0ffffh
        tgt_base_add DB  ?,?,?; To be filled by pointer
        tgt_dat_ri   DB  93H
        tgt_dat_res  DW  00H

        bios_cs      DQ  0

        temp_ss      DQ  0
;
;
        tgt_data DW  ?     ; Variable to hold pixel value
;
        gdt          DD  dummy ; gdt points (seg,offset) to top
;                         of table
        read_add     DW  src_base_add ; points(offset) to source
;                            base address
        point_tgt        DW      tgt_base_add ; pointer defined for
;                            filling target address
;-------------------------------------------------------------------------
;   Write descriptor table now setup
;
        wdummy       DQ  0
;
        wgdt_loc DQ  0
 ;
        wsrc_seg_limit   DW  0ffffH
        wsrc_base_add    DB  ?,?,?; Equivalent to tgt_base_add
        wsrc_dat_ri  DB  93H
        wsrc_dat_res DW  00H

        wtgt_seg_limit   DW  0ffffh
        wtgt_base_add    DB  ?,?,?; Dt board;To be filled by pointer
        wtgt_dat_ri  DB  93H
        wtgt_dat_res DW  00H

        wbios_cs DQ  0

        wtemp_ss DQ  0
;
;
        write_add    DW  wtgt_base_add ;pointer to DT board
;                            address location
        wgdt         DD  wdummy ; wgdt points (seg,offset) to top
;                         of table
        wrt_scr_add  DW  wsrc_base_add ; points(offset) to source
;                            base address
        tgt_dat_add  DW  ?      ;variable to hold pixel value
;                        address
;-------------------------------------------------------------------------

            ; define variables for control purposes
;
    incl_val DW  ?   ; value in INCSR1
    inc2_val DW  ?   ; value in INCSR2
    out_val      DW  ?   ; value in OUTCSR
;
    ENDDS
;
    PSEG     ; program segment starts
;
; The following functions capture(ip_buff,ILUT,op_buff,OLUT)
; display(buffer,OLUT), camera(buffer,inlut,outlut) capture and display
```

```
; the specified buffer or camera through the specified LUTs
;
; This procedure captures one frame
;
    BEGIN    capture
    pushbp
    mov bp,sp
;                   Busy must be clear to proceed
    mov dx,incsr1
    mov ax,08H
    out dx,ax          ; Set ENSTOP
    out dx,ax          ; Clear BUSY
check_b:
    in  ax,dx
    and ax,080H        ; Check BUSY is clear
    jnz check_b
;
;                   value for inscr2 now calculated
    mov cl,7
    mov bx,[bp+6]      ; fetch input buffer number
    shl bx,cl          ; shift bx 7 places left
    mov dx,incsr2
    mov ax,0ff10H      ; MODE = 001, no write protect
    add ax,bx
        out  dx,ax
    mov dx,outcsr
    mov bx,[bp+10]     ; fetch display buffer
    add bx,[bp+12]     ; fetch output LUT
    mov ax,0000111110100000b; Set DISPLAY,EXTERNAL TIMING,O/P BUF=0
    add ax,bx
    out dx,ax              ;
;
    mov dx,incsr1
    mov bx,[bp+8]      ; fetch INPUT LUT
    mov ax,0ff88H      ; Set BUSY,ENSTOP; Input LUT = 000
    add ax,bx
    out dx,ax
;
check_by:
    in  ax,dx
    and ax,080H
    jnz check_by
;                   Frame captured, return to caller
    pop bp
    ret
capture  ENDP
;-------------------------------------------------------------------------------
;
; This procedure displays the captured frame
;
    BEGIN    show_buf
    pushbp
    mov bp,sp
    mov dx,outcsr
    mov bx,[bp+6]
    mov cl,4
    shl bx,cl
    add bx,[bp+8]
    mov ax,0000111110100000b
    add ax,bx
    out dx,ax              ; CURSOR OFF,BUF = 0,SYNC INT,O/P LUT=0
    pop bp
    ret
show_buf ENDP
;
;-------------------------------------------------------------------------------
;    This is function "read_pix(x,y,buffer)" which obtains the pixel value
;    at position (x,y) in bufferand returns this value (in ax) to the caller.
;    It utilises BIOS INT 15H, Function 87H, to access the frame store.

;       This function returns the pixel value in AL; AH defines status
;    of BIOS call:see IBM tech. ref.

    BEGIN    read_pix
    pushbp       ; save caller's stack frame pointer
    mov bp,sp        ; establish this function's stack
    mov ax,200H      ; length of one line i.e 512 pixels
    mov bx,[bp+8]    ; L memory model used; bx = y position
    mul bx       ; calculate y position in memory,result in DX:AX
    mov bx,[bp+6]    ; L memory model used; bx = x position
    add ax,bx    ; calculate true location in memory:include x
    mov cx,[bp+10]   ; fetch buffer no.
    jcxz     b0
    add      dx,0a4H      ; start of buffer 1
    jmp b1
b0: add dx,0a0H      ; include offset of DT2853 base address
b1: mov bx,read_add  ; bx points to base address in GDT
    mov [bx+2],dl    ; put hi byte pixel address calculated into GDT
    mov [bx+0],ax    ; put low word of pixel address into GDT
;                   ; 24 bit address of target now calculated
    mov ax,ds        ; ax:si loaded with (segment:offset) of
    lea si,tgt_data  ; intended location of pixel value
;
    mov bx,10H       ; ax now shifted 4 places left and result
    mul bx       ; placed in DX:AX
;
    add ax,si        ; offset address now added to lower word
    adc dx,0H        ; and any carry resulting added to DX
;
    mov     bx,point_tgt ; bx points to target base address
    mov [bx],ax      ; tgt_base_add now filled with
    mov [bx+2],dl    ; low word, high byte

    les si,gdt       ; es:si now points to top of GDT
;                     BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,1     ; number of pixels to be moved
    int 15H      ; BIOS interrupt
    mov ax,tgt_data      ; pixel value put into AL for return to C
    xor ah,ah        ; clear AH
    pop bp
    ret
 read_pix    ENDP
;
    BEGIN    wrt_pix
    pushbp       ; save caller's stack frame pointer
    mov bp,sp        ; establish this function's stack
;                   First read the memory word starting
;                   at the pixel location required
```

```
        mov ax,200H       ; length of one line i.e 512 pixels
        mov bx,[bp+8]     ; L memory model used; bx = y position
        mul bx        ; calculate y position in memory,result in DX:AX
        mov bx,[bp+6]     ; L memory model used; bx = x position
        add ax,bx     ; calculate true location in memory:include x
        mov cx,[bp+12]
        jcxz bu0
        add dx,0a4H
        jmp bu1
bu0:    add dx,0a0H       ; include offset of DT2853 base address
bu1:    mov bx,read_add   ; bx points to SOURCE address in READ GDT
        mov [bx+2],dl     ; put hi byte pixel address calculated into GDT
        mov [bx+0],ax     ; put low word of pixel address into GDT
            mov  bx,write_add ; bx points to TARGET address in WRITE GDT
        mov [bx+2],dl     ; put hi byte pixel address calculated into GDT
        mov [bx+0],ax     ; put low word of pixel address into GDT
;                   ; 24 bit address of target now calculated
        mov ax,ds         ; ax:si loaded with (segment:offset) of
        lea si,tgt_data   ; intended location of pixel value
        mov tgt_dat_add,si    ; store location in tgt_dat_add
;
        mov bx,10H        ; ax now shifted 4 places left and result
        mul bx        ; placed in DX:AX
;
        add ax,si         ; offset address now added to lower word
        adc dx,0H         ; and any carry resulting added to DX
;
        mov     bx,point_tgt  ; bx points to TARGET base address in READ GDT
        mov [bx],ax       ; tgt_base_add now filled with
        mov [bx+2],dl     ; low word, high byte
        lea bx,wsrc_base_add; bx points to SOURCE address in WRITE GDT
        mov [bx],ax       ; wsrc_base_add now filled with
        mov [bx+2],dl     ; low word, high byte

        les si,gdt        ; es:si now points to top of READ GDT
;                   BIOS call now set up
        mov ah,87H        ; BIOS function 87H
        mov cx,1      ; number of pairs of pixels to be moved
        int 15H       ; BIOS interrupt
        mov bx,[bp+10]    ; value to be written to frame store in bx
        lea si,tgt_data   ;
        mov [si],bl       ; byte overwritten in tgt_data
        les si,wgdt       ; es:si now points to top of WRITE GDT
;                   BIOS call now set up
        mov ah,87H        ; BIOS function 87H
        mov cx,1      ; number of pixels to be moved
        int 15H       ; BIOS interrupt
        pop bp
        ret
wrt_pix ENDP
;
;-------------------------------------------------------------------------
;
        ENDPS
        END
```

These routines move data to and from the DT2853.

Dtmoves.asm :

```
;-------------------------------------------------------------------------
;    This program moves DATA between the frame store and system memory.
;    A 256KB area must first be allocated before any transfers and must be
;    freed at the end.
;    It is to be called from a C program which supplies the values for
;    buffer moves.
;-------------------------------------------------------------------------
;
    TITLE    Data access
    SUBTTL   22/2/88, revised 25/11/88
    NAME dt_moves
;
    INCLUDE DOS.MAC
;
;       DATA items now declared and descriptor tables set up.
;    For details of descriptor table see IBM tech. ref. pp 5-149,5-150.
;
    DSEG
;
;    The following is the descriptor table for reading from the frame store
;
        dummy         DQ  0

        gdt_loc       DQ  0

        src_seg_limit     DW  0ffffH
        src_base_add DB  ?,?,?; DT base address = A00000H
;                        To be filled by pointer
        src_dat_ri    DB  93H
        src_dat_res   DW  00H

        tgt_seg_limit     DW  0ffffh
        tgt_base_add DB  ?,?,?; To be filled by pointer
        tgt_dat_ri    DB  93H
        tgt_dat_res   DW  00H

        bios_cs       DQ  0

        temp_ss       DQ  0
;
;
;
        gdt           DD  dummy ; gdt points (seg,offset) to top
;                          of table
        read_add      DW  src_base_add ; points(offset) to source
;                             base address
        point_tgt         DW      tgt_base_add ; pointer defined for
;                            filling target address
;-------------------------------------------------------------------------
;    Write descriptor table now setup
;
        wdummy        DQ  0
;
        wgdt_loc DQ  0

        wsrc_seg_limit    DW  0ffffH
        wsrc_base_add     DB  ?,?,?; Equivalent to tgt_base_add
        wsrc_dat_ri  DB  93H
        wsrc_dat_res DW  00H
;
        wtgt_seg_limit    DW  0ffffh
        wtgt_base_add     DB  ?,?,?; Dt board;To be filled by pointer
        wtgt_dat_ri  DB  93H
        wtgt_dat_res DW  00H
;
        wbios_cs DQ  0
;
        wtemp_ss DQ  0
;
;
        write_add     DW  wtgt_base_add ;pointer to DT board
;                             address location
        wgdt          DD  wdummy ; wgdt points (seg,offset) to top
;                          of table
        wrt_src_add   DW  wsrc_base_add ; points(offset) to source
;                             base address
;
;
        buff_seg DW  ?    ; word to hold buffer seg for FREE
;
    ENDDS
;
;-------------------------------------------------------------------------
;
    PSEG     ; program segment starts
;
;    allocate(KBYTES) :  This allocates KBYTES kbytes of memory and returns
;                          segment address

    BEGIN    allocate
    pushbp        ; save caller's stack frame pointer
    mov bp,sp         ; establish this function's stack
    mov ax,[bp+6]     ; get number of kbytes
                      ; and convert to number of paragraphs
    mov cl,06h
    shl ax,cl
    mov bx,ax         ; no. of paragraphs in bx
    mov ah,48h
    int 21h
    jnc got_add
;                     error so set ax = ffff
    mov ax,0ffffh
;
got_add:
    pop bp
    ret
allocate ENDP

;-------------------------------------------------------------------------
;
;    free_mem(segment) :  This frees memory previously allocated at
;                          SEGMENT address
```

```
        BEGIN    free_mem
        pushbp       ; save caller's stack frame pointer
        mov bp,sp        ; establish this function's stack
        mov ax,[bp+6]    ; get segment
        mov buff_seg,ax
        mov es,buff_seg
        mov ah,49h
        int 21h
        jnc done_ok
;                        error so set ax = c000
        mov ax,0c000h
        jc  no_zero
;
done_ok:
        mov ax,0h
no_zero:
        pop bp
        ret
free_mem ENDP


;-------------------------------------------------------------------------
;   This is function "mov_buf(SEGMENT,BUFF)" which moves DT buffer BUFF
;   to system memory starting at segment SEGMENT.
;   It utilises BIOS INT 15H, Function 87H, to access the frame store.


        BEGIN    mov_buf
        pushbp       ; save caller's stack frame pointer
        mov bp,sp        ; establish this function's stack
        mov bx,read_add  ; bx points to base address of DT
        mov ax,00H
        mov [bx],ax      ; put low word of pixel address into GDT
        mov ax,[bp+8]    ; fetch buffer no.
        mov cl,2
        shl ax,cl        ; Do adjustment for buffer number
        add al,0a0H
        mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
        mov cl,4     ; Need to shift segment to insert paragraph
        mov bx,point_tgt ; bx points to target base address
        mov ax,[bp+6]
        mov buff_seg,ax  ; put start into buff_seg
        shl ax,cl
        mov [bx],ax      ;
        mov cl,0ch
        mov ax,buff_seg
        shr ax,cl
        mov [bx+2],al    ; low word, high byte

        les si,gdt       ; es:si now points to top of GDT
;                      BIOS call now set up
        mov ah,87H       ; BIOS function 87H
        mov cx,8000H ; number of pixels to be moved:64k
        int 15H      ; BIOS interrupt
;
        mov bx,read_add  ; bx points to base address in          GDT
        mov ax,00h
        mov [bx],ax      ; put low word of pixel address into GDT
        mov ax,[bp+8]    ; fetch buffer no.
        mov cl,2
        shl ax,cl        ; Do adjustment for buffer number
        add al,0a1H
        mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
        mov cl,4     ; Need to shift segment to insert paragraph
        mov bx,point_tgt ; bx points to target base address
        mov ax,buff_seg
        shl ax,cl
        mov [bx],ax      ;
        mov cl,0ch
        mov ax,buff_seg
        shr ax,cl
        add al,1
        mov [bx+2],al    ; low word, high byte

        les si,gdt       ; es:si now points to top of GDT
;                      BIOS call now set up
        mov ah,87H       ; BIOS function 87H
        mov cx,8000H ; number of pixels to be moved:64k
        int 15H      ; BIOS interrupt
;
        mov bx,read_add  ; bx points to base address in GDT
        mov ax,00h
        mov [bx],ax      ;
        mov ax,[bp+8]    ; fetch buffer no.
        mov cl,2
        shl ax,cl        ; Do adjustment for buffer number
        add al,0a2H
        mov [bx+2],al    ; put hi byte pixel address calculated into GDT
        mov cl,4     ; Need to shift segment to insert paragraph
        mov bx,point_tgt ; bx points to target base address
        mov ax,buff_seg
        shl ax,cl
        mov [bx],ax      ;
        mov cl,0ch
        mov ax,buff_seg
        shr ax,cl
        add al,2
        mov [bx+2],al    ; low word, high byte

        les si,gdt       ; es:si now points to top of GDT
;                      BIOS call now set up
        mov ah,87H       ; BIOS function 87H
        mov cx,8000H ; number of pixels to be moved:64k
        int 15H      ; BIOS interrupt
;
        mov bx,read_add  ; bx points to base address in GDT
        mov ax,00h
        mov [bx],ax  ; put low word of pixel address into GDT
        mov ax,[bp+8]    ; fetch buffer no.
        mov cl,2
        shl ax,cl        ; Do adjustment for buffer number
        add al,0a3H
        mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
```

```
    mov cl,4     ; Need to shift segment to insert paragraph
    mov bx,point_tgt ; bx points to target base address
    mov ax,buff_seg
    shl ax,cl
    mov [bx],ax      ;
    mov cl,0ch
    mov ax,buff_seg
    shr ax,cl
    add ax,3
    mov [bx+2],al    ; low word, high byte

    les si,gdt       ; es:si now points to top of GDT
;                  BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,8000H ; number of pixels to be moved:64k
    int 15H      ; BIOS interrupt
    pop bp
    ret
mov_buf  ENDP
;
;-------------------------------------------------------------------------
;   This is function "res_buf(SEGMENT,BUFF)" which restores the buffer from
;   system memory at address SEGMENT to buffer BUFF on DT2853.
;   It utilises BIOS INT 15H, Function 87H, to access the frame store.
;-------------------------------------------------------------------------
;
    BEGIN    res_buf
    pushbp       ; save caller's stack frame pointer
    mov bp,sp        ; establish this function's stack
    mov bx,write_add ; bx points to base address in GDT
    mov ax,00H
    mov [bx+0],ax    ; put low word of pixel address into GDT
    mov ax,[bp+8]    ; fetch buffer no.
    mov cl,2
    shl ax,cl        ; Do adjustment for buffer number
    add al,0a0H
    mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
    mov bx,wrt_src_add    ; bx points to target base address
    mov cl,4              ; Need to shift segment to insert paragraph
    mov ax,[bp+6]
    mov buff_seg,ax
    shl ax,cl
    mov [bx],ax      ;
    mov cl,0ch
    mov ax,buff_seg
    shr ax,cl
    mov [bx+2],al    ; low word, high byte

    les si,wgdt      ; es:si now points to top of GDT
;                  BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,8000H ; number of pixels to be moved:64k
    int 15H      ; BIOS interrupt
;
    mov bx,write_add ; bx points to base address in GDT
    mov ax,00h
    mov [bx],ax  ; put low word of pixel address into GDT
    mov ax,[bp+8]    ; fetch buffer no.
    mov cl,2
    shl ax,cl        ; Do adjustment for buffer number
    add al,0a1H
    mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
    mov bx,wrt_src_add    ; bx points to target base address
    mov cl,4              ; Need to shift segment to insert paragraph
    mov ax,buff_seg
    shl ax,cl
    mov [bx],ax      ;
    mov cl,0ch
    mov ax,buff_seg
    shr ax,cl
    add ax,1
    mov [bx+2],al    ; low word, high byte

    les si,wgdt      ; es:si now points to top of GDT
;                  BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,8000H ; number of pixels to be moved:64k
    int 15H      ; BIOS interrupt
;
    mov bx,write_add ; bx points to base address in GDT
    mov ax,00h
    mov [bx],ax
    mov ax,[bp+8]    ; fetch buffer no.
    mov cl,2
    shl ax,cl        ; Do adjustment for buffer number
    add al,0a2H
    mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
    mov bx,wrt_src_add    ; bx points to target base address
    mov cl,4              ; Need to shift segment to insert paragraph
    mov ax,buff_seg
    shl ax,cl
    mov [bx],ax      ;
    mov cl,0ch
    mov ax,buff_seg
    shr ax,cl
    add ax,2
    mov [bx+2],al    ; low word, high byte

    les si,wgdt      ; es:si now points to top of GDT
;                  BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,8000H ; number of pixels to be moved:64k
    int 15H      ; BIOS interrupt
;
    mov bx,write_add ; bx points to base address in GDT
    mov ax,00h
    mov [bx],ax  ; put low word of pixel address into GDT
    mov ax,[bp+8]    ; fetch buffer no.
    mov cl,2
    shl ax,cl        ; Do adjustment for buffer number
    add al,0a3H
    mov [bx+2],al    ; put hi byte pixel address calculated into GDT
;
    mov bx,wrt_src_add    ; bx points to target base address
    mov cl,4              ; Need to shift segment to insert paragraph
    mov ax,buff_seg
```

```
    shl ax,cl
    mov [bx],ax      ;
    mov cl,0ch
    mov ax,buff_seg
    shr ax,cl
    add ax,3
    mov [bx+2],al    ; low word, high byte

    les si,wgdt      ; es:si now points to top of GDT
;                 BIOS call now set up
    mov ah,87H       ; BIOS function 87H
    mov cx,8000H ; number of pixels to be moved:64k
    int 15H      ; BIOS interrupt
    pop bp
    ret
res_buf  ENDP
;
;-------------------------------------------------------------------------
;   Function wrtpixl(x,y,z,seg): writes value z to location x,y in buffer
;   currently in system memory at segment seg.
;
    BEGIN   wrtpixl
    pushbp
    mov bp,sp
    mov ax,200H      ;length of one line
    mov bx,[bp+8]
    mul bx       ;result in DX:AX
    mov cl,12
    shl dx,cl
    mov bx,ax        ;temp store of ax
    mov cl,4
    shr ax,cl
    add dx,ax
    mov ax,[bp+0ch]  ; get segment
    add dx,ax        ; include memory offset
    mov ax,bx        ; restore ax
    and ax,000fh ; clear top 12 bits of ax
    mov bx,[bp+6]
    add bx,ax        ;
    mov al,[bp+10]   ;value to be written
    push  ds
    mov ds,dx
    mov [bx],al
    pop ds
    pop bp
    ret
wrtpixl  ENDP
;
;-------------------------------------------------------------------------
;   Function readpixl(x,y,seg): reads value at location x,y in buffer
;   currently in system memory.
;
    BEGIN readpixl
    push bp
    mov bp,sp
    mov ax,200H      ;length of one line
    mov bx,[bp+8]
    mul bx       ;result in DX:AX
    mov cl,12
    shl dx,cl
    mov bx,ax        ; temp store of ax
    mov cl,4
    shr ax,cl
    add dx,ax
    mov ax,[bp+0ah]  ; get segment
    add dx,ax        ; include memory offset
    mov ax,bx        ; restore ax
    and ax,000fh ; clear top 12 bits of ax
    mov bx,[bp+6]
    add bx,ax        ;
    push ds
    mov ds,dx
    mov ax,[bx]
    xor ah,ah
    pop ds
    pop bp
    ret
readpixl ENDP
;
;-------------------------------------------------------------------------
    ENDPS
    END
```

APPENDIX B.1

PAL DESIGNS FOR OFCS

PAL Designs for OFCS

Timing.pld :

```
Name        timing;
Partno      IC1;
Date        30/07/89;
Revision    05;
Designer    Greg;
Company     University of Liverpool;
Assembly    Fibre Analysis Board;
Device       p22V10;
Location    XX;

/*************************************************/
/*  This PLD interfaces the camera sync timings    */
/*  with the counters for capture/display      */
/*************************************************/
/*  Allowable Target Device Types:  22V10      */
/*************************************************/

/**  Inputs  **/

Pin 1    = clock  ;    /*       clock - 20 MHz                */
Pin 2    = !cmp_sync  ;    /*  composite sync                      */
Pin 3    = !vcl_sync  ;    /*  vertical syncs                      */
Pin 6    = h_count    ;    /*  horizontal count done               */
Pin 7    = hrco       ;    /*  horizontal ripple carry out  */
Pin 8    = v_count    ;    /*  vertical count done    */
Pin 9    = vrco       ;    /*  vertical ripple carry out    */
Pin 10   = CAPTURE    ;
Pin 11   = ACCESS ;
Pin 13   = OddnotEven ;
Pin 17   = RESET  ;    /*  Reset to S0                 */

/**  Outputs  **/

Pin 23   = data_valid ;    /*  A/D output valid      */
Pin 19   = QA     ;
Pin 21   = hclr       ;    /*  horizontal clear      */
Pin 20   = hclk       ;    /*  horizontal clock      */
Pin 22   = vclr       ;    /*  vertical clear         */
Pin 18   = QC         ;    /*  data latch clock      */
Pin 15   = Cap_data   ;
Pin 14   = QB         ;

/** Declarations and Intermediate Variable Definitions **/

FIELD states = [hclk,hclr,QC,QB,QA];


$define  S0 'b' 00000 /*  0    */
$define  S1 'b' 11100 /*  28   */
$define   D4 'b' 01100     /*  12   */
$define  S2 'b' 01000 /*  8    */
$define  S3 'b' 11101 /*  29   */
$define  S4 'b' 10000 /*  16   */
$define  S5 'b' 10100 /*  20   */
$define  S6 'b' 10101 /*  21   */
$define  S7 'b' 00100     /*  4    */
$define  S8 'b' 01001 /*  9    */
$define  S9 'b' 00101 /*  5    */
$define S10 'b' 00110 /*  6    */
$define S11 'b' 00111 /*  7    */
$define S12 'b' 00001 /*  1    */
$define S13 'b' 10110 /*  22   */
$define L1  'b' 10111 /*  23   */
$define L2  'b' 00010 /*  2    */
$define L3  'b'  00011    /*  3    */
$define S14 'b' 10001 /*  17   */
$define S15 'b' 01010 /*  10   */
$define S16 'b' 01011 /*  11   */

$define D5  'b'  01101    /* 'd' 13 */

$define D6  'b'  01110    /* 'd' 14 */
$define D7  'b'  01111    /* 'd' 15 */
$define D8  'b'  10010    /* 'd' 18 */
$define D9  'b'  10011    /* 'd' 19 */
$define D10 'b'  11000    /* 'd' 24 */
$define D11 'b'  11001    /* 'd' 25 */
$define D12 'b'  11010    /* 'd' 26 */
$define D13 'b'  11011    /* 'd' 27 */
$define D14 'b' 11110 /* 'd' 30 */
$define D15 'b' 11111 /* 'd' 31 */

/**  Logic Equations  **/

vclr.ar = RESET;
```

```
data_valid.ar     = RESET;
Cap_data.ar  = RESET;
hclr.ar = RESET;
hclk.ar = RESET;
QC.ar = RESET;
QB.ar     = RESET;
QA.ar     = RESET;

vclr.sp = 'b' 0;
data_valid.sp     = 'b' 0;
Cap_data.sp  = 'b' 0;
hclr.sp = 'b' 0;
hclk.sp = 'b' 0;
QC.sp = 'b'0;
QB.sp     = 'b'0;
QA.sp     = 'b'0;

sequence states
{
    present S0        next S1;

    /*  Wait For vertical sync    */

    present S1   if !vcl_sync next S1;
                 if  vcl_sync next D4;

present D4   if !vcl_sync next S2;
             if  vcl_sync next D4
                 out vclr;

/*  Wait For Start Of First Line i.e wait for VCOUNT  */

present S2   if (!v_count &  CAPTURE & !OddnotEven)   next S2
             out Cap_data;
             if (!v_count &  CAPTURE &  OddnotEven)  next S2;
             if (!v_count & !CAPTURE &  OddnotEven)  next S2;
             if (!v_count & !CAPTURE & !OddnotEven)  next S2;

             if v_count        next S3
                 out vclr;

present S3   if  cmp_sync next S3;
             if !cmp_sync next S4; /* hclk */

/*  Now at start line                    */

present S4                          next S5; /* hclk */
present S5   if !h_count            next S6;
             if  h_count       next S8; /* hclr */
present S6                          next S7;
present S7                          next S4;
present S8                          next S9;
present S9                          next S10 /* data latch */
                  out data_valid;
present S10                         next S11 /* data latch */
                  out data_valid;
present S11                         next S12
                  out data_valid;
present S12                         next S13
                  out data_valid;
present S13                         next L1    /* data latch, hclk */
                  out data_valid;
present L1   if !hrco               next L2  /* data latch, hclk */
                  out data_valid;
             if hrco                next S14
                  out data_valid;

present L2                          next L3
                  out data_valid;
present L3                          next S13
                  out data_valid;
present S14  if !cmp_sync next S14;
             if cmp_sync            next S15;
present S15  if !vrco     next S16;
             if  vrco               next S1;
present S16  if !cmp_sync next S16;
             if  cmp_sync next S3;
present D6        next S0;
present D7        next S0;
present D8        next S0;
present D9        next S0;
present D10       next S0;
present D11       next S0;
present D12       next S0;
present D13       next S0;
present D14       next S0;
present D15       next S0;
}
```

Clut_io.pld :

```
Name        clut_io;
Partno      IC60;
Date        1/11/89;
Revision    01;
Designer    Greg;
Company     University of Liverpool;
Assembly    Fibre Analysis Board;
Device       p22v10;
Location    XXXXX;

/*************************************************/
/*  This one controls I/O from transputer        */
/*  via C011 to the G178                         */
/*************************************************/
/*  Allowable Target Device Types:  22V10        */
/*************************************************/

/**  Inputs  **/

Pin 1    = clock  ;    /*  clock - 5 MHz                */
Pin 2    = A2;
Pin 3    = QVALID;
Pin 4    = Sys_reset;
Pin 5    = clut_acc;
Pin 6    = reset;

Pin 7    = RCO_LO;
Pin 8    = RCO_HI;
Pin 9    = CCLK;
Pin 10   = ADD0;

/**  Outputs  **/

Pin 23   = !WRITE;
Pin 22   = QACK   ;
Pin 20   = ctlclk;    /*  control latch clock        */
Pin 19   = clutclk;
Pin 18   = QB;
Pin 17   = QA;
Pin 15   = CLUT_RESET;
Pin 14   = IO_RESET;

Pin 16   = Sys_rco;

/** Declarations and Intermediate Variable Definitions **/

FIELD states = [WRITE,QACK,QB,QA];

$define boot_up          'b' 0000  /* 0 */
$define start       'b' 0100  /* 4 */
$define go          'b' 0010  /* 2 */
$define register              'b' 0110  /* 6 */
$define data              'b' 0011  /* 3 */
$define wait        'b' 0001  /* 1*/
$define write_to_clut     'b' 1000  /* 8 */
$define wait_again        'b' 0101  /* 5 */
$define qack        'b' 0111  /* 7*/

$define  S11              'b' 1011  /* 11*/
$define S5          'b' 1001  /* 9 */
$define S10                   'b' 1010  /* 10*/
$define S12         'b' 1100  /* 12*/
$define S13         'b' 1101  /* 13*/
$define S14         'b' 1110  /* 14*/
$define S15         'b' 1111  /* 15*/

QACK.ar = reset;
WRITE.ar = reset;
QA.ar    = reset;
QB.ar    = reset;

QACK.sp = 'b' 0;
WRITE.sp = 'b' 0;
QA.sp    = 'b' 0;
QB.sp    = 'b' 0;

QACK.oe = clut_acc;
ctlclk.oe = clut_acc;
clutclk.oe = clut_acc;

/**  Logic Equations  **/

IO_RESET   = (Sys_reset #  clut_acc );
CLUT_RESET = (Sys_reset # !clut_acc );

Sys_rco  = (RCO_LO & RCO_HI);

/*
1. pixel mask
2. write address
3. RGB

*/

sequence states
{

present boot_up
     if !QVALID next boot_up;
     if  QVALID next start
     out ctlclk
     out clutclk;

present start          /* Sends QACK    */
     if  QVALID   next start;
     if !QVALID   next go;

present go
     if  QVALID   next register
     out clutclk;
     if !QVALID   next go;

present register
```

```
    if ( A2 &  QVALID)    next register;
    if  !A2       next boot_up;
    if ( A2 & !QVALID)    next data;

present data
    if  QVALID   next wait;
    if !QVALID   next data;

present wait           next write_to_clut;
present write_to_clut  next wait_again;
present wait_again           next qack;

present qack
    if  QVALID   next qack;
    if !QVALID   next go;

present S5             next boot_up;
present S10            next boot_up;
present S11            next boot_up;
present S12            next boot_up;
present S13            next boot_up;
present S14            next boot_up;
present S15            next boot_up;

}
```

LUT_CNTRL.pld :

```
Name        lut_ctrl;
Partno      01;
Date        25/04/89;
Revision    01;
Designer    Greg;
Company     University of Liverpool;
Assembly    Fibre Analysis Board;
Location    XXXXX;

/************************************************/
/*  This PLD uses the control bits to control  */
/*  access to the Look Up Table                */
/************************************************/
/*  Allowable Target Device Types:  EP600      */
/************************************************/

/**  Inputs  **/

Pin 1    = clock  ;    /*       clock - 20 MHz          */
Pin 2    = pixel_clk  ;    /*  Pixel clock - 5MHz           */
Pin 3    = 10MHz_clk  ;    /*  Reference 10 Mhz clock       */
Pin 4    = !cmp_sync  ;    /*  composite sync                    */
Pin 5    = !vcl_sync  ;    /*  vertical syncs               */
Pin 6    = h_count    ;    /*  horizontal count done             */
Pin 7    = hrco       ;    /*  horizontal ripple carry out  */
Pin 8    = v_count    ;    /*  vertical count done          */
Pin 9    = vrco       ;    /*  vertical ripple carry out    */
Pin 10   = !even      ;    /*  even field                   */
Pin 11   = !EN        ;    /*  Device enable                */

/**  Outputs  **/

Pin 23   = vclk            ;    /*  vertical clock                */
Pin 22   = hclk            ;    /*  horizontal clock         */
Pin 21   = Q3              ;
Pin 20   = Q2              ;
Pin 19   = Q1              ;
Pin 18   = Q0         ;
Pin 17   = data_valid ;    /*  A/D data valid               */
Pin 16   = vclr       ;    /*  vertical clear               */
Pin 15   = hclr            ;    /*  horizontal clear         */
Pin 14   = data_lch_clk    ;    /*  data latch clock         */

/** Declarations and Intermediate Variable Definitions **/

vclk.oe = EN;
hclk.oe = EN;
vclr.oe = EN;
hclr.oe = EN;
data_lch_clk.oe = EN;
data_valid.oe = EN;

vclk.ar = 'b' 0;
hclk.ar = 'b' 0;
vclr.ar = 'b' 0;
hclr.ar = 'b' 0;
data_lch_clk.ar = 'b'0;
data_valid.ar    = 'b' 0;

Q3.ar    = 'b' 0;
Q2.ar    = 'b' 0;
Q1.ar    = 'b' 0;
Q0.ar    = 'b' 0;


vclk.sp = 'b' 0;
hclk.sp = 'b' 0;
vclr.sp = 'b' 0;
hclr.sp = 'b' 0;
data_lch_clk.sp = 'b'0;
data_valid.sp    = 'b' 0;

Q3.sp    = 'b' 0;
Q2.sp    = 'b' 0;
Q1.sp    = 'b' 0;
Q0.sp    = 'b' 0;

FIELD states = [Q3..Q0];

$define  S0 'b' 0000
$define  S1 'b' 0001
$define  S2 'b' 0011
$define  S3 'b' 0010
$define  S4 'b' 0110
$define  S5 'b' 0111
$define  S6 'b' 0101
$define  S7 'b' 0100
$define  S8 'b' 1100
$define  S9 'b' 1000
$define S10 'b' 1001
$define S11 'b' 1011
$define S12 'b' 1010
$define S13 'b' 1110
$define S14 'b' 1111
$define S15 'b' 1101

/**  Logic Equations  **/

sequence states
{
    /*  Wait For Enable  */
    present S0   if !EN                          next S0;
                 if EN                           next S1;

    /*  Wait For vertical sync    */
    present S1   if !EN                               next S0;

                 if (!vcl_sync & EN)                  next S1
                     out hclr
                     out vclr;

                 if (EN & vcl_sync)                   next S2
                     out hclr
                     out vclr;
```

```
                default                                         next S0;

/*  Wait For Even Frame  and for vertical sync to go away */
present S2      if !EN                                          next S0;
                if (EN & !even & !vcl_sync)                     next S1;
                if (EN & !even &  vcl_sync)                     next S2;
                if (EN &  even & !vcl_sync)                     next S3;
                if (EN &  even &  vcl_sync)                     next S2;
                default                                         next S0;

/*  Wait For Start Of First Line  */
/*  Output one VCLK per cmp_sync  */
/*  and wait for VCOUNT                 */
present S3      if !EN                                          next S0;
                if (EN & cmp_sync & !v_count)                   next S4
                    out vclk;

                if (EN & !cmp_sync & !v_count)                  next S3;
                if (EN & !cmp_sync & v_count)                   next S5
                    out vclr;

                if (EN & cmp_sync & v_count)                    next S4;
                default                                         next S0;

present S4      if !EN                                          next S0;
                if (EN & cmp_sync)                              next S4;
                if (EN & !cmp_sync)                             next S3;
                default                                         next S0;

/*  Now at first line                   */
/*  Wait For porch and state before     */
/*  HI -> LO on pixel clock             */
present S5      if !EN                                          next S0;
                if (EN & !10MHz_clk & pixel_clk)                next S6;
                if (EN & (10MHz_clk # !pixel_clk))              next S5;
                default                                         next S0;

present S6      if !EN                                          next S0;
                if (EN & 10MHz_clk & pixel_clk)                 next S7
                    out hclk;

                if (EN & !(10MHz_clk & pixel_clk))              next S5;
                default                                         next S0;

/* Wait for count*/
present S7      if !EN                                          next S0;
                if EN                                           next S8;

present S8      if (EN & !h_count)                              next S5;

                if (EN & h_count)                               next S9
                    out hclr;

                default                                         next S0;

/*  count 256 pixels */
present S9      if !EN                                          next S0;
                if EN                                           next S10
                    out data_valid
                    out data_lch_clk;

present S10     if !EN                                          next S0;
                if EN                                           next S11
                    out data_valid
                    out hclk;

/*  Wait for VRCO,HRCO if any */
present S11     if !EN                                          next S0;
                if EN                                           next S12
                    out data_valid;

present S12     if !EN                                          next S0;
                if (EN & hrco)                                  next S13
                    out hclr
                    out vclk;

                if (EN & !hrco)                                 next S9
                    out data_valid;

/*  wait for VRCO    */
present S13     if !EN                                          next S0;
                if EN                                           next S14;

/*  Wait for line sync   */
present S14     if !EN                                          next S0;
                if (EN & !vrco & !cmp_sync)                     next S14;
                if (EN & !vrco &  cmp_sync)                     next S15;
                if (EN &  vrco & !cmp_sync)                     next S1
                    out vclr;
                if (EN &  vrco &  cmp_sync)                     next S1
                    out vclr;
                default                                         next S0;

present S15     if !EN                                          next S0;
                if (EN &  cmp_sync)                             next S15;
                if (EN & !cmp_sync)                             next S5;
}
```

Hcounter.pld :

```
Name      hcounter;
Partno     IC7;
Date      30/07/89;
Revision  03;
Designer  Greg;
Company   Liverpool University;
Assembly  Fibre Analysis Board;
Device     p22v10;
Location  None yet;

/*************************************************************/
/*                                                           */
/* Octal Counter                                             */
/*                                                           */
/* 8-bit synchronous counter with CLEAR and ENABLE.          */
/* The CLEAR operation is asynchronous and resets            */
/* the output register to all LOWs.                          */
/* Setting output enable tri-states all outputs.             */
/* A variable number of clock pulses can be counted          */
/* for timing delays.                                        */
/*************************************************************/
/**       Allowable Target Device Types :  22V10            **/
/*************************************************************/

/**  Inputs  **/

PIN 1         = clock    ;   /* Register Clock       */
Pin 2         = !enable   ;    /* Enable count (Hold) */
Pin 3         = clear    ; /* Clear Counter      */
PIN 13        = !OE      ;   /* Output enable        */

/**  Outputs  **/

PIN [15..22] = [Q7..0]  ;     /* Register Outputs    */
Pin 23        = rco       ;   /* Ripple carry out    */
Pin 14        = done      ;   /* Variable count done */


/** Declarations and Intermediate Variable Definitions **/

FIELD value = [Q7..Q0];

Q0.ar = clear;              Q0.sp = 'b' 0;
Q1.ar = clear;              Q1.sp = 'b' 0;
Q2.ar = clear;              Q2.sp = 'b' 0;
Q3.ar = clear;              Q3.sp = 'b' 0;
Q4.ar = clear;              Q4.sp = 'b' 0;
Q5.ar = clear;              Q5.sp = 'b' 0;
Q6.ar = clear;              Q6.sp = 'b' 0;
Q7.ar = clear;              Q7.sp = 'b' 0;

done.ar = clear;      done.sp = 'b' 0;
rco.ar = clear;           rco.sp = 'b' 0;

Q0.oe = OE ;
Q1.oe = OE ;
Q2.oe = OE ;
Q3.oe = OE ;
Q4.oe = OE ;
Q5.oe = OE ;
Q6.oe = OE ;
Q7.oe = OE ;

done.oe = OE;
rco.oe = OE;

/** Logic Equations **/

done.d = value:'d'24;

rco.d = value:'d'255;

Q0.d =    ((!Q0 & enable ) # (Q0 & !enable) );

Q1.d = ((( Q0 & !Q1)
        # (!Q0 &  Q1)) & enable)
        # (Q1 & !enable);

Q2.d = ((( Q0 &  Q1 & !Q2)
        # (!(Q0 & Q1) &  Q2) ) & enable)
        # (Q2 & !enable);

Q3.d = ((( Q0 &  Q1 &  Q2 & !Q3)
        # (!(Q0 & Q1 & Q2) &  Q3) ) & enable)
        # (Q3 & !enable);

Q4.d = (((Q0 &  Q1 &  Q2 &  Q3 & !Q4)
       # (!(Q0 & Q1 & Q2 & Q3) &  Q4) ) & enable)
       # (Q4 & !enable);

Q5.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 & !Q5)
       # (!(Q0 & Q1 & Q2 & Q3 & Q4) &  Q5) ) & enable)
       # (Q5 & !enable);

Q6.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 & !Q6)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5) &  Q6) ) & enable)
    # (Q6 & !enable);

Q7.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 &  Q6 & !Q7)
       # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6) &  Q7)) & enable)
       # (Q7 & !enable);
```

Vcounter.pld :

```
Name      vcounter;
Partno     IC8;
Date      30/07/89;
Revision  03;
Designer  Greg;
Company   Liverpool University;
Assembly  Fibre Analysis Board;
Device     p22v10;
Location  None yet;

/*************************************************************/
/*                                                           */
/* Octal Counter                                             */
/*                                                           */
/* 8-bit synchronous counter with CLEAR and ENABLE.          */
/* The CLEAR operation is asynchronous and resets            */
/* the output register to all LOWs.                          */
/* Setting output enable tri-states all outputs.             */
/* A variable number of clock pulses can be counted          */
/* for timing delays.                                        */
/*************************************************************/
/**       Allowable Target Device Types :  22V10            **/
/*************************************************************/

/**  Inputs  **/

PIN 1         = clock    ;   /* Register Clock       */
Pin 2         = !enable   ;    /* Enable count (Hold) */
Pin 3         = clear    ; /* Clear Counter        */
PIN 13        = !OE      ;   /* Output enable        */

/**  Outputs  **/

PIN [15..22] = [Q7..0]  ;     /* Register Outputs     */
Pin 23        = rco       ;   /* Ripple carry out     */
Pin 14        = done      ;   /* Variable count done */


/** Declarations and Intermediate Variable Definitions **/

FIELD value = [Q7..Q0];

Q0.ar = clear;              Q0.sp = 'b' 0;
Q1.ar = clear;              Q1.sp = 'b' 0;
Q2.ar = clear;              Q2.sp = 'b' 0;
Q3.ar = clear;              Q3.sp = 'b' 0;
Q4.ar = clear;              Q4.sp = 'b' 0;
Q5.ar = clear;              Q5.sp = 'b' 0;
Q6.ar = clear;              Q6.sp = 'b' 0;
Q7.ar = clear;              Q7.sp = 'b' 0;

done.ar = clear;      done.sp = 'b' 0;
rco.ar = clear;           rco.sp = 'b' 0;

Q0.oe = OE ;
Q1.oe = OE ;
Q2.oe = OE ;
Q3.oe = OE ;
Q4.oe = OE ;
Q5.oe = OE ;
Q6.oe = OE ;
Q7.oe = OE ;

done.oe = OE;
rco.oe = OE;

/** Logic Equations **/

done.d = value:'d'30;

rco.d = value:'d'255;

Q0.d =    ((!Q0 & enable ) # (Q0 & !enable) );

Q1.d = ((( Q0 & !Q1)
        # (!Q0 &  Q1)) & enable)
        # (Q1 & !enable);

Q2.d = ((( Q0 &  Q1 & !Q2)
        # (!(Q0 & Q1) &  Q2) ) & enable)
        # (Q2 & !enable);

Q3.d = ((( Q0 &  Q1 &  Q2 & !Q3)
        # (!(Q0 & Q1 & Q2) &  Q3) ) & enable)
         # (Q3 & !enable);

Q4.d = (((Q0 &  Q1 &  Q2 &  Q3 & !Q4)
       # (!(Q0 & Q1 & Q2 & Q3) &  Q4) ) & enable)
       # (Q4 & !enable);

Q5.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 & !Q5)
       # (!(Q0 & Q1 & Q2 & Q3 & Q4) &  Q5) ) & enable)
       # (Q5 & !enable);

Q6.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 & !Q6)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5) &  Q6) ) & enable)
    # (Q6 & !enable);

Q7.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 &  Q6 & !Q7)
       # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6) &  Q7)) & enable)
       # (Q7 & !enable);
```

Hi_count.pld :

```
Name      HI Count;
Partno     02;
Date      12/04/89;
Revision  02;
Designer  Greg;
Company   Liverpool University;
Assembly  Fibre Analysis Board;
Device     p22v10;
Location  None yet;

/*************************************************************/
/*                                                           */
/* Octal Counter                                             */
/*                                                           */
/* 8-bit synchronous counter with CLEAR and ENABLE.          */
/* The CLEAR operation is asynchronous and resets            */
/* the output register to all LOWs.                          */
/* Setting output enable tri-states all outputs.             */
/*-----------------------------------------------------------*/
/*  Also enables Frame buffer outputs                        */
/*************************************************************/
/**       Allowable Target Device Types :  22V10            **/
/*************************************************************/

/**  Inputs  **/

PIN 1    = clock    ;   /* Register Clock       */
Pin 2    = enable   ; /* Enable count (Hold)    */
Pin 3    = clear    ; /* Clear Counter      */

Pin 7    = ACCESS;
Pin 8    = !count_FB1;
Pin 9    = Buff_no;
Pin 10   = R_not_W;
Pin 11   = LUT_not_BUF;

PIN 13   = OE      ;   /* Output enable        */

/**  Outputs  **/

PIN [15..22] = [Q7..0]  ;     /* Register Outputs    */
Pin 23       = rco      ;     /* Ripple carry out    */

Pin 14   = !FB1_oe;

/** Declarations and Intermediate Variable Definitions **/

FIELD value = [Q7..Q0];

Q0.ar = clear;            Q0.sp = 'b' 0;
Q1.ar = clear;            Q1.sp = 'b' 0;
Q2.ar = clear;            Q2.sp = 'b' 0;
Q3.ar = clear;            Q3.sp = 'b' 0;
Q4.ar = clear;            Q4.sp = 'b' 0;
Q5.ar = clear;            Q5.sp = 'b' 0;
Q6.ar = clear;            Q6.sp = 'b' 0;
Q7.ar = clear;            Q7.sp = 'b' 0;
rco.ar = clear;           rco.sp = 'b' 0;

Q0.oe = OE ;
Q1.oe = OE ;
Q2.oe = OE ;
Q3.oe = OE ;
Q4.oe = OE ;
Q5.oe = OE ;
Q6.oe = OE ;
Q7.oe = OE ;

rco.oe = OE;

/** Logic Equations **/

rco.d = value:'d'255;

Q0.d =    ((!Q0 & enable ) # (Q0 & !enable) );

Q1.d = ((( Q0 & !Q1)
        # (!Q0 &  Q1)) & enable)
        # (Q1 & !enable);

Q2.d = ((( Q0 &  Q1 & !Q2)
        # (!(Q0 & Q1) &  Q2) ) & enable)
        # (Q2 & !enable);

Q3.d = ((( Q0 &  Q1 &  Q2 & !Q3)
        # (!(Q0 & Q1 & Q2) &  Q3) ) & enable)
        # (Q3 & !enable);

Q4.d = (((Q0 &  Q1 &  Q2 &  Q3 & !Q4)
        # (!(Q0 & Q1 & Q2 & Q3) &  Q4) ) & enable)
        # (Q4 & !enable);

Q5.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 & !Q5)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4) &  Q5) ) & enable)
        # (Q5 & !enable);

Q6.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 & !Q6)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5) &  Q6) ) & enable)
    # (Q6 & !enable);

Q7.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 &  Q6 & !Q7)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6) &  Q7)) & enable)
        # (Q7 & !enable);

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

FB1_oe = (count_FB1 # (ACCESS & (!LUT_not_BUF & R_not_W & Buff_no)));
```

Lo_count.pld

```
Name     LO_Count;
Partno    02;
Date      12/04/89;
Revision  02;
Designer  Greg;
Company   Liverpool University;
Assembly  Fibre Analysis Board;
Device    p22v10;
Location  None yet;

/*************************************************************/
/*                                                           */
/* Octal Counter                                             */
/*                                                           */
/* 8-bit synchronous counter with CLEAR and ENABLE.          */
/* The CLEAR operation is asynchronous and resets            */
/* the output register to all LOWs.                          */
/* Setting output enable tri-states all outputs.             */
/*-----------------------------------------------------------*/
/*  Also enables Framw buffer outputs                        */
/*************************************************************/
/**       Allowable Target Device Types :  22V10            **/
/*************************************************************/

/**  Inputs  **/

PIN 1     = clock     ;    /* Register Clock       */
Pin 2     = enable    ; /* Enable count (Hold)    */
Pin 3     = clear     ; /* Clear Counter      */

Pin 7     = ACCESS;
Pin 8     = !count_FB0;
Pin 9     = Buff_no;
Pin 10    = R_not_W;
Pin 11    = LUT_not_BUF;

PIN 13    = OE       ;   /* Output enable         */

/**  Outputs  **/

PIN [15..22] = [Q7..0]  ;      /* Register Outputs    */
Pin 23         = rco       ;   /* Ripple carry out    */

Pin 14    = !FB0_oe;

/** Declarations and Intermediate Variable Definitions **/

FIELD value = [Q7..Q0];

Q0.ar = clear;              Q0.sp = 'b' 0;
Q1.ar = clear;              Q1.sp = 'b' 0;
Q2.ar = clear;              Q2.sp = 'b' 0;
Q3.ar = clear;              Q3.sp = 'b' 0;
Q4.ar = clear;              Q4.sp = 'b' 0;
Q5.ar = clear;              Q5.sp = 'b' 0;
Q6.ar = clear;              Q6.sp = 'b' 0;
Q7.ar = clear;              Q7.sp = 'b' 0;

rco.ar = clear;             rco.sp = 'b' 0;

Q0.oe = OE ;
Q1.oe = OE ;
Q2.oe = OE ;
Q3.oe = OE ;
Q4.oe = OE ;
Q5.oe = OE ;
Q6.oe = OE ;
Q7.oe = OE ;

rco.oe = OE;

/** Logic Equations **/

rco.d = value:'d'255;

Q0.d =    ((!Q0 & enable ) # (Q0 & !enable) );

Q1.d = ((( Q0 & !Q1)
        # (!Q0 &  Q1)) & enable)
         # (Q1 & !enable);

Q2.d = ((( Q0 &  Q1 & !Q2)
        # (!(Q0 & Q1) &  Q2) ) & enable)
         # (Q2 & !enable);

Q3.d = ((( Q0 &  Q1 &  Q2 & !Q3)
        # (!(Q0 & Q1 & Q2) &  Q3) ) & enable)
         # (Q3 & !enable);

Q4.d = (((Q0 &  Q1 &  Q2 &  Q3 & !Q4)
        # (!(Q0 & Q1 & Q2 & Q3) &  Q4) ) & enable)
        # (Q4 & !enable);

Q5.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 & !Q5)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4) &  Q5) ) & enable)
        # (Q5 & !enable);

Q6.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 & !Q6)
         # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5) &  Q6) ) & enable)
    # (Q6 & !enable);

Q7.d = ((( Q0 &  Q1 &  Q2 &  Q3 &  Q4 &  Q5 &  Q6 & !Q7)
        # (!(Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6) &  Q7)) & enable)
        # (Q7 & !enable);

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

FB0_oe = (count_FB0 # (ACCESS & (!LUT_not_BUF & R_not_W & !Buff_no)));
```

Run.pld :

```
Name    run;
Partno      06;
Date        06/07/89;
Revision    05;
Designer    Greg;
Company     University of Liverpool;
Assembly    Fibre Analysis Board;
Device       ep600;
Location    XXXXX;

/*************************************************/
/*  This one controls the continuous running   */
/*************************************************/
/*  Allowable Target Device Types:  EP600      */
/*************************************************/

/**  Inputs  **/

Pin 2     = RUN;
Pin 3     = FB0_A15;
Pin 4     = FB1_A15;
Pin 5     = CAPTURE;
Pin 6     = data_valid;
Pin 7     = write_buffer;  /*  output from flip-flop clocked by S_of_Field   */
Pin 11    = DISPLAY;
Pin 14   = buffer_no;
Pin 23    = hclk;      /*  counter clock     */

/**  Outputs  **/

Pin 22    = !FB0_w;
Pin 21    = !FB1_w;
Pin 20    = !FB0_64_cs;
Pin 19    = !FB0_32_cs;
Pin 18    = !FB1_64_cs;
Pin 17    = !FB1_32_cs;
Pin 16    = !COUNT_FB1;
Pin 15    = !COUNT_FB0;
Pin 10    = !LUT_1_oe;
Pin 9     = !LUT_0_oe;
Pin 8     = O_EN;

/** Declarations and Intermediate Variable Definitions **/

FB0_w.oe = O_EN;
FB1_w.oe = O_EN;
FB0_64_cs.oe = O_EN;
FB0_32_cs.oe = O_EN;
FB1_64_cs.oe = O_EN;
FB1_32_cs.oe = O_EN;

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

RUN	DISPLAY	CAPTURE	BUF|no	LUTnotBUF	READ!WRITE	Buffer 0 64|CS	Buffer 0 32|CS	Buffer 0 EN|OP	Buffer 0 LUT|OP	Buffer 1 64|CS	Buffer 1 32|CS	Buffer 1 EN|OP	Buffer 1 LUT|OP	
1	0	0	0	1	1	**********				**********				Reconstruct thru LUT
1	1	1	0	1	1	1<>0	1	0		1<>0	0	1		Dis buffer 0; Cap 1
1	1	1	1	1	1	1<>0	0	1		1<>0	1	0		Dis buffer 1; Cap 0
1	1	0	0	0	0	1<>0	1	0		0<>1	0	0		Display buffer 0
1	1	0	1	0	0	0<>1	0	0		1<>0	1	0		Display buffer 1
0	0	1	0	1	1	1<>0	0	1		0<>1	0	0		Capture into buffer 0
0	0	1	1	1	1	0<>1	0	0		1<>0	0	1		Capture into buffer 1

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

/** Logic Equations       **/

O_EN = (CAPTURE # DISPLAY # RUN);

/*-----------------------------------------------------------------------*/

FB1_64_cs =  ( FB1_A15 &
               ((!RUN &  DISPLAY &  CAPTURE &  buffer_no) #
                ( RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                ( RUN &  DISPLAY & !CAPTURE &  buffer_no) #
                (!RUN & !DISPLAY &  CAPTURE &  buffer_no)
                # (RUN & !DISPLAY & !CAPTURE)));

FB1_32_cs =  (!FB1_A15 &
               ((!RUN &  DISPLAY &  CAPTURE &  buffer_no) #
                ( RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                ( RUN &  DISPLAY & !CAPTURE &  buffer_no) #
                (!RUN & !DISPLAY &  CAPTURE &  buffer_no)
                # (RUN & !DISPLAY & !CAPTURE)));

FB0_64_cs =  ( FB0_A15 &
               ((!RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                ( RUN &  DISPLAY &  CAPTURE &  buffer_no) #
                ( RUN &  DISPLAY & !CAPTURE & !buffer_no) #
                (!RUN & !DISPLAY &  CAPTURE & !buffer_no)
                # (RUN & !DISPLAY & !CAPTURE)));

FB0_32_cs =  (!FB0_A15 &
               ((!RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                ( RUN &  DISPLAY &  CAPTURE &  buffer_no) #
```

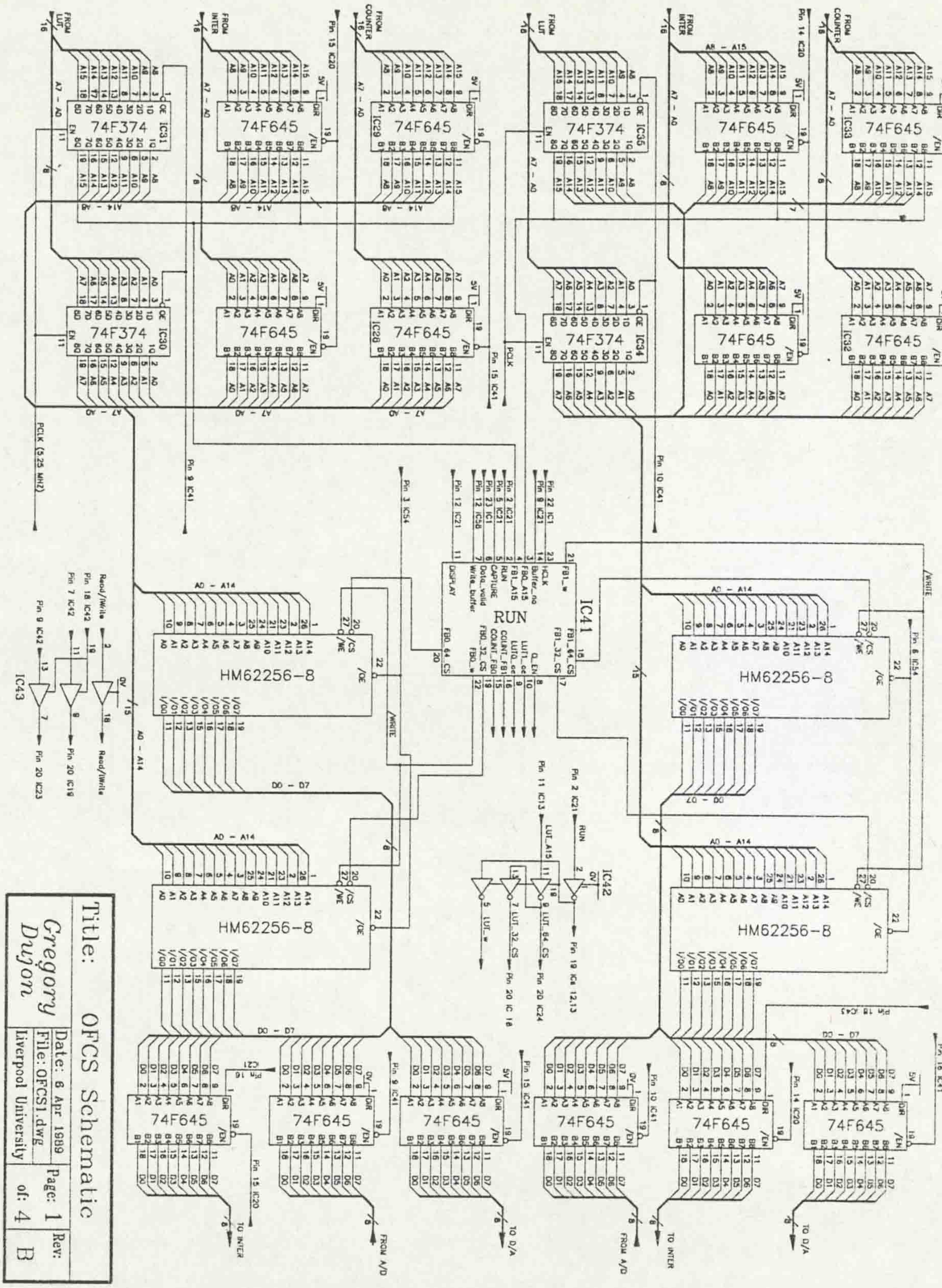
```
                ( RUN &  DISPLAY & !CAPTURE & !buffer_no) #
                (!RUN & !DISPLAY &  CAPTURE & !buffer_no)
                # (RUN & !DISPLAY & !CAPTURE)));

/*-------------------------------------------------------------------*/

COUNT_FB1 =     ( (( RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                  ( RUN &  DISPLAY & !CAPTURE &  buffer_no))
                  # (( RUN & !DISPLAY & !CAPTURE) & !write_buffer) );

COUNT_FB0 =     ( (( RUN &  DISPLAY &  CAPTURE &  buffer_no) #
                  ( RUN &  DISPLAY & !CAPTURE & !buffer_no))
                  # (( RUN & !DISPLAY & !CAPTURE) &  write_buffer) );

/*-------------------------------------------------------------------*/

LUT_1_oe =      ( (( RUN &  DISPLAY &  CAPTURE &  buffer_no) #
                  (!RUN & !DISPLAY &  CAPTURE &  buffer_no))
                  # (( RUN & !DISPLAY & !CAPTURE) & write_buffer) );

LUT_0_oe =      ( (( RUN &  DISPLAY &  CAPTURE & !buffer_no) #
                  (!RUN & !DISPLAY &  CAPTURE & !buffer_no))
                  # (( RUN & !DISPLAY & !CAPTURE) & !write_buffer) );

/*-------------------------------------------------------------------*/

FB1_w =           ( data_valid &
                ((( RUN &  DISPLAY &  CAPTURE & buffer_no & !write_buffer) #
                  (!RUN & !DISPLAY &  CAPTURE & buffer_no & !write_buffer) #
                  ( RUN & !DISPLAY & !CAPTURE & write_buffer))
                    & !hclk);

FB0_w =           ( data_valid &
                ((( RUN &  DISPLAY &  CAPTURE & !buffer_no & !write_buffer) #
                  (!RUN & !DISPLAY &  CAPTURE & !buffer_no & !write_buffer) #
                  ( RUN & !DISPLAY & !CAPTURE & !write_buffer))
                    & !hclk);
```

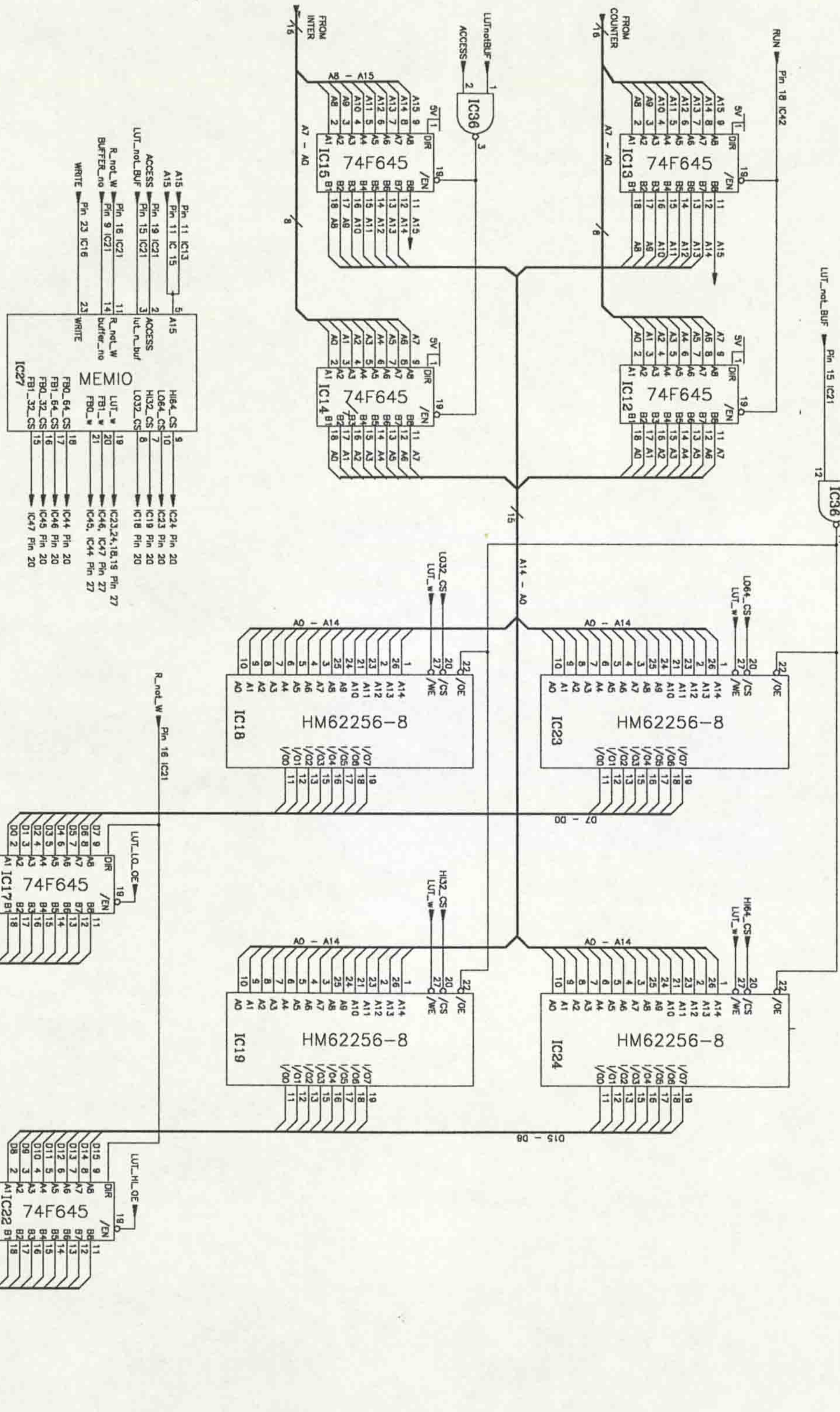
Memio.pld :

```
Name        memio;
Partno      IC27;
Date        30/07/89;
Revision    02;
Designer    Greg;
Company     University of Liverpool;
Assembly    Fibre Analysis Board;
Device       ep600;
Location    XX;

/*************************************************/
/*  This controls I/O to/from LUT                */
/*************************************************/
/*  Allowable Target Device Types:  EP600        */
/*************************************************/

/**  Inputs  **/

Pin 2    = access;
Pin 3    = lut_not_buf;
Pin 5    = A15;
Pin 11   = R_not_W;
Pin 14   = buffer_no;
Pin 23   = WRITE;

/**  Outputs  **/

Pin 21   = !FB0_w;
Pin 20   = !FB1_w;
Pin 19   = !LUT_w;
Pin 18   = !FB0_64_cs;
Pin 17   = !FB1_64_cs;
Pin 16   = !FB0_32_cs;
Pin 15   = !FB1_32_cs;
Pin 10   = !lo64_cs;
Pin 9    = !hi64_cs;
Pin 8    = !lo32_cs;
Pin 7    = !hi32_cs;
/*

    LUT chip select truth table

    Buffer number indicates whether high or low byte coming thru

      INPUTS          OUTPUTS
    ------------------------
    A           h  l  h  l
    C           i  o  i  o
    C           3  3  6  6
    E   L  A  B 2  2  4  4
    S   U  1  n
    S   T  5  o
    ------------------------
    1   1  0  0 0  1  0  0
    1   1  1  0 0  0  0  1
    1   1  0  1 1  0  0  0
    1   1  1  1 0  0  1  0

    FB chip select truth table

      INPUTS      OUTPUTS
    ------------------------
    A           1  1  0  0
    C           |  |  |  |
    C           6  3  6  3
    E   L  A  B 4  2  4  2
    S   U  1  n
    S   T  5  o
    ------------------------
    1   0  0  0 0  0  0  1
    1   0  1  0 0  0  1  0
    1   0  0  1 0  1  0  0
    1   0  1  1 1  0  0  0

*/
/** Declarations and Intermediate Variable Definitions **/

FB0_w.oe  = access;
FB1_w.oe  = access;
LUT_w.oe  = access;
FB0_64_cs.oe = access;
FB1_64_cs.oe = access;
FB0_32_cs.oe = access;
FB1_32_cs.oe = access;
lo64_cs.oe = access;
hi64_cs.oe = access;
lo32_cs.oe = access;
hi32_cs.oe = access;

/**  Logic Equations  **/

    lo32_cs =  (access & lut_not_buf & !A15 &  !buffer_no);
    lo64_cs =  (access & lut_not_buf &  A15 &  !buffer_no);
    hi32_cs =  (access & lut_not_buf & !A15 &  buffer_no);
    hi64_cs =  (access & lut_not_buf &  A15 &  buffer_no);
    FB0_64_cs =  (access & !lut_not_buf &  A15 & !buffer_no);
    FB0_32_cs =  (access & !lut_not_buf & !A15 & !buffer_no);
    FB1_64_cs =  (access & !lut_not_buf &  A15 &  buffer_no);
    FB1_32_cs =  (access & !lut_not_buf & !A15 &  buffer_no);


    LUT_w = (WRITE & lut_not_buf & access & !R_not_W);
    FB0_w = (WRITE & !lut_not_buf & access & !R_not_W & !buffer_no);
    FB1_w = (WRITE & !lut_not_buf & access & !R_not_W &  buffer_no);
```

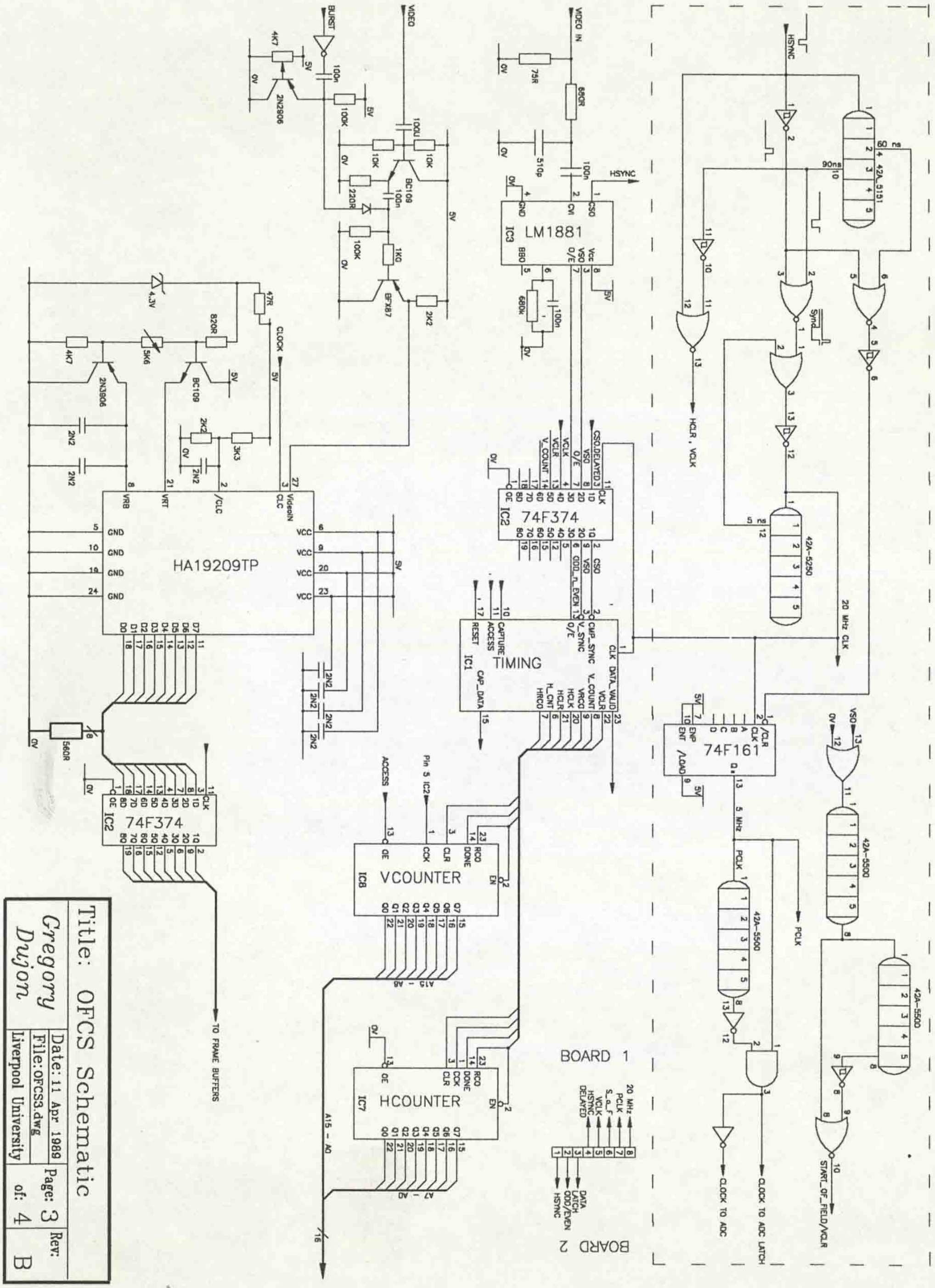
APPENDIX B.2

OFCS SCHEMATICS

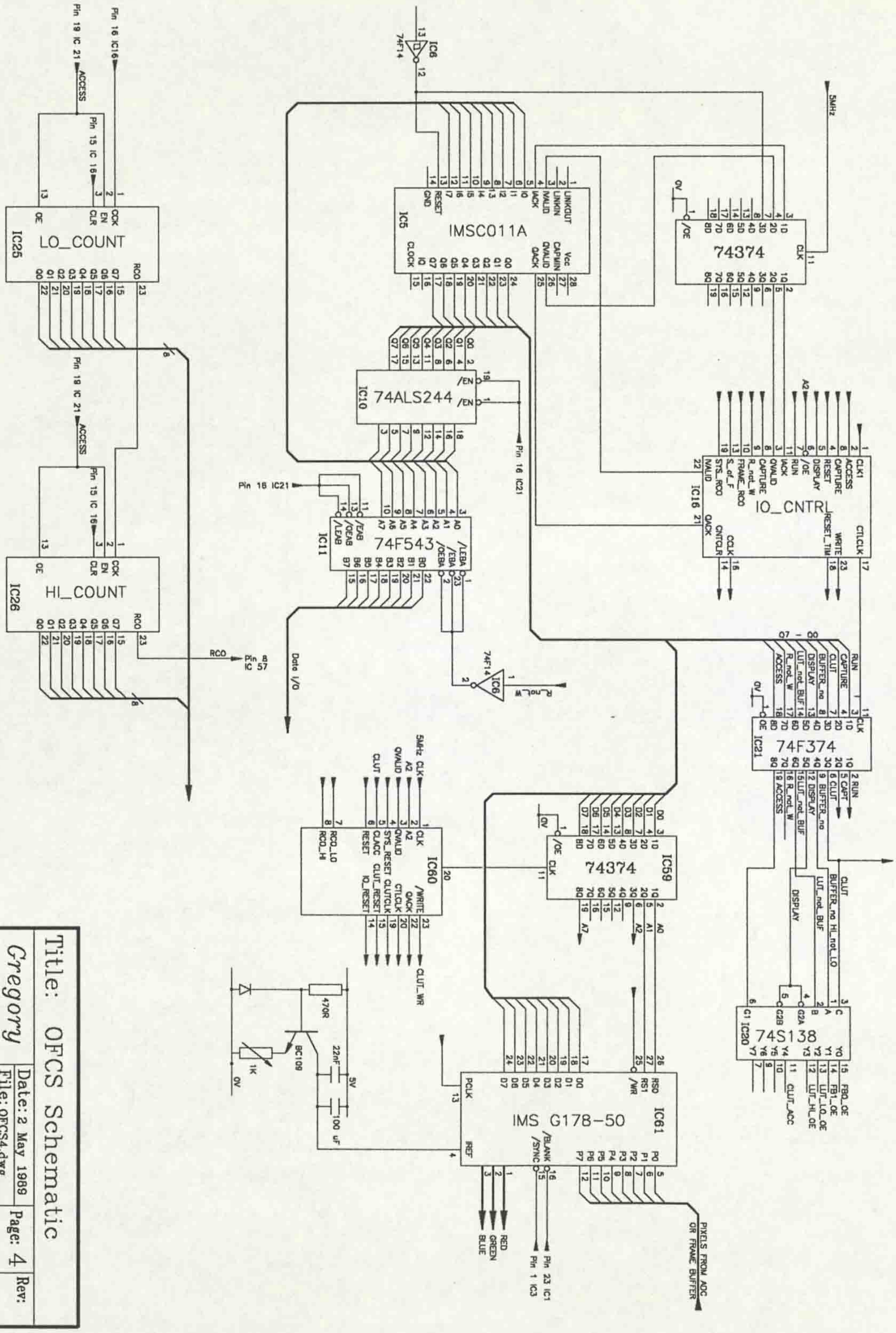

Title: OFCS Schematic
Gregory Dujon
Date: 6 Apr 1989
File: OFCS1.dwg
Liverpool University
Page: 1 of: 4
Rev: B
74F645
74F374
HM62256-8
RUN
IC41
IC42
IC43
FROM LUT
FROM INTER
FROM COUNTER
TO INTER
TO D/A
FROM A/D
PCLK (5.25 MHZ)
Read/Write



Title: OFCS Schematic	
Gregory Dujon	Date: 4 May 1989
	File: OFCS2.dwg
Liverpool University	Page: 2 of: 4
	Rev: A1



Title: OFCS Schematic		
Gregory Dujon	Date: 11 Apr 1989	Page: 3
	File: OFCS3.dwg	Rev:
	Liverpool University	of: 4
		B



Title: OFCS Schematic		
Gregory Dujon	Date: 2 May 1989	Page: 4
	File: OFCS4.dwg	Rev:
	Liverpool University	of: 4 B

APPENDIX B.3

CONTROL PROCEDURES FOR OFCS

Control Procedures for OFCS

To reset the system :

```
PROC reset.board()
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  INT s.s.reset :
  PLACE s.s.reset AT #20000000 :
  TIMER clock :
  VAL INT reset.time IS 15625/2 :
  INT time :
  SEQ
    s.s.reset := 1
    SEQ
      clock ? time
      clock ? AFTER time PLUS reset.time
    s.s.reset := 0
    SEQ
      clock ? time
      clock ? AFTER time PLUS reset.time
:
```

To initialise the display :

```
PROC init.G178()
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  BYTE control.byte :
  SEQ
    SEQ
      SEQ
        control.byte := #4(BYTE) -- CLUT access
        send.data ! control.byte
      SEQ
  control.byte := #6(BYTE) -- (CLUT access,A3),pixel mask
  send.data ! control.byte
  send.data ! #FF(BYTE)    -- No masking
SEQ
  SEQ
    control.byte := #4(BYTE) -- CLUT access,A3, write starting address
    send.data ! control.byte
    send.data ! #0(BYTE)
SEQ
  INT count :
  INT count.rgb :
  SEQ
    control.byte := #5(BYTE) -- CLUT access,A3 and !A1,A0,i.e write colour value
    SEQ do.it = 0 FOR 255
      SEQ
        count := 0
        WHILE count <= 255
          SEQ
            count.rgb := 0
            WHILE count.rgb < 3
              SEQ
                send.data ! control.byte
                send.data ! (BYTE (count))
                count.rgb := count.rgb + 1
            count := count + 1
SEQ
  control.byte := #10(BYTE) -- display B0
```

```
        send.data ! control.byte
:
```

To read the contents of a frame buffer :

```
PROC read.buffer(INT buffer.number,[256][256] BYTE buffer,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  BYTE control.byte :
  SEQ
    IF
      (buffer.number = 0)
        SEQ
          BYTE pixel :
          SEQ
            SEQ
              control.byte := #C0(BYTE)
              send.data ! control.byte
            SEQ y = 0 FOR 256
              SEQ x = 0 FOR 256
                SEQ
                  get.data ? pixel
                  buffer[y][x] := pixel
      (buffer.number = 1)
        SEQ
          BYTE pixel :
          SEQ
            SEQ
              control.byte := #C8(BYTE)
              send.data ! control.byte
            SEQ y = 0 FOR 256
              SEQ x = 0 FOR 256
                SEQ
                  get.data ? pixel
                  buffer[y][x] := pixel
      TRUE
        SEQ
          SEQ
            OK := FALSE
:
```

To write to a frame buffer :

```
PROC write.buffer(INT buffer.number,[256][256] BYTE data,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  BYTE control.byte :
  SEQ
    IF
      (buffer.number = 0)
        SEQ
          BYTE pixel :
          SEQ
            SEQ
              control.byte := #80(BYTE)
              send.data ! control.byte
            SEQ y = 0 FOR 256
              SEQ x = 0 FOR 256
                SEQ
                  pixel := data[y][x]
                  send.data ! pixel
      (buffer.number = 1)
        SEQ
          BYTE pixel :
          SEQ
            SEQ
              control.byte := #88(BYTE)
              send.data ! control.byte
            SEQ y = 0 FOR 256
              SEQ x = 0 FOR 256
                SEQ
                  pixel := data[y][x]
                  send.data ! pixel
      TRUE
        SEQ
          SEQ
            OK := FALSE
:
```

To read the contents of the LUT :

```
PROC read.lut([256][256][2] BYTE lut)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
```

```
VAL link2.in  IS  6:
VAL link3.in  IS  7:

VAL link0.out  IS  0:
VAL link1.out  IS  1:
VAL link2.out  IS  2:
VAL link3.out  IS  3:
CHAN OF BYTE send.data:
PLACE send.data AT link1.out :

CHAN OF BYTE get.data:
PLACE get.data AT link1.in :
BYTE control.byte :
SEQ
  SEQ
    SEQ
      control.byte := #E0(BYTE)
      send.data ! control.byte
    SEQ y = 0 FOR 256
      SEQ x = 0 FOR 256
        get.data ? lut[y][x][0]
  SEQ
    SEQ
      control.byte := #E8(BYTE)
      send.data ! control.byte
    SEQ y = 0 FOR 256
      SEQ x = 0 FOR 256
        get.data ? lut[y][x][1]
:
```

To set up the LUT :

```
PROC write.lut([256][256][2] BYTE lut)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  BYTE control.byte :
  SEQ
    SEQ
      SEQ
        control.byte := #A0(BYTE)
        send.data ! control.byte
      SEQ y = 0 FOR 256
        SEQ x = 0 FOR 256
          SEQ
            send.data ! lut[y][x][0]
    SEQ
      SEQ
        control.byte := #A8(BYTE)
        send.data ! control.byte
      SEQ y = 0 FOR 256
        SEQ x = 0 FOR 256
          SEQ
            send.data ! lut[y][x][1]
:
```

To capture a frame :

```
PROC capture.buffer(INT buffer.number,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE send.data:
  PLACE send.data AT link1.out :

  CHAN OF BYTE get.data:
  PLACE get.data AT link1.in :
  BYTE control.byte, captured.byte :
  SEQ
    IF
      (buffer.number = 0)
        SEQ
          control.byte := #2(BYTE)
          send.data ! control.byte
          get.data ? captured.byte

      (buffer.number = 1)
        SEQ
          control.byte := #A(BYTE)
          send.data ! control.byte
          get.data ? captured.byte

      TRUE
        OK := FALSE
:
```

To display a frame :

```
PROC display.buffer(INT buffer.number,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
```

```
VAL link2.in  IS  6:
VAL link3.in  IS  7:

VAL link0.out  IS  0:
VAL link1.out  IS  1:
VAL link2.out  IS  2:
VAL link3.out  IS  3:
CHAN OF BYTE send.data:
PLACE send.data AT link1.out :

CHAN OF BYTE get.data:
PLACE get.data AT link1.in :
BYTE control.byte :
SEQ
  IF
    (buffer.number = 0)
      SEQ
        control.byte := #71(BYTE)
        send.data ! control.byte

    (buffer.number = 1)
      SEQ
        control.byte := #79(BYTE)
        send.data ! control.byte

    TRUE
      OK := FALSE
:
```

APPENDIX C

SOFTWARE FOR SPOT CALIBRATION ON OFCS

Spot Calibration on OFCS

```
#USE ioconv
#USE interf
#USE msdos
#USE filerhdr
#USE krnlhdr
#USE userio
VAL X IS 0 :
VAL Y IS 1 :
BOOL OK :
INT key :
INT fibre.diameter, step.size, threshold,area :
INT window.size :
[250] INT square :
[250] INT travel.limits :
INT number.of.steps, number.of.steps.in.radius :
INT x.travel, y.travel :
INT y.in, x.in :
[9] BYTE x.go.forward, x.go.backward:
[9] BYTE x.move.forward, x.move.backward:
[9] BYTE y.go.forward, y.go.backward:
[9] BYTE y.move.forward, y.move.backward:
[9] BYTE x.step, y.step :
[256][256][2] BYTE output.points :
[256][256] BYTE clear.buffer :
TIMER clock :
VAL INT character.time IS (15625/960) :
INT start.time, end.time,time :
VAL link0.in  IS  4:
VAL link1.in  IS  5:
VAL link2.in  IS  6:
VAL link3.in  IS  7:

VAL link0.out  IS  0:
VAL link1.out  IS  1:
VAL link2.out  IS  2:
VAL link3.out  IS  3:
CHAN OF ANY to.oriel,from.oriel:
PLACE to.oriel   AT link3.out:
PLACE from.oriel AT link3.in:
CHAN OF BOOL is.going:

PROC open.file.write(CHAN OF ANY screen,keyboard,from.filer,to.filer)
  #USE userhdr
  #USE interf
  #USE msdos
  #USE filerhdr
  #USE krnlhdr
  #USE userio
  BOOL abort :
  INT open.error,result,char,any :
  [63]BYTE file.name:
  [63]BYTE dos.name:
  INT file.name.len,dos.name.len:
  SEQ
    abort := FALSE
    result := fi.ok
    open.error := fi.ok

    goto.xy(screen,0,22)
    write.full.string(screen,"Filename : ")
    read.echo.text.line(keyboard,screen,file.name.len,file.name,char)

    dos.name.len := file.name.len - 1   --ignore carriage ret
    [dos.name FROM 0 FOR dos.name.len] := [file.name FROM 0 FOR dos.name.len]
    open.tkf.file(from.filer,to.filer,tkf.open.write,
                  dos.name.len,dos.name,open.error)

    IF
      (open.error = fi.ok)AND( result = fi.ok)
        SEQ
          goto.xy(screen,0,23)
          clear.eos(screen)
          goto.xy(screen,0,23)
          write.full.string(screen,"File Opened O.K ... ")
      TRUE
        SEQ
          goto.xy(screen,0,23)
          clear.eos(screen)
          goto.xy(screen,0,23)
          write.full.string(screen,"Unable to open FILE -- ABORTING!!!")
          abort := TRUE
          goto.xy(screen,40,23)
          write.full.string(screen,"Err. No. ")
          write.int(screen,open.error,6)
          keyboard ? any
:
PROC open.file.read(CHAN OF ANY screen, keyboard,from.filer,to.filer)
  #USE interf
```

```
#USE msdos
#USE filerhdr
#USE krnlhdr
#USE userio
BOOL abort :
INT open.error,result,char,any :
SEQ
  [63]BYTE file.name:
  [63]BYTE dos.name:
  INT file.name.len,dos.name.len:
  SEQ
    abort := FALSE
    result := fi.ok
    open.error := fi.ok

    goto.xy(screen,0,22)
    write.full.string(screen,"Filename : ")
    read.echo.text.line(keyboard,screen,file.name.len,file.name,char)

    dos.name.len := file.name.len - 1   --ignore carriage ret
    [dos.name FROM 0 FOR dos.name.len] := [file.name FROM 0 FOR dos.name.len]

    open.tkf.file(from.filer,to.filer,tkf.open.read,
                  dos.name.len,dos.name,open.error)

    IF
      (open.error = fi.ok)AND( result = fi.ok)
        SEQ
          goto.xy(screen,0,23)
          clear.eos(screen)
          goto.xy(screen,0,23)
          write.full.string(screen,"File Opened O.K ... ")
      TRUE
        SEQ
          goto.xy(screen,0,23)
          clear.eos(screen)
          goto.xy(screen,0,23)
          write.full.string(screen,"Unable to open FILE -- ABORTING!!!")
          abort := TRUE
          goto.xy(screen,40,23)
          write.full.string(screen,"Err. No. ")
          write.int(screen,open.error,6)
          keyboard ? any
:
PROC close.file(CHAN OF ANY from.filer,to.filer)
  #USE interf
  #USE msdos
  #USE filerhdr
  #USE krnlhdr
  #USE userio
  INT result :
  SEQ
    result := fi.ok
    close.tkf.file(from.filer,to.filer,result)
:
PROC file.buffer(CHAN OF ANY screen,keyboard,from.filer,to.filer,
                 [256][256] BYTE buffer,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE to.fab :
  PLACE to.fab AT link2.out:
  CHAN OF BYTE from.fab:
  PLACE from.fab AT link2.in:
  #USE interf
  #USE msdos
  #USE filerhdr
  #USE krnlhdr
  #USE userio
  VAL INT data.length IS (64 * 1024) :
  BOOL abort :
  INT open.error,result,char,any :
  [data.length] BYTE data RETYPES buffer :
  SEQ
    abort := FALSE
    result := fi.ok
    open.error := fi.ok
    IF
      abort = FALSE
        SEQ
          SEQ
            write.full.string(screen,"Waiting for DATA ... ")
          VAL block.size IS 512:
          SEQ
            SEQ x = 0 FOR ( data.length / block.size )
              SEQ
                IF
                  (open.error = fi.ok)AND( result = fi.ok)
                    SEQ
                      write.tkf.block(from.filer,to.filer,block.size,
                                      [data FROM (x * block.size) FOR block.size],
                                       result)
                  TRUE
                    SKIP
          SEQ
            goto.xy(screen,10,13)
            write.full.string(screen,"File CLosed ... Any Key?")
      TRUE
        OK := FALSE
:
PROC file.lut(CHAN OF ANY screen,keyboard,from.filer,to.filer,
                [256][256][2] BYTE lut,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE to.fab :
  PLACE to.fab AT link2.out:
```

```
CHAN OF BYTE from.fab:
PLACE from.fab AT link2.in:
#USE interf
#USE msdos
#USE filerhdr
#USE krnlhdr
#USE userio
VAL INT data.length IS (64 * 1024) :
BOOL abort :
INT open.error,result,char,any :
[data.length] BYTE x.from.fab :
[data.length] BYTE y.from.fab :
SEQ
  abort := FALSE
  result := fi.ok
  open.error := fi.ok
  SEQ a = 0 FOR 256
    SEQ b = 0 FOR 256
      SEQ
        x.from.fab[((a*256)+b)] := lut[a][b][0]
        y.from.fab[((a*256)+b)] := lut[a][b][1]
  IF
    abort = FALSE
      SEQ
        SEQ
          write.full.string(screen,"Waiting for DATA ... ")
        VAL block.size IS 512:
        SEQ
          SEQ x = 0 FOR ( data.length / block.size )
            SEQ
              IF
                (open.error = fi.ok)AND( result = fi.ok)
                  SEQ
                    write.tkf.block(from.filer,to.filer,block.size,
                                    [x.from.fab FROM (x * block.size) FOR block.size],
                                    result)
          SEQ x = 0 FOR ( data.length / block.size )
            SEQ
              IF
                (open.error = fi.ok)AND( result = fi.ok)
                  SEQ
                    write.tkf.block(from.filer,to.filer,block.size,
                                    [y.from.fab FROM (x * block.size) FOR block.size],
                                    result)
              TRUE
                SKIP
        SEQ
          goto.xy(screen,10,13)
          write.full.string(screen,"File CLosed ... Any Key?")
    TRUE
      OK := FALSE
:
PROC get.buffer.file(CHAN OF ANY screen,keyboard,from.filer,to.filer,
                     [256][256] BYTE buffer,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE to.fab :
  PLACE to.fab AT link2.out:
  CHAN OF BYTE from.fab:
  PLACE from.fab AT link2.in:
  #USE interf
  #USE msdos
  #USE filerhdr
  #USE krnlhdr
  #USE userio
  VAL INT data.length IS (64 * 1024) :
  BOOL abort :
  INT open.error,result,char,any :
  [data.length] BYTE data :
  SEQ
    abort := FALSE
    result := fi.ok
    open.error := fi.ok
    IF
      abort = FALSE
        SEQ
          SEQ
            write.full.string(screen,"Reading DATA ... ")
          INT block.size :
          INT  number.of.block :
          VAL block.length IS 512 :
          SEQ
            block.size := 512
            SEQ number.of.block = 0 FOR ( data.length / block.length)
              IF
                open.error = fi.ok
                  SEQ
                    read.tkf.block(from.filer,to.filer,block.size,
                    [data FROM (number.of.block * block.length) FOR block.length],
                    result)
                TRUE
                  SKIP
          [256][256] BYTE image RETYPES data :
          SEQ
            buffer := image
          SEQ
            goto.xy(screen,10,13)
            write.full.string(screen,"File CLosed ... Any Key?")
      TRUE
        OK := FALSE
:
PROC exe.to.prog.buffer([256][256] BYTE buffer)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE to.fab :
```

```
  PLACE to.fab AT link2.out:
  CHAN OF BYTE from.fab:
  PLACE from.fab AT link2.in:
  SEQ
    SEQ y = 0 FOR 256
      SEQ x = 0 FOR 256
        to.fab ! buffer[y][x]
:
PROC get.lut.file(CHAN OF ANY screen,keyboard,from.filer,to.filer,
                  [256][256][2] BYTE lut,BOOL OK)
  VAL link0.in  IS  4:
  VAL link1.in  IS  5:
  VAL link2.in  IS  6:
  VAL link3.in  IS  7:

  VAL link0.out  IS  0:
  VAL link1.out  IS  1:
  VAL link2.out  IS  2:
  VAL link3.out  IS  3:
  CHAN OF BYTE to.fab :
  PLACE to.fab AT link2.out:
  CHAN OF BYTE from.fab:
  PLACE from.fab AT link2.in:
  #USE interf
  #USE msdos
  #USE filerhdr
  #USE krnlhdr
  #USE userio
  VAL INT data.length IS (64 * 1024) :
  BOOL abort :
  INT open.error,result,char,any :
  [data.length] BYTE x.fab.data :
  [data.length] BYTE y.fab.data :
  SEQ
    abort := FALSE
    result := fi.ok
    open.error := fi.ok
    IF
      abort = FALSE
        SEQ
          SEQ
            write.full.string(screen,"Reading DATA ... ")
          INT block.size :
          INT  number.of.block :
          VAL block.length IS 512 :
          SEQ
            block.size := 512
            SEQ number.of.block = 0 FOR ( data.length / block.length)
              IF
                open.error = fi.ok
                  SEQ
                    read.tkf.block(from.filer,to.filer,block.size,
                    [x.fab.data FROM (number.of.block * block.length) FOR block.length],
                    result)
            block.size := 512
            SEQ number.of.block = 0 FOR ( data.length / block.length)
              IF
                open.error = fi.ok
                  SEQ
                    read.tkf.block(from.filer,to.filer,block.size,
                    [y.fab.data FROM (number.of.block * block.length) FOR block.length],
                    result)
                TRUE
                  SKIP
          SEQ a = 0 FOR 256
            SEQ b = 0 FOR 256
              SEQ
                lut[a][b][0] := x.fab.data[((a*256)+b)]
                lut[a][b][1] := y.fab.data[((a*256)+b)]
          SEQ
            goto.xy(screen,10,13)
            write.full.string(screen,"File CLosed ... Any Key?")
      TRUE
        OK := FALSE
:
SEQ
  [256][256][2] BYTE LUT :
  SEQ
    reset.board()
    SEQ count = 0 FOR 8
      init.G178()
    INT count1,count2 :
    BYTE data :
    SEQ
      count1 := 0
      WHILE count1 <= 255
        SEQ
          count2 := 0
          WHILE count2 <= 255
            SEQ
              data := BYTE(count2)
              LUT[count1][count2][0] := data
              count2 := count2 + 1
          count1 := count1 + 1
    count1 := 0
    WHILE count1 <= 255
      SEQ
        count2 := 0
        WHILE count2 <= 255
          SEQ
            data := BYTE(count1)
            LUT[count1][count2][1] := data
            count2 := count2 + 1
        count1 := count1 + 1

    write.lut(LUT)
SEQ
  goto.xy(screen,0,0)
  clear.eos(screen)
  goto.xy(screen,15,1)
  write.full.string(screen,"Optical Fibre Calibration by Single Spot Method")
INT char :
SEQ
  char := 0
  goto.xy(screen,5,3)
  write.full.string(screen,"Enter Fibre Bundle Diameter ( mm/10 ): ")
  read.echo.int(keyboard,screen,fibre.diameter,char)
```

```
  char := 0
  goto.xy(screen,5,4)
  write.full.string(screen,"Enter pinhole step size (microns) : ")
  read.echo.int(keyboard,screen,step.size,char)

  char := 0
  goto.xy(screen,5,5)
  write.full.string(screen,"Enter pixel brightness threshold ( 0 -> 255 ) : ")
  read.echo.int(keyboard,screen,threshold,char)

  char := 0
  goto.xy(screen,5,6)
  write.full.string(screen,"Enter typical single fibre pixel area : ")
  read.echo.int(keyboard,screen,area,char)
SEQ a = 0 FOR 256
  SEQ b = 0 FOR 256
    SEQ c = 0 FOR 2
      output.points[a][b][c] := 0(BYTE)
SEQ
  SEQ a = 0 FOR 256
    SEQ b = 0 FOR 256
      clear.buffer[a][b] := 0(BYTE)
PAR
  SEQ
    clock ? start.time
INT square.of.radius,radius :
INT square.y.step,square.x.limit :
INT square.window :
INT index :
SEQ
  SEQ entry = 0 FOR 250
    square[entry] := (entry * entry)
  goto.xy(screen,0,0)
  clear.eos(screen)
  radius := (fibre.diameter/2)
  number.of.steps.in.radius := (((radius * 100) / step.size) + 1)
  goto.xy(screen,5,0)
  write.full.string(screen,"Number of steps in radius is : ")
  write.int(screen,number.of.steps.in.radius,5)
  goto.xy(screen,5,0)
  square.of.radius := square[number.of.steps.in.radius]
  SEQ y.step = 0 FOR (number.of.steps.in.radius + 1)
    SEQ
      square.y.step := square[y.step]
      square.x.limit := (square.of.radius - square.y.step)
      index := 0
      WHILE (square.x.limit > square[index])
        index := index + 1
      travel.limits[y.step] := (index + 1)
  SEQ
    index := 0
    square.window := area
    WHILE (square.window > square[index])
      index := index + 1
    window.size := (index - 1)
  SEQ
    SEQ count = 0 FOR (number.of.steps.in.radius + 1)
      SEQ
        IF
          (count < 20)
            SEQ
              goto.xy(screen,0,(count+2))
              write.int(screen,count,3)
              write.int(screen,travel.limits[count],5)

          ((count >= 20) AND (count < 40))
            SEQ
              goto.xy(screen,12,(count - 18))
              write.int(screen,count,3)
              write.int(screen,travel.limits[count],5)

          ((count >= 40) AND (count < 60))
            SEQ
              goto.xy(screen,24,(count - 38))
              write.int(screen,count,3)
              write.int(screen,travel.limits[count],5)

          TRUE
            SEQ
              goto.xy(screen,36,(count - 58))
              write.int(screen,count,3)
              write.int(screen,travel.limits[count],5)
  keyboard ? key
  SEQ
    goto.xy(screen,0,0)
    clear.eos(screen)
    goto.xy(screen,5,5)
    write.full.string(screen,"Initialising Oriel ... ")
[4] BYTE string.steps :
SEQ
  PAR
    PAR
      SEQ
        x.go.forward[0] := 'A'
        x.go.forward[1] := 'G'
        x.go.forward[2] := '+'
        x.go.forward[7] := ','
        x.go.forward[8] := 'S'
      SEQ
        x.go.backward[0] := 'A'
        x.go.backward[1] := 'G'
        x.go.backward[2] := '-'
        x.go.backward[7] := ','
        x.go.backward[8] := 'S'
      SEQ
        y.go.forward[0] := 'B'
        y.go.forward[1] := 'G'
        y.go.forward[2] := '+'
        y.go.forward[7] := ','
        y.go.forward[8] := 'S'
      SEQ
        y.go.backward[0] := 'B'
        y.go.backward[1] := 'G'
        y.go.backward[2] := '-'
        y.go.backward[7] := ','
        y.go.backward[8] := 'S'
  PAR
    SEQ count = 3 FOR 4
      x.go.forward[count] := '0'
```

```
    SEQ count = 3 FOR 4
      x.go.backward[count] := '0'
    SEQ count = 3 FOR 4
      y.go.forward[count] := '0'
    SEQ count = 3 FOR 4
      y.go.backward[count] := '0'
  PAR
    SEQ
      x.move.forward[0] := 'A'
      x.move.forward[1] := 'T'
      x.move.forward[6] := ','
      x.move.forward[7] := '+'
      x.move.forward[8] := 'S'
    SEQ
      x.move.backward[0] := 'A'
      x.move.backward[1] := 'T'
      x.move.backward[6] := ','
      x.move.backward[7] := '-'
      x.move.backward[8] := 'S'
    SEQ
      y.move.forward[0] := 'B'
      y.move.forward[1] := 'T'
      y.move.forward[6] := ','
      y.move.forward[7] := '+'
      y.move.forward[8] := 'S'
      SEQ
        y.move.backward[0] := 'B'
        y.move.backward[1] := 'T'
        y.move.backward[6] := ','
        y.move.backward[7] := '-'
        y.move.backward[8] := 'S'
      PAR
        SEQ count = 2 FOR 4
          x.move.forward[count] := '0'
        SEQ count = 2 FOR 4
          x.move.backward[count] := '0'
        SEQ count = 2 FOR 4
          y.move.forward[count] := '0'
        SEQ count = 2 FOR 4
          y.move.backward[count] := '0'
      INT len :
      [8] BYTE str :
      SEQ
        SEQ count = 0 FOR 4
          string.steps[count] := '0'
        number.of.steps := (step.size / 2)
        INTTOSTRING(len,str,number.of.steps)
        [string.steps FROM (4 - len) FOR len] := [str FROM 0 FOR len]
    SEQ
      x.step := x.move.forward
      [x.step FROM 2 FOR 4] := string.steps

      y.step := y.move.forward
      [y.step FROM 2 FOR 4] := string.steps
SEQ
  BOOL quit :
  SEQ
    quit := FALSE
    SEQ
  to.oriel ! 'A'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! 'E'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
SEQ
  to.oriel ! 'B'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! 'E'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
SEQ
  to.oriel ! 'R'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! '7'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
SEQ
  SEQ command.index = 0 FOR 9
    SEQ
      to.oriel ! x.go.forward[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
  BYTE response :
  SEQ
    ALT
      from.oriel ? response
        SEQ
          from.oriel ? response
      keyboard ? key
        SEQ
          IF
            key = (INT('q'))
              quit := TRUE
            TRUE
              SKIP
SEQ
  SEQ command.index = 0 FOR 9
    SEQ
      to.oriel ! y.go.forward[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
  BYTE response :
  SEQ
    ALT
      from.oriel ? response
        SEQ
          from.oriel ? response
```

```
      keyboard ? key
        SEQ
          IF
            key = (INT('q'))
              quit := TRUE
            TRUE
              SKIP
[8] BYTE str :
INT len :
SEQ
  INT back.index.str :
  SEQ
    back.index.str := (travel.limits[number.of.steps.in.radius] *
                                                    number.of.steps)
    INTTOSTRING(len,str,back.index.str)
    [x.go.backward FROM (7 - len) FOR len] := [str FROM 0 FOR len]
  SEQ
    SEQ command.index = 0 FOR 9
      SEQ
        to.oriel ! x.go.backward[command.index]
        SEQ
          clock ? time
          clock ? AFTER time PLUS character.time
    BYTE response :
    SEQ
      ALT
        from.oriel ? response
          SEQ
            from.oriel ? response
        keyboard ? key
          SEQ
            IF
              key = (INT('q'))
                quit := TRUE
              TRUE
                SKIP
[8] BYTE str :
INT len :
SEQ
  INT back.index.str :
  SEQ
    back.index.str := (number.of.steps.in.radius * number.of.steps)
    INTTOSTRING(len,str,back.index.str)
    [y.go.backward FROM (7 - len) FOR len] := [str FROM 0 FOR len]
  SEQ
    SEQ command.index = 0 FOR 9
      SEQ
        to.oriel ! y.go.backward[command.index]
        SEQ
          clock ? time
          clock ? AFTER time PLUS character.time
    BYTE response :
    SEQ
      ALT
        from.oriel ? response
          SEQ
            from.oriel ? response
        keyboard ? key
          SEQ
            IF
              key = (INT('q'))
                quit := TRUE
              TRUE
                SKIP
  y.travel := 0
  x.travel := number.of.steps.in.radius
CHAN OF BOOL arrived,captured,end :
CHAN OF INT coordinates :
PAR
  INT back.index, forward.index :
  [8] BYTE str :
  [9] BYTE x.back :
  INT len, char :
  BOOL go.on,here :
  BOOL finished :
  SEQ
    finished := FALSE
    SEQ
      to.oriel ! 'B'
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
      to.oriel ! '+'
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
    SEQ steps = 0 FOR number.of.steps.in.radius
      SEQ
        SEQ
          forward.index :=
                    travel.limits[(number.of.steps.in.radius - steps)]
          back.index :=
                    travel.limits[(number.of.steps.in.radius - (steps+1))]
        INT back.index.str :
        SEQ
          back.index.str := (back.index * number.of.steps)
          INTTOSTRING(len,str,back.index.str)
          x.back := x.go.backward
          [x.back FROM (7 - len) FOR len] := [str FROM 0 FOR len]
        SEQ
          to.oriel ! 'A'
          SEQ
            clock ? time
            clock ? AFTER time PLUS character.time
          to.oriel ! '+'
          SEQ
            clock ? time
            clock ? AFTER time PLUS character.time
        BYTE response :
        SEQ
          go.on := TRUE
          SEQ count = 0 FOR (2 * forward.index)
            SEQ
              here := FALSE
              go.on := TRUE
              SEQ
                WHILE ((go.on = TRUE) AND (here = FALSE))
                  SEQ
                    SEQ command.index = 0 FOR 9
```

```
                SEQ
                  to.oriel ! x.step[command.index]
                  SEQ
                    clock ? time
                    clock ? AFTER time PLUS character.time
              BYTE carriage :
              SEQ
                from.oriel ? response
                from.oriel ? carriage
              INT any :
              SEQ
                IF
                  response = 'a'
                    SEQ
                      here := TRUE
                      arrived ! here
                  TRUE
                    arrived ! here
            x.travel := (x.travel + 1)
            SEQ
              coordinates ! x.travel
              coordinates ! y.travel
            SEQ
              captured ? go.on
SEQ
  to.oriel ! 'A'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! '-'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
BYTE response :
SEQ
  SEQ command.index = 0 FOR 9
    SEQ
      to.oriel ! x.back[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
  BYTE carriage :
  SEQ
    from.oriel ? response
    from.oriel ? carriage
x.travel := (x.travel - (forward.index + back.index))
SEQ
  to.oriel ! 'B'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! '+'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
BYTE response :
SEQ
  SEQ command.index = 0 FOR 9
    SEQ
      to.oriel ! y.step[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
    BYTE carriage :
    SEQ
      from.oriel ? response
      from.oriel ? carriage
  y.travel := (y.travel + 1)
SEQ steps = 1 FOR (number.of.steps.in.radius-1)
  SEQ
    forward.index :=
                travel.limits[steps]
    back.index :=
                travel.limits[(steps + 1)]
    INT back.index.str :
    SEQ
      back.index.str := (back.index * number.of.steps)
      INTTOSTRING(len,str,back.index.str)
      x.back := x.go.backward
      [x.back FROM (7 - len) FOR len] := [str FROM 0 FOR len]
    SEQ
      to.oriel ! 'A'
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
      to.oriel ! '+'
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
    BYTE response :
    SEQ
      go.on := TRUE
      SEQ count = 0 FOR (2 * forward.index)
        SEQ
here := FALSE
SEQ
  WHILE ((go.on = TRUE) AND (here = FALSE))
    SEQ
      SEQ command.index = 0 FOR 9
        SEQ
          to.oriel ! x.step[command.index]
          SEQ
            clock ? time
            clock ? AFTER time PLUS character.time
      BYTE carriage :
      SEQ
        from.oriel ? response
        from.oriel ? carriage
      INT any :
      SEQ
        IF
          response = 'a'
            SEQ
              here := TRUE
              arrived ! here
          TRUE
            arrived ! here
x.travel := (x.travel + 1)
SEQ
```

```
        coordinates ! x.travel
        coordinates ! y.travel
      SEQ
        captured ? go.on
        IF
          go.on = TRUE
            SKIP
          TRUE
            go.on := FALSE
SEQ
  to.oriel ! 'A'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! '-'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
BYTE response :
SEQ
  SEQ command.index = 0 FOR 9
    SEQ
      to.oriel ! x.back[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
      BYTE carriage :
      SEQ
        from.oriel ? response
        from.oriel ? carriage
      x.travel := (x.travel - (forward.index + back.index))
      SEQ
        to.oriel ! 'B'
        SEQ
          clock ? time
          clock ? AFTER time PLUS character.time
        to.oriel ! '+'
        SEQ
          clock ? time
          clock ? AFTER time PLUS character.time
      BYTE response :
      SEQ
        SEQ command.index = 0 FOR 9
          SEQ
            to.oriel ! y.step[command.index]
            SEQ
              clock ? time
              clock ? AFTER time PLUS character.time
        BYTE carriage :
        SEQ
          from.oriel ? response
          from.oriel ? carriage
      y.travel := (y.travel + 1)
  finished := TRUE
  end ! finished
BOOL snap.image, done :
BOOL stop :
INT buffer.no :
[256][256] BYTE image :
INT x.in,y.in :
INT total.output :
SEQ
  SEQ
    goto.xy(screen,0,0)
    clear.eos(screen)
    goto.xy(screen,10,1)
    write.full.string(screen,"The Image Processing ")
    goto.xy(screen,5,3)
    write.full.string(screen,"Input coords       Chosen Output")
    goto.xy(screen,5,7)
    write.full.string(screen,"Total ")
  buffer.no := 0
  total.output := 0
  stop := FALSE
  WHILE (stop = FALSE)
    SEQ
      ALT
        arrived ? snap.image
          SEQ
            SEQ
              coordinates ? x.in
              coordinates ? y.in
              goto.xy(screen,5,5)
              write.int(screen,x.in,5)
              write.int(screen,y.in,5)
            SEQ
              IF
                snap.image = TRUE
                  SEQ
                    INT any :
                    SEQ
                      OK := FALSE
                      capture.buffer(buffer.no,OK)
                      IF
                        OK = TRUE
                          SEQ
                            done := TRUE
                            captured ! done
                            read.buffer(buffer.no,image,OK)
                        TRUE
                          SEQ
                            done := FALSE
                            captured ! done
[512][2] INT illuminated.pixels :
[512] INT sum.of.intensity :
INT number.illuminated :
INT brightest,found.brightest :
SEQ
  SEQ a = 0 FOR 2
    SEQ b = 0 FOR 512
      illuminated.pixels[b][a] := 0
  INT entry :
  INT pixel.value :
  SEQ
    entry := 0
    SEQ y = 0 FOR 256
      SEQ x = 0 FOR 256
        SEQ
          pixel.value := (INT(image[y][x]))
```

```
            IF
              pixel.value >= threshold
                SEQ
                  IF
                    (entry < 511)
                      SEQ
                        illuminated.pixels[entry][X] := x
                        illuminated.pixels[entry][Y] := y
                        entry := entry + 1
                    TRUE
                      SKIP
              TRUE
                SKIP
    number.illuminated := entry
  INT x.centre,y.centre :
  INT x.start,y.start :
  SEQ
    SEQ a = 0 FOR 512
      sum.of.intensity[a] := 0
    SEQ pixel = 0 FOR number.illuminated
      SEQ
        x.centre := illuminated.pixels[pixel][X]
        y.centre := illuminated.pixels[pixel][Y]
        x.start := (x.centre - (window.size/2))
        y.start := (y.centre - (window.size/2))
        SEQ
          IF
            x.start < 0
              x.start := 0
            TRUE
              SKIP
          IF
            x.start > (255 - window.size)
              x.start := (255 - window.size)
            TRUE
              SKIP
          IF
            y.start < 0
              y.start := 0
            TRUE
              SKIP
          IF
            y.start > (255 - window.size)
              y.start := (255 - window.size)
            TRUE
              SKIP
        SEQ y = y.start FOR window.size
          SEQ x = x.start FOR window.size
            SEQ
              sum.of.intensity[pixel] :=
                                  ( sum.of.intensity[pixel] + (INT(image[y][x])) )
  IF
    number.illuminated <> 0
      SEQ
        SEQ
          brightest := 0
          SEQ pixel = 0 FOR number.illuminated
            SEQ
              IF
                sum.of.intensity[pixel] > brightest
                  SEQ
                    brightest := sum.of.intensity[pixel]
                    found.brightest := pixel
                TRUE
                  SKIP
        SEQ
          output.points[y.in][x.in][X] :=
                                  BYTE(illuminated.pixels[found.brightest][X])
          output.points[y.in][x.in][Y] :=
                                  BYTE(illuminated.pixels[found.brightest][Y])
          total.output := total.output + 1
          goto.xy(screen,24,5)
          write.int(screen,illuminated.pixels[found.brightest][X],5)
          write.int(screen,illuminated.pixels[found.brightest][Y],5)
          goto.xy(screen,20,7)
          write.int(screen,total.output,7)
        [256][256] BYTE dis.buffer :
        INT out.buffer :
        INT x,y :
        INT any :
        SEQ
          out.buffer := 1
          SEQ a = 0 FOR 256
            SEQ b = 0 FOR 256
              dis.buffer[a][b] := 0(BYTE)
          y := illuminated.pixels[found.brightest][Y]
          x := illuminated.pixels[found.brightest][X]
          dis.buffer[y][x] := 200(BYTE)
          write.buffer(out.buffer,dis.buffer,OK)
          display.buffer(out.buffer,OK)
          keyboard ? any
      TRUE
        SKIP
  TRUE
                  SEQ
                    done := TRUE
                    captured ! done
        end ? stop
          SKIP
BYTE response :
SEQ
  SEQ count = 3 FOR 4
    SEQ
      y.go.forward[count] := '0'
      x.go.forward[count] := '0'
  SEQ
    SEQ command.index = 0 FOR 9
      SEQ
        to.oriel ! x.go.forward[command.index]
        SEQ
          clock ? time
          clock ? AFTER time PLUS character.time
    BYTE carriage :
    SEQ
      from.oriel ? response
      from.oriel ? carriage
  SEQ
    SEQ command.index = 0 FOR 9
      SEQ
```

```
      to.oriel ! y.go.forward[command.index]
      SEQ
        clock ? time
        clock ? AFTER time PLUS character.time
  BYTE carriage :
  SEQ
    from.oriel ? response
    from.oriel ? carriage
SEQ
  to.oriel ! 'A'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! 'D'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
SEQ
  to.oriel ! 'B'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
  to.oriel ! 'D'
  SEQ
    clock ? time
    clock ? AFTER time PLUS character.time
INT time.lapse :
SEQ
  clock ? end.time
  time.lapse := ((end.time - start.time) / 15625)
  goto.xy(screen,20,10)
  write.full.string(screen,"Time : ")
  write.int(screen,time.lapse,6)
  write.full.string(screen," seconds")
BOOL abort :
INT open.error,result,char,any :
[63]BYTE dos.name:
INT file.name.len :
VAL  file.name IS "c:\data\spot.lut" :
SEQ
  abort := FALSE
  result := fi.ok
  open.error := fi.ok
  file.name.len := SIZE file.name
  [dos.name FROM 0 FOR file.name.len] := file.name
  open.tkf.file(from.filer,to.filer,tkf.open.write,
                    file.name.len,dos.name,open.error)
  IF
    (open.error = fi.ok)AND( result = fi.ok)
      SEQ
        [256][256] BYTE data :
        VAL block.size IS 256:
        SEQ
          SEQ a = 0 FOR 256
            SEQ b = 0 FOR 256
              data[a][b] := output.points[a][b][X]
          SEQ x = 0 FOR 256
            SEQ
              IF
                (open.error = fi.ok)AND( result = fi.ok)
                  SEQ
                    write.tkf.block(from.filer,to.filer,block.size,data[x],result)
                TRUE
                  SKIP

        SEQ a = 0 FOR 256
          SEQ b = 0 FOR 256
            data[a][b] := output.points[a][b][Y]
        SEQ x = 0 FOR 256
          SEQ
            IF
              (open.error = fi.ok)AND( result = fi.ok)
                SEQ
                  write.tkf.block(from.filer,to.filer,block.size,data[x],result)
              TRUE
                SKIP
    TRUE
      SKIP
result := fi.ok
close.tkf.file(from.filer,to.filer,result)
```

APPENDIX D

SOFTWARE FOR TEST SCREEN CALIBRATION

Test Screen Calibration

This software algorithm displays the test screens on the plasma panel of the Toshiba T5200 :

```
SCREENS.c :
#include <stdlib.h>
#include <stdio.h>
#include <graph.h>
int screen_mask[8] = { 0x80,0x40,0x20,0x10,0x8,0x4,0x2,0x1 };
main()
{
int x,y;
int result,key;
int x_origin,y_origin,y_end,x_end;
int screen_count;

     y_origin = 112;
     x_origin = 192;
     x_end = (x_origin + 256);
     y_end = (y_origin + 256);

     _setvideomode(_VRES16COLOR);
     _setcolor(16);

     result = system("cls");
      rectangle( GFILLINTERIOR,192,112,448,368);
     while(!kbhit());    /* wait for key */
     key = getch();        /*clear kbhit()  */
     result = system("cls");
for ( screen_count = 1; screen_count <= 8; screen_count++)
{
     for ( y = y_origin; y <= y_end; y++)
     {
          moveto(x_origin,y);
         If ( ((y-y_origin) & screen_mask[(screen_count-1)]) != 0)
         {
             _lineto(x_end,y);
         }
     }
     while(!kbhit());    /* wait for key */
     key = getch();        /*clear kbhit()  */
     result = system("cls");
}

for ( screen_count = 1; screen_count <= 8; screen_count++)
{
     for ( x = x_origin; x <= x_end; x++)
     {
          moveto(x,y_origin);
         If ( ((x-x_origin) & screen_mask[(screen_count-1)]) != 0)
         {
             _lineto(x,y_end);
         }
     }
     while(!kbhit());    /* wait for key */
     key = getch();        /*clear kbhit()  */
     result = system("cls");
}
 setvideomode( DEFAULTMODE);
result = system("cls");            }
```

The plasma panel calibration program :

```
...  Libraries
...  chans
...  channel addresses
...  PROCs
INT char :
INT any :
BOOL waiting :
BOOL exit :
INT answer :

[256] INT number.of.pixels.at.intensity :
[256] INT pixels :
INT highest,max.intensity :
INT threshold,bits :
INT brightest,darkest :
INT pixel.value,lut.index :
INT char,response :
BOOL OK,good :
INT frame :

VAL BYTE black IS 0(BYTE) :
VAL BYTE white IS 200(BYTE) :
VAL clear IS 0(BYTE) :
VAL lut.size IS (64 * 1024) :
VAL X IS 0 :
VAL Y IS 1 :

[lut.size] BYTE y.data.map :
[lut.size] BYTE x.data.map :

[256][256] INT input.image :
[256][256] INT output.image :
[256][256] BYTE light.image :
[256][256] BYTE dark.image :
[256][256] INT thresholds :

VAL []INT data.mask.table IS [#80,#40,#20,#10,#8,#4,#2,#1]:

SEQ
  SEQ
    [256][256][2] BYTE LUT :
    SEQ
      reset.board()
      SEQ count = 0 FOR 8
        init.G178()
      INT count1,count2 :
      BYTE data :
      SEQ
        count1 := 0
        WHILE count1 <= 255
          SEQ
            count2 := 0
            WHILE count2 <= 255
              SEQ
                data := BYTE(count2)
                LUT[count1][count2][0] := data
                count2 := count2 + 1
          count1 := count1 + 1

    count1 := 0
    WHILE count1 <= 255
      SEQ
        count2 := 0
        WHILE count2 <= 255
          SEQ
            data := BYTE(count1)
            LUT[count1][count2][1] := data
            count2 := count2 + 1
        count1 := count1 + 1

    write.lut(LUT)

SEQ
  goto.xy(screen,0,0)
  clear.eos(screen)
  goto.xy(screen,10,0)
  write.full.string(screen,"Calibration Using The Plasma Display")

SEQ a = 0 FOR lut.size
  SEQ
    y.data.map[a] := clear
    x.data.map[a] := clear

SEQ a = 0 FOR 256
  SEQ b = 0 FOR 256
    SEQ
      thresholds[a][b] := 0
      input.image[a][b] := 0
      output.image[a][b] := 0

INT bits :
SEQ
  INT char :
  SEQ
    char := 0
    clear.eos(screen)
    goto.xy(screen,10,2)
    write.full.string(screen,"Enter Number of Bits for resolution : ")
    read.echo.int(keyboard,screen,bits,char)

  SEQ
    SEQ
      goto.xy(screen,0,4)
      clear.eos(screen)
      goto.xy(screen,10,4)
      write.full.string(screen,"Space to capture dark image ... ")
    INT bar :
    SEQ
      bar := 0
      frame := 1
      WHILE bar <> 32
        SEQ
          keyboard ? bar
          IF
```

```
            bar = 32
              SEQ
                SEQ cap = 0 FOR 20
                  SEQ
                    capture.buffer(frame,OK)
                    read.buffer(frame,dark.image,OK)
                    SEQ a = 0 FOR 256
                      SEQ b = 0 FOR 256
                        output.image[a][b] :=
                          (output.image[a][b] +
                            (INT(dark.image[a][b])) )
                SEQ a = 0 FOR 256
                  SEQ b = 0 FOR 256
                    dark.image[a][b] := (BYTE(output.image[a][b] / 20))
                write.buffer(frame,dark.image,OK)
                display.buffer(frame,OK)
            TRUE
              SKIP

SEQ
          goto.xy(screen,0,4)
  clear.eos(screen)
  goto.xy(screen,10,4)
  write.full.string(screen,"Space to capture light image ... ")
INT bar :
SEQ
  bar := 0
  frame := 1
  WHILE bar <> 32
    SEQ
      keyboard ? bar
      IF
        bar = 32
          SEQ
            SEQ cap = 0 FOR 20
              SEQ
                capture.buffer(frame,OK)
                read.buffer(frame,light.image,OK)
                SEQ a = 0 FOR 256
                  SEQ b = 0 FOR 256
                    input.image[a][b] :=
                      (input.image[a][b] +
                        (INT(light.image[a][b])))
            SEQ a = 0 FOR 256
              SEQ b = 0 FOR 256
                light.image[a][b] := (BYTE(input.image[a][b] / 20))
            write.buffer(frame,light.image,OK)
            display.buffer(frame,OK)
        TRUE
          SKIP

SEQ y = 0 FOR 256
  SEQ x = 0 FOR 256
    SEQ
      IF
        light.image[y][x] > dark.image[y][x]
          thresholds[y][x] :=
          ((INT(dark.image[y][x])) +
            (( (INT(light.image[y][x])) -
              (INT(dark.image[y][x])) ) / 2))
        TRUE
          thresholds[y][x] := 255

SEQ screen.count = 0 FOR (bits * 2)
  SEQ
    SEQ
      goto.xy(screen,10,4)
      write.full.string(screen,"Test PLATE No. ")
      write.int(screen,(screen.count+1),3)
      write.full.string(screen," ( 1 -> 16 ")
      clear.eos(screen)

    INT bar :
    SEQ
      bar := 0
      frame := 1
      {{{  clear images
      SEQ a = 0 FOR 256
        SEQ b = 0 FOR 256
          SEQ
            input.image[a][b] := 0
            output.image[a][b] := 0
      goto.xy(screen,20,12)
      write.full.string(screen,"SPACE bar to Capture")
      WHILE bar <> 32
        SEQ
          keyboard ? bar
          IF
            bar = 32
              SEQ
                goto.xy(screen,0,12)
                clear.eos(screen)
                goto.xy(screen,20,12)
                write.full.string(screen,"Capturing into Buffer 1")
                SEQ cap = 0 FOR 20
                  SEQ
                    capture.buffer(frame,OK)
                    read.buffer(frame,light.image,OK)
                    SEQ a = 0 FOR 256
                      SEQ b = 0 FOR 256
                        input.image[a][b] :=
                          (input.image[a][b] +
                            (INT(light.image[a][b])) )
                SEQ a = 0 FOR 256
                  SEQ b = 0 FOR 256
                    light.image[a][b] := (BYTE (input.image[a][b] / 20))
                write.buffer(frame,light.image,OK)
                display.buffer(frame,OK)
            TRUE
              SKIP

INT light, dark :
INT above, below :
SEQ
  SEQ y = 0 FOR 256
    SEQ x = 0 FOR 256
      SEQ
        pixel.value := (INT(light.image[y][x]))
        IF
```

```
            pixel.value >= thresholds[y][x]
              dark.image[y][x] := 150(BYTE)
            TRUE
              dark.image[y][x] := 0(BYTE)

    SEQ
      frame := 0
      write.buffer(frame,dark.image,OK)
      display.buffer(frame,OK)

INT data.mask,mask.result :
SEQ
  IF
    screen.count < bits
      SEQ
        data.mask := data.mask.table[screen.count]
        lut.index := 0
        SEQ y = 0 FOR 256
          SEQ x = 0 FOR 256
            SEQ
              pixel.value := (INT(dark.image[y][x]))
              IF
                pixel.value = 150
                  SEQ
                    mask.result := ((INT(y.data.map[lut.index])) \/ data.mask)
                    y.data.map[lut.index] := (BYTE(mask.result))
                TRUE
                  SKIP
              lut.index := (lut.index + 1)
    TRUE
      SEQ
        data.mask := data.mask.table[(screen.count - bits)]
        lut.index := 0
        SEQ y = 0 FOR 256
          SEQ x = 0 FOR 256
            SEQ
              pixel.value := (INT(dark.image[y][x]))
              IF
                pixel.value = 150
                  SEQ
                    mask.result := ((INT(x.data.map[lut.index])) \/ data.mask)
                    x.data.map[lut.index] := (BYTE(mask.result))
                TRUE
                  SKIP
              lut.index := (lut.index + 1)

[256][256][2] BYTE LUT :
[256][256] BYTE x.data RETYPES x.data.map :
[256][256] BYTE y.data RETYPES y.data.map :
SEQ
  SEQ a = 0 FOR 256
    SEQ b = 0 FOR 256
      SEQ
        LUT[a][b][X] := x.data[a][b]
        LUT[a][b][Y] := y.data[a][b]

  open.file.write(screen,keyboard,from.filer,to.filer)
  file.lut(screen,keyboard,from.filer,to.filer,LUT,OK)
  close.file(from.filer,to.filer)
```