Inspection of the Integrity of Surface Mounted Integrated Circuits on a Printed Circuit Board Using Vision

By

Imad Yakoub (B. Eng.)

This Thesis is Submited as the Fulfilment of the Recquirement for the Award of the Degree of

Master of Engineering (M. Eng.)

Supervisor: Dr. Charles McCorkell

School of Electronic Engineering

Dublin City University

Dublin, Ireland

Sponsoring Establishment:

Scientific Studies and research Centre (SSRC)

Damascus, Syria

AUGUST 1991

Acknowledgements

For the completion of this thesis I would like to thank my supervisor, Dr. Charles McCorkell for his invaluable academic guidance which helped in carrying this project to completion.

I would also like to thank Mr. Noel Murphy for his invaluable help during this research.

I also wish to thank all the technical staff of the school for their support at various stages of this work.

I would also like to express my deep gratitude to the Scientific Studies & Research Centre (SSRC) in Syria for the provision of the grant to carry out this work.

I Would also like to thank my parents' for their blessings and prayers.

Finally, many thanks to my wife for continued moral support and patience.

Declaration

I hereby declare that, unless stated otherwise in the text, the work presented herein is my own and has not been submitted for another degree or qualification at this or any other university.

Signed : [illegible]

Date: 8 August 1991

Contents

Preface

Computer vision and image processing is becoming an increasingly important tool for the production engineer involved in the automation of manufacturing, offering a fresh approach to problems, some of which may have been impossible to solve previously. Vision applications in automation may be divided into further subject areas: robotic arm guidance, parts identification, process monitoring and inspection.

An industrial vision system is merely a computer with some extra electronics and software to take pictures, analyze them and take some decision from its analysis. The computer makes an image of the object to analyze, detects its characteristics and, compares them to the tolerances of the standard product. With these values, corrections can be fed to the production tool and statistics can be computed to follow the manufacturing process.

The aim of the present research is to develop a method for the inspection of J-Lead Surface Mount Integrated Circuits (*SMICs*) by means of a combination of a solid-state camera, a general purpose vision system and a proper illumination method, and a personal computer.

It was considered appropriate to begin the thesis with a general introduction to computer vision, image processing and pattern recognition discussing the basic elements.

Chapter 2 is concerned with an overview of current applications of computer vision in industry. The purpose of this overview is to stress the importance of vision for manufacturing in general and inspection in particular.

Chapter 3 deals with optics which include illumination, lenses, camera specification and inspection system interfacing.

In Chapter 4 the surface mount process and inspection algorithms are highlighted, the soldering defects are mentioned, and the lighting used is illustrated.

Chapter 5 is concerned with a discussion of aspects of quality control, a statement of aims, how to achieve these aims, and the inspection strategy.

Chapter 6 details program structure, and organization and implementation of the inspection algorithms for soldering defects introduced in Chapter 4. The recognition algorithm uses gray-scale images to locate IC leads in the image as a first step. Having leads located, the program looks for the defects by using a fixed threshold in the first line containing leads. A full listing of the program is given in appendix A.

In chapter 7 the thesis is completed with a discussion on and conclusions of the work undertaken.

Abstract

Machine vision technology has permeated many areas of industry, and automated inspection systems are playing increasingly important roles in many production processes. Electronic manufacturing is a good example of the integration of vision based feedback in manufacturing and the assembly of surface mount PCBs is typical of the technology involved. There are opportunities to use machine vision during different stages of the surface mount process. The problem in the inspection of solder joints on surface mount printed circuit board is much more difficult than many other inspection problems.

In this thesis, an approach for inspecting surface mounted integrated circuits (SMICs) is presented. It is based on the variance of intensity values of pixels in an image. This method is able to cope with 4 kinds of soldering defects in SMICs.

A set of modules for the system is proposed. The computer program which performs the image processing and analyzing has been written in C. It has been linked with a number of image processing routines from MAVIS[1] to perform some image processing tasks, and the result is a compact executable module which works under MS-DOS[2] 3.30.

(1) MAVIS is a registered trademark of CAPTEC, Computer Applied Techniques Ltd.
(2) MS-DOS is a registered trademark of Microsoft Corp.

Chapter

1

An Introduction to Computer Image Processing and Recognition

An Introduction to Computer Image Processing and Recognition

1.1 *Introduction*

Computer vision is a relatively new and fast growing field. With its origins in the early 50s. Most of the important concepts have been developed during last few years. A great deal of research in the domain of Artificial Intelligence (AI) is dedicated to computer vision (*AI is a science that enables computers to perform tasks which are related to mental and associated activities*) [1]. Such tasks include computer vision, understanding languages and visual environments, medical diagnosis etc. [2].

Computer vision is a multidisciplinary field, interacting with optics and sensing, image processing and analysis, computer architectures, pattern recognition, and robotics. Applications of computer vision are found in many fields such as medicine, manufacturing, science, and defence.

Computer vision is a collection of techniques, software and hardware for use in the above mentioned applications. The techniques involved in computer vision include optics sensing and image capture, algorithms, and architectures.

1.2 *Digital Image Processing*

Computers have been used to process pictures for approximately two decades, and the technology of digital image processing is continuing to expand rapidly. The increasing number of digital image sources include remote sensing and satellite systems, film scanners,video digitizers, biomedical systems, and military.

Digital image processing includes the processes of storage, manipulation, display, and use of digital images. These processes are entwined, dependent upon each other, and upon the initial image acquisition methods and technology. Digital Image processing has found its own niche, developing into a technology in its own right, due to its advantages over conventional optical processing techniques.

Image processing as it stands today found its roots during the heyday of space travel and exploration [3]. It's main applications, were at that time, the processing and enhancement of images returned by the satellites. The work and development done in this era has had lasting influence and effects on the area as it stands today. Needless to say however the technology, techniques and applications of this new science have evolved considerably since those times.

Initially the hardware and systems centred around monochrome camera and display technology. This soon developed to include multi-spectral image capturing and transmission systems. This was where the computers advantages over conventional camera/processing technology became apparent.

1.2.2 Digital Image Processing Elements

Any digital image processing system basically consists of a computer which is used to process images. Also the system must have two pieces of special input/output equipment, an image digitizer (*camera*) and an image display device (*monitor*).

1.2.2.1 Digitizers

Digitizers are used to convert continuous images to a numerical form which a computer can analyze. Most commonly used input devices are vidicon cameras and solid-state arrays (CCD). Vidicons have photo-sensitive targets which are scanned by an electron beam that reads off a charge which is accumulated on the target by exposure to light. Solid state arrays are composed of silicon imaging elements which produce a voltage output proportional to the incident light intensity. Figure 1.1 illustrates the digitizing process.

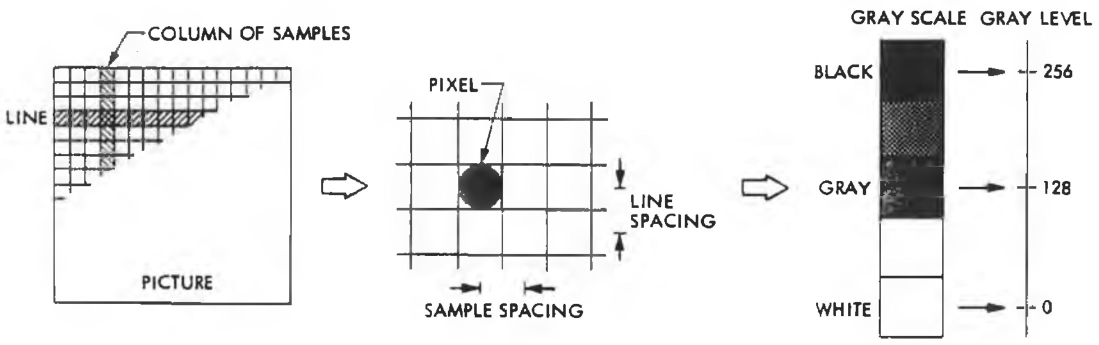


Fig. 1.1 Digitizing an image [10].

The image is divided into very small regions called *picture elements* - or *pixels* for short - which its number indicates to the image resolution. At each pixel location, the image brightness is sampled and quantized. This step generates an integer at each pixel to represent the brightness or darkness of the image at that point, Fig. 1.2 shows the relationship between the display image and the numeric values in a digital image. When this has been done for all pixels, the image is represented by a rectangular array of integers. Each pixel has a location or address (*row number and column number*) and an integer value called the *grey level*. This array of digital data is now ready for Computer processing. The resolution of the digitizer, expressed in number of bits, determines the number of intensity levels each pixel can represent. For video cameras, the signal is digitized to 8 bits of resolution, producing 256 grey levels ranging from black to white [4].

1.2.2.2 Image Processors

The digital image processor is considered to be the heart of any image processing system. It consists of a set of hardware modules which can perform the following:

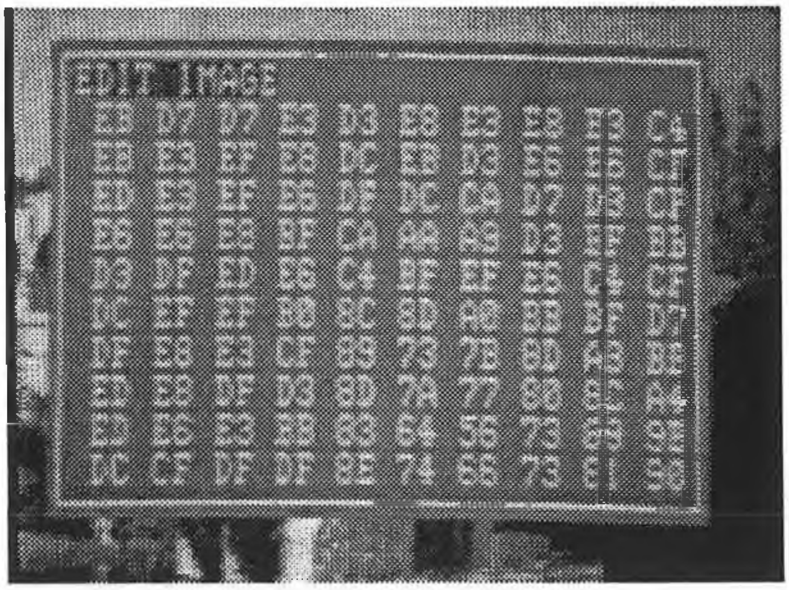
1 - Image acquisition.

2 - Input data storing .

3 - Processing.

4 - Displaying.

(a) Image with a patch.



```
EDIT IMAGE
 E8 D7 D7 E3 D3 E8 E3 E8 E3 C4
 E8 E3 EF E8 DC E8 D3 E6 E6 CF
 ED E3 EF E6 DF DC CA D7 D3 CF
 E6 E6 E8 BF CA AA A9 D3 EF BB
 D3 DF ED E6 C4 BF EF E6 C4 CF
 DC EF EF B0 8C 8D A0 B3 BF D7
 DF E8 E3 CF 89 73 7B 8D A3 BE
 ED E8 DF D3 8D 7A 77 88 8[illegible] A4
 ED E6 E3 BB 83 64 56 73 69 9E
 DC CF DF DF 8E 74 66 73 E1 98
```

(b) Numeric values equivalent to the patch displayed in a.

Fig. 1.2 The relationship between the pictorial image and its digital representation.

The digital image which is produced by the digitizer (*Image acquisition tool or frame grabber*) goes into temporary storage (*high-speed memory, often referred to as a frame buffer, with typical dimensions of 512 x 512 pixels or bytes*). In response to a control input (*or by user control*), the computer executes image processing programs. During execution, the input image (*in the frame buffer*) is read into the computer line by line. Operating upon one or several lines, the computer generates the output image pixel by pixel, and it is written on the output data storage device (*computer memory*) line by line. During the processing, the pixels may be modified (*enhancing, encoding and filtering etc.*). After processing, the resulting product is displayed by converting the stored digital image into an analog video signal, and this signal is output to a video device (*TV monitor*). The grey level of each pixel is used to determine the brightness (*or darkness*) of the corresponding point on a display screen. The processed image is therefore made visible and hence amenable to human interpretation. A typical digital image processing system is shown in Fig. 1.3.

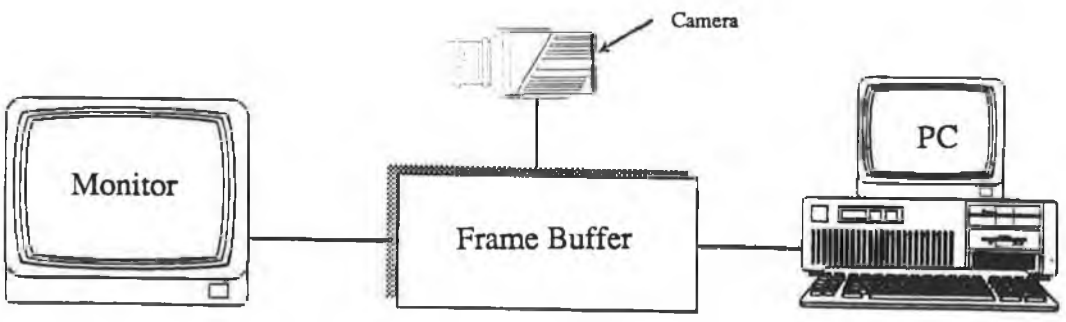


Fig. 1.3 Typical Low-Cost PC-Based Image Processing System.

1.2.3 Digital Image Processing Fundamentals

The digital image which the computer deals with, is represented by a two-dimensional function (*See Fig. 1.4*), where the value of the function $f(x, y)$ at spatial coordinates (x, y) in the x-y plane defines a measure of light intensity or brightness at that point.

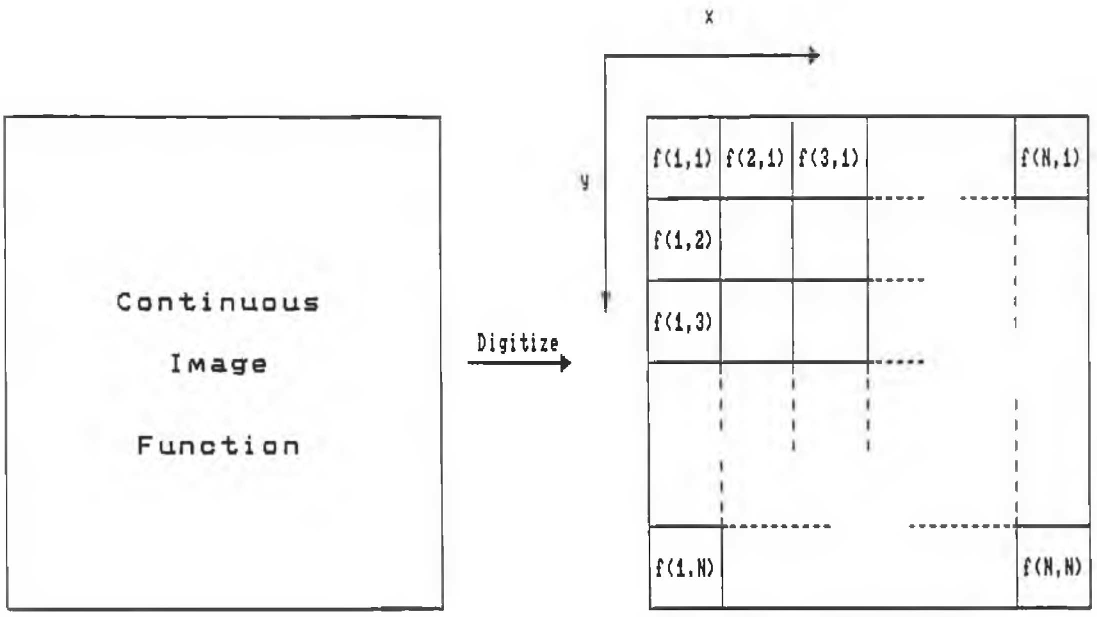


Fig. 1.4 Definition of a pixel reference grid in image digitization [7].

Digital images are approximated in two ways, corresponding to spatial, and amplitude digitization [5]. In general the spatially digitized image consists of N x N equally distributed samples, where each point is identified as a picture element or pixel. In amplitude digitizing, each pixel represent a grey level in the range 0 to $M - 1$, where $M = 2^m$. In binary images $m = 1$ (*that gives two grey levels*), while $m = 8$ provides 256 distinct grey levels, requiring one byte per pixel storage.

The digitization parameters - *the number of grey levels and the number of pixels* - will determine the degree of resolution of an image, and the amount of storage which is required to hold and process an image (*Figures 1.5 and 1.6 show some examples of varying these two parameters*). So, increased resolution further increases the storage requirements.

1.2.3.1 The Grey Level Histogram

Perhaps the most useful of image processing tools is the image histogram. The histogram reveals the distribution of digitized intensity within an image (*Figures 1.7, 1.8, and 1.9 show images and their histograms*). The histogram is unique for any particular image, but the reverse is not true. A well exposed image will tend to approximate a normal distribution around some mean. Images whose histograms contain two or more peaks tend to be high contrast images, with say a lot of dark areas and a lot of brighter ones within the same image. Under and over-exposed images on the other hand tend to have most of their pixels within a certain small range at one end of the histogram. This will result in a very poor quality of image, from which little can usually be detected directly. It is possible to perform contrast manipulation on images that display this form of distorted histogram, and the manipulation process usually uses information from the histogram itself. The objective of such manipulation is to produce an optimal display of the information contained in the image. Contrast enhancement therefore is the process of turning a poor contrast image into a display that makes more information visible.

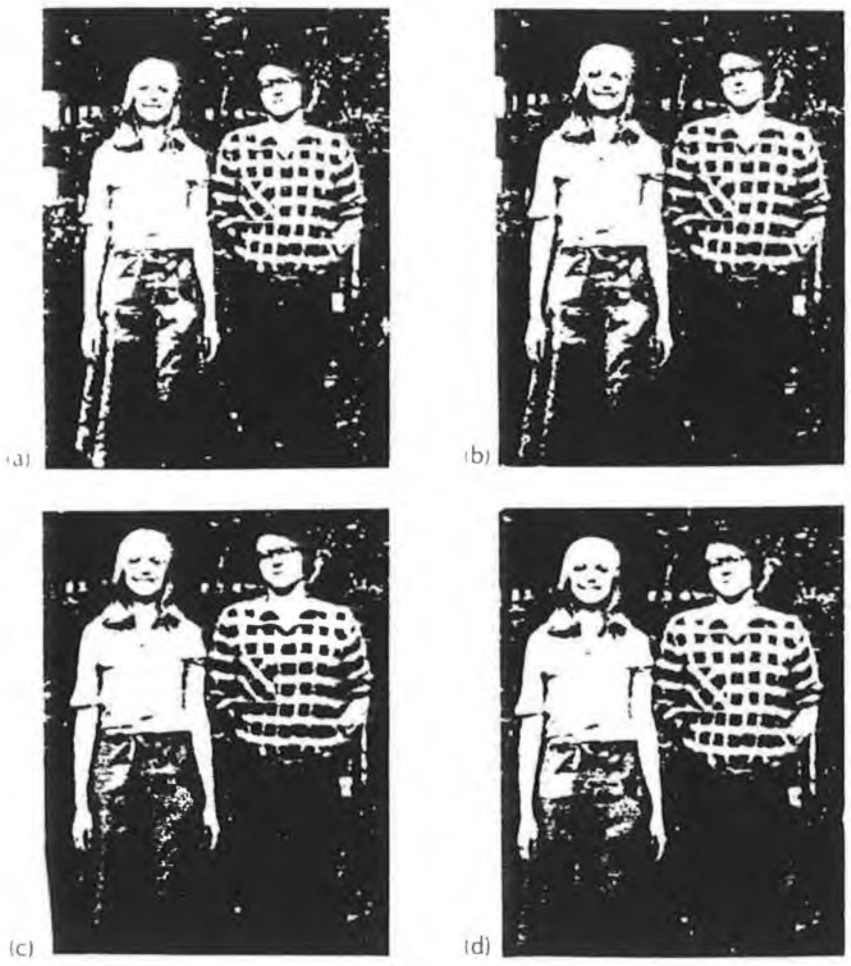


Fig. 1.5 Illustration of the visual effect of varying amplitude resolution. m increases from 1 to 8 between (a) and (d) [8].

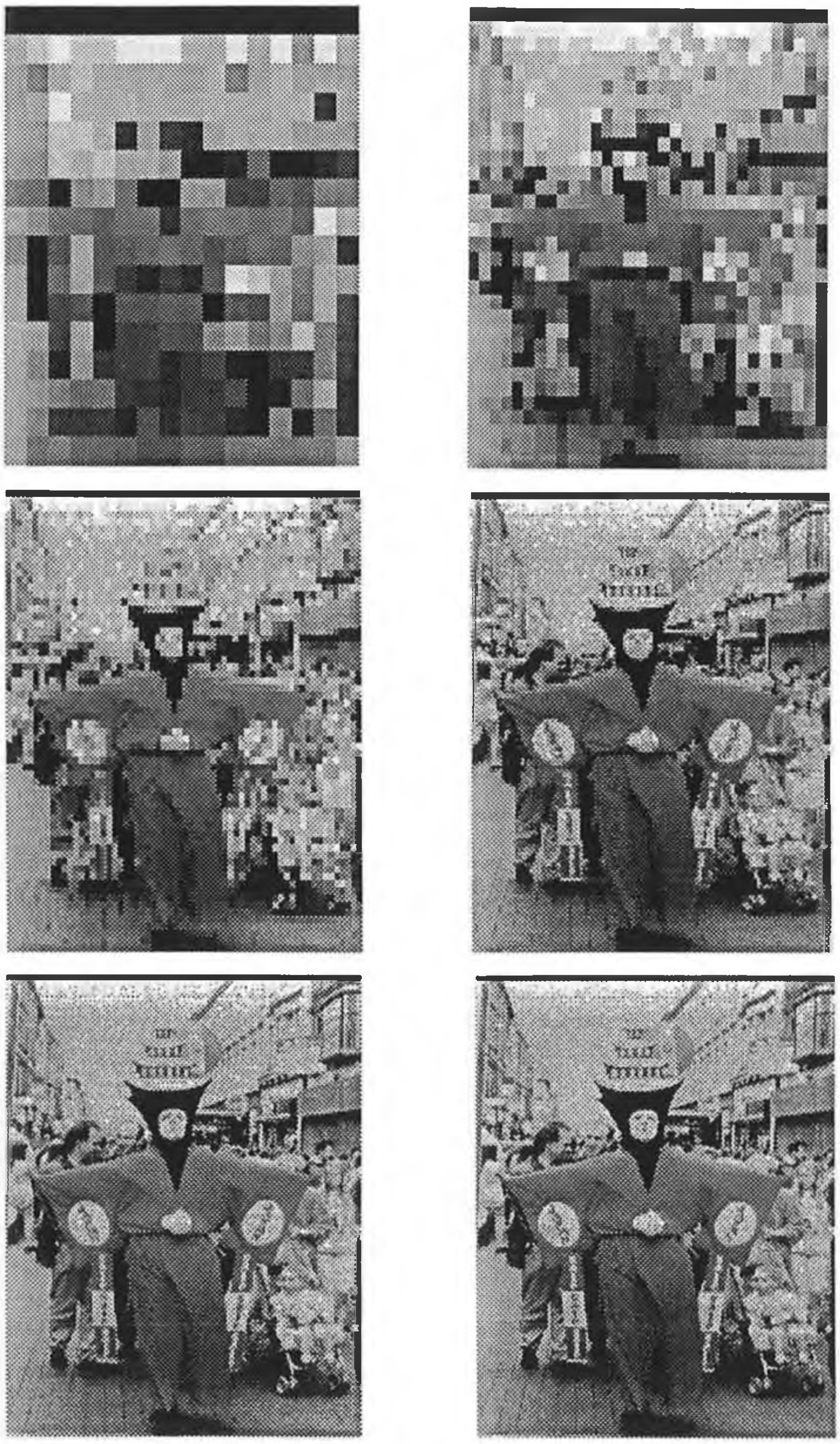
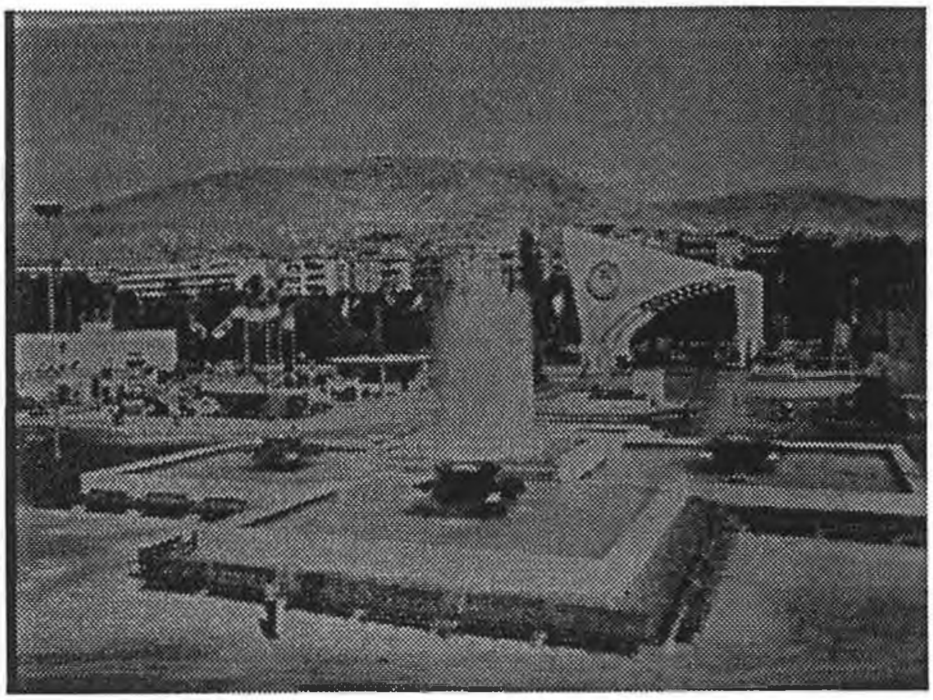


Fig. 1.6 Illustration of the visual effect of varying spatial resolution. N increases from 16 to 512 between (a) and (f).

1.2.3.2 Image Enhancement

Digital images often have limited contrast due to poor lighting during capturing or unfocused lenses. The objective of enhancement techniques is either to improve the appearance of image or highlight specific information. Most digital imaging systems can resolve more levels of intensity than can be presented on display devices.

Grey scale levels ranging from 0 to 255 are typical, requiring 8 bits of resolution per pixel. If a digital image has an intensity distribution such that the majority of pixels fall within a subset of digitized intensity value range, straightforward display of the unprocessed image will yield a very low-contrast result that does not fully display the information available in the image (***Fig.*** *1.7 shows an image and its histogram*). One method of improving on this situation is to stretch the values that exist in the image over the available value range. This is known as a *contrast stretch*, or *histogram stretch*. The results of this process can transform an image into a far more useful source of information, with far greater contrast and visibility of information in the image. An intensity histogram on the result of such a stretch can approximate the normal distribution far better, and even if it doesn't the histogram is at least better distributed than before (*Fig. 1.8 shows a stretched image and its histogram*). The nature of this stretch can necessitate the loss of some information at the extreme values of the



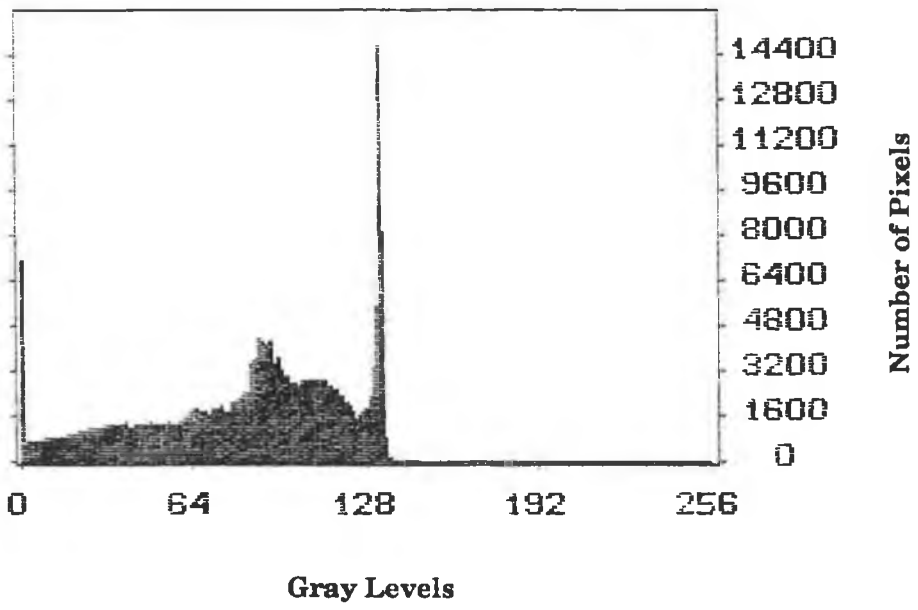
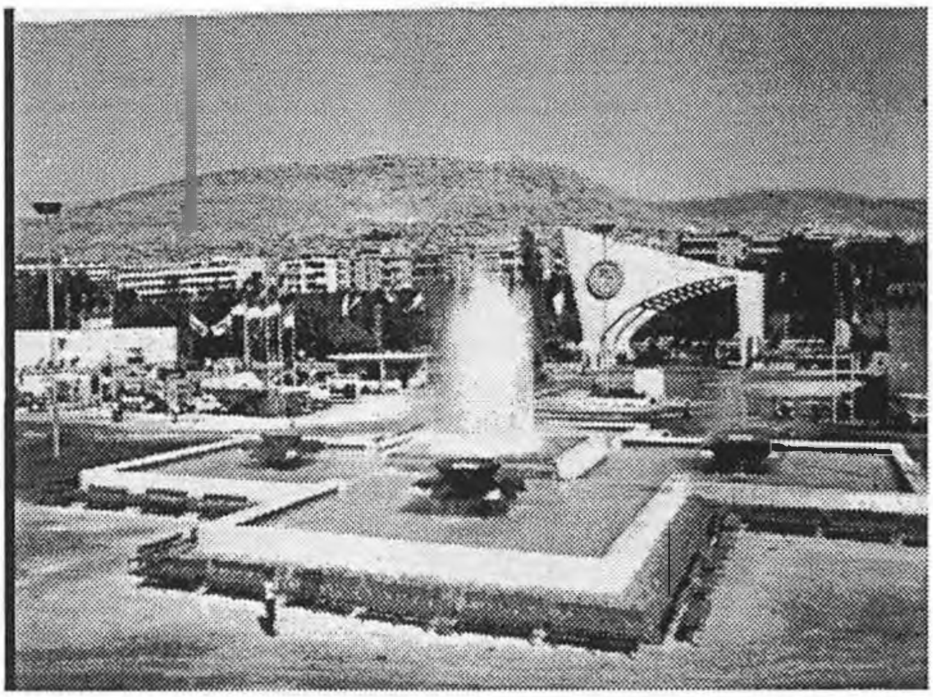


Fig. 1.7 Low contrast image and its histogram.



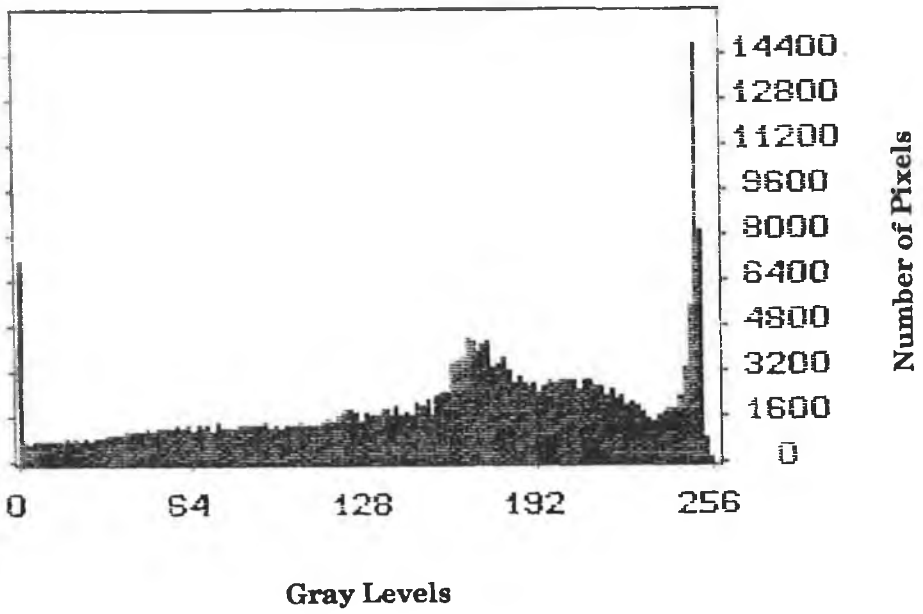
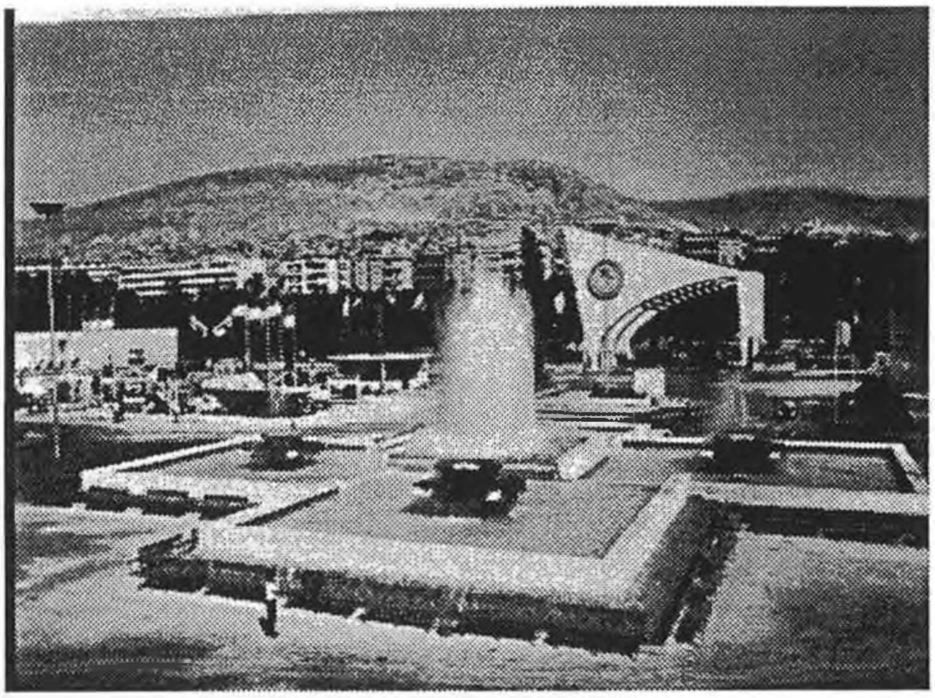


Fig. 1.8 Stretched image and its histogram.



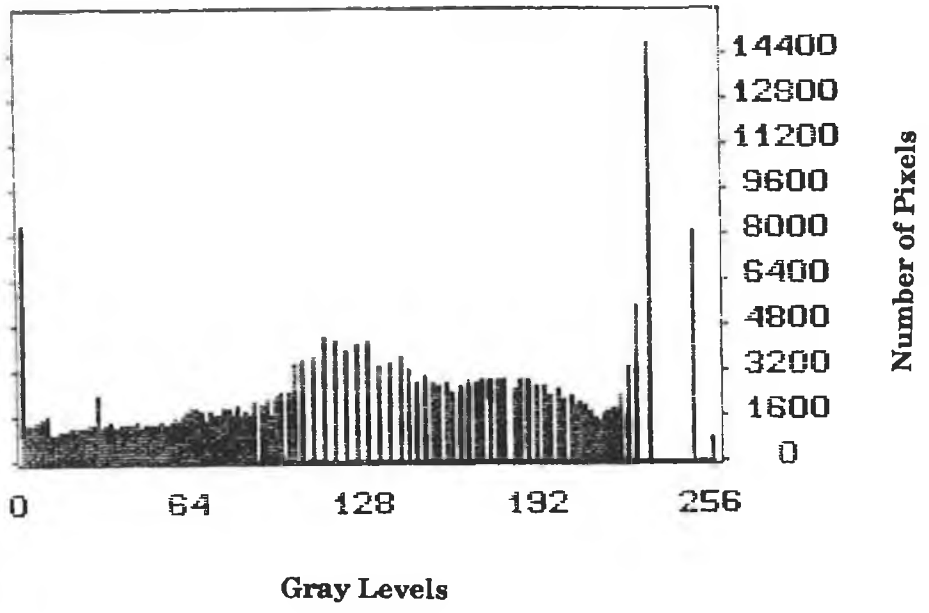


Fig. 1.9 Equalized image and its histogram.

stretch. These extreme intensity values are specified by the user and the algorithm then divides the existing intensity values between the two points distributed, those above the high extreme are set to the maximum available value, and those below the low value are set to the minimum available value.

One of the simplest and most effective techniques of image enhancement is *histogram equalization* where the idea is to enhance the use of "*underused*" grey levels and damp down use of "*overused*" ones. If the image consists of N pixels square, we will be aiming for an image in which each grey level is occupied by N^2 / M pixels (*where M is the number of grey levels*), and is thus equalized (*Fig. 1.9 shows an equalized image and its histogram*). Another approach to histogram equalization based on human visual model can be used [6], as the ordinary histogram equalization technique doesn't consider the human visual response to light. The human visual response to light is non-linear, and is somewhat like a natural logarithmic function (See Fig. 1.10). So the grey levels of histogram equalized image is not really equally distributed for human eyes.

Another useful technique for image enhancement is *spatial filtering*. An image will contain information at a wide range of spatial frequencies. Overall gradual transitions from light to dark with an image can be interpreted as low frequency components. Rapid, local variations in contrast represent information at a higher frequency. Local texture is an example of image information that is high frequency information.

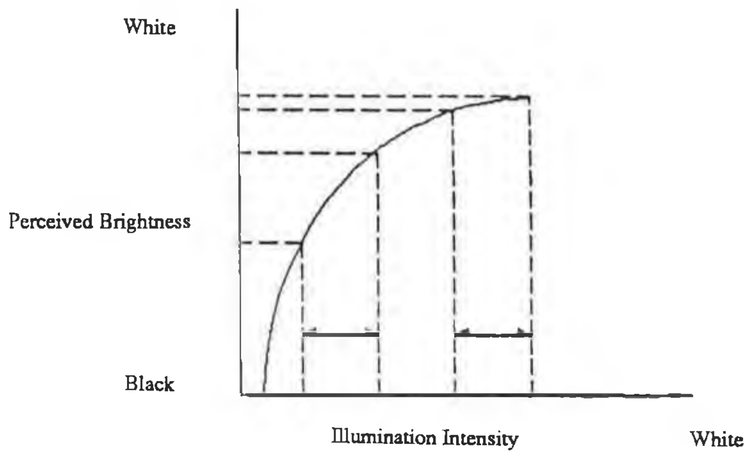


Fig. 1.10 The eye's logarithmic response to brightness.

It is possible to analyze the spatial frequencies present in digital images using Fourier transforms and other techniques. Such techniques are used to highlight areas of rapid change within an image, or areas of gradual change ignoring the rapid changes.

The topic of image filtering can become quite complex and very hard to follow. One filtering technique however is very useful and provides some interesting results. High Pass Filtering is performed on an image to remove slowly varying information and to retain only rapid changes in the image. It is used for analysis of surface structure and local image detail.

Other interesting filtering techniques include mean filtering, where each pixel is set to the average of those pixels around it, thus removing noise, but inducing blurriness. Median filtering on the other hand replaces each pixel by the median value of those pixels around it. This also reduces impulsive and noise as well in performing edge-preserving smoothing. It creates less of a blurring effect and also has the advantage that it does not "*create*" intensity values that were not in the original image. This can be very important when using colourized images.

1.2.3.3 Image Segmentation

Segmentation is one of the most important elements in image analysis because it enables the extraction of those regions or edges from an image that correspond to separate objects in a scene for subsequent processing such as description and recognition.

Segmentation algorithms are generally based on one of the two basic properties of grey-scale values: similarity and discontinuity. Images that are based on abrupt changes in grey level are classed in the second category. The basic areas of interest in this category are isolated points detection and the detection of lines and edges in an image.

The main approaches in the first category are based on thresholding, region growing, region clustering, and splitting and merging.

Thresholding

Thresholding is a particularly useful technique for scenes containing solid objects resting on contrasting background. It is computationaly simple and never fails to define disjoint regions with closed connected boundaries In thresholding technique, any pixel with a grey level at or above a selected threshold fall inside the object; whereas any pixel with grey level below the selected threshold fall outside the object. The boundary is then a set of interior points, each of which has at least one neighbour outside the object. Thresholding works well if the grey level of the object is uniform and the object rests on a background of a different grey level, which is also uniform.

Global Thresholding

By thresholding in a boundary location , a constant grey level threshold can be used throughout the image. If the background grey level is reasonably constant throughout the image, and the object contrast nearly equal against the background, a fixed global threshold will usually work well.

Adaptive Thresholding

In many cases the background grey level is not constant and the object contrast varies with the image. In such cases, a threshold that works well in one area might not work properly in other areas of the image. It is then convenient to use a grey level threshold that is a slowly varying function of the position in the image.

Optimal Thresholding

The exact value of the threshold can have a considerable effect on the boundary position and overall size of the extracted object, unless the object in the image has very steep sides. So, there is a need for an optimal, or at least consistent, method for establishing the threshold. An image containing an object on a contrasting background has a bimodel grey level histogram. The two peaks correspond to the relatively larger number of points inside and outside the object. The dip between the peaks corresponds to the relatively few points around the edge of the object and is commonly used to establish the threshold.

Region Growing

Region growing is a process that groups pixels or sub-regions into larger regions (*a region is a connected set of pixels*). In the simplest form, pixel aggregation, it may start with a set of 'seed' points within a region and grow it by grouping all neighbours which possess a similar property. In this scheme neighbouring pixels whose attribute values are within a fixed predefined range are grouped together to form the first clusters. Then neighbouring clusters are examined and eventually merged together based on suitable properties and relations, such as grey level, texture, and colour. For instance, in applications of infra red imaging, hot targets appear brighter than the background. Choosing the brightest pixels is then a natural starting point for a region growing algorithm.

The selection of similar criteria depends not only on the problem under consideration, but also on the type of image data available. For example; the analysis of satellite imagery is dependent mainly on the use of colour. The analysis would be much more difficult if only monochrome images were available.

All boundaries between adjacent regions are examined. A measure of the boundary strength is computed from the differences of the averaged properties of the adjacent regions. A given boundary is strong if the properties differ significantly on either side of the boundary and weak if they do not.

Strong boundaries are allowed to stand, while weak boundaries are dissolved and the adjacent regions merged. The process is repeated by first computing the object membership properties of the enlarged region again and then dissolving the weak boundaries. The process is then continued until a point is reached where the boundaries are weak enough to be dissolved.

Region Splitting and Merging

The region growing process starts from a set of 'seed' points. An alternative is to split large regions into smaller ones, and the process is repeated iteratively to obtain a selected level of uniformity.

Given a digital image containing several objects, the pattern recognition process consists of three major phases. The first of these is object isolation, in which each object must be found and its image isolated from the rest of the scene. The second is called feature extraction. The features are formed by a set of measurable properties. The extraction phase measures these properties from which it produces a set of measurements called the feature vector. This drastically reduced amount of information represents all the knowledge on which the subsequent classification must be based. The third phase is object classification, which is merely a decision as to which class the object belongs.

When a human observer views a scene, the neurological process that takes place in the retina and the optic cortex essentially segments the scene for him. This is done so effectively that he sees not a complex scene, but rather something he thinks of as a collection of objects. With digital processing, however, the objects in the image must be isolated by breaking up that image into sets of pixels, each of which is the image of one object.

Point Detection

The problem of detecting and segmenting isolated points in an image applies in noise removal. The basic mask used for detecting isolated points in an image is shown in Fig. 1.11.

-1	-1	-1
-1	8	-1
-1	-1	-1

Fig. 1.11 A mask used for detecting isolated points.

At each mask location, the vector product is:

$$-x1-x2-x3-x4+8x5-x6-x7-x8-x9.$$

In an area of constant grey level, the result of this operation would have been zero. Since in the example the image is centred at an isolated point (x5). Where the intensity is greater than at the other locations, the result is greater than zero.

In practice, where one is interested only in strong responses, we say an isolated point, whose intensity is significantly different from the background, has been detected if the vector product is greater than some non-negative threshold.

Line Detection

The next level of complexity involves the detection of lines in an image. If the mask shown in Fig. 1.12*a* were moved around on an image, it would respond more strongly to horizontal lines; with constant-level background, the maximum response would obtain when the line passed through the middle row of the mask.

(*a*)

-1	-1	-1
2	2	2
-1	-1	-1

(*b*)

-1	2	-1
-1	2	-1
-1	2	-1

(*c*)

-1	-1	2
-1	2	-1
2	-1	-1

(*d*)

2	-1	-1
-1	2	-1
-1	-1	2

Fig. 1.12 Line masks.

A similar experiment would reveal that the mask in Fig. 1.12*b* would respond to vertical lines; while the mask in Fig. 1.12*c* to lines at +45, and the mask in Fig. 1.12*d* to lines at -45.

The direction of the lines may also be established by noting that the preferred direction of each mask is weighted by a larger coefficient (i.e., 2) than other possible directions.

Edge Detection

Edge detection is the most commonly used approach for detecting meaningful discontinuities in grey level, although point and line detection are elements of any discussion on segmentation. The reason for this is that isolated points and thin lines are not frequent occurrences in most applications of practical interest.

In this approach an edge can be defined as the boundary between two regions with relatively distinct grey-level properties. It is assumed that the two regions are sufficiently homogeneous for the transition from one to the other to be determined on the basis of grey-level discontinuities alone. When this assumption is not valid, line or point techniques are generally more suitable than edge detection.

Most of the edge detection techniques involve the computation of a local derivative operator. The first derivative of an edge is zero in all regions of constant grey level and assumes a constant value during a grey level transition. The second derivative of an edge is zero in all locations, except at the onset and termination of grey level transition.

Gradient Operation

The gradient of an image $f(x, y)$ at location (x, y) is defined as the two-dimensional vector:

$$G[f(x, y)] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

It is well-known from vector analysis that the vector **G** points in the direction of maximum rate of change of *f* at location *(x, y)*. For edge detection, we are interested in the magnitude of this vector, generally referred to simply as the gradient and denoted by **G**[*f(x, y)*]. Where:

$$G[f(x, y)] = \left[Gx^2 + Gy^2 \right]^{1/2}$$

This quantity is equal to the maximum rate of increase *f(x, y)* per unit distance in the direction of **G**. The direction of the gradient vector is also an important quantity. If *(x, y)* represents the direction angle of **G** at location *(x, y)*, it follows from vector analysis that:

$$\alpha(x, y) = \tan(Gx/Gy).$$

where the angle is measured with respect to the *x*-axis.

If we take the sub-image area shown in Fig. 1.13 where x5 represents the grey level at location *(x, y)* and the other mask locations represent the grey levels of the neighbours of *f(x, y)*. The component of the gradient vector in the *x* direction can be defined as:

$$\mathbf{G}x = (x7 + 2x8 + x9) - (x1 + 2x2 + x3);$$

and in the *y* direction as:

$$Gy = (x3 + 2x6 + x9) - (x1 + 2x4 + x7).$$

The use of a 3x3 area in the computation of the gradient has the advantage of increased smoothing over 2x2 operators, tending to make the derivative operations less sensitive to noise.

x1	x2	x3
x4	x5	x6
x7	x8	x9

Fig. 1.13 A sub-image area.

Weighting the pixels closest to the centre by 2 also produces additional smoothing. It is possible to base gradient computations over larger neighbourhoods, but 3x3 neighbourhoods are by far the most popular because of the advantage in computational speed and modest hardware requirements.

G*x* =

-1	-2	-1
0	0	0
1	2	1

(*a*)

G*y* =

-1	0	1
-2	0	2
-1	0	1

(*b*)

Fig. 1.14 Sobel Operators.

From the previous discussion, **G***x* and **G***y* can be computed by using the masks shown in Fig. 1.14(*a*), (*b*) respectively.

These two masks are commonly known as the SOBEL operators. The responses of these operators at any point (x, y) are combined to obtain the gradient at that point. Convolving these masks with an image $f(x, y)$ yields the gradient at all points in the image; the result is often referred to as a *gradient image*.

Another useful approximation, called the ROBERTS gradient, makes use of the cross differences given by the equation:

$$G[f(x, y)] \equiv |f(x, y) - f(x+1, y-1)| + |f(x+1, y) - f(x, y-1)|$$

Laplacian Operation

The Laplacian is a second-order derivative operator, which can be implemented by convolving the mask shown in Fig. 1.15 with an image.

0	1	0
1	-4	1
0	1	0

Fig. 1.15 Mask used to compute the Laplacian.

Although the Laplacian responds to transitions in intensity, it is seldom used by itself for edge detection, because being a second derivative operator, the Laplacian is typically sensitive to noise. Therefore, it is usually delegated to the secondary role of serving as a detector for establishing whether a given pixel is at the dark or the light side of an edge.

1	$\sqrt{2}$	1
0	0	0
-1	$-\sqrt{2}$	-1

(*a*)

1	0	-1
$\sqrt{2}$	0	$-\sqrt{2}$
1	0	-1

(*b*)

0	-1	$\sqrt{2}$
1	0	-1
$-\sqrt{2}$	1	0

(*c*)

$\sqrt{2}$	-1	0
-1	0	1
0	1	$-\sqrt{2}$

(*d*)

0	1	0
-1	0	-1
0	1	0

(*e*)

-1	0	1
0	0	0
1	0	-1

(*f*)

1	-2	1
-2	4	-2
1	-2	1

(*g*)

-2	1	-2
1	4	1
-2	1	-2

(*h*)

1	1	1
1	1	1
1	1	1

(*i*)

Fig. 1.16 **Masks used in the Frei-Chen technique.**

The vector formation for the detection of points, lines and edges has the important advantage because it can be used to detect combinations of these features. The technique was developed by Frei and Chen. The masks *a*, *b*, *c* and *d* in Fig. 1.16 are suitable for detecting edges; while masks *e*, *f*, *g* and *h* represents templates suitable for line detection and the mask *i*, is proportional to the average of the pixels in the region at which the mask is located in an image.

1.3 Pattern Recognition

Pattern recognition is the science that concerns the description or classification of measurements, usually based on an underlying model. The two major approaches to pattern recognition are the statistical (*or decision theoretic*) and the syntactic (*or structural*) approaches. The structure of a typical pattern recognition system is shown in *FIG. 1.17*. The first stage in any pattern recognition task is usually referred to as feature extraction. Feature extraction is nothing more than a process of measurement, but this process can often be so complicated that it constitutes the main work of the pattern recogniser. The result of the feature extraction stage is a set of numbers that are fed to the classification or decision stage of the recogniser.

A pattern can be defined by a feature vector **C** whose components describe the pattern. In general, the feature vector can be represented as a n-dimensional vector:

$$C = \{ c_1, c_2, c_3, \ldots, c_n \}$$

Where c_1, c_2, ..., c_n are measurements of specified characteristics of the pattern, usually referred to as *features* or *pattern descriptors*.

1.3.1 Features Extraction

The most difficult problem here is to find suitable pattern features. Proper selection of the features is important, since only they will be used to distinguish the objects.

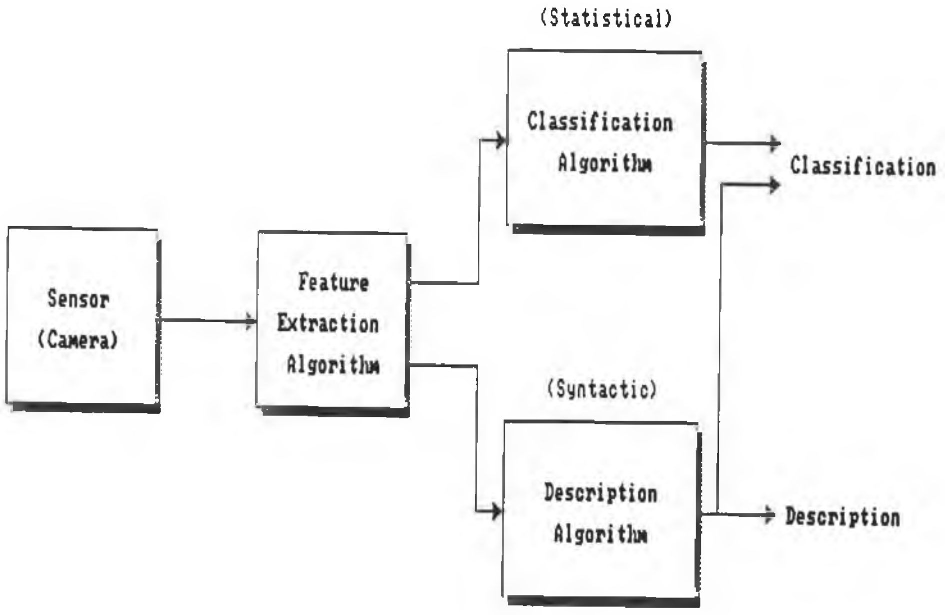


Fig. 1.17 Typical Pattern Recognition System.

Features used for classification should be reliable, independent, discriminating, and few in number; that means, features should take on similar values for all objects of the same class; the features used should be uncorrelated with each other; also features should take on significantly different values for objects belonging to different classes; and finally the complexity of a pattern recognition system increases rapidly with the number of features used, as adding more features that are either noisy or correlated with existing features can actually degrade the performance of the classifier.

Template Matching

The fastest method of classifying a pattern is to compare it with stored models of known patterns and choose the best match. In template matching, the comparison is performed directly on images, so this makes it possible to find a certain object in the presence of several objects or in distorted picture. Template matching is very sensitive to the orientation and size, as using different scales, or orientation is computationally expensive.

In summary, template matching techniques are useful for applications where the number of classes and the variability within a class is small; it is not an approach for general situations.

1.3.2 Classification methods

The classification stage follows the feature extraction stage where a decision is made on the pattern class involved. That means that the n-dimensional feature vector **C** generated in the first stage is assigned a specific class.

There are a large number of procedures given to solve the classification problem, but only a few of them can be used in practical applications because of the requirements of economic, viability and simple solutions.

There are several methods of classification: linear, nonlinear, sequential and nearest neighbour classifier. Linear classifier are by far the most common sort of classification rule used in pattern recognition. They are especially interesting because of their easy implementation. Linear classifiers are usually introduced by an appeal to the common-sense observation that two well separated groups can be classified using a linear rule (*Baye's rule for example*). The linear classifier works well either with groups that are widely separated or when the groups are elliptically symmetrical or approximately so.

The linear classifier fails when it comes to problems where the features of one class are completely surrounded by those of another class, so the nonlinear classifiers are used.

In the sequential classifier, the basic idea is to order the features according to their discriminating effect and to compare the features of the vector to be classified in this order.

The spatial distance from one pattern to other patterns for classification is used in the nearest neighbour classifier, therefore the distance of high-dimensional feature vector to all stored feature vectors is necessary to be calculated.

1.4 Image Analysis

Image analysis operation includes elements of image processing, but image processing often means simply improving features or characteristics in an image somehow, to make it easier to interpret by the human eye, without any measurement or quantification taking place.

Image analysis systems perform measurements or determination of features in an image (*counting number of holes in a PCB, or determination the shape of an object, etc.,*). One of the most developed areas where image analysis is used is in industrial machine vision, in which the image analyzer contributes to the automation of an industrial processing system (*inspecting parts, measuring certain features, recognizing and classifying parts, and so on*).

1.5 Conclusion

In fields such as machine vision, robotic guidance, medical imaging, military applications, and many others, the processing of images by computer is a key element. The growth of all stages of computer technology in recent years has enabled the development of vision systems and digital image processing technology. The trend toward faster, more powerful and cheaper computational elements, and the advent of special-purpose hardware, have made possible the processing and manipulation of large volumes of digital imagery.

Chapter

2

An Overview of Computer Vision Applications in Industry

An Overview of Computer Vision Applications in Industry

2.1 Introduction

Computer vision started gaining attention more than *20* years ago, and this interest has increased dramatically eversince. It is widely accepted that computer vision applied to industry, often called machine vision, is a key technology for the automated factory of the future [25,26]. Computer vision has already invaded many areas of the industrial world. Up to a short time ago, computer vision was considered primarily an element of robotic systems. However, today, it has developed into an industry in its own right [27]. It is estimated that the machine vision industry growth rate is between *40%* and *50%* annually worldwide [28]. In Europe the growth rate is estimated at *52%* [29]. This could be largely attributed to the high success rate reported by end product users. Another reason for the rapid growth of machine vision is the growing emphasis on industrial automation - machine vision is recognized as one of the most important, if not the most important, tool that helps pave the way for industrial automation [30,31]. Automation, when carefully implemented, can result in lower cost and better quality [32].

The decreasing price and increasing processing speed and computing power of personal computers (*eg., the IBM PC/AT or Apple MAC or compatibles*), as well as the large pool of hardware and software available, have prompted developers in recent years to design small vision systems based on personal computers [32,17]. *(Many of the reported successful systems are implemented on the PC/AT class of computers due to its relatively high power compared to the PC)* [33].

2.2 Computer Vision Applications

Computer vision applications in industry fall into two major categories; inspection or quality control, *(the usual meaning of the term 'Machine Vision'*) and adaptive control of robot systems (*also referred to as 'Robot Vision'*) [34,35]. The first category often involves measurement either of geometric dimensions or surface features to detect the presence of defects. The second application area is the guidance of robot manipulators for locating and taking up parts, and visual servo control [36]. Systems have been developed that are able to locate the position, orientation, and identification of parts on a moving conveyor or parts presenter. Most of these systems are special purpose, being heavily dependent on application specific constraints and techniques.

Demand from industry is growing for automatic inspection and control, using vision as one of the prime sensors. In response, systems are beginning to emerge, based upon improved hardware technology and some innovative software.

This underlines the importance of both hardware and software in computer vision.

Available systems for non-contact sensing, use one of three basic approaches: X-ray, laser and visible light technology. Each approach is radically different from the others, but all are quite workable with current technical abilities. However, one approach may be more suitable to a particular application than another - it remains for the manufacturing engineer to decide which approach is best for his or her individual requirements [37].

The present applications can be split into five main subjects:

1 - Automation of manufacturing processes.

2 - Quality control.

3 - Remote sensing.

4 - Medical applications.

5 - Military applications.

The benefits of computer vision to these areas include increased speed of processing, improved efficiency, consistency of data interpretation and improved control of the operation [38]. The two main branches of industrial automation are guidance and inspection.

2.2.1 Guidance

Vision systems for robot guidance take basically two forms. One is verification of parts or determination of position for assembly, and so on in material sorting and related tasks. Such applications include automated calibration of air gauges [39]; the use of vision to guide a robot to align a pick-and-place head over an X-Y table in electronics assembly [40], or to advise a robot in order that it can pick up a part for subsequent assembly (*in the correct orientation*) [41]; the placement of fine-lead-pitch surface mount electronic devices using vision guidance [42]; and chips placement process monitoring [43]. The second class of robotic vision guidance task is the visual servo control *(such as in weld guidance)*, i.e. using vision to provide position information of fabrication parts to the servo control of the robot to adjust its path [36]. The major successful applications of robot manipulators in manufacturing has been the positioning of spot welding electrodes for car bodies [11].

In the case of robots, typical tasks that a robot is expected to perform include materials handling, product assembly *(pick-and-place machines)*. At present, the limitations of the use of robots in industry are due mainly to low flexibility and high cost. Increased flexibility usually further increases the cost. Nevertheless, the need for high-performance, intelligent robots does exit; for example, they can be used for sophisticated assembly, mining and exploration.

In summary, the requirements for robot vision are different in many respects from the need of inspection, and quite different types of systems have been developed to deal with it.

2.2.2 Inspection

Inspection represents by far the largest category of applications for machine vision systems to-date. It has been estimated that over 50% of the potential applications of vision in industry are in the area of inspection [38], and there is a great diversity of applications. These include not only visual examination for faults such as is normally done by eye, but also measurement of dimensions or other characteristics, counting checking of orientation, and so on.

Inspection criteria can range from a simple pass or fail decision about the general appearance of an object or item to a diagnostic procedure which may be used to recover or control errors, and can be implemented with varying degrees of required accuracy. Traditionally, inspection has been carried out by human inspectors. The human visual system is extremely sophisticated and versatile. Machine vision systems developed so far are less sophisticated. However machine vision systems don't get tired of inspecting as humans do. Also, despite its inherent sophistication inspection by humans is prone to error in the case of repetitive tasks, at high speeds, or over a long period of time.

Now many systems have been developed where machine vision is used in inspection of industrial parts [44,45]. This operation, which is still labour-intensive, has been estimated to represent *10%* of the total labour cost. This is second only to the cost of assembly which is estimated to be *22%* of the labour cost [46]. Thus, significant savings can be achieved in high-volume production lines by automating manual inspection. The use of human labour for inspection is becoming less cost effective since wages tend to increase on regular basis while human performance remains constant. This has led many industrial engineers to investigate vision systems for use in manufacturing. Examples include automated inspection of connectors [47], automated crack detection in metal and other workpieces [48], and many other inspection tasks. In other cases, inspecting discrete parts is a monotonous job and human inspectors do not perform well. As a further complication, this work is often done in unhealthy environments, or ones that cannot be 'invaded' such as a clean-room. Another motivation occurs in processes where the inspection task is simple but the high speed required prohibits the use of human labour.

2.3 Inspection in high volume manufacturing

Industrial automation is required for high volume electronics manufacturing. Automation will increase productivity and improve quality. Leaded components and integrated circuits can be automatically inserted into printed circuit boards, but sometimes the lead fails to go through the hole. Leadless components *(Surface Mounted Devices)* can be automatically assembled to boards, but they may not be placed in the right location to make a solderable connection, or they might fall before being soldered. Printed circuit boards are automatically soldered by wave solder machines or reflow soldered, but sometimes, solder bridges can form, or solder balls can sputter onto other areas of the circuit. Always, all of those automatic processes are followed by inspection to catch these defects. Therefore inspection is very important for the following reasons [49]:

1 - At the high rates of production, there is a probability to producing large quantities of scrap if the process is out of control and the parts are not inspected immediately after the various stages of assembly or solder.

2 - Electrical test alone will not detect all defects that could cause field failure. Often these types of defects which pass factory electrical tests but quickly degrade during the product use, can be spotted by visual characteristics.

2.4 Machine Vision Task

The task of machine vision can be summarized as one of verification of the process. However, depending upon the type of technology used *(Surface Mount, Thick Film, Leaded Components or Printed Circuit Board)*, the optimal inspection procedure or processing method will vary substantially. In our case, *(the surface mount)* visual inspection should occur after soldering.

2.5 AVI in Electronic Manufacturing*

Recent advances in machine vision coupled with a demonstrated need for automatic visual inspection of electronic circuits has caused an increased interest in developing inspection systems for industry [50]. Since automatic inspection processes are playing increasingly important roles in many industries, several methods developed for automatic inspection, as in electronic circuits inspection tasks which range from integrated circuits to printed circuit boards.

Although there are many advantages to automatic visual inspection, there are, however, some problems and several disadvantages. The problems of machine vision inspection systems in real-time automated inspection, may be solved by adopting sampling approaches [51]. Vision systems are expensive, and are rarely available off the shelf [52].

(*) Automatic Visual Inspection

They must be developed especially for the application, and special fixturing and handling equipment may be needed.

2.5.1 Printed Circuit Boards Inspection

Electronics, is considered to be the backbone of many of today's products. Printed circuits are still the most common assembly vehicle for the fabrication of electronic systems. Most electronic components *(leaded components)*, can be automatically inserted, while other components *(such as power transistors and transforms)* have to be manually inserted, so it is necessary to check the board for proper assembly before solder to keep the cost of repair low. It is very costly, and sometimes impossible to remove soldered components and rework the board [49]. Therefore it has been found necessary in many instances to apply vision technology to the inspection of bare printed circuit boards *(to inspect for defect classes as, breaks, mousebites, shorts, etc.)* [53,54], and assembled boards as well *(for missing or misoriented components, and missing leads through the board)*. Automated visual inspection systems presents a method to solve many problems, as some defect areas are almost impossible to tackle by manual means. The most common manufacturing defects in printed circuit assembly can be classified into three main categories:

1 - Board defects (etching & drilling).

2 - Incorrect insertion.

3 - Solder defects.

Studies based on U.S. manufacturers of electronic products indicate that, presently, 78% of all inspection is human visual. With today's significant labour costs, it is clear that automated inspection techniques are needed in order to minimize the number of defects. Inspection systems developed for the inspection of printed circuit boards are usually based on grey-scale analysis, template operations, transition operations, and histogramming [55].

2.5.2 Solder Joints Inspection

Solder joints are the critical electrical connections between electronic devices and the printed wiring boards on which they are mounted. In addition to its electrical duties, the solder joint has a mechanical fastening role as well. The criticality of this role is increasing with the widespread use of surface mounted devices (SMD) in which solder is often the only means of attachment between the device and the circuit board. The human inspection operation is performed by using human inspectors looking through microscopes, or video monitors. This method is labor intensive, and inaccurate. The solder joint inspection problem is more difficult than many other visual inspection problems. It is a difficult area to apply machine vision techniques for automatic inspection, because of the variability in the appearance of acceptable solder joints. It has been found that approaches based on edge detection or template matching are not adequate for this purpose. Also laser heating techniques are not preferred because of possible damage to solder joints. However, by using all the grey scale information present in the image, and using well-designed image processing algorithms, the vision systems can work with a broad range of lighting configurations [56], and avoid the problems which results from the non-uniform lighting.

Vision systems have been developed to inspect solder joints on pin-through-hole component leads for voids, cracks, pin-holes etc. For example, an integrated robot/machine vision system with colored illumination has been developed for solder joint inspection provided by two concentrically mounted fluorescent lights [57]. Also a solder joint inspection system have been implemented using grey scale image processing techniques [58]. Flaws are identified using features based on the characteristics of intensity surfaces and classification is by way of a minimum distance algorithm.

Another approach used to inspect solder joints is based on tiered-colour illumination, which generates colour contours on the solder joint for the detection and classification of defects [59]. Based on structured light *(light of a known shape)*, a 3-dimensional approach has been developed for the inspection of solder joints on surface mounted printed circuit boards [60].

2.5.3 Wire Bonders Inspection

While computer vision inspection of surface mount and through-hole components is becoming common, it has been only very recently that technology has been developed to inspect semiconductor components. Automatic wire bond machines are one of the dominant means by which semiconductor dies are attached to their packages. As with any machine, these wire bonders make mistakes more frequently as the density of these packages goes up to meet an increased number of input/outputs. In order to monitor the output of wire bonders and provide feedback for process control, a 3-dimensional inspection station with structured lighting can be attached immediately following the bonder [61].

2.5.4 Surface Mount Inspection

The development of surface mount printed circuit board technology has generated a need for new inspection systems to deal with its problems. 2-Dimensional and 3-dimensional approaches have been developed for various inspection tasks on surface mount printed circuit boards. The following are some of these tasks:

1 - Inspection of circuit patterns and conductive layers comprising PCB substrates for pattern match, alignment, positioning and breaks.

2 - Inspection of components prior to soldering for appropriate location, presence and alignment.

3 - Inspection of solder paste deposits *(pre-reflow)* and fillets *(post-reflow)* for presence, correct height and width and internal characteristics [62].

4 - Inspection of leads on components themselves, before soldering, for defects in alignment, coplanarity, spacing and count. Also inspection carried out after soldering for solder bridges, misplacement of leads, insufficient soldering and excessive soldering (*which is the aim of the project*) .

2.6 Other Applications

In the previous sections, the use of machine vision in automatic inspection in electronic manufacturing has been described . There are a number of other applications in manufacturing industry, which can be described very briefly.

- Part number reading

Bar-code and character readers are used for reading part and serial numbers on parts and containers, for routing and for controlling test and assembly equipment [7,11].

- Assembly step verification

Machine vision is used in a number of ad hoc applications to check for the presence and configuration of parts in an assembly [11].

- Surface finish inspection

Machine vision is being used to check the uniformity of texture and colour of painted surfaces, and also for processed materials such as paper and plastics for mixing and colour inspection [63].

- Print quality monitoring

Machine vision systems used to check the quality of printed output from printers and typewriters to ensure conformance to certain subjective standards for "*contrast*" and "*legibility*" [11].

- Part integrity checking

Machine vision is used in a number of applications to detect qualitative flaws (*cracks, pits, voids, scuffs*) in mechanical parts [11].

- Inspection of analog and digital meters

Vision system modules are developed for inspecting various types of meters, both analog and digital, by using industrial robot having multisensory input capability [64]. The main task performed by the vision system include: location of the test panel, positioning of the robot arm, recognition of the objects on the panel and determination of their 3-dimensional location.

- Inspection of integrated circuits

Due to the requirement of an ultra clean environment in the semiconductor industry, it is desirable to employ automation wherever feasible. An approach on a knowledge based system for on-line, automatic inspection of semiconductor wafers/ICs is developed to inspect wafer defects such as, warpage, surface damage, non-flatness of wafer, and crystal-plane slippage [65,66].

Finally, many new applications of machine vision for inspection have been developed over the last few years, and many more applications are still to come. However, there are many more applications areas, some of which we will just list, such as automatic inspection of automobile displays [67], automated optical inspection of knitted fabrics [68], print verification and label position inspection [69].

2.7 Conclusion

Vision technology is a powerful tool for inspecting assembly steps in electronic manufacturing, due to its non-contact nature, reliability, and high speed operation. In general, vision technology is applied to the inspection of mechanical parts, robot guidance, and parts sorting and counting. Electronic manufacturing inspection places special requirements on vision system resolution, processing speed, grey scale analysis, lighting and optics, as well as user interface software. The following are some advantages that make vision systems desirable:

1 - Cost saving; labor requirements for inspection tasks are much lower.

2 - Accuracy and field repair reduction as machine vision systems do not get tired of inspecting as humans do, and inspection with humans is prone to error for the case of repetitive inspection, at hight speeds over a long period of time.

3 - Elimination of undesirable jobs - visually monitoring parts or process is a very monotonous task.

4 - Line speed increase - vision systems are quicker than humans.

For the reasons mentioned above, vision technology will be used in this project in the inspection of surface mount integrated circuits on a PCB, and J-Lead ICs in particular.

Chapter

3

Illumination and Sensors

Illumination and Sensors

3.1 Introduction

Good illumination of the object is the first step to achieve a quality image in a computer vision system, and it is essential for successful application of that system. It is impossible to rectify poor quality image by further image processing stages. The vast majority of machine vision decisions are based on light intensity contrasts. This mean that any techniques for improving control of lighting would be beneficial in many machine vision applications.

One area of machine vision applications where there are often engineering difficulties is, the area of image acquisition, i.e., in the lighting, optics, and sensor selection. Many good applications have failed due to improper lighting and optics design. The problem of designing a suitable physical image acquisition configuration is one which requires a lot of experience to overcome the difficulties and so for example, expert systems have been designed to solve lighting and optics problems for machine vision applications [1].

3.2 Illumination

Illumination is one of the most important factors regarding success or failure for many machine vision applications. As many machine vision systems will base much of their analysis on the intensity of the incident light, the control of illumination is very important to ensure consistent data from the image. This is particularly the case, if image processing algorithms are being made with reference to a fixed intensity threshold. Similarly the same problem will occur if the illumination, and the reflected light, changes.

With appropriate lighting, much useless information can be eliminated, contrast of significant image features can be enhanced, and the sensor signal-to-noise ratio improved. However, with many current systems in use today, additional constrains are placed on the lighting and optics associated with the image sensor due to requirements for uniform and temporally constant levels of illumination. For these reasons, many grey-scale processing functions have been provided in more advanced systems to alleviate such problems. Since image features of interest such as edges predominate at high frequencies, the effects of illumination variation can be reduced prior to further processing such as thresholding [70].

The diffused light is the most suitable source for most machine vision applications, since it is non-directional and produces a minimum amount of shadows. The common method to obtain diffused illumination is fluorescent lighting. Fluorescent tubes of different types have different spectral distributions and need to be matched to the application. (*In high speed applications the flicker of fluorescent tubes may be important*). The tungsten filament bulbs also used for lighting, but it provides directional illumination and produces strong shadows.

Object illumination and image acquisition are always of crucial importance for automated visual inspection [71]. The image processing algorithm should not be expected to compensate for poor image quality, caused by the use of inadequate lighting/viewing techniques. It is cheaper to use better object illumination than to add sophistication to the image processing algorithm. All parts of an automated visual inspection machine should be individually optimized, to make the overall system as efficient and as robust as possible.

The increasing use of automated inspection has elevated the importance of measuring the spectral distribution of light [72], and the type of lighting required for automated visual inspection depends on the specific task and may vary from simple overall illumination of an object to the use of patterned, colored, polarized light.

3.3 Sensors

The task of the image sensor is to convert an optical picture which is usually 3-dimensional or 2-dimensional into a form suitable for use with electrical systems (*time sequence of electrical signals*). The sensor (*camera*) represents the direct link between the environment being examined and the information processing system, so, it is of particular importance.

3.3.1 Lenses

Lenses are one of the basic elements of image formation, and they are required to form the image on the sensor's light sensitive device. It is possible to form images without lenses, by using a collimated light source to project a shadow of an object onto the sensor but, such a configuration is not adequate for a vision system.

Lenses are defined by their focal length *(quoted in millimeters mm)* and their aperture *(the f number)*.

In order to find out the actual focal length of lens to use to achieve a given magnification and system length, the simple lens equation is used:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v} \qquad (3.1)$$

where f is the focal length of the lens, u the distance between the object and the lens *(the object conjugate)*, v the distance between the image and the lens *(the image conjugate)*, and m the magnification, defined as image size divided by object size, so :

$$m = \frac{\text{image size}}{\text{object size}} \qquad (3.2)$$

and, by symmetry :

$$m = \frac{\text{image distance}}{\text{object distance}} \qquad (3.3)$$

Thus, from (3.1) :

$$f = \frac{u \times m}{m + 1} \qquad (3.4)$$

So, if the required magnification factor and the distance between the object and the lens are known, the required focal length can be computed.

Choosing Focal length

The focal length of the lens is chosen to obtain the required magnification [71] Fig. 3.1(a-b) shows that image size can be reduced by choosing a shorter focal length lens. This might be done to match the image to a smaller sensor, or to image a larger object on to the original sensor. Most importantly, the choice of magnification is made to match the detail in the object to the resolution of the sensor. Most image sensors have a resolution limit expressible in terms of a number of picture elements *(pixels)* per picture height or width and, other constraints allowing, the optical magnification is adjusted to match this to the required inspection resolution. Fig. 3.1(c) shows that another reason for reducing the focal length can be to achieve a shorter system at the same magnification. From this it is clear why the shorter focal length lenses in a commercial range *(such as camera lenses)* are called wide angle lenses.

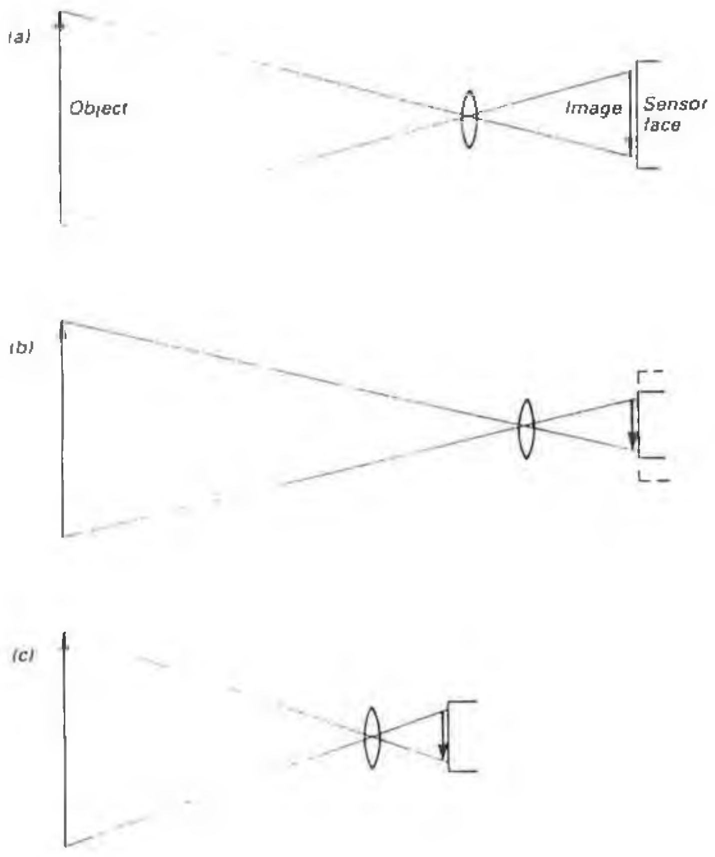


Fig. 3.1 Choosing focal length: (a) original design, (b) shorter focal length gives smaller image or puts more of scene on more of scene on to original sensor, and (c) shorter focal length shortens system length for same image size.

For example, in our case, if a 3cm wide object (*an IC with 21 leads/side*) is to be imaged on a 6.4 x 4.8mm sensor (*which was used in the project*) [73] from a distance of 33cm this implies a magnification factor of:

$$m = \frac{6.4}{30} = 0.2$$

so, using (3.4):

$$f = \frac{330 \times 0.2}{1.2} = 55$$

Thus, an 55mm focal length lens required.

Depth of Focus and Depth of Field

The depth of focus, is shown in Fig. 3.2(b). If the image plane can in fact be held rigid with respect to the lens, then all this focus tolerance can be used to accommodate object movement. The object may move by an amount equal to the depth of focus divided by the square of the magnification of the system. This is called the depth of field *(the distance between the nearest and the furthest points of a scene with acceptable focus at any aperture setting)*. Because of the fact that *u* and *v* have a non-linear relationship through the lens formula, the depth of field is not symmetrically disposed about the object plane. Also, the depth of field increases as the aperture closes *(as the f number increases)*.

Finally, many of the visual problems which may cause difficulty in interpreting a scene can often be solved by using some simple filters on the camera lens. Polarizing filter, for example, eliminates all light waves except those in one plane, so it reduces glinting *(spectacular reflection)* when viewing brightly reflecting materials *(i.e. metal, plastics, ceramics)*, gloss-paint, wet and oily surfaces [74]. Zoom lenses and lenses which are used with "*teleconverters*" and extension tubes suffer more distortion than fixed-focal length lenses used alone [4].

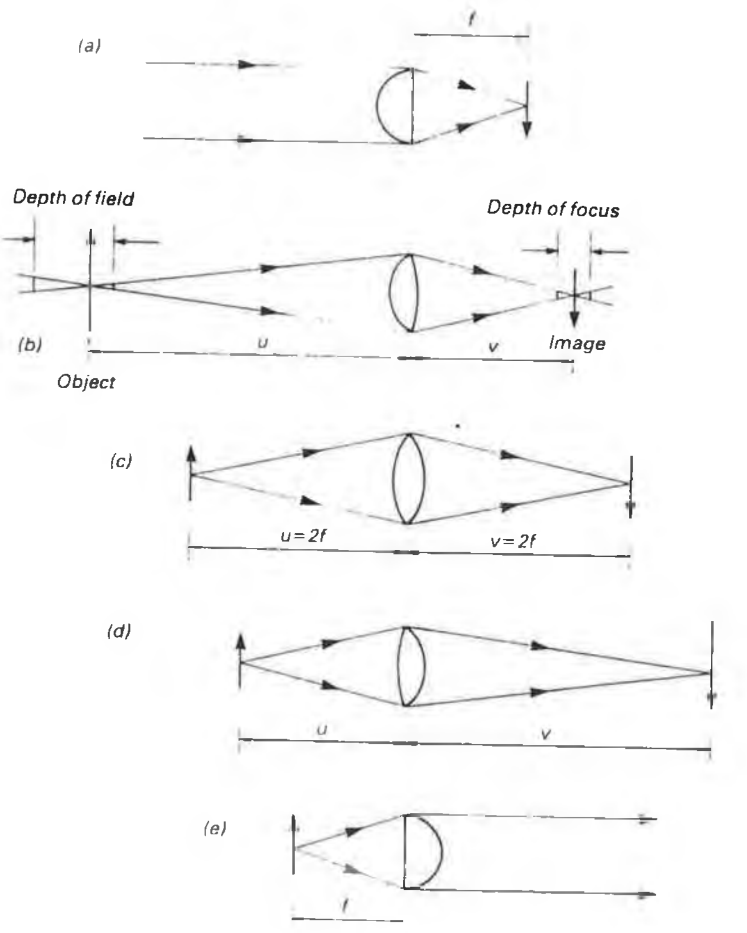


Fig. 3.2 Image formation basics: (a) image of distant object, (b) demagnification, (c) unit magnification, (d) magnification, and (e) collimation of light source or distant image forming.

3.3.2 Cameras

Camera sensors which used as inputs for processors in machine vision systems fit into two categories: linear devices and area *(2-dimensional)* devices. Each has strengths that can be used to its advantage. Linear solid state arrays, either CCD or photo diode, and scanned laser devices can provide single lines of video in high spatial detail in relatively short periods of time. Machine vision systems using linear input sensors are in use in lumber, paper, textile, steel and aluminum forming industry [75].

The area array category is served by a somewhat broader selection of devices. Tube sensors, principally vidicons, charge coupled devices, charge injection devices, and other matrix addressed devices such as photo diodes and MOS sensors.

A distinction is made between the following camera types depending on the sensor element: the standard vidicon has a sensor element made of antimony sulfide (Sb_2S_3), the silicon diode vidicon has a sensor element made from silicon (Si), and the plumbicon has a sensor element made from lead oxide (PbO), while the selfscanning CCD TV camera has a sensor element made from silicon (Si) [12].

Vidicon Tube

This is one of the two main methods in use today *(in addition to the orthicon)*. The target *(the inner surface of the face plate)* is coated with a transparent conducting film which forms a video signal electrode *(Fig. 3.3)*.

A thin photosensitive layer is deposited on the film, consisting of a large number of tiny resistive globules whose resistance decreases on illumination. This layer is scanned with an electron beam over 625 lines in accordance with the television standard. The beam is deflected magnitically by a set of deflecting coils outside the tube bulb. This electron beam makes up the charge lost through the incidence of light in individual mosaic cells and so generates the video signal at the sensor element. Synchronisation pulses which inserted by the camera electronics indicates scan lines, fields, and frame ends .

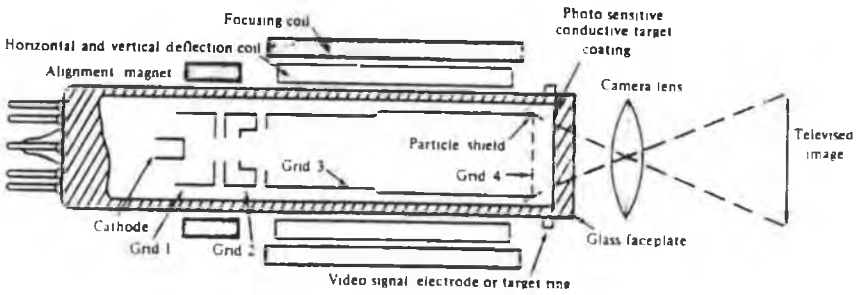

Focusing coil
Horizontal and vertical deflection coil
Alignment magnet
Photo sensitive conductive target coating
Camera lens
Televised image
Particle shield
Grid 3
Grid 4
Cathode
Grid 1
Grid 2
Glass faceplate
Video signal electrode or target ring

Fig. 3.3 The Vidicon.

Charge Transfer Devices

Solid state sensors, which present the recent development in image formation, known as charge transfer devices *(CTDs)*. There are two main classes of CTDs: charge coupled devices *(CCDs)* and charge injection devices *(CIDs)*.

CCD technology is the most widespread. In their structure, charge coupled devices resemble MOSFETs (*metal-oxide semiconductor field-effect transistor*) in that they contain a "*source*" and "*drain*" regions coupled by a depletion-region channel *(See Fig. 3.4)*. Charges in the depletion region are transferred to the output by applying a series of clocking pulses to a row of electrodes between the source and the drain.

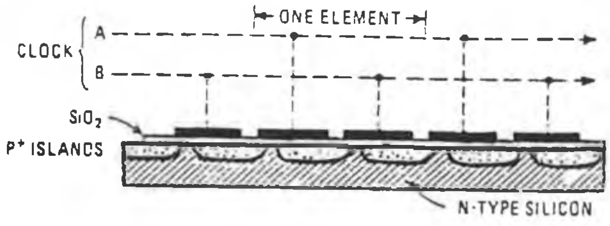
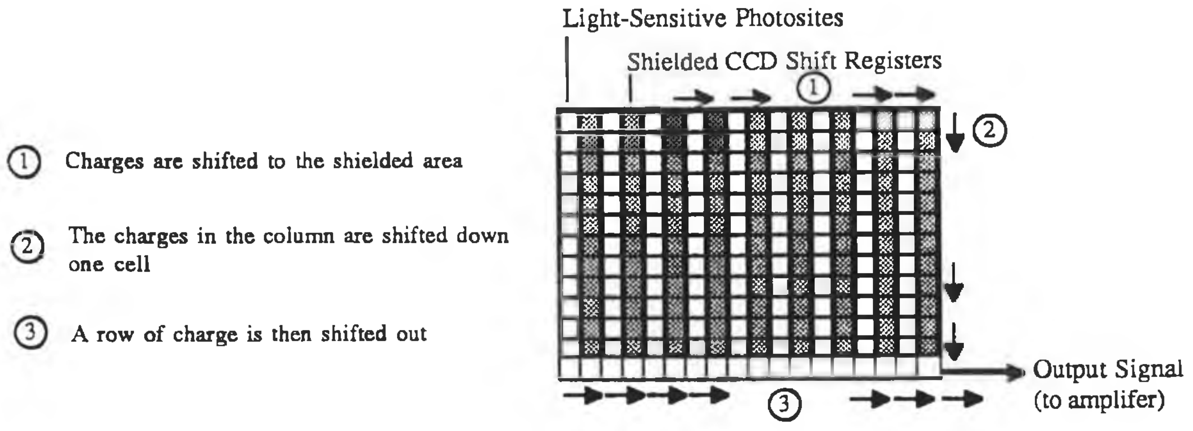


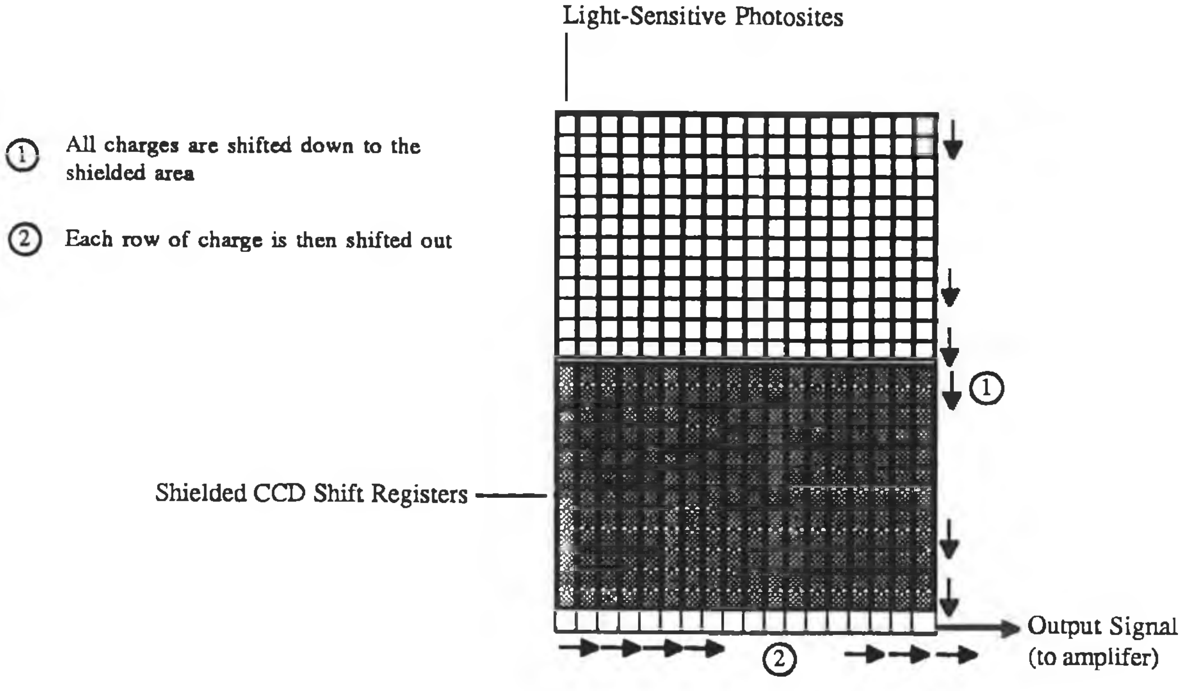
Fig. 3.4 Charge coupled device.

Photons incident on the semiconductor generate a series of charges on the CCD array. They are transferred to an output register either directly one line at a time *(interline transfer)* or via a temporary storage area *(frame transfer), as the last method is much more versatile. Fig. 3.5* illustrates both of the two methods.

In the frame transfer system, electrodes are grouped into two sections - an upper image section and a lower, optically-shielded store section. This enables a complete field of charge packets to be collected in the upper image area and rapidly transferred en-masse to the storage area. Whilst the charge is being transferred via a read-out register to an on chip amplifier, the next image is collected in the image section. The storage area is needed in frame transfer because the CCD array is scanned more rapidly than the output can be directly accommodated.


Light-Sensitive Photosites
Shielded CCD Shift Registers
①
②
① Charges are shifted to the shielded area
② The charges in the column are shifted down one cell
③ A row of charge is then shifted out
③
Output Signal (to amplifer)

3.5a Line transfer of charge in CCD sensors.


Light-Sensitive Photosites
① All charges are shifted down to the shielded area
② Each row of charge is then shifted out
①
Shielded CCD Shift Registers
②
Output Signal (to amplifer)

3.5b Frame transfer of charge in CCD sensors.

Fig. 3.5 Charge transfer methods.

CCD Sensors Characterizations

Charge couple devices have numerous advantages over conventional vidicon sensors:

1 - Lower blooming: Blooming is the spreading of incident illumination from an intensely illuminated pixel location to adjacent pixel locations, which produces an imaged spot whose apparent size is larger than its true size. Phosphor-based vidicon imagers are particularly prone to blooming; the physical separation of charge sensing sites in the CCD sensor tend to limit this effect.

2 - Faster response to incident illumination changes: The resultant bright streaks that occur (*e.g., when a vidicon television camera is panned through bright lights*) are due to the persistence of the vidicon phosphor. CCD (*or CID*) response time is governed primarily by the rate at which minority carriers may recombine in the substrate.

3 - Smaller size and lower consumption power: The compact size of solid-state cameras in general results because there is no need for an elongated vidicon tube.

4 - Light weight, high sensitivity, wide spectral and dynamic range.

3.3.3 Digitizers

Since with the current technology, the representation of an image as an analog electrical waveform is not suitable for further processing by the computer, the camera output should be converted to digital form (a *series of numbers which the computer can analyze*). For binary systems, the number is '1' or '0' and the digitizer is usually a comparator (*a comparator is a circuit which produces a '1' if the input voltage is greater than a threshold value and a '0' if the input voltage is less than the threshold*). For the grey scale systems, the digitizer is usually a circuit called an analog-to-digital *(A/D)* converter. The function of an A/D converter is to take as input a voltage such as a video signal and to produce as output a representation of the voltage in digital form, suitable for reading by an interface to a digital computer.

3.3.4 Camera Interface

The monochrome video signal used in the USA and Japan is RS-170, a subset of the NTSC (*National Television Systems Committee*) standard. In Europe the standard used, is the CCIR (*International Radio Consultative Committee*), which is similar to, but incompatible with RS-170. Since television video signal standards are not suitable for the requirements of machine vision, both standards present the same problem to machine vision applications.

Approximately 50 pictures per second are necessary for a flicker-free television picture. The same scene is therefore scanned twice. First in 1/50th of a second only the odd lines of a picture are scanned and then in the following 1/50th of a second the even lines are scanned. This means that a complete picture is established in 1/25th of a second. This procedure whereby one line is skipped in each frame is called interlaced scanning or skipping line scanning. So the number of complete pictures per second, i.e. the picture frame frequency is 25 Hz; raster field frequency is then 50Hz. With 25 pictures each having 625 lines, 625 x 25 = 15625 lines will be scanned in a second, i.e. the line frequency is 15625 Hz.

The signal from a line consists of the picture content and the synchronizing mark at the end of a line. With a line frequency of 15625 the time available for one line is:

$$T = \frac{1}{f} = \frac{1}{15625} \text{ s} = 64.10^{-6} \text{ s} = 64 \ \mu\text{s}$$

Of this, 11.5 μs are used for the blanking and synchronizing signal. It consists of the so-called "*porch*" and the synchronizing pulse lasting about 5 μs. The synchronizing pulse signals the beginning and end of a line so the poarch prevents the beam jumping back to the beginning of the line flyback from appearing as a bright line. The end of a picture consisting of 625 lines is characterised by several picture pulses which are significantly different from the line pulses. During these picture pulses the picture is blanked. Fig. 3.6 shows the video signal of a television camera.

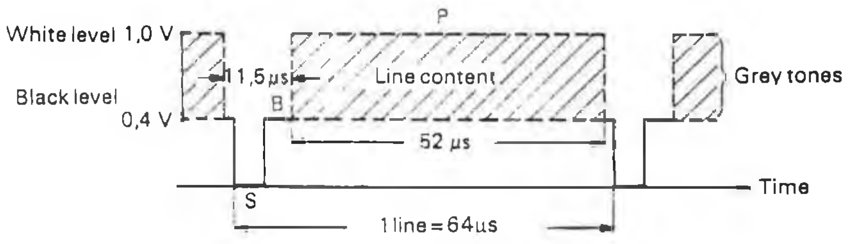


Fig. 3.6 Video signal of a television camera.

3.3.5 Camera Specifications

There are several measurements of the reliability and usefulness of a camera sensor. It is found that the camera characteristics that most directly relate to accurate transduction abilities are its sensitivity, noise, and resolution, all of which depend on details of the pixel arrangement and construction.

Resolution is an important camera characteristic. The resolution is often specified by the size and number of pixels. Machine vision systems often need to make measurements at or near the camera's resolution limit. Even when small feature detection is not important, improved camera resolution resulted in improved accuracy in determining object location, orientation, size, etc. The sensitivity is measured by sensor saturation illumination or, by recommended and scene illuminance. The camera noise is characterized by a signal-to-noise ratio *(SNR)*. The most usual *SNR* is a comparison of the maximum signal level with an *RMS* or time-averaged dark-level signal.

Finally, camera specifications usually omit crucial information about a camera's adherence to or improvement upon RS-170. For example, a strict reading of RS-170 allows geometric distortions and line-edge shifts of several pixels whiten an image, far beyond what would be acceptable in most machine vision applications. There is also ambiguity in RS-170 about whether the video dark level is to be set relative each horizontal line's "*back porch*" or relative to some time-averaged level throughout the frame; this could allow a dark-level shift that is equivalent to many least-significant bits of grey-level information [76].

3.4 Conclusion

Vision systems are mainly composed of a video camera, lightning source, frame grabber and image processor, ranging from the simple to the highly sophisticated. Both vidicon and solid-state matrix type cameras are used. Vidicon cameras have a wider illumination intensity working range than matrix cameras, but they lack the light wight, and simplicity of solid-state cameras. As a consequence, solid-state cameras are widely used in the industrial work place [77]. These cameras can be obtained in line scan and matrix models. The matrix type is best for general inspection applications. By introducing digital output cameras [78], accurate imaging can be achieved, as well as reducing system cost and complexity. *Television cameras* are cheap but are rather expensive to interface to a computer. *Solid-state cameras* are the most widely used image sensors for industrial applications, on account of their precise pixel geometry, smaller size, and improved robustness, compared to CRT cameras.

As lighting and camera sensors are among the most important factors in the vision systems, an important consideration is the choice of these and as such this review has been undertaken.

Chapter

4

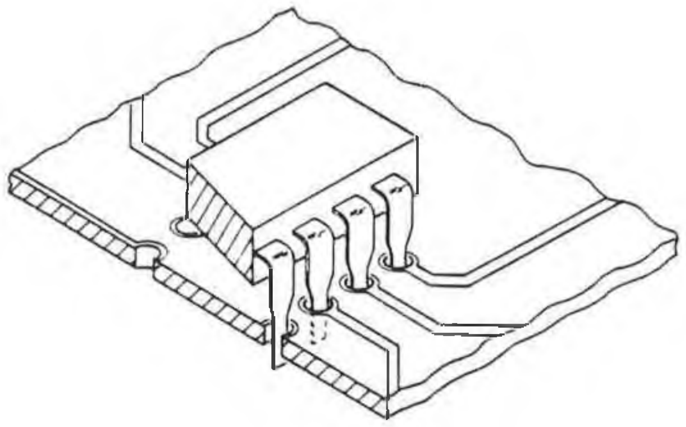
Surface Mount Inspection

Surface Mount Inspection

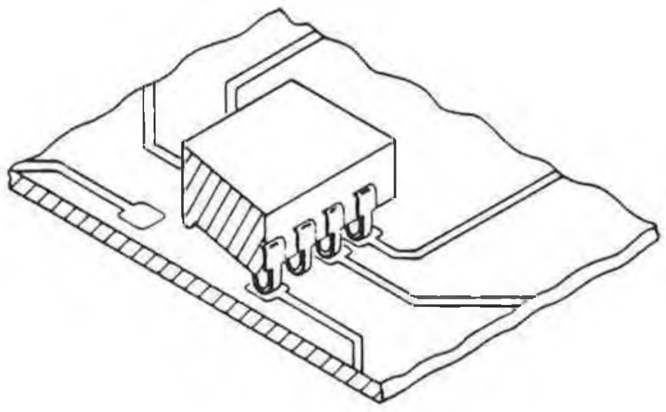
4.1 Introduction

Surface mount devices (SMD), which represent a recent development in the miniaturization of electronic components [79], are rapidly evolving into certain electronic industries such as aerospace and communications [55]. Many manufacturers of electronic products are migrating to the use of leadless and surface mount devices. Component complexity with larger pinouts and increased packaging density have made quality control and reliability major competitive issues, as product features and performance are less distinguishable across vendor lines.

Surface mount is a relatively new method of solid-state assembly, which helps to reduce the size of the components with the same functional capability (*or more functions*), thus reducing cost. Surface mount technology provides new techniques for assembling packaged integrated circuits and related components onto printed circuit boards to obtain reduction in PCB size of *40%* to *60%* for the same density. It also provides increased reliability, and allows reduction in assembly cost of up to *50%* [79]. In addition to the previous advantages, there are benefits for warehousing and manufacturing. Less warehouse space is required both for the storage of components, and for the storage of the final assemblies because of the smaller size. Fig. 4.1 illustrates the difference in mounting between a typical DIP[*] component and its surface mounted counterpart.



a. DIP Through-Hole Assembly.



b. SMD Surface-Mount Assembly.

Fig. 4.1 Comparison of Through-Hole and SMT[#].

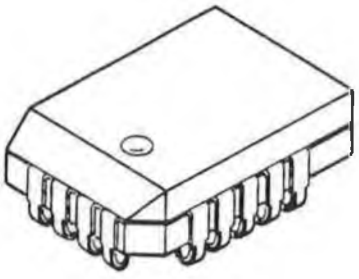
4.2 Surface Mount Process

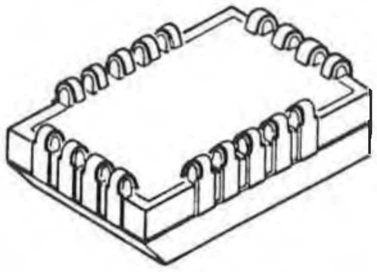
Surface mounting is a process in which a packaged integrated circuit or other component is physically mounted on the surface of a PCB instead of inserting the leads into

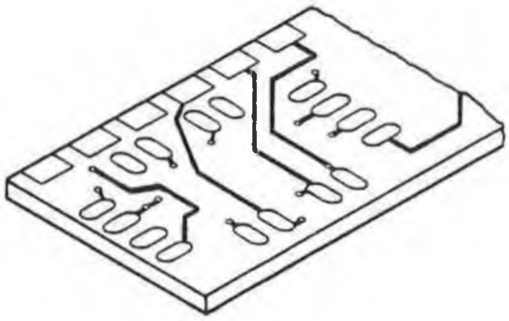
(*) Dual In-line package.
(#) Surface Mount Technology.

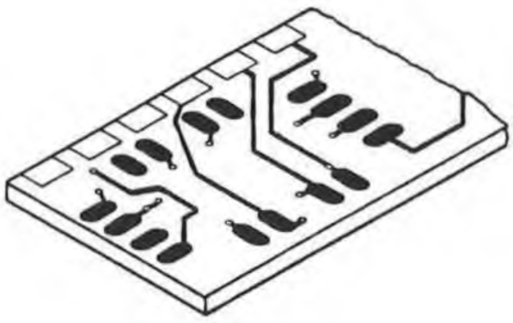
plated holes through the PCB. Fig. 4.2 illustrates the surface mounting process. Prior to mounting the leads should be tinned to provide a better solder joint. Extra solder is therefore required to form this joint. The next step in PCB preparation is to "*print*" the solder paste onto the pads as shown in Fig. 4.2d. This is normally done by stenciling on the solder in paste form. The paste provides the necessary solder to form the joint fillets so important to electrical and mechanical connection. Then the component is placed on the fresh solder paste (Fig. 4.2e) and the operation is completed through a vapour phase reflow soldering process which melts the solder and bonds the component to the PCB as shown in Fig. 4.2f.

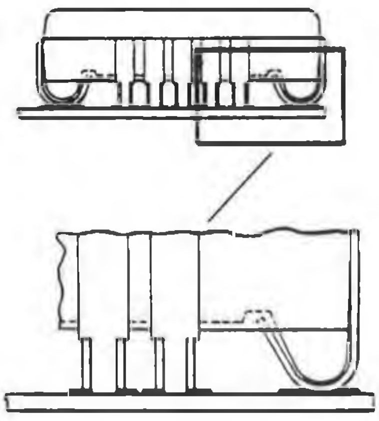
There are opportunities to use machine vision during different stages of this process. For example, to align the silk screen during stencilling, to check location and volume of solder paste, placement process of the component, and inspection of the solder joints. By eliminating the need to pass pins through holes in the PCB, the pad spacing can be reduced dramatically as we have seen. This can increase the component density, so that new and more difficult performance goals are placed on the inspection systems in terms of resolution and speed requirements [62].


a. SMD Top View.


b. SMD Bottom View.


c. PCB Footprint.


d. Solder Paste Applied.


e. SMD Placed.

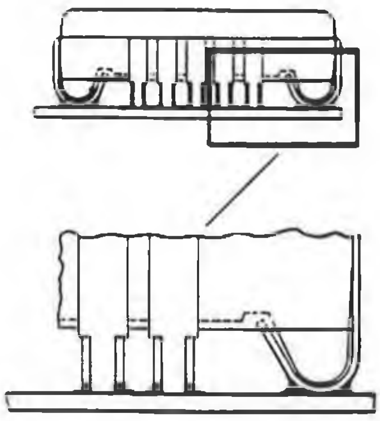

f. SMD Soldered.

Fig. 4.2 The Surface-Mounting Process.

4.3 Surface Mount Inspection

The problem in the inspection of solder joints on surface mount printed circuit board is much more difficult than other inspection problems, and it is not easy to apply machine vision systems for that task because of the variability in the appearance of acceptable joints. It is not simply the process of detecting the presence or absence of a feature, but of determining whether the solder joint is good or belonging to one of several defect classes based on the appearance of its surface. Also what makes this task more complicated is that the joints surface behaves as a mirror which reflects illumination *(specular reflection)* due to the angles of illumination and viewing. In addition, there are many types of bad joints, and different types of leads.

Many systems have been developed using several different approaches; 2-dimensional intensity analysis; X-ray techniques; laser heating; acoustics; and techniques based on grey-scale images [80]. It has been found that approaches based on edge detection or template matching are not adequate for this purpose [58]. Also laser heating techniques are often not preferred because of possible damage to solder joints. Approaches based on grey-scale images, will hopefully provide a widely applicable solution to the inspection problem.

4.3.1 Surface Mount Integrated Circuits Faults

There are some common defects which might be encountered when inspecting any solder joint. Any of these defects can prevent the product from performing according to its specification so great care must be exercised in performing the inspection. In general, faulty solder joints are very costly defects both for electronic equipment vendors and customers. Some reports estimate that even a solder joint failure rate as low as 0.05% can cost a large manufacturing organisation over three million dollars per year [58]. Any effective techniques to improve solder joint inspection will be welcomed by industry and should have widespread beneficial results.

Fig. 4.3 shows the solder joint types in the surface mount technology and the through-hole technology. As we can see, there is two different types of solder joints in the surface mount technology, Gull-Wing and J-Lead. Examples of approaches used in Gull-Wing inspection, are based on grey-scale images using 2-dimensional system [80], and structured light using 3-dimensional vision [60].

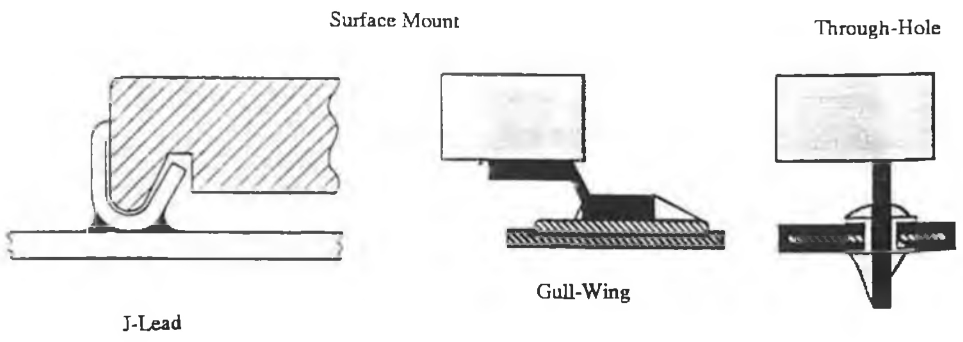


Fig. 4.3 Solder Joint Types.

Based on samples of surface mount assembled printed circuit boards from DEC[*], which contain surface mount integrated circuits with 'J' leads, there are various classes of soldering faults. Frequent production faults are:

1 - Insufficient solder *(leads bent inside the vertical axis without sufficient solder)*. See Fig. 4.4.

2 - Bent leads *(leads bent to the left or right of the vertical axis)*. See Fig. 4.5.

3 - Misplaced leads *(no correspondence between leads and pads)*. See Fig. 4.6.

4 - Solder bridges *(shorts between two leads or more)*. See Fig. 4.7.

5 - Excess solder *(too much solder on the leads)*. See Fig. 4.8.

There are a number of other defects that may not affect performance immediately, but may affect the long term reliability of the product [79]. Flux residues, for example, may induce corrosion after a period of time. Too much solder makes a proper inspection impossible and voids may not be detected. Too little solder results in a weak joint or an insufficiently defined fillet and can cause

(*) Digital Equipments Corporation .

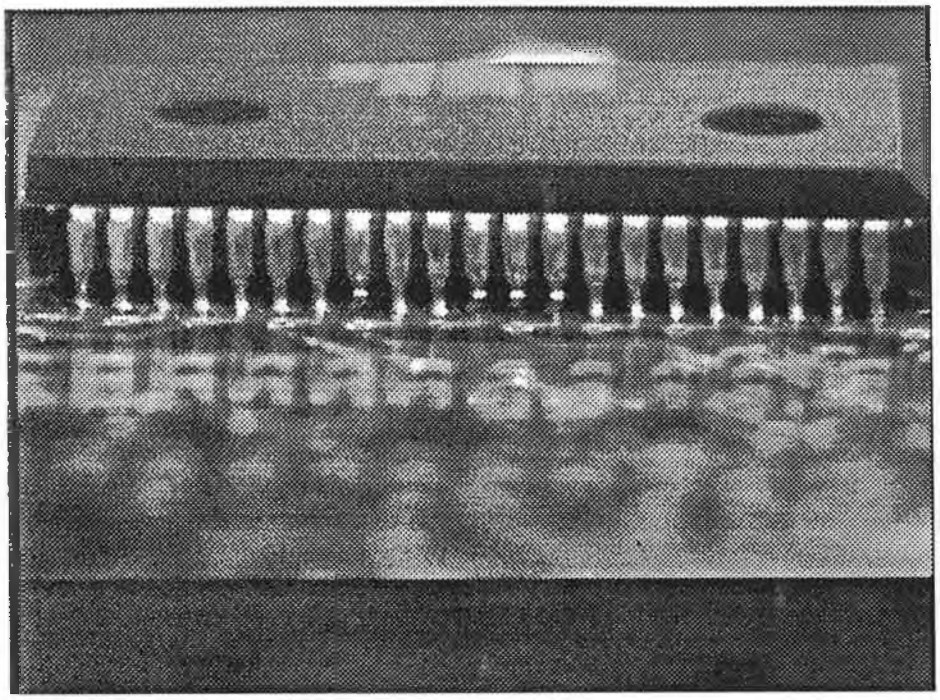


Fig. 4.4 Insufficient Solder (leads No. 8, 11, 12, 13).

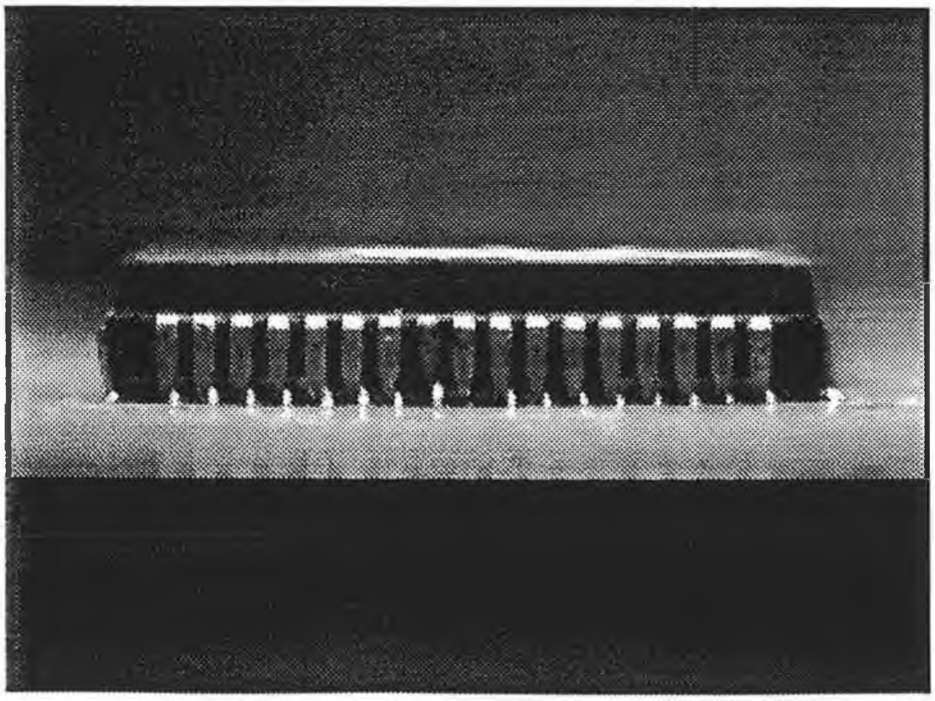


Fig. 4.5 Bent Leads (lead No. 8).

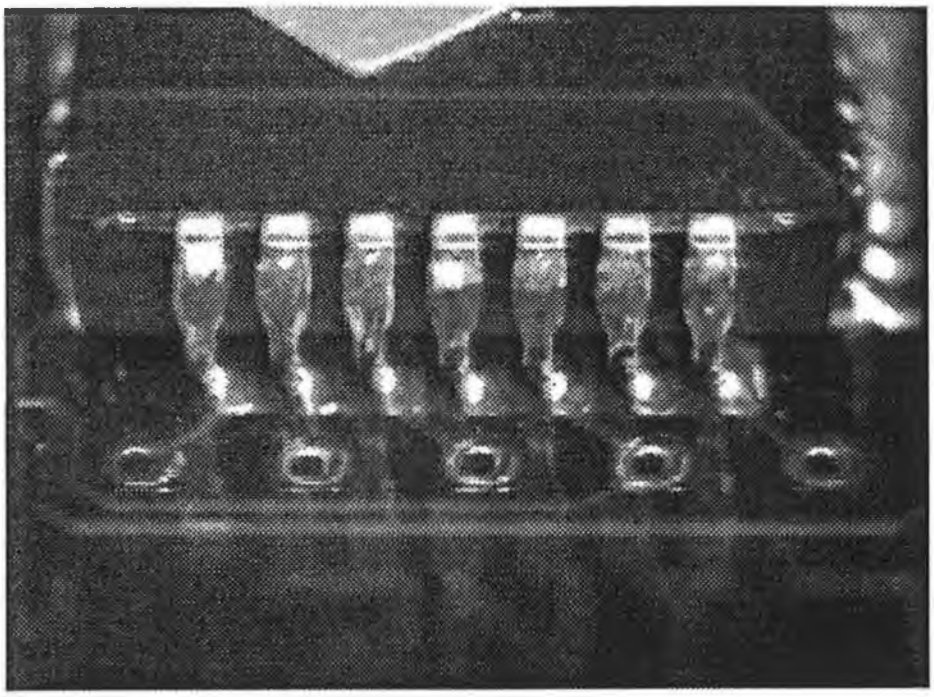


Fig. 4.6 Shifted Leads.

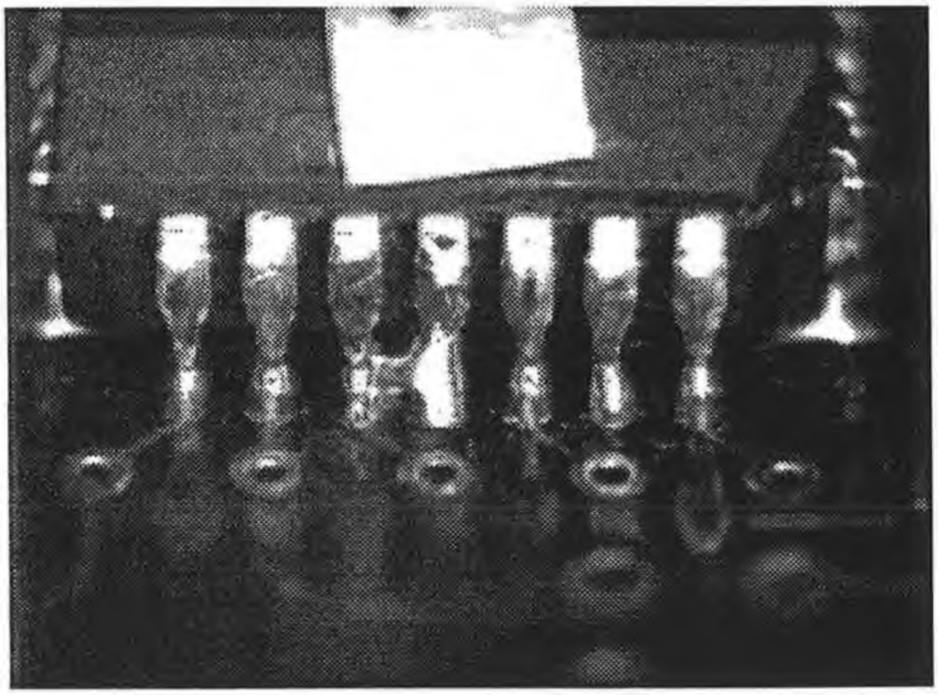


Fig. 4.7 Solder Bridge Between lead No. 3 and 4.

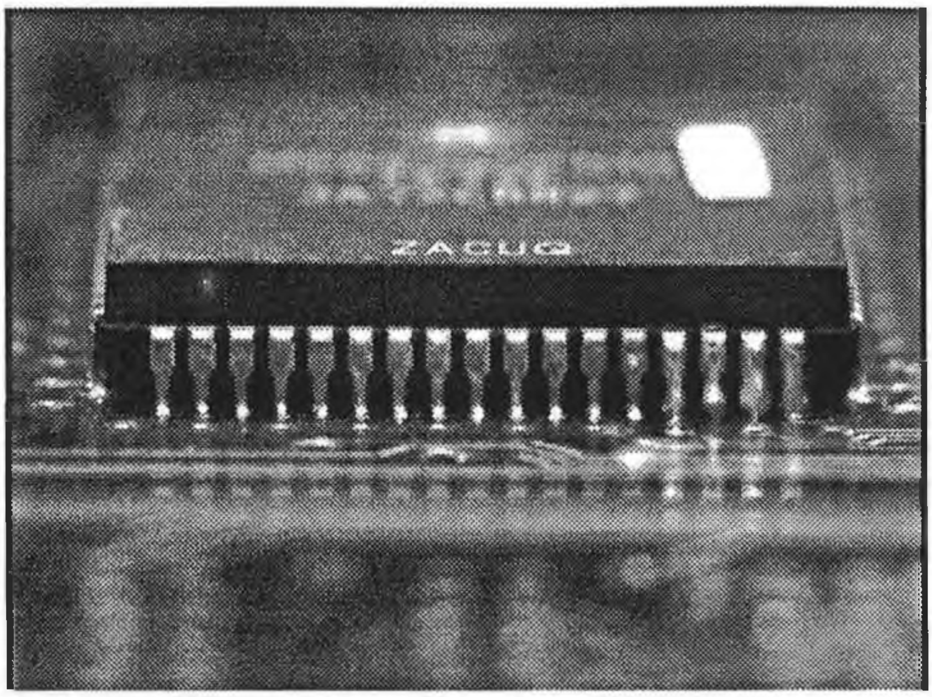


Fig. 4.8 Too much solder on leads No. 14, 15, 16 and 17.

reliability problems, especially if the assembly is subjected to high vibration.

4.3.2 Inspection Task

There are several tasks of surface mount inspection [37], some of these are:

- ***Pattern inspection***: which includes, inspection of the circuit patterns and conductive layers composing PCB substrates for pattern match, alignment, positioning and breaks.

- ***Alignment inspection***: this includes, inspection of components location, presence and alignment before soldering.

- ***Lead inspection*:** which include inspection of leads on components for defects in alignment, spacing, count and so on.

- ***Solder inspection*:** includes, inspection of solder paste deposits volume, and solder joints (which represents our case).

The above named faults and the demands on function assembly bring about the following inspection tasks:

1 - Location of leads in the image (*single frame*).
2 - Determination of lead width, length, number of leads, and other parameters which are necessary for the recognition functions.
3 - Detection and classification of the above named faults.

4.3.3 The Need for Automated Inspection

The inspection operation is carried out by a microscope especially one with a zoom lens and different powers of magnification. It costs less than some other inspection systems, but microscopes have the disadvantage of causing operator fatigue rather quickly. In addition these methods are limited to defects that are external and are not hidden from view of the operator by a component.

Since, even with the aid of microscopes, visual inspection no longer may be adequate for inspection of solder joints, different technologies, including X-rays and lasers, must be employed. Also automated visual system will need to be employed in order to eliminate human involvement in the inspection task.

4.4 Inspection of "J" Leads

In many applications, attention has been given to inspection as a problem separate from the manufacturing process. It appears that the detection of a fault is not enough if we want to develop a completely automatic manufacturing system. It may be necessary to find the type of defect and to decide the reason for this defect. If there is a particular stage in manufacturing that may be responsible for the defect, then corrective action can be taken.

A 2-Dimensional system based on grey-scale image data, has been used to classify Gull-Wing solder joints defects [80]. This system can be driven from layout information available from CAD data, and produces results for in-line use. From Fig. 4.3 it is very clear that the inspection of 'J' leads, is much more difficult than the inspection of Gull-Wing leads because the different type of the solder joint, and cannot be seen from above, as well as the uselessness of CAD layout informations here, but it may help in an early stage, for example to find chip locations on the PCB (See chapter 6).

4.4.1 Inspection Algorithm

2-Dimensional machine vision, the most common type, views objects by comparing differences in light reflectance often referred to as grey scale. Thus the machine can detect the boundaries of objects, of course there must be sufficient contrast between the object of interest and its surroundings. An approach based on 2-dimensional grey-scale analysis as well as thresholding technique, is developed for the inspection of 'J' leads integrated circuits.

The detection procedures* are simple and reliable. After loading an image of SMD chip to the memory (*one side of the chip each time*), the program looks for the leads of the integrated circuit. The location of leads performed by computing variance of intensity values for each line in the image, until the line which has the maximum variance is found, the assumption in this line is supposed to contain the top edges of leads. Having found the first line of leads, it takes this line and by using a threshold, it finds the start lead, lead width, number of leads, the space between each two leads, and estimates leads length and other different parameters, which are used by the other detection functions. By using a similar algorithm the program was able to detect the faults mentioned early in this chapter.

(*) See chapter 6 for details.

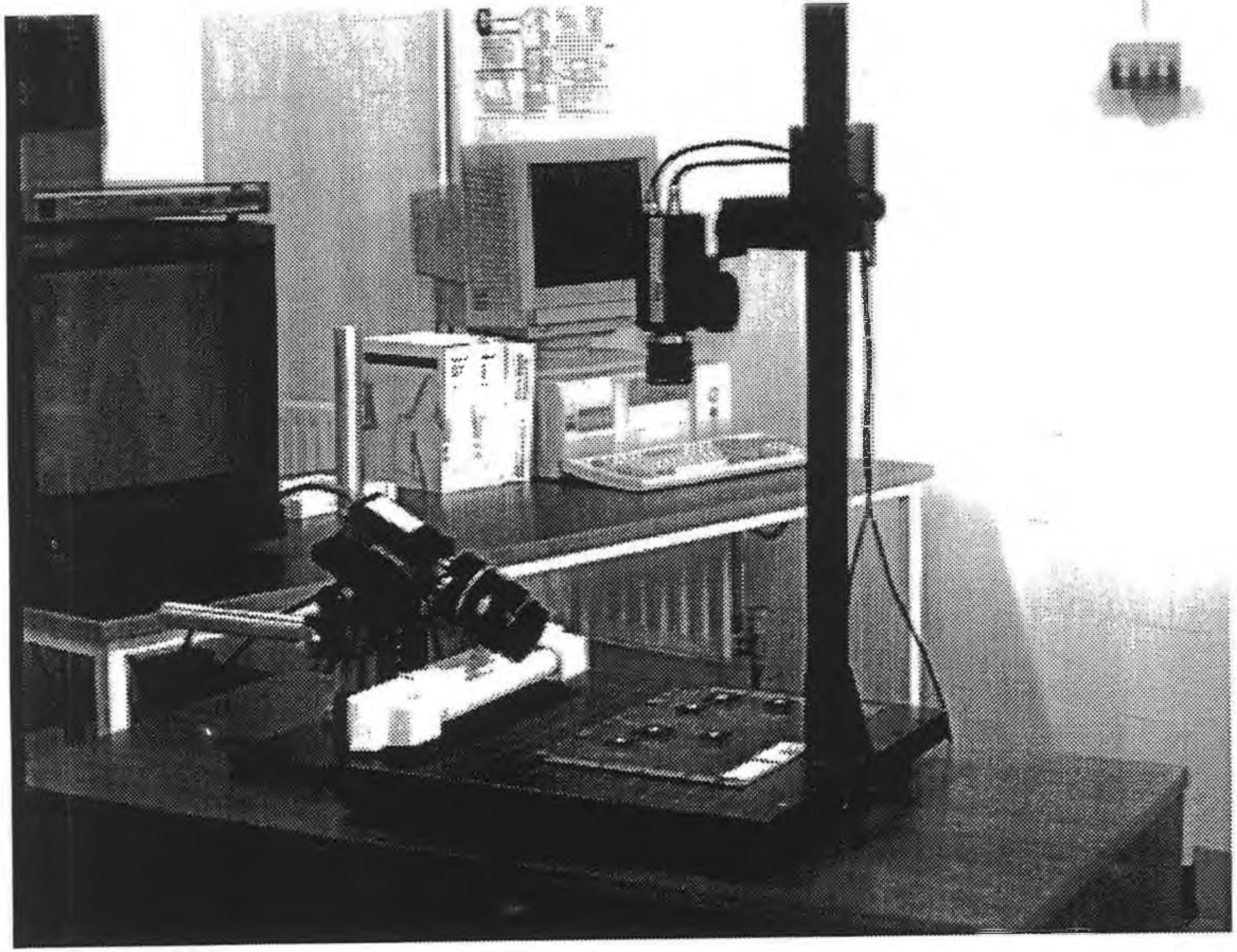


Fig. 4.9 The Lighting and Camera Setup.

4.4.2 Lighting

Lighting is a difficult point in the optical systems. Several lighting methods were tested (*tungsten filament bulb, circular fluorescent tube*) and different angles has been tried. A linear fluorescent light *(10 inches long tube)* at low level was found the best way to obtain the uniform image of leads. With a single camera at 30 degrees to the inspected board, it was possible to observe a good solder joint as a bright spot, while a bad joint produces no such spot. The inspection algorithm works well with the board aligned horizontally. Fig. 4.9 shows the lighting and the camera setup.

4.5 Conclusion

Surface mount technology is being widely used, and, as this technology continues to grow in popularity, more new or improved devices will be added to the product lines. Each development in this area has required new techniques, not only for product design and assembly, but also for the inspection function. Therefore many techniques and different approaches have been developed for the inspection process during different stages of surface mount assembly.

Chapter

5

Computer Assisted Quality Control

Computer Assisted Quality Control

5.1 Introduction

With increasing competition in manufacturing industries and product quality becoming even more important, the latest developments in automated manufacturing have generated a need for Computerized Quality Control. Controlling quality by inspection means performing measurements on each required characteristic against specified limits by using calibrated measuring equipment or predefined features for the product. This is to ensure that all characteristics, including performance, are within desired limits.

Computer systems are particularly appropriate in seven key areas of quality control:

* Data accumulation.
* Statistical analysis.
* Information retrieval.
* Real-time process control.
* Automatic testing and inspection.
* Quality management-related techniques.
* Data reduction, analysis, and reporting.

The use of machine vision in automatic inspection helps in quality control where the concept of total inspection (*or 100% inspection*) is introduced. Total on-line inspection is defined as the inspection of all parts as they come down the line. If faulty parts could be detected right on the production line, the remedy could be implemented immediately. Future generations of inspection systems should be able to do recovery action when it detects faults. Presently, on-line inspection helps to keep the risk of producting faulty objects low [31]. In some industries there are stringent requirements for the reliability of products, and *100%* inspection may be the only answer [30].

The earlier systems tend to be inflexible and newer generations of machine vision systems try to overcome this by incorporating programmability into the system. The extreme case of the inflexible system is a turnkey system where the equipment manufacturer must tailor fit exactly into the predefined environment. Many manufacturers of machine vision systems prefer the off-the-shelf systems so that they can reach a bigger market.

5.2 Inspection Models

A simple inspection model is shown in Fig. 5.1. In this model, every part is inspected, and parts that the inspection system classifies as defective are scrapped. A computerized and more realistic model of inspection is shown a Fig. 5.2. In this model, every part is inspected, defective parts can be reworked and reinspected by the system (*or by human inspector*) or subjected to another

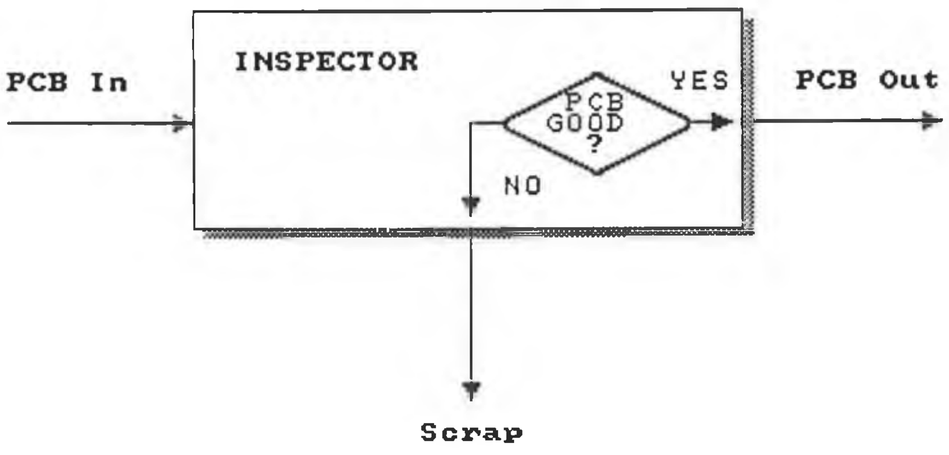


Fig. 5.1 Inspection with simple model.

inspection processes, instead of being scrapped. Parts that can not be repaired (*or reworked*) are scrapped. Samples of good parts are inspected by human inspector to ensure that the inspection system works well. More important, inspection information can be used in a number of ways in addition to screening or sorting parts, for example:

1 - It can be fed back to reduce or minimize false alarms on good parts, and to remove or prevent the false pass on bad parts.

2 - It can be fed back "strategically" for the purpose of process learning, for example, historical data on defects might be used to develop material handling regimes that introduce less contamination.

3 - It can be fed forward to control subsequent processes (i.e., for rework).

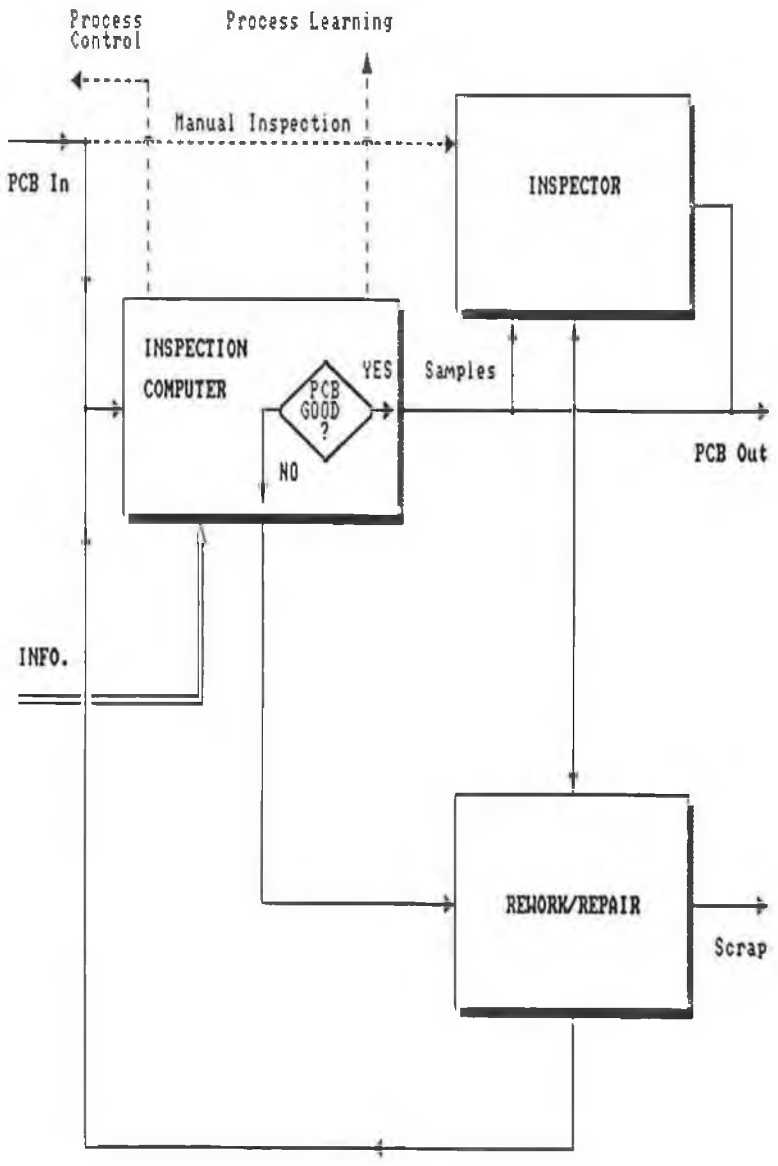


Fig. 5.2 Computerized Inspection System.

This model is much better than the first, because: Point 1 reduces yield loss, point 2 helps maintain yield, and point 3 helps accelerate the improvement of yield as new products and processes are being introduced.

5.3 Quality Control Aims

Quality must be implemented to aim for the elimination of waste in order to increase productivity. Automated inspection systems using machine vision can offer *100%* on line inspection; labor savings; improved productivity and reliability; faster changeovers; and reduced downtime.

5.3.1 The Inspection Operation

The ideal inspection system will increasingly be considered as part of the process; it should provide outputs which as far as possible can be used for automatic feedback and process control. Using machine vision systems in inspection should minimize the number of human inspectors. Also a good informative human interface will improve the interaction between the system and the operators which should lead to best results.

5.3.2 The Vision System

The performance, efficiency, and the reliability of the inspection system should increase as machine vision systems do not get tired of inspecting as humans do. Inspection with humans is prone to error for the case of repetitive work for a long period of time.

Another important point is the increase of flexibility and capability as automatic inspection systems work in a wide variety of circumstances and environments. Also labour requirements are lower which means cost saving as well as product speed increase as vision systems are typically quicker than humans.

5.3.3 The Product Quality

Quality is the totality of features and characteristics of a product that bear on its ability to satisfy a given need [81]. It is a complex and multifaceted concept, its basic elements include conformance, durability, reliability, and performance.

Conformance is concerned with the degree to which products design and operating characteristics match the standards (*CAD standardizes drawings, CNC and Robots reduces process variabilities*). The introduction of new technology will improve the degree of conformance within a manufacturing environment.

Durability is concerned with product's useful life. In other words, the length of the time a product is expected to operate. So good inspection means keep the durability high.

Reliability is concerned with the probability of a product failing within some specified period of time. Thus, inspection is very important stage which will offer greater potential for defects detection. This in turn will lead to achieve a higher degree of product conformance and reliability.

Performance is concerned with primary operating characteristics of the product. Automatic visual inspection can help to improve product performance by eliminating several defects on the product.

5.3.4 How to Achieve Aims

To achieve the aims listed above, of quality control it is important to highlight the problems and give good description of these problems which include, the printed circuit board and the environment (*inspection system*).

Also another important thing required, is effective algorithms should be carefully designed to reliably perform the inspection function at the required speed, as the speed depends on the computer and the processor used.

Most inspection systems in use nowadays are using CAD informations [62,80,82], and some of them are using Knowledge based systems [65]. In an assembled PCB inspection if the system is a stand alone system (*without CAD informations or knowledge base or any other interface*), and the system is capable of inspecting different types of PCBs, different leads type and size and so on, then the system will be more flexible.

The inspection system should be provided by an indication of errors and their description as well as statistics on errors (record or list of each type of faults and locations). Finally a suitable strategy should be used to get better results.

5.4 Inspection Strategy

The inspection of Surface Mounted Integrated Circuits (*IC*) on a printed circuit board (*PCB*) requires the ICs locations on that PCB as a first step. After that the camera moves to each IC on the PCB for the inspection of each side against the defined defects. This can be implemented in several ways. First approach is to use multiple cameras, that require special hardware as well as CAD information. It is very clear that this approach is expensive due to the number of cameras, software, and hardware required to control the movement of the board. The second approach can be setup by using a single camera mounted on a robot arm. With CAD information about the locations of the ICs on the PCB, the camera can move and reach any part of the PCB, which allows the system to look at each side of the ICs sequentially and independently. This approach like the first one is very expensive.

For the inspection of J-Lead surface mounted integrated circuits there is another approach based on two cameras (*over head camera and side camera*) without any other interface (*CAD information for example*). The view from the fixed overhead camera can provide information about the ICs location on the PCB. The second camera (*should be mounted on a robot arm*) moves to each IC location on the fixed PCB (*by using location informations provided from the first camera*) and detects possible defects on the leads.

Lighting from above may be used for the illumination of the PCB to find out ICs locations. Lighting from the side* of the IC by means of a fluorescent lamp is used for the illumination of leads for defects detection.

5.5 Conclusion

Vision has brought to industry miscellaneous benefits. Increased quality, better knowledge of the production process, better productivity and "the solutions" to industrial automation. With an increased concern over quality control and long term issues, vision technology can bring innovative tools to manufacturing on the factory floor.

(*) See chapter 4 for details.

Chapter

6

J-Lead Integrated Circuits Inspection

J-Lead ICs Inspection

6.1 Introduction

Computers are at the heart of machine vision. Whether the application requires a generalized or specialized system, programming is the key to the success or failure of the project.

The first programming option is to use low level programming (or hardware programming) with *PROMS*. In this case, any future changes require a new chip. Another option would be to use an *EPROM*, as the changes can be made to the chip, without the need for a new one. The advantage of hardware programming is the dramatic increase in the processing speed, but on other hand, this option is rather inflexible, especially in the case of product design development, where frequent revisions may be necessary.

High level programming is the second option. Where the application program is written in one of the programming languages and the hardware stays fixed. The biggest advantage of this approach is flexibility , which is one of automation's main requirements. Here the inspection will be slower, as the processing speed decreases considerably. This approach would be suitable in the case where having a flexible system is essential, or the inspection operation is a stand-alone system and not part of an automated line.

The main program, *INSPECT* is an interactive program to apply various inspection routines to an image. After snapping an image the program reads it line by line and calculates intensity variance for each line to locate the first edge of the leads. When the first line containing leads is located the program looks for the start lead by computing the variance of each pixel in the line. Having found the first lead, lead width, number of leads and lead gap width can be calculated and lead length estimated.

The most important algorithms (badleads, balleads, bridges, shiftleg) embedded in INSPECT are described in this chapter. All programs are listed in appendix A. The "*badleads*" performs the task of detecting the leads which do not have enough solder in the soldering point. The "*balleads*" detects the leads which have excessive solder on the leads. The "*bridges*" performs the task of detecting any solder bridge between two leads or more. The "*shiftleg*" detects the misplacement of the integrated circuit on the printed circuit board. In interactive programs, the need for flexible and efficient methods for manipulating and accessing large amounts of data is central to both execution time and storage. Therefore it was felt that the program should have the following properties:

1 - It should be easy to use.

2 - It should be able to operate in automatic and manual mode.

3 - It should include comprehensive hardware control utilities to allow it to perform image acquisition and processing and other operations.

6.2 Structure of the program

In general, there are two programming methods that can be used when designing a computer program. The first of these is to write the whole program as a single program unit. The main limitations of this approach are as follows:

1 - Every time a minor modification is added to one part of the program the whole program has to be recompiled.

2 - It is difficult to design test cases for large programs, particularly because of the large numbers of different types of input data they may have to deal with and the immense number of different paths through the code.

3 - Only a single programming language can be used throughout the program.

The second approach, known as modular approach, allows the programmer to break a large program into a number of independent routines which can then be compiled to form a complete program. Each routine performs a specific function or set of functions on a defined set of data.

This technique has the following benefits:

1 - Routines could be used many times from different parts of the program. In addition, these routines may be used in other programs without the need to rewrite them.

2 - Each routine can be tested individually before incorporation into the final program.

3 - Many different programming languages (*such as C, Pascal, Assembly, etc.*) can be used to write different routines for a program.

4 - When an error is detected in the program, then only one of the routines needs to be recompiled.

5 - Large programs can be constructed in small steps.

6 - The readdability of the program is improved enormously.

7 - Adaptation to other systems is made easier.

From the above discussion, it is apparent why the modular programming approach was adopted for implementing the software of the project.

The equipment used is based around the '*PCVisionplus*(*)' board located on the Arc personal computer with 80386 processor. The computer hosts the driving software for the image processing board, and the processing software which has been developed for the system as C routines. The camera

(*) A trademark of Imaging Technology Inc.

used is a *Photon* CCD from EEV yielding 512x512 pixel resolution in the frame buffer [73]. A 55mm lens is used for good operating distance.

The interface card connects directly to the expansion ports of the computer. The camera and a colour monitor are connected to the video input on the interface card.

6.3 Software Description

Two types of software contribute to the computer as an automatic visual inspection workstation. The first type includes the commonly used image processing languages such as *C,* which is a popular programming language, especially for process control. The second type consists of all the tools for manipulating image and graphical data. Most of these were written by Computer Applied Techniques Ltd. (CAPTEC).

6.3.1 Inspection Program.

The computer program that performs the inspection, called *INSPECT,* has been written in C and compiled using Borland's Turbo C V2.0 . It has been linked with some image processing routines *('MAVIS'* from *CAPTEC*) for driving the hardware and the result is a compact executable module which runs under MS-DOS 3.30.

To simplify user interaction, a keyboard-controlled menu (*and mouse as well*) was developed for the operation of the system. Also the program places coloured areas in the examined places. There are many options in the main menu, but the most important option is the *AUTOMATIC* inspection. The automatic option starts and performs all the necessary operations for inspection which includes grabbing the image, locating the leads of the IC, detecting all the soldering faults, and so on. There are another four options which allow the user performing the inspection operation for each defect individually. The other options allow tasks including savings special images as a record of some faults to be performed.

6.3.2 Software Interface to the system:

A number of functions interface the image processing hardware to *INSPECT* (See Table 1). These *'MAVIS'* functions are supplied with the hardware [83]. The 'fginit()' function for example initializes the hardware (PCVision*plus* registers) in the system to a known state.

Table 6.1: Interface Functions

fgconfig	reconfigures MAVIS for non-default board settings.
fgclear	sets the pixels in an area to a single value.
fgsync	selects the video input synchronisation.
fgmode	sets the access mode to frame memory.
setup_menu	initializes a menu with appropriate data.
fgmask	selects write enabled bit-planes.
run_menu	an interactive menu driver.
clear_menu	clears the defined menu from frame memory.

fgvmask - selects bit-planes for image acquisition.

fgzoom - sets the position and the zoom factor.

fggrab - acquires images from the camera continuously.

fggain - adjusts the gain of the framestore's video input signal.

fgsnap - acquires a single frame from the camera.

fgwait - waits for the next frame.

fgsave512 - saves a 512x512 area to file at high speed.

6.4 Inspection System

The inspection system of the J-Lead surface mounted integrated circuits on a PCB is shown in Fig. 6.1. It consists of two CCD cameras, fluorescent lighting tube, table, Personal Computer, and TV monitor. (*It is worth mentioning here that one camera is adequate in the case of Gull-Wing leads because the possibility of using CAD information*).

The first camera (*fixed over-head with 16mm lens*) can be used to detect[*] the presence of the PCB, then it determines orientation of PCB, and the locations of the ICs on PCB. This information is then sent to the inspection process which is performed using the second camera. This second could be mounted on a robot arm[#]. The robot arm moves the camera to the sides of each IC, and looks for possible defects on the leads.

(*) Not implemented.
(#) A fixed camera has been used.

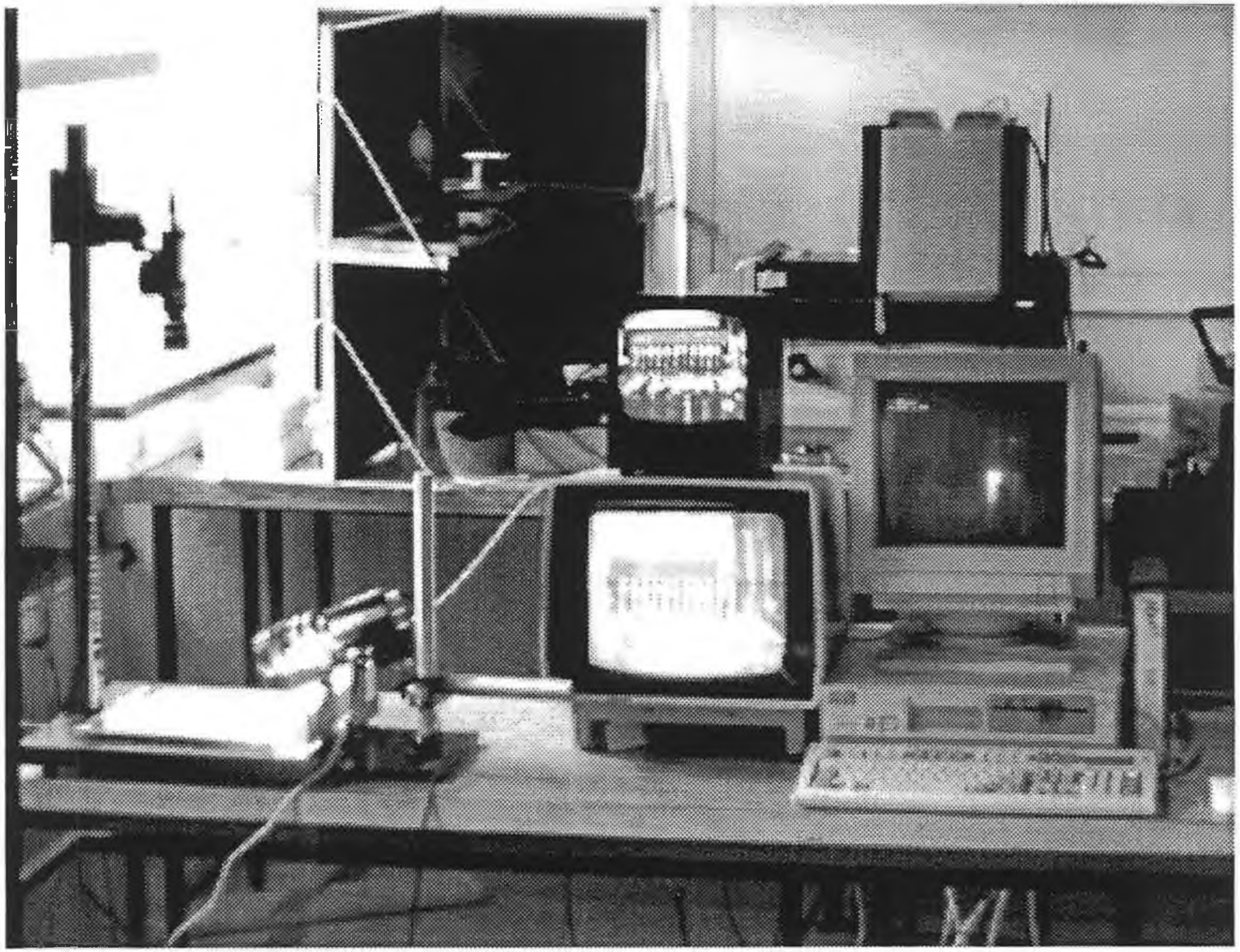


Fig. 6.1 The Inspection System.

Lighting is also an important part of the inspection process. Lighting from above can be used for illumination of the whole PCB (*for the location of ICs*). The most important point is the illumination of leads. That is implemented by a fluorescent lamp as illustrated in chapter 4.

6.5 Inspection Procedure

After locating ICs on the PCB, the sequence of inspecting IC leads should be known, the camera is then moved to an appropriate location and the inspecting process is carried out. Inspection procedure includes locating IC leads in the image, and detecting several soldering defects sequentially.

6.5.1 leads Locating:

A flow chart for lead recognition is shown in Fig. 6.2. The procedure is divided into three stages. In the first stage the program computes intensity values for each line in the image. Preliminary experiments indicated that the brightest point of the picture is always found in the leads and solder joints even with the presence of bright background. In the second stage the variance of grey level values of each line computed. In this stage it is found that the variance of the lines which contains leads is larger than the variance of other lines. For example, Fig. 6.3 shows an image of surface mounted IC with 7 leads, and Fig. 6.4 shows a diagram of the gray level variance of

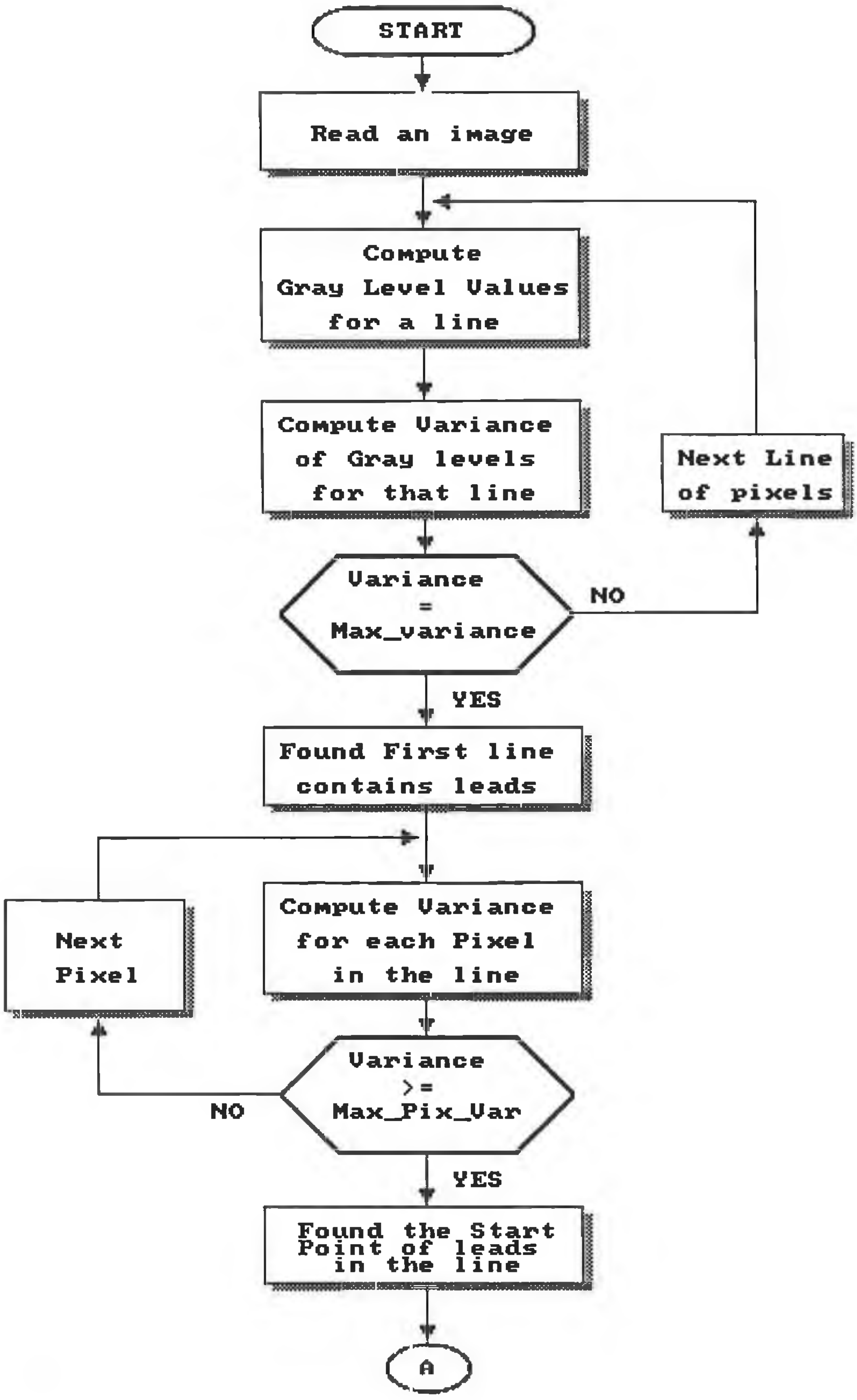

START
Read an image
Compute Gray Level Values for a line
Compute Variance of Gray levels for that line
Variance = Max_variance
NO
Next Line of pixels
YES
Found First line contains leads
Compute Variance for each Pixel in the line
Next Pixel
Variance >= Max_Pix_Var
NO
YES
Found the Start Point of leads in the line
A

Fig. 6.2 Leads locating flow chart.

the lines for the previous image, the largest value refers to the line containing leads (*which is line NO. 247 in this image*). This has led to leads being located in the next stage. In the third stage the variance for each pixel in the line containing leads is computed, as the variance of the lead pixels differs than that in the background pixels. Fig. 6.5 illustrates the gray level variance of the line containing leads in the image shown in Fig. 6.3. By using a threshold, leads can be isolated from the background as shown in Fig. 6.6. In this stage leads are located, and the number of leads is known.

6.5.2 Solder Bridges Detection

A flow chart for solder bridge detection is shown in Fig. 6.7. In the first stage the program takes the gap between two leads and examines the intensity values along the central pixel column for a number of lines equal to the last 2/5 of the lead length. In the second stage it compares intensity values to a threshold and decides if there is any solder bridges. For example in the image shown in Fig. 6.3, leads are located in from line No. 247 and the first gap starts at column No. 187. Lead length can be estimated by using information from data sheets, as:

lead length/(lead width + gap width) = constant.

This constant equals to 2.4 in this case, thus lead length will be 40 for this image. The program examines the intensity values along the central pixel's column of the gap for the last 16 lines of lead's length for each lead.

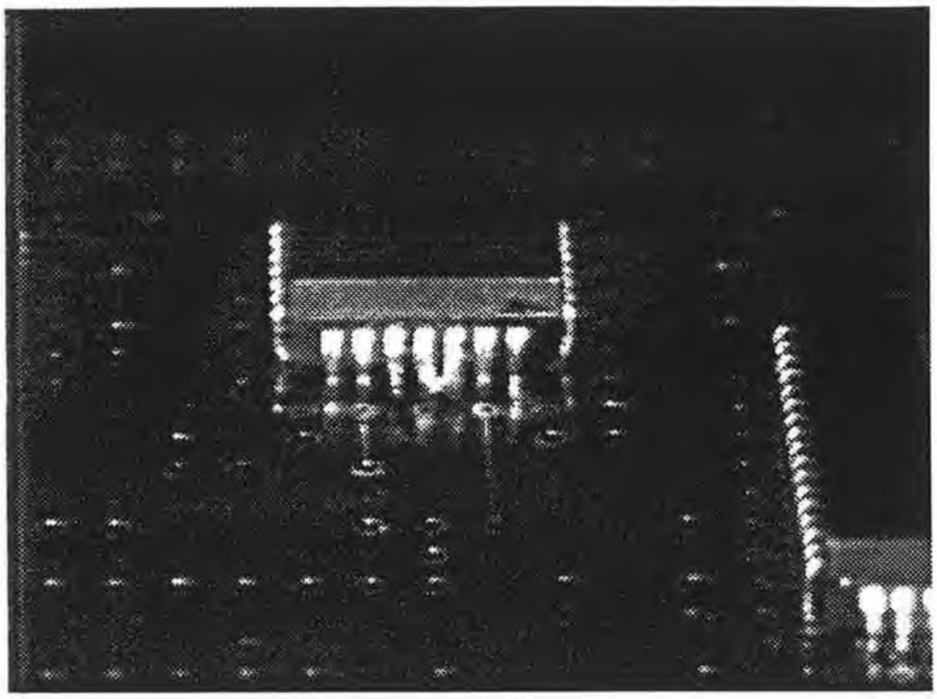


Fig. 6.3 A 7 Leads/side SMIC with solder bridge.

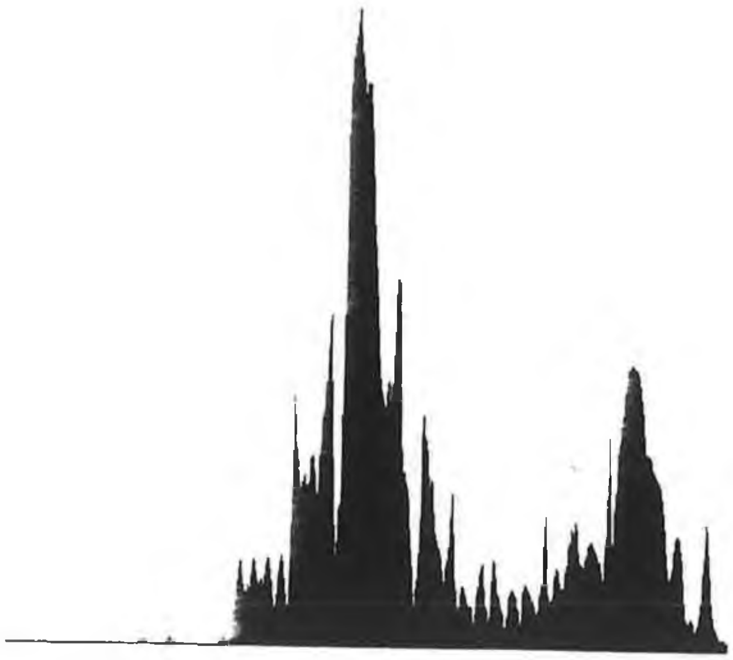


Fig. 6.4 Gray level variance chart of the image shown in Fig. 6.3.

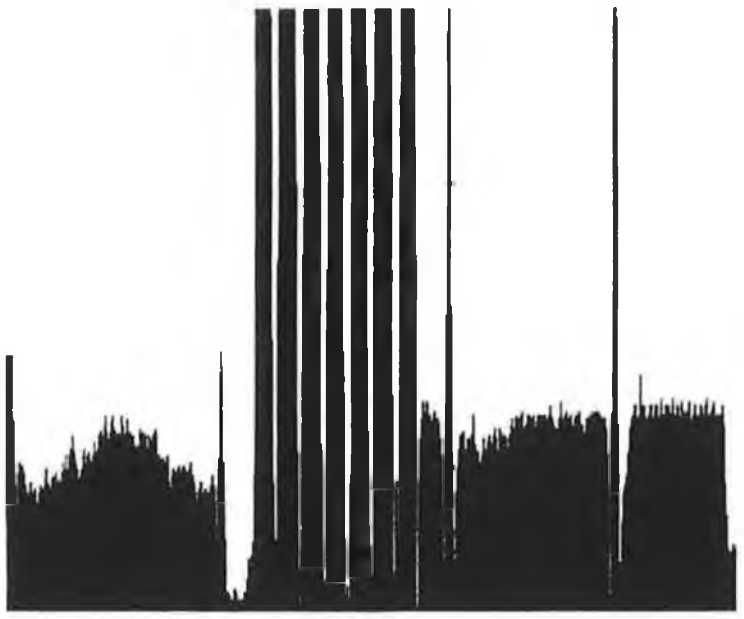


Fig. 6.5 Gray level variance chart of the line containing leads for the image shown in Fig. 6.3.

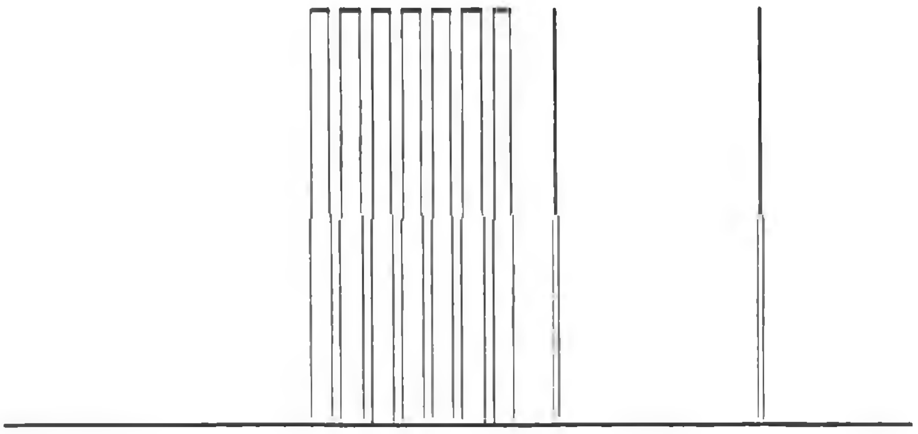


Fig. 6.6 The detected leads chart (for illustration only).

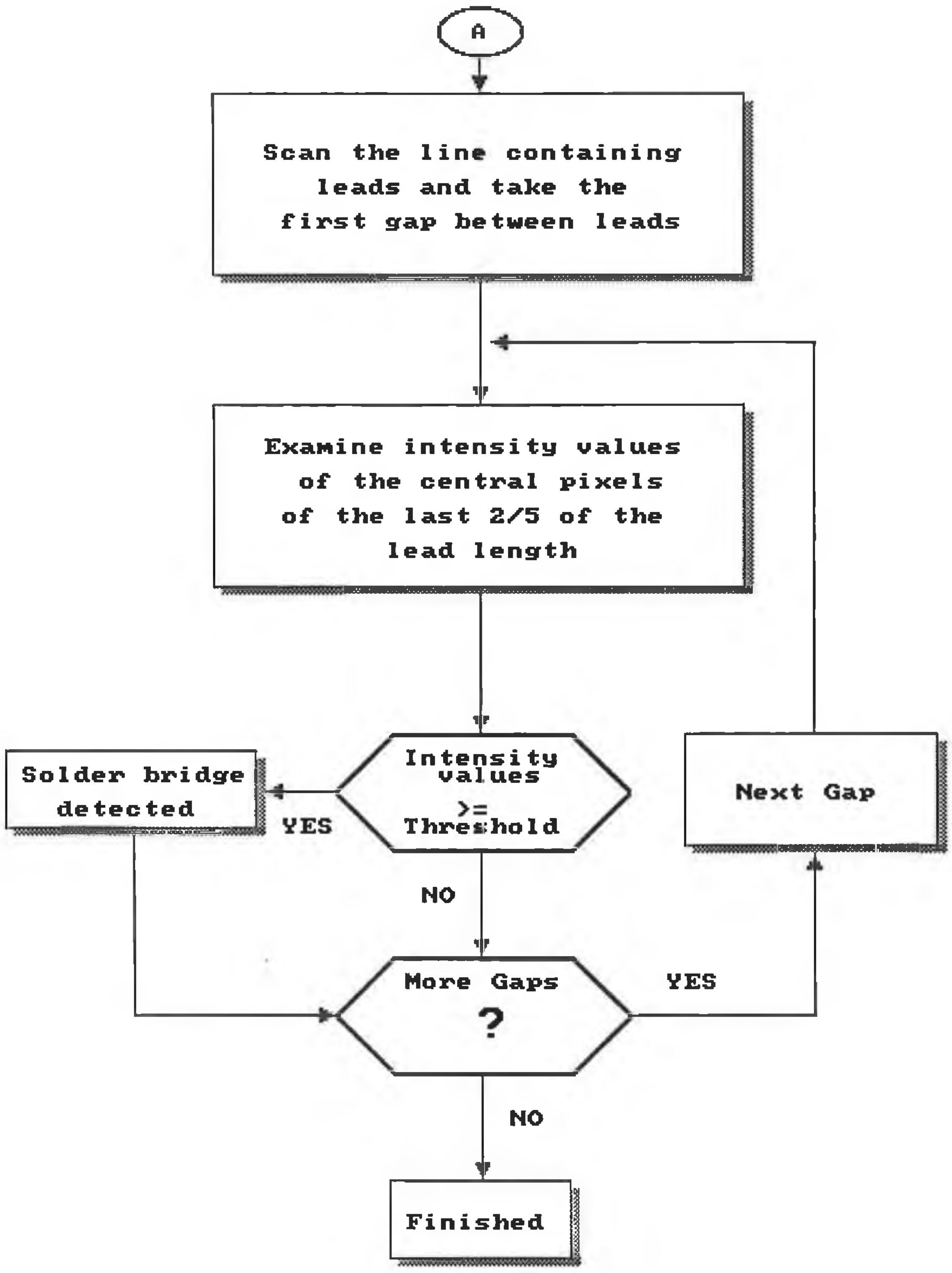

A
Scan the line containing leads and take the first gap between leads
Examine intensity values of the central pixels of the last 2/5 of the lead length
Intensity values >= Threshold
YES
Solder bridge detected
NO
More Gaps ?
YES
Next Gap
NO
Finished

Fig. 6.7 Solder bridges detection flow chart.

Comparing these values to a threshold will indicate if there are any bridges. Fig 6.8 shows the examined areas.

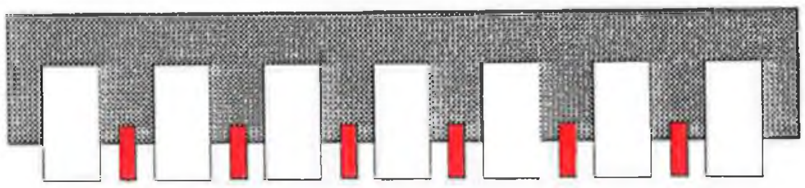


Fig. 6.8 Illustration of solder bridges detection.

6.5.3 Excessive Soldering detection:

Fig. 6.9 shows the flow chart for excessive soldering detection. In the first stage the program takes the nearest pixel around each lead and examines the intensity value of both pixel's column for a number of lines equal to the last 2/5 of the lead length. In the second stage it compares intensity values to a threshold and decides if there is excessive solder on the leads.

In the image shown in Fig. 6.10 for example, leads are located on line NO. 169, and starts at column No. 215, the lead length estimated to be 46 pixels. So the program compares intensity values around leads to a threshold and decides if there is any excessive solder. The examined area is shown in Fig. 6.11.

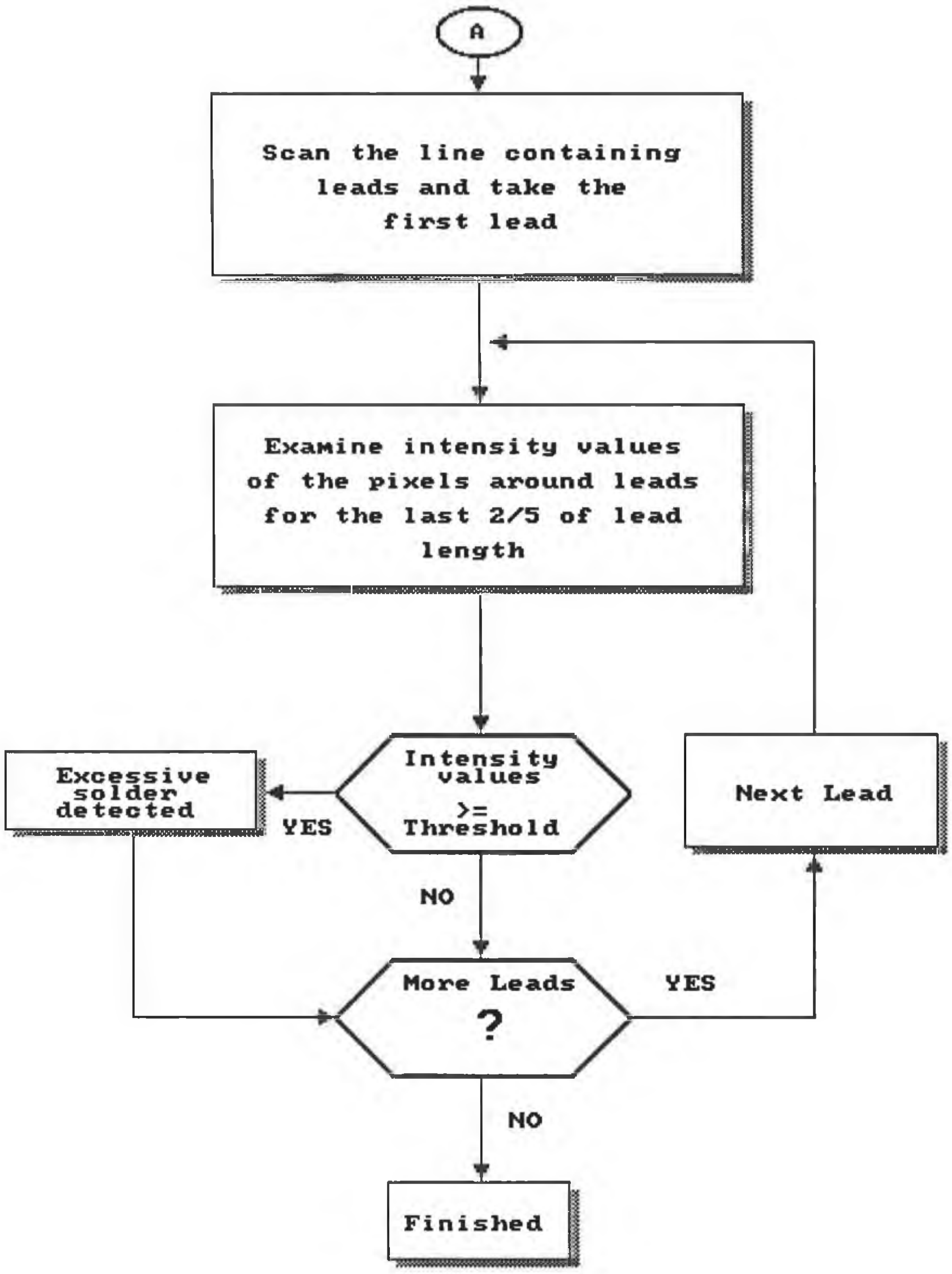

A
Scan the line containing leads and take the first lead
Examine intensity values of the pixels around leads for the last 2/5 of lead length
Intensity values >= Threshold
YES
Excessive solder detected
NO
More Leads ?
YES
Next Lead
NO
Finished

Fig. 6.9 Excessive soldering detection flow chart.

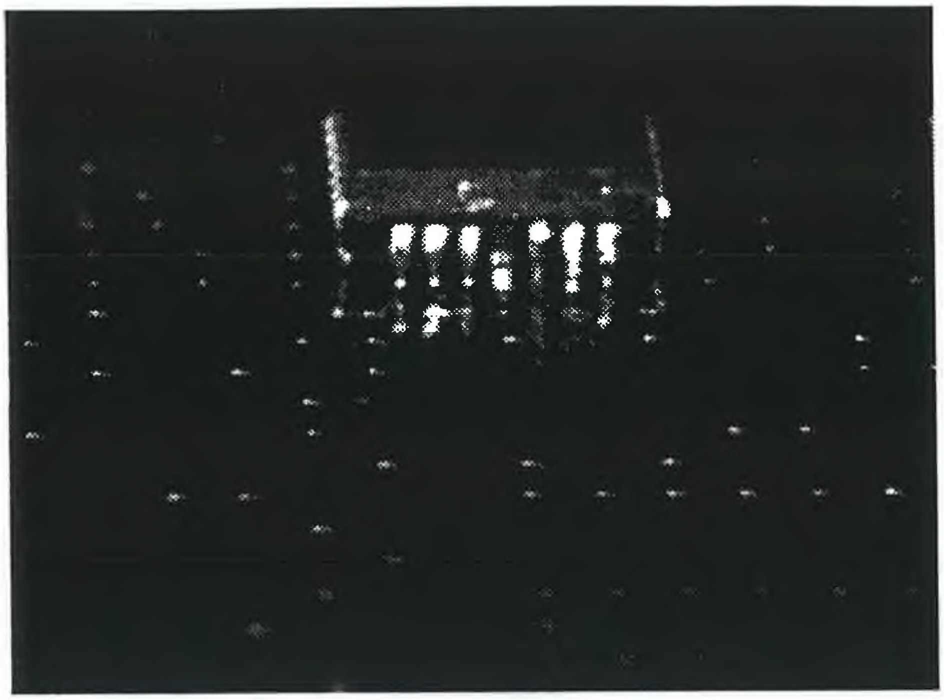


Fig. 6.10 An image with excess solder on lead No. 4.

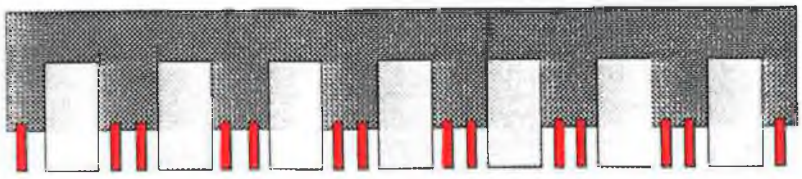


Fig. 6.11 Illustration of excessive solder detection

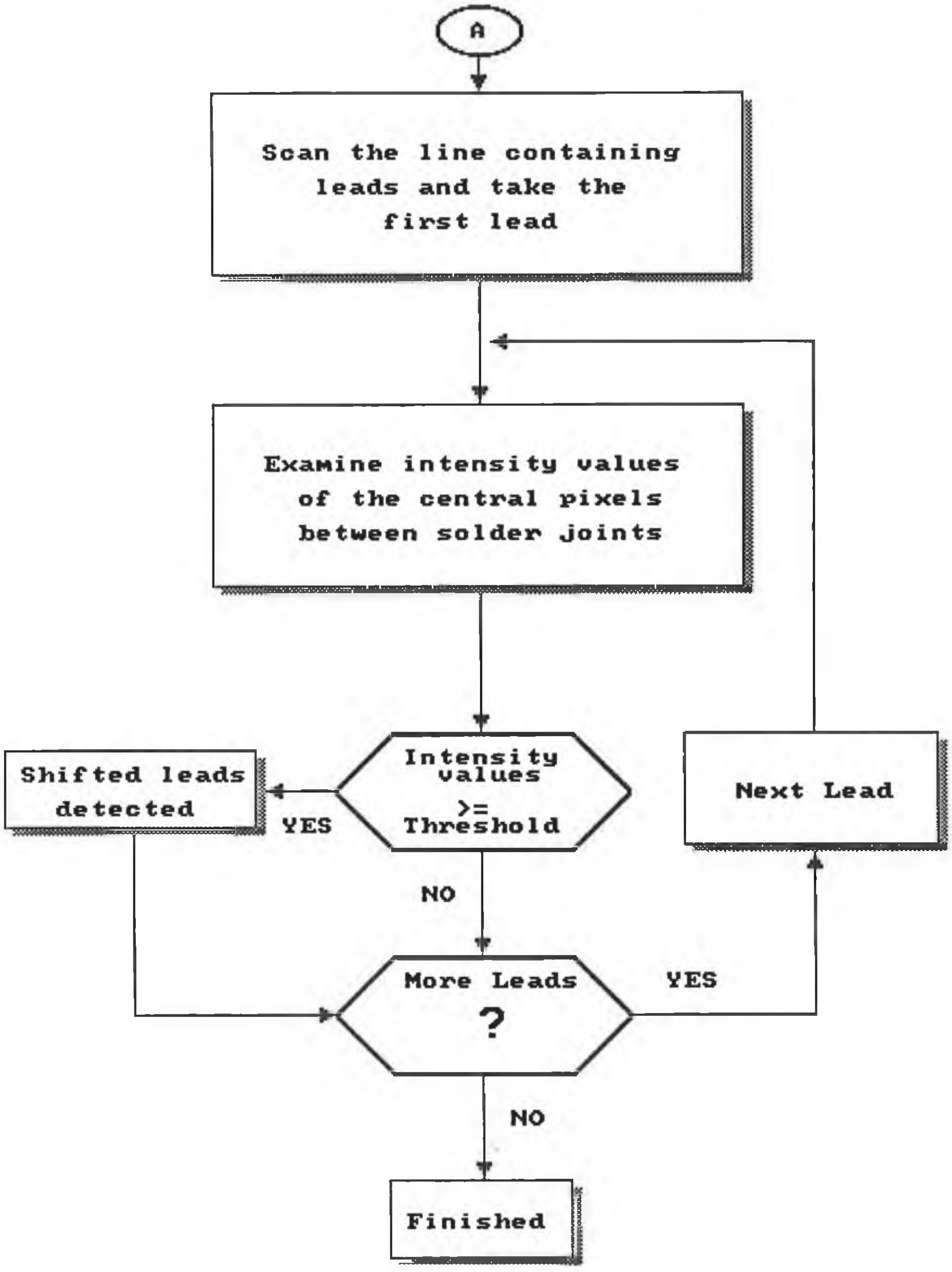

A
Scan the line containing leads and take the first lead
Examine intensity values of the central pixels between solder joints
Intensity values >= Threshold
YES
Shifted leads detected
NO
More Leads ?
YES
Next Lead
NO
Finished

Fig. 6.12 Shifted leads detection flow chart.

6.5.4 Shifted leads Detection

A flow chart for this task is shown in Fig. 6.12. In the first stage the program takes the side pixels of each lead along the last 8/10 of the lead length and examines the intensity values of these pixels. In the second stage and by comparing intensity values to a threshold it indicates if there is a shift.

Fig. 6.14 illustrates the detection areas between leads for the image shown in fig. 6.13. Here leads are located from line No. 320, and first gap starts at column No. 137, the program looks for the central pixels between solder joints (pads) and compares its intensity value to a threshold, and indicates if there is any shifted lead.

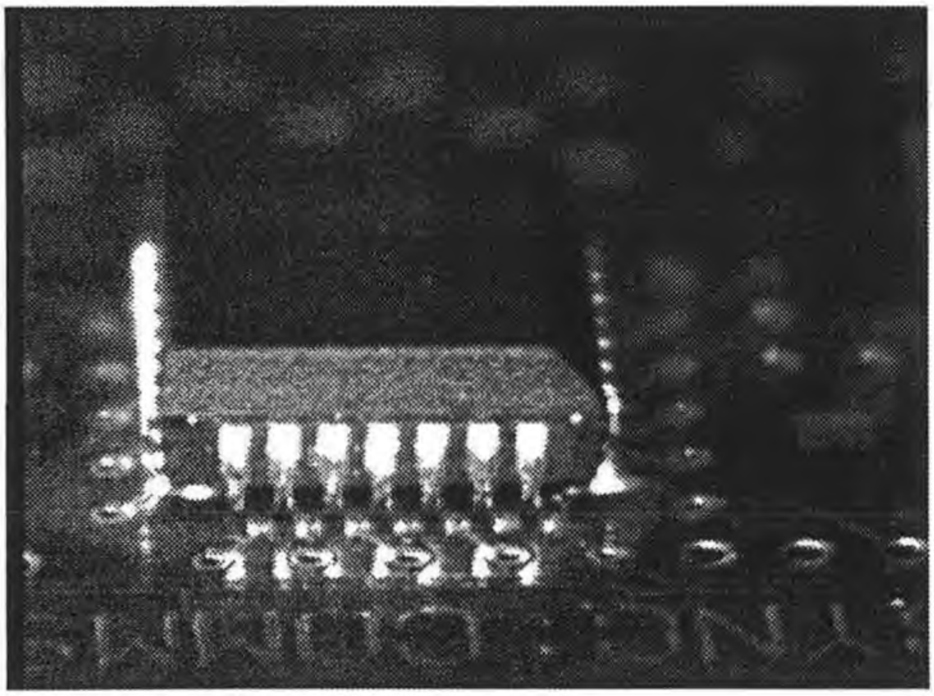


Fig. 6.13 An image of IC with shifted leads.

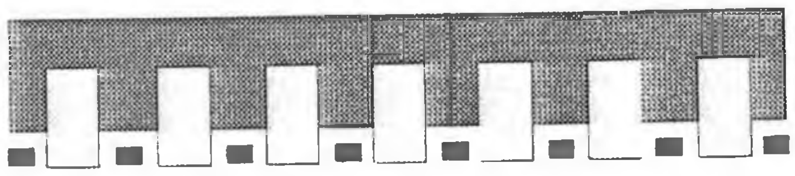


Fig. 6.14 Illustration of Shifted leads detection.

6.5.5 Insufficient Soldering Detection

A flow chart for this task is shown in Fig. 6.15. In the first stage the program takes the central pixel of each lead and examines intensity values of the last 9/10 of the lead length. In the second stage it compares these values to a threshold and then decides if leads are ok or have insufficient solder.

For example, in the image shown in Fig. 6.16 where leads are located from line No. 198, and first lead starts at column No. 100, the program examines intensity values of the central pixels for the last 5 lines, and gives a decision after comparing these values to the threshold. The examined area is shown in fig. 6.17.

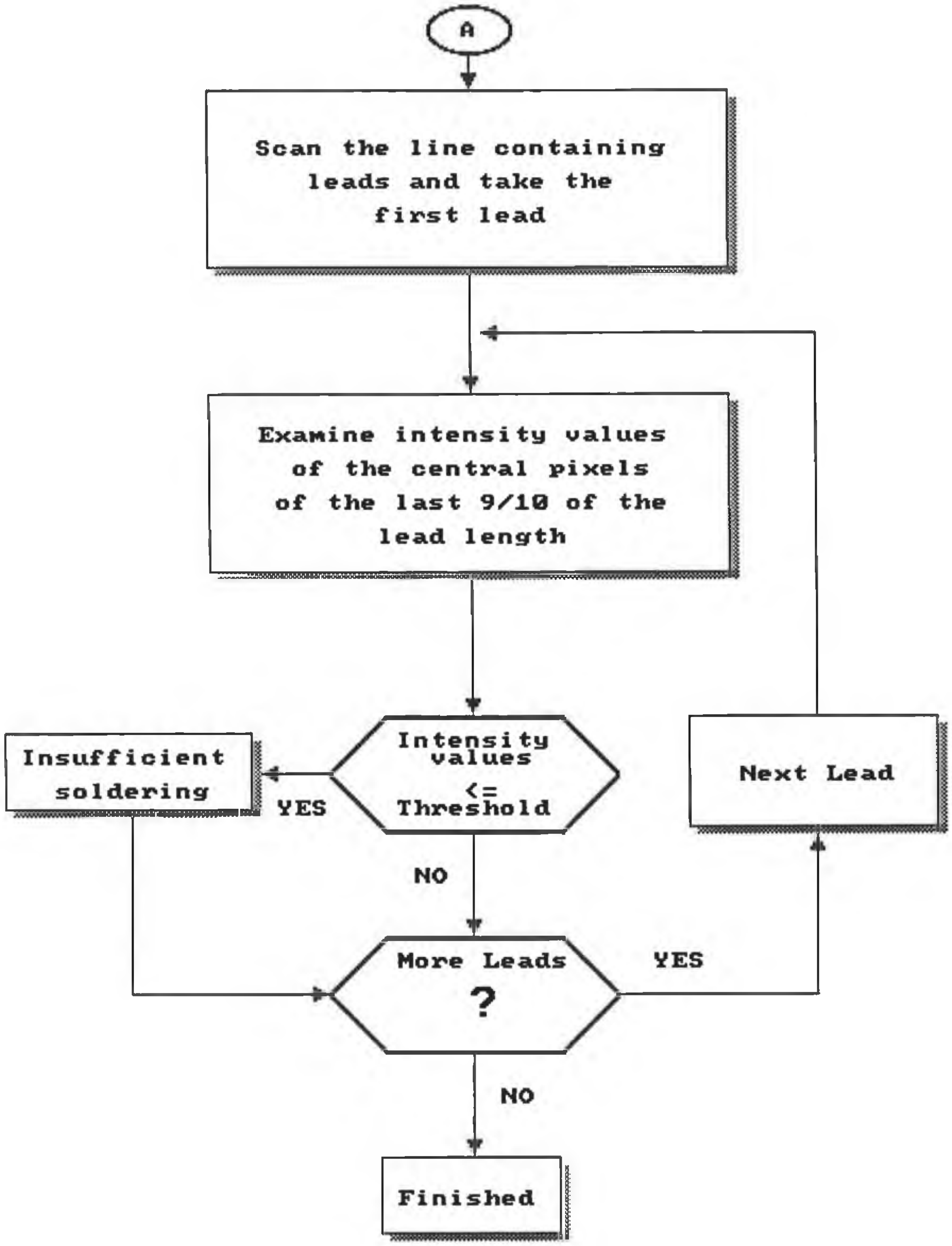

A
Scan the line containing leads and take the first lead
Examine intensity values of the central pixels of the last 9/10 of the lead length
Intensity values <= Threshold
YES
Insufficient soldering
NO
Next Lead
More Leads ?
YES
NO
Finished

Fig. 6.15 Insufficient soldering detection flow chart.

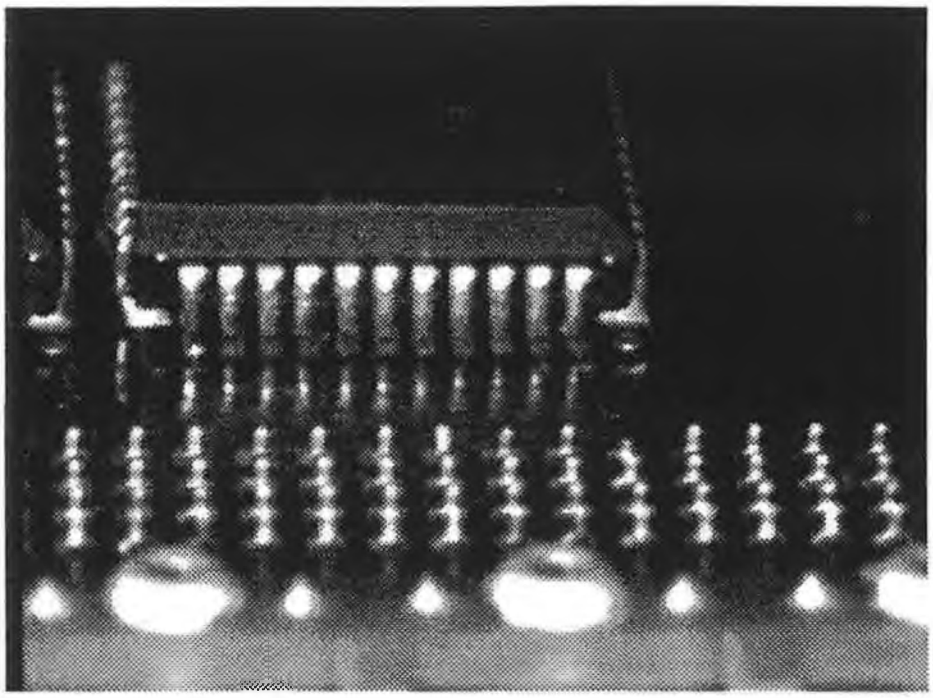


Fig. 6.16 An image with bad leads.

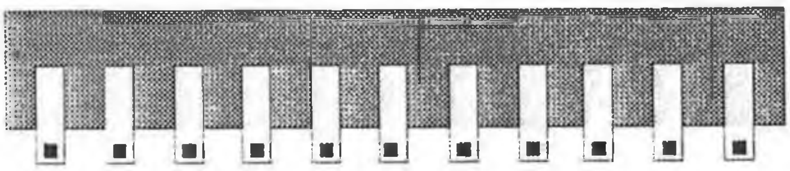


Fig. 6.17 Diagram shows areas examined in for the detection of insufficient solder joints.

6.6 Experimental results

The integrity of Surface Mount Integrated Circuits on a PCB is demonstrated with the results from inspection of some PCBs manufactured in DEC.

A number of PCBs containing examples of Integrated Circuits with solder joints in the categories of good, solder bridges, excess solder, insufficient solder and misplaced leads were examined.

For each type of defect, accuracy and repeatability test were carried out, which consisted of processing 4 PCBs containing in total 5 ICs with solder bridges, 4 ICs with insufficient solder, 5 ICs with misplaced leads, 3 ICs with excess solder, and more than 7 ICs with good leads. These defective ICs were tested 25 times, and comparing the results obtained for each IC.

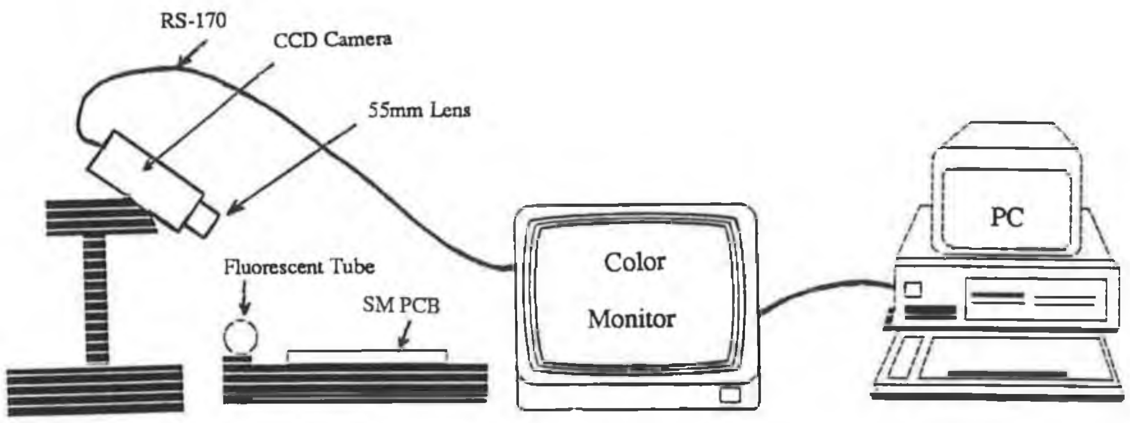


Fig. 6.18 Inspection System Hardware.

As shown in Fig. 6.18, a test rig was setup to test the inspection software. The camera was about 6 inches above the PCB and at about 30 degrees from the horizon. The light source, a 10 inches long fluorescent tube, was placed at about 1 inch above the board and along side the board. The PCB was tested at different angles: horizontal, at 10 degrees rotation around its vertical axis and at 30 degrees rotation around its horizontal axis.

The first test was based on aligning the PCB horizontally as shown in Fig. 6.19 and the software was run 25 times.

In table 6.2 the actual defect class is shown by row with the identification reported by the inspection system shown by column. For example, reading across the third row the percentage of solder bridges correctly detected was 25; the number of solder bridges classified as having excess solder was 24 and so on for other defects.

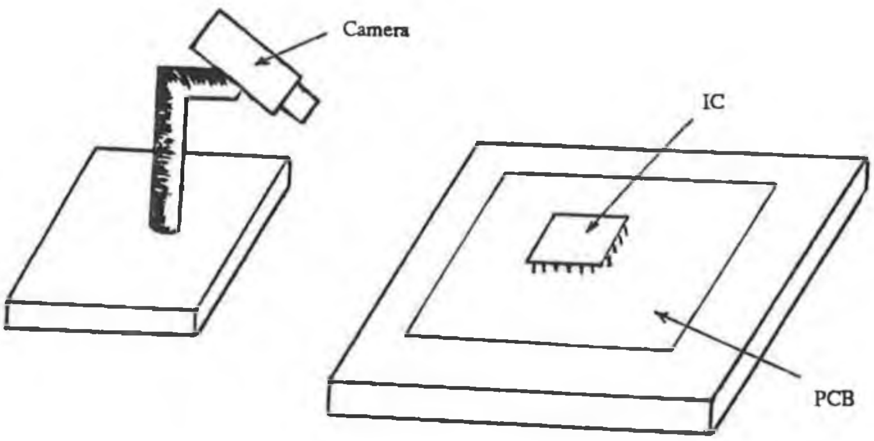


Fig. 6.19 Inspection with the PCB aligned horizontally.

The solder bridge here was classified as excess solder because the bridge has caused by the connection of the excess solder on two neighbouring leads. The solder bridge also classified as shifted solder because the bridge was between the two leads near the solder joints. The shifted leads classified as insufficient because the leads were shifted over the pads so that no correspondence existed between leads and pads.

Horizontal Orientation for the PCB					
Identified Class					
Joint class	Good	Insufficient	Bridges	Excess	Shifts
Good	25	0	0	0	0
Bad	0	25	0	0	0
Bridges	0	0	25	24	1
Excess	0	0	0	25	0
Misplaced	0	1	0	0	25
False Alarms: 1					

Table 6.2 Results for horizontal orientation of the PCB, Percentage of defects identified and performance for 25 times repeated test.

Another test involved aligning the PCB rotated about 10 degrees around its vertical axis as shown in Fig. 6.20 and the software has been run also 25 times.

In the same way in table 6.3, for example, reading across the first row the number of good joints correctly detected was 24; whereas the number of good joints classified as insufficient solder (*bad leads*) was about 10 and so on for other defects. The good joint here reported as insufficient solder because the rotation angle has moved the begining line up or down for a number of lines (*depending on the rotation angle orientation*) which caused changing the scanning area. Fig. 6.21 illustrates this point. For the same reason the shifted leads reported as insufficient solder.

Orientation = 10 Degrees around vertical axis					
	Identified Class				
Joint class	Good	Insufficient	Bridges	Excess	Shifts
Good	24	10	0	0	0
Bad	0	20	0	0	0
Bridges	0	0	25	0	0
Excess	0	0	0	20	0
Misplaced	0	7	0	0	20
False Alarms: up to 10					

Table 6.3 Results for rotated position of the PCB for about 10 degrees, Number of defects identified and performance.

The other test involved aligning the PCB rotated about 30 degrees around its horizontal axis as shown in Fig. 6.22 and the software has been run also 25 times.

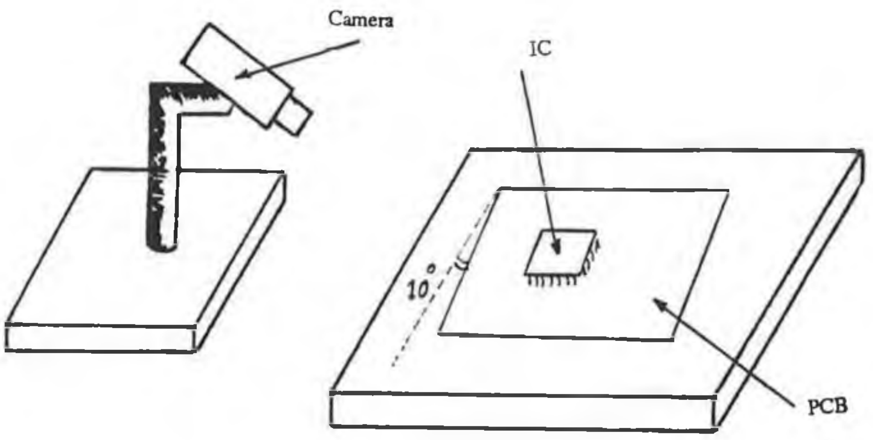


Fig. 6.20 Inspection with the PCB rotated 10 degrees around its vertical axis.

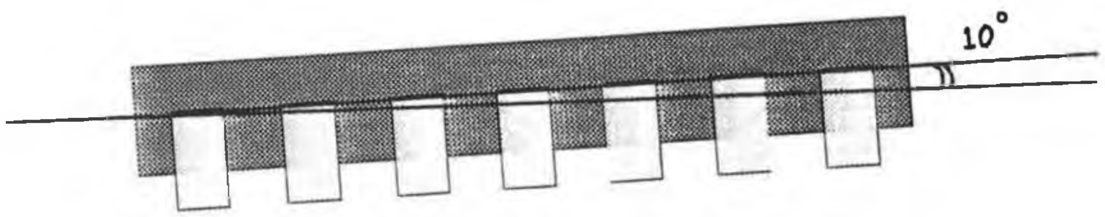


Fig. 6.21 The appearance of the IC leads with the PCB rotated 10 degrees around its vertical axis.

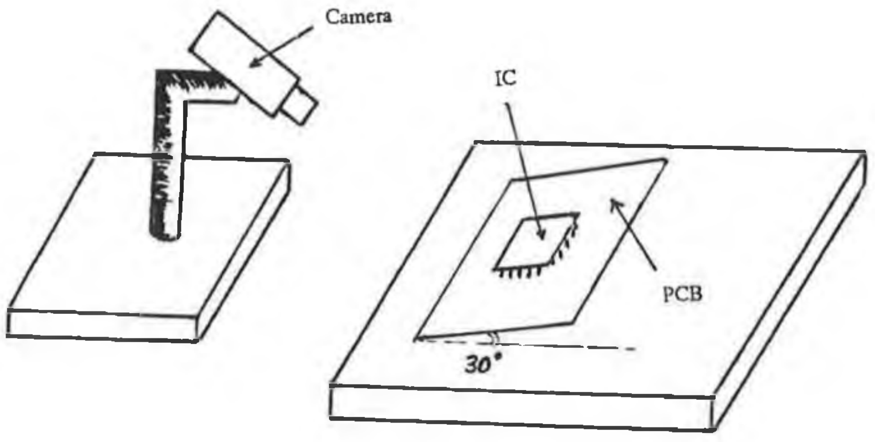


Fig. 6.22 Inspection with the PCB rotated 30 degrees around its horizontal axis.

Similarly in table 6.4, for example, reading across the third row the number of bridges correctly detected was 25; whereas the number of bridges classified as insufficient was about 11 and so on for other defects. The good joint, as well as the other classes, reported as insufficient solder because the rotation angle has reduced the estimated lead length for a number of lines which caused changing the scanning area. Fig. 6.23 illustrates this point. For the same reason the shifted leads reported as bridges.

Orientation = 30 Degrees about Horizontal Axis					
Identified Class					
Joint class	Good	Insufficient	Bridges	Excess	Shifts
Good	0	10	0	0	0
Bad	0	0	0	0	0
Bridges	0	11	25	23	10
Excess	0	0	0	24	0
Misplaced	0	18	14	0	20
False Alarms: up to 18					

Table 6.4 Results for vertical rotation of the PCB Number of defects identified and performance.

6.7 Conclusion

The number of features to be detected affects the complexity of software but is not in itself prohibitive. Another thing is, how much time is available for each inspection ?. A certain exposure time is necessary to obtain the image, depending on the level of lighting. Time is also needed to interpret the image, and this may well be the limiting factor. Processing time is one of the things that is being cut dramatically with the introduction of parallel processing and the development of special hardware which can handle the time consuming computations and could also help to speed up the system . But there is a limit to the throughput rate which the vision software can handle, and the more complicated the analytical work needed the more time it will require.

A number of PCBs containing examples of Integrated Circuits with solder joints in the categories of good, solder bridges, excess solder, insufficient solder and shifted leads were examined.

The inspection system was able to cope with different types of chips (*28, 44, 68, and 84 leads chips*), examining critical areas of the IC, if these are OK, that can eliminate the possibility of certain defects. Also with this approach it is not necessary to take all of the lead, but only the critical areas of the lead or the gap (*such as the central pixels of the lead and the gap between each two leads*). As can be seen from the tables 6.2, 6.3, 6.4 the PCB should be horizontally oriented to get accurate results. However the system works well for small rotations (*5 degrees horizontally and about 15 degrees vertically*). The rotation of the PCB changes the scanning areas which may affect the decisions given by the system.

The inspection process was carried out using fixed camera and by changing the video input gain between 24 and 27 (*this is equal to the changing of the light intensity*). Feature extraction was based on gray level images and fixed threshold in some stage, where the last eliminated the accuracy and flexibility of the system.

Chapter

7

Conclusion and Further Work

Conclusion an Further Work

7.1 Introduction

As surface mount is being more widely used, there is a requirement for new techniques for inspection. Consequently many techniques and different approaches have been developed for the inspection during different stages of surface mount assembly development.

Having spent considerable time in discussion with manufacturing staff in **Digital Equipment Corporation** (Clonmel) and having observed the manufacturing and inspection systems in operation for surface mount assembly, it is apparent that automated fault detection in assembly is a complex problem. It is clear that the partial detection of faults is possible. However it is unlikely that vision can provide a complete solution. A combination of several techniques may be necessary. In proposing and implementing a solution here it is acknowledged that a definitive answer cannot be given as to it's 'real-time' application. However insofar as the sample boards available allow it, an empirical approach has yielded a working system.

Vision systems are mainly composed of video camera, lighting source and image processor. Lighting is the most critical factor regarding the success or failure of the vision application, as many vision systems base much of their analysis on the light intensity.

In this project one of the approaches for the inspection of surface mount integrated circuits on a printed circuit board has been attempted. This approach is based generally on gray-scale images, and the use of a fixed threshold.

7.2 System Performance

When it comes to performance, the expectations are different. Some may prefer speed over accuracy, or vice versa, and others may want to have both. Basically, vision system performance depends on the hardware configuration, resolution of the input device, and complexity of the inspection area.

For each class of defect accuracy and repeatability tests were carried out, which consisted of processing a number of defective ICs several times (*from the available PCBs*), and comparing the results obtained for each IC.

The inspection time is relatively slow . The inspection functions which consisted of taking the pictures, extracting the features, and testing the defects took about 40 seconds. The algorithm could be run faster with a more powerful computer. Alternatively, parallel processing and the development of special hardware to handle the consuming computations could also help to speed up the system. The overall system inspection accuracy differs according to each type of defect, varying between 100% for solder bridges detection and 86% for insufficient solder detection. It also depends on lighting (*which is the most important factor*), and the orientation of the PCB (*the best results were with video input gain 25 and PCB aligned horizontally*).

7.3 Conclusion and further work

The inspection approach was able to cope with different types of chips (*28, 44, 68, and 84 leads chips*), examine critical areas of the image, check if these were OK, then eliminate the possibility of certain defects. Also with this approach it is not necessary to examine all of the lead, but only the critical areas of the lead or the gap (*such as the central pixels of the lead and the gap between each two leads are examined*).

The inspection process was carried out using a fixed camera. Feature extraction based on using a fixed threshold in the second stage of locating leads, which eliminated the accuracy and flexibility of the system.

Further work would involve evaluation of the system in an actual production environment. To increase the flexibility of the inspection system, the configuration described in chapter 5 should be implemented (two cameras and robot arm). Threshold may be calculated automatically or by histograming. The solution for PCB rotation could be to rotate the image in the opposite direction to compensate for the changes in scanning areas. Most importantly, a learning process should be developed to make use of experience and to allow for a growth in reliability and confidence in decision making.

References

References

[1] **A. Novini,** "*Light and Optics Expert Systems for Machine Vision*", Proc. SPIE Int. Opt. Eng., Vol. 1005, pp. 131-136, 1989.

[2] **D. D. Majumder,** "*Computer Vision and Knowledge Based Computer Systems*", J. Instn. Electronics & Telecom. Eng. Vol. 34, No. 3, 1988.

[3] **B. M. Dawson, K. Sheldon, J. F. Asmus, C. McManis,** "*Image Processing*", Byte, Vol. 12, No. 3, MAR. 1987, pp.141-195.

[4] **S. N. Lapidus, A. C. Englander,** "*Understanding How Images are Digitized*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 4/22 -33.

[5] **R.D. Boyle, R.C. Thomas,** "*Computer Vision a first course*", Blackwell S. LTD., 1988.

[6] **Z. Yu-Cheng, Z. seng,** "*A Human Visual Model Based Image Histogram Modification Technique*", IEEE Int. Conf. on Systems Man, Cybernetics, China, 8 - 12 AUG. Vol. 1, pp. 151 - 154, 1988.

[7] **M.C. Fairhurst,** "*Computer vision for robotic systems an introduction*", Prentice-Hall 1988.

[8] **D. H. Ballard, C. M. Brown,** "*Computer vision*" Prentice-Hall, 1982.

[9] **M. James,** "*Pattern recognition*", BSP, 1987.

[10] **K. R. Castleman,** "*Digital image processing*", Prentice Hall, 1979.

[11] **H. Freeman,** "*Machine Vision Algorithms, Architectures, and Systems*", Academic Press 1988.

[12] **J. Bretschi,** "*Automated Inspection Systems for Industry*", IFS Ltd., U.K., 1981.

[13] **R. Nevatia,** "*Machine Perception*", PRENTICE-HALL, 1982.

[14] **R.C. Gonzalez, P. Wintz,** "*DIGITAL IMAGE PROCESSING*", 2nd Ed. Addison-Wesley Publishing CO. 1987.

[15] **J.M. Juran, F.M. Gryna, R.S. Bingham,** "*Quality Control Handbook*", McGraw-Hill, 1978.

[16] **A.K. Jain,** "*Fundamentals of Digital Image Processing*",

Prentice-Hall 1989.

[17] **R.J. Schalkoff,** "*Digital Image Processing and Computer Vision*", John Wiley & Sons, Inc. 1989.

[18] **A. Rosenfield, A.C. Kak,** "*Digital Picture Processing*", 2nd Ed. Academic Press, 1982.

[19] **E.L. Hall,** "*Computer Image Processing & Recognition*" Academic Press 1979.

[20] **M. P. Manson,** "*Introduction to Image Processing*", Proc. Electronic Imaging '88 Conf. 28-31 MAR. 1988, Anaheim, CA USA, pp. 1-6.

[21] **G. J. Rutledge,** "*An Introduction to Gray Scale Vision Machine Vision*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 3/1 -10.

[22] **R.M. Haralick, L.G. Shapiro,** "*Segmentation & Its Place in Machine Vision*", Scanning Microscopy Supplement 2, 1988, pp. 39-54.

[23] **E.R. Davies,** "*Efficient Image Analysis Techniques for Automated Visual Inspection*", Proc. Computer Vision & Image Processing, 29 - 30 NOV. 1988, London, UK, pp. 2 -19.

[24] **H. H. Arsenault, L. leclerc, Y. Sheng,** "*Similarity & Invariance in pattern Recognition*", SPIE, 1988, Vol. 960, pp. 2-17.

[25] **F.P. Murphy,** "*Special Purpose hardware Modules: A Bridge to Increasing the Practicality of machine Vision Applications*", Int. Elec. Imag. Exposition & Conf. 3-6 OCT. 1988, MA. USA. pp. 861-865.

[26] **F. Perchais,** "*Artificial Vision: A Tool to improve Quality & productivity*", Proc. of The 19th Int. Symposium on Automotive Technology & Automation, 24-28 OCT. 1988, Monaco, France, pp. 261-267.

[27] **S.F. Shapiro,** "*Machine Vision Finds a Niche in Automated Inspection*", Computer Design, 1986, Vol. 25 No. 6, pp.46-50.

[28] **R.J. Ahlers,** "*Industrial Applications of Automated Optical Inspection*", Automated Inspection and Measurement: Proceedings of SPIE held in Cambridge, MA, 28-30 OCT. 1986, Bellingham, WA, 730, pp.28-31.

[29] **D.W. Braggins,** "*Machine Vision in Europe*", Proc. of VISION '87 held in Detroit, MI, 8-11 JUN. 1987, Dearborn, MI, pp.1-37 1-46.

[30] **J. Hollingum,** *"Machine Vision; The Eyes of Automation"* IFS Publications Ltd., U.K., Springer-Verlag, 1984.

[31] **G. Winkler,** *"Automated Visual Inspection"*, Proc. of a workshop on Industrial Applications of Image Analysis held in Antwerp, Belgium, 17-19 OCT 1983, D.E.B. Publis., Pijnacker, The Netherlands, 1983, pp.13-30.

[32] **K.J. Heng, J.C. Even,** *"A Vision System for Quality Control"*, Int. Jnl. Appl. Eng. Ed. U.K. Vol. 5, No.3, 1989, pp.403-410.

[33] **E.L. Hall,** *"A PC-Based Machine Vision System"*, Proc. of VISION '86 held in Detroit, MI, 3-5 JUN. 1986, Dearborn, MI, pp.10-1 10-13.

[34] **M. Narayanan,** *"Applications of Vision systems"*, Electron Conventions Management 1987, pp. 1-3.

[35] **A. Pugh,** *"Robot Vision"*, IFS (Publications) Ltd., U.K., Springer-Verlag, 1983.

[36] **P. Villiers,** *"Present Industrial Use of Vision Sensors for Robot Guidance"*, Robot Vision, ed. A. Pugh, IFS (Publications) Ltd., U.K., Spring-Verlag, 1983.

[37] **M.L. Martel,** *"Automated Inspection Roundup"*, Circuits Manufacturing, 1989, Vol. 29, No. 7, pp. 24-28.

[38] **T. Kehoe,** *"Giving Computers the Eye"*, Computerized Manufacturing, JUNE 1988, pp.32-35.

[39] **J. Hollingum,** *"Vision Sensing Provides Automated Calibration"*, Sensors review, Vol.3, No.3, JULY 1983, p.137.

[40] **R. Keeler,** *"Vision Guides Machines in Automated Manufacturing"* Electronic Packaging & Production, 1985, Vol. 25, No. 5, pp. 58-65.

[41] **M.R. Ward,** *"Consight: A Practical Vision-Based Robot Guidance System"*, Proc. 9th International Symposium on Industrial Robots, Society of Manufacturing Engineers and Robot Institute of America, Washington, D.C., MAR. 1979, pp. 195-212.

[42] **D. M. A. Oughton,** *"Automated Placement of Fine Lead Pitch Surface Mount Electronic Device Using Vision Guidance"*, Adv. Manuf. Eng. Vol. 1, OCT. 1988, pp. 6 -10.

[43] **M.J. Carlson,** *"Machine Vision: A Chip Placement*

Monitor", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.168-1-5.

[44] **G.J. Agin,** "*Computer Vision Systems for Industrial Inspection and Assembly*", Computer IEEE, Vol.13, No.5, MAY 1980, pp.11-20.

[45] **W.A. Perkins,** "*A Model-Based Vision System for Industrial parts*", IEEE Trans. Comput., Vol.C-27, No.2, FEB.1978, pp.126-143.

[46] **C.A. Rosen, D. Nitzan,** "*Use of Sensors in programmable Automation*", Computer IEEE, Vol.10, No.12, DEC. 1977, pp.12-23.

[47] **R.G. Wort, J.D.T. Tannock,** "*Automated real time visual inspection for integrated quality control*", Proc. Int. Conf. Robot Vision & Sensory Controls, Swiss, 2-4 FEB. 1982, pp. 271 - 280.

[48] **J. Hollingum,** "*Automating Crack Detection*", Sensors review, Vol.3, No.3, JULY 1983, pp.130-131.

[49] **Valerie C. Bolhouse,** "*Machine vision application for high volume electronics manufacturing*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 5/75 - 97.

[50] **P.F. Leonard, D.J. Svetkoff, R.W. Kelley, D.K. Roher,** "*Machine vision applications in electronic manufacturing*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 5/99 -116.

[51] **C.A. Chang, Y. Lee, M.S. Leonard,** "*A Sampling Approach for Computer Vision Inspection in Automatic Production Systems*", Recent Developments in Production research, OH. USA, 17-20 AUG. 1987, Elseview 1988.

[52] **R.A. Brook,** "*Automatic Inspection in Industry Today* ", Proc. of Industrial Inspection, 19-20 SEP. 1988, SPIE, Vol. 1010, pp. 2-7.

[53] **R.H. Thibadeau,** "*Automated visual inspection as skilled perception*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 5/1 -18.

[54] **M.D. Briggs,** "*Printed Wiring Board Inspection*", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.167-1-14.

[55] **A. Abramovich,** "*Advanced vision technology for printed circuit inspection*", Proc. of VISION '85 conf. Detroit, MI, USA, 25 - 28 MAR. 1985, pp. 5/168 - 179.

[56] **S.N. Lapidus,** "*New Techniques for Industrial Vision Inspection*", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.162-1-12.

[57] **W. E. McIntosh,** "*Automating the inspection of printed circuit boards*", Robotics Today, June 1983, pp. 75-78.

[58] **P. Besl, E. Delp, R. Jain,** "*Automated visual solder joint inspection*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 5/54 -74.

[59] **D. W. Capson, S. Eng,** "*A Tiered-Color Illumination Approach for Machine Inspection of Solder Joints*", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-10, No. 3, pp. 387-393, MAY 1988.

[60] **S. Chen,** "*3-D Machine Vision for Solder Joint Inspection and process Control*", Proc. of The Technical Program Nepcon East 89, 13-15 JUN. 1989, Boston, USA, Publ. Channers Exposition Group 1989, pp. 394-404.

[61] **M. Goharla'ee,** "*A 3-D approach to wire bond inspection*", Hybrid Circuit Technology. (USA), 1988, Vol. 5, No. 6, pp. 10-12.

[62] **J. Mahon,** "*Auto 3-D inspection of solder paste on Surface Mount PCBs*", Proc. IMC-6, Conference of the Irish Manufacturing Committee, Vol. 1, pp.665 -679, 1989.

[63] **M.L. Rhodes,** "*On-Line Color Sensing for Industrial Automation*", Proc. SPIE Industrial Opt. Sensing Vol. 961, pp. 35-47, 1988.

[64] **M.M. trivedi, S.Marapane, C. Chen,** "*Automatic Inspection of Analog & Digital Meters in a Robot Vision System*", Proc. of 4th Conf. on Artificial Intelligence for Space Applications, pp. 233-242, 15-16 NOV. 1988.

[65] **A. E. Kayaalp, R. Jain,** "*Knowledge based automatic on-line wafer (IC) inspection system*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985 pp. 5/117 - 130, 1985.

[66] **J.C.H. Chung, M. Litt, W. Cobb, D. Bard, G. Leiniger,** "*A Vision System for Automated Dopant Wafer Inspection*", Proc. of VISION '89 conf. Chicago 24-27 APR, pp. 5.25-41.

[67] **R.M. Atkinson, J.F. Claridge, S.R. Hattersley,** "*Automatic Inspection of Automobile Displys*", Proc. of 'AIPC-7' The 7th Int. Conf. on Automated

Inspection & Product Control, 26-28 MAR. 1985, Birmingham, UK, pp.139-148.

[68] **A. Hyman, N.D. Burns,** "*Automated Inspection of Knitted Fabrics*", Proc. of 'AIPC-7' The 7th Int. Conf. on Automated Inspection & Product Control, 26-28 MAR. 1985, Birmingham, UK, pp.177-184.

[69] **P. Naylor, C. Loughlin,** "*Print Verification & Label Position Inspection*", Proc. of The 7th Int. Conf. on Automated Inspection & Product Control, 26-28 MAR. 1985, Birmingham, UK, pp.349-356.

[70] **J. W. V. Miller,** "*Illumination Invariant Image processing techniques*", Proc. of VISION '85 conf. Detroit, MI, USA, 25-28 MAR. 1985, pp. 3/24 -31.

[71] **B.G. Batchelor, D.A. Hill, D.C. Hodgson,** "*Automated Visual Inspection*", IFS (Publications) Ltd, UK, 1985.

[72] **D. Hill,** "*The Measurement of light for Inspection*", Proc. of 'AIPC-7' The 7th Int. Conf. on Automated Inspection & Product Control, 26-28 MAR. 1985, Birmingham, UK, pp.219-230.

[73] *EEV CCD Cameras and Components*, EEV Inc. 1987.

[74] **S.H. Mersch,** "*Polarized Lighting for Machine Vision Applications*", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.163-1-14.

[75] **F.A. Sachs,** "*Imaging Devices and Cameras for Automation Processors*", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.171-1-8.

[76] **N. Wittels, J.R. McClellan, K. Cushing, W.Howard, A. Palmer,** "*How to Select Cameras for Machine Vision*", Proc. SPIE Int. Opt. Eng., Vol. 1005, pp. 44-53, 1989.

[77] **M.M. Bahn, J. Harned,** "*Flexible Dimensional Gauging System*", Proc. of Applied Machine Vision Conf. FEB.27-MAR.1, 1984, Chicago, USA, pp.161-1-21.

[78] **A. Prabhala,** "*Digital Output Cameras Enhance Image processing Systems*", Proc. of VISION '89 conf. Chicago 24-27 APR, pp. 10.41-50.

[79] **J. Mullen,** "*How to Use Surface Mount Technology*", Texas Instruments Inc., 1984.

[80] **J. Mahon,** "*Automatic Inspection of Solder Joints on Surface Mount PCBs*", Proc. IMC-7, Conference of the

Irish Manufacturing Committee, pp. 575 - 589, 1990.

[81] **BS 4778,** "*Glossary of Terms Used in Quality Assurance*", BST, 1979.

[82] **M.T. Traband, D.J. Medeiros,** "*CAD-directed Programming of a vision-based Inspection System*", Journal of Manufacturing Systems, 1989, Vol. 8, No. 3, pp. 215-223.

[83] **Computer Applied Techniques (CAPTEC),** "*Mavis User Manual*", DEC. 1990.

Appendix

A

Inspection Software

```
/*
** Inspect.prj
*/

smd_insp.c (mavis.h)
mavis.lib
auto_run.c
badleads.c
bridges.c
balleads.c
shiftleg.c

/* End of inspect.prj */


/*
** Smd_insp.C
*/

#include <io.h>
#include <dos.h>
#include <bios.h>
#include <math.h>
#include "mavis.h"
#include <fcntl.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>

static struct menu_struct inspectmenu;
static pixel pixbuf[0x2FFF];
static char *options[] = {"Clear", "Grab", "Snap", "Save
                           File", "Load File", "Badleads",
                           "Bridges", "Misplaced",
                           "Excessive", "Automatic",
                           "Exit"};
                           /* Setup menu options */

void main(ac,av)
int ac;
char *av[];
{
  long addr;
  static char buf[32768L];
  short base,tbuf[6*512],finished = 1;
  int select, g = 24;

  /* check for optional framestore address parameters */
  addr = (long)(fg_fma)<<6;
  base = fg_reg;
  if (ac>2)
    sscanf(av[2],"%x",&base);
  if (ac>1)
  sscanf(av[1],"%lx",&addr);
printf("mem %lx, reg %x\n",addr,base);

if (fgconfig((short)(addr>>6),base) < 0)
  {
   printf("invalid framestore address specified\n");
   exit(0);
  }

fginit(); /* initialise frame store */
fgclear(0, 0, 1024, 512, 0); /* clear frame store */
fgsync(0); /* Internal source */
fgmode(1); /* setup graphic overlay mode */

setup_menu(&inspectmenu,"-INSPECT-", options, 11, 0, 25,
                25, 1, 2, 8, fg_white, fg_red, fg_white,
                   fg_blue, fg_white, fg_blue, pixbu f);
                                  /* setup menu options */

while (finished) {

fgmask(fg_overl); /* access lower 2 bits of frame memory */
select = run_menu(&inspectmenu);   /* run inspect menu */
clear_menu(&inspectmenu);

switch (select)
      {
       case -1: /* ESC option */
          select = -1;
          finished = 0;
          break;

       case 0:
          fgmask(fg_image); /* access image data only */
          fgclear(0, 0, 1024, 512, 0);
                                  /* clear frame store */
          finished = 1;
          break;

       case 1:
          fgmask(fg_image); /* access image data only */
          fgvmask(fg_image); /* access the 8 bit image plane */
          fgzoom(0, 0, 0); /* position & zoom factor */
          fggrab(0); /* grab an image from camera 0 */
          fggain(g); /* set the gain of the framestore's
video input signal */
          finished = 1;
          break;

       case 2:
          fgmask(fg_image); /* access image data only */
          fgvmask(fg_image); /* access the 8 bit image plane */
          fgzoom(0, 0, 0); /* position & zoom factor */
          fgsnap(0); /* snap an image from camera 0 */
```

```
        fgwait(2); /* wait for next frame */
        finished = 1;
        break;

    case 3:
        fgmask(fg_image); /* access image data only */
        fgsaveb512("smd.ima", 0, 0, buf);
        /* 512x512 right side of image is saved in smd.ima */
        finished = 1;
        break;

    case 4:
        fgmask(fg_image); /* access image data only */
        fgloadb512("smd.ima", 0, 0, buf); /* 512x512 right side of
        image is loaded from smd.ima */
        finished = 1;
        break;

    case 5:
        fgclf(0); /* clear framestore */
        fgmask(fg_image); /* access image data only */
        fgclear(0, 0, 1024, 512, 0); /* clear frame store */
        fgvmask(fg_image); /* access the 8 bit image plane */
        fgzoom(0, 0, 0); /* position & zoom factor */
        fggrab(0); /* grab an image from camera 0 */
        fggain(g); /* set the gain of the framestore's  video input signal */
        fgmask(fg_image); /* access image data only */
        fgvmask(fg_image); /* access the 8 bit image plane */
        fgzoom(0, 0, 0); /* position & zoom factor */
        fgsnap(0); /* snap an image from camera 0 */
        fgwait(2); /* wait for next frame */
        fgmask(fg_image); /* access image data only */
        fgsaveb512("smd.ima", 0, 0, buf);
        /* 512x512 right side of image is saved in smd.ima */
        badleads(); /* call bad leads function */

        finished = 1;
        break;


    case 6:
        fgclf(0); /* clear framestore */
        fgmask(fg_image); /* access image data only */
        fgclear(0, 0, 1024, 512, 0); /* clear frame store */
        fgvmask(fg_image); /* access the 8 bit image plane */
        fgzoom(0, 0, 0); /* position & zoom factor */
        fggrab(0); /* grab an image from camera 0 */
        fggain(g); /* set the gain of the framestore's  video input signal */
        fgmask(fg_image); /* access image data only */
        fgvmask(fg_image); /* access the 8 bit image plane */
        fgzoom(0, 0, 0); /* position & zoom factor */
        fgsnap(0); /* snap an image from camera 0 */
        fgwait(2); /* wait for next frame */
        fgmask(fg_image); /* access image data only */
```

```
            fgsaveb512("smd.ima", 0, 0, buf);
            /* 512x512 right side of image is saved in smd.ima */
            bridges(); /* call bridges function */

            finished = 1;
            break;

        case 7:
            fgclf(0); /* clear framestore */
            fgmask(fg_image); /* access image data only */
            fgclear(0, 0, 1024, 512, 0); /* clear frame store */
            fgvmask(fg_image); /* access the 8 bit image plane */
            fgzoom(0, 0, 0); /* position & zoom factor */
            fggrab(0); /* grab an image from camera 0 */
            fggain(g); /* set the gain of the framestore's  video input signal */
            fgmask(fg_image); /* access image data only */
            fgvmask(fg_image); /* access the 8 bit image plane */
            fgzoom(0, 0, 0); /* position & zoom factor */
            fgsnap(0); /* snap an image from camera 0 */
            fgwait(2); /* wait for next frame */
            fgmask(fg_image); /* access image data only */
            fgsaveb512("smd.ima", 0, 0, buf);
            /* 512x512 right side of image is saved in smd.ima */
            balleads(); /* call balleads function */

            finished=1;
            break;

        case 8:
            fgclf(0); /* clear framestore */
            fgmask(fg_image); /* access image data only */
            fgclear(0, 0, 1024, 512, 0); /* clear frame store */
            fgvmask(fg_image); /* access the 8 bit image plane */
            fgzoom(0, 0, 0); /* position & zoom factor */
            fggrab(0); /* grab an image from camera 0 */
            fggain(g); /* set the gain of the framestore's  video input signal */
            fgmask(fg_image); /* access image data only */
            fgvmask(fg_image); /* access the 8 bit image plane */
            fgzoom(0, 0, 0); /* position & zoom factor */
            fgsnap(0); /* snap an image from camera 0 */
            fgwait(2); /* wait for next frame */
            fgmask(fg_image); /* access image data only */
            fgsaveb512("smd.ima", 0, 0, buf);
            /* 512x512 right side of image is saved in smd.ima */
            shiftleg(); /* call shift leads function */

            finished = 1;
            break;

        case 9:
            fgclf(0); /* clear framestore */
            fgmask(fg_image); /* access image data only */
            fgclear(0, 0, 1024, 512, 0); /* clear frame store */
            fgvmask(fg_image); /* access the 8 bit image plane */
```

```
                fgzoom(0, 0, 0); /* position & zoom factor */
                fggrab(0); /* grab an image from camera 0 */
                fggain(g); /* set the gain of the framestore's  video input signal */
                fgmask(fg_image); /* access image data only */
                fgvmask(fg_image); /* access the 8 bit image plane */
                fgzoom(0, 0, 0); /* position & zoom factor */
                fgsnap(0); /* snap an image from camera 0 */
                fgwait(2); /* wait for next frame */
                fgmask(fg_image); /* access image data only */
                fgsaveb512("smd.ima", 0, 0, buf);
                /* 512x512 right side of image is saved in smd.ima */
                auto_run(); /* run automatic inspection */
                finished = 1;
                break;

            case 10:
                fgmask(fg_overl); /* access lower 2 bits of frame memory */
                clear_menu(&inspectmenu); /* clear the menu from frame  memory */

                fgmask(fg_image); /* access image data only */
                fgclear(0, 0, 1024, 512, 0); /* clear frame store */
                finished = 0;
                break;
            }
    }
fgexit(); /* shutdown frame store */
}

/* End of smd_insp.c */
/*
** auto_run.c
*/

auto_run()

{

     int i = 1;

     if (i == 1)
        {
         badleads();

         bridges();

         balleads();

         shiftleg();
         i = 0;
        }
}

/* End of auto_run() */
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                    Bad Leads Detection Routine
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

#include <io.h>
#include <dos.h>
#include <bios.h>
#include <math.h>
#include "mavis.h"
#include <fcntl.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>

#define BUF_SIZE        512
#define BUF1_SIZE       1
#define NO_LEVELS       256
#define NO_LINS         512
#define NO_COLS         512
#define JMAX            192
#define L_FACTOR        2.4
#define N               512
#define cls             printf("\033[2J\033[;H")

badleads()
{

           int i, j, n, k, line, row, l_start, g_start;
           int start_y, l_width, g_width, Threshold, flag,
               counter, f_h;
           int l_end, g_end, lead_num, first_l_end,
               start_col, end_col;
           int l_scan_field, g_scan_field, l_step,
               start_row, end_row;
           int gray_level, max_var_line;
           int g_center, l_center, l_length, l_numbers,
               g_numbers;
 unsigned long variance, max_var, mean_pix, sum, total,
               var[NO_COLS];
 unsigned long no_pix[NO_LEVELS], variance_of[NO_LINS],
               max_pix_var;
 unsigned long max_no_pix, mean_no_pix, ref[NO_COLS],
               offset, value;
 unsigned char buf[ BUF_SIZE ], buf1[ BUF1_SIZE ];
          char *file_name[20];

 *file_name = "smd.ima";
 if ((f_h = open( *file_name, O_RDONLY | O_BINARY )) < 0 )
    {
     puts("\7\7ERROR file name TRY AGAIN");
     exit(0);
    }
```

```
cls;
puts("\t\t\t\t\working...");
Threshold = 255;
start_y = 0;
max_var = 0;
for (line = start_y; line < NO_LINS; line++)
    variance_of[line] = 0;

for (line = start_y; line < NO_LINS; line++)
    {
     read( f_h, &buf, BUF_SIZE);
     for (i= 0;i < NO_LEVELS; i++)
         no_pix[i] = 0 ;

     for (i = 0; i< NO_COLS; i++)
         no_pix[(int)( buf[i])]++;

     sum = 0;
     for (i = 0; i < NO_COLS; i++)
          sum += buf[i];

     mean_no_pix = sum / N;
     variance = 0;
     for (i = 0; i< NO_COLS; i++)
         variance += abs((buf[i] - mean_no_pix));

     variance_of[line] = variance;
    }

for (line = 0; line < NO_LINS; line++)
    {
     if (max_var < variance_of[line])
        {
         max_var = variance_of[line];
         max_var_line = line;
        }
    }
line = max_var_line; /* first line contains leads */

for (row = line; row < line + 1; row++)
    {
     lseek(f_h, 512L * row, 0);
     read( f_h, &buf, BUF_SIZE);
     for (i= 0;i < NO_LEVELS;i++)
         no_pix[i] = 0;
        /* Intensity values computing */
        for (i = 0; i< NO_COLS; i++)
          {
           buf[i] = (buf[i] >= Threshold) ? = 255 : 0;
           no_pix[(int)( buf[i])]++;
          }

     total = 0; /* Variance Computing */
     for (i = 0; i < NO_COLS; i++)
     total += buf[i];

mean_pix = total / N;
variance = 0;
for (i = 0; i< NO_COLS; i++)
    var[i] = 0;
for (i = 0; i< NO_COLS; i++)
    {
     variance = ((buf[i] - mean_pix)
                * (buf[i] -mean_pix));
                if (variance > 9999)
         {
          variance = 1;
         }
     else variance = 0;
          var[i] = variance;
    }

l_width = 0;
for (i = 0; i< NO_COLS; i++)
    {
     if (var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
        l_start = i; /* start of lead */
     if (var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
        l_end = i; /* end of lead */
     l_width = l_end - l_start + 1;
    }
g_width = 0;
for (i = 0; i< NO_COLS; i++)
    {
     if (var[i] == 0 & var[i-1] == 1
                     & var[i+l_width] == 1)
        g_start = i; /* start of gap */
     if (var[i] == 0 & var[i+1] == 1
                     & var[i-l_width] == 1)
        g_end = i; /* end of gap */
     g_width = g_end -g_start;
    }
l_length = ceil((float)(l_width + g_width)
                             * L_FACTOR);
                             l_step =
                             ceil((float)(l_length
                             * 9 / 10));


l_width = 0;
l_start = 0;
l_end = 0;
for (i = 0; i< NO_COLS; i++)
    {
     if (var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
        l_start = i; /* start of lead */
     if (var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
        l_end = i; /* end of lead */
```

```
                l_width = l_end - l_start + 1;
                if (l_width < 20 & l_width > 4)
                   {
                    l_center = ceil((float)l_width / 2); /* lead
                    width / 2 */
                    start_row = line + l_step+2;
                    /* start line for scaning */
                    end_row = line + l_length-1;
                    /* last row to scan */
                    start_col = l_start + l_center - 1;
                    end_col = l_start + l_center - 1;
                    fgmask(fg_overl);
                    xorrect(start_col, start_row, 2, (end_row -
                                         start_row), fg_red);
                    for (j = start_row; j <= end_row + 2; j++)
                        {
                         for (k = start_col; k <= end_col; k++)
                             {
                              flag = 0 ;
                              offset = (512L * j) + k;
                              lseek(f_h, offset, 0); read( f_h,
                              &buf1, BUF1_SIZE );
                              value = buf1[ 0 ];
                              flag = (value >= 64) ? 1 : 0;
                              if (flag == 0)
                                 printf("Bad Lead Detected\n");
                             }
                        }
                    l_width = 0;
                    l_start = 0;
                    l_end = 0;
                   }
               }
          }
  puts("Finished Bad Leads Detection....");
}
/*  End of Badleads Detection Routine */


/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                 Solder Bridges Detection Routine
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

#include <io.h>
#include <dos.h>
#include <bios.h>
#include <math.h>
#include "mavis.h"
#include <fcntl.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>
```

```
#define BUF_SIZE        512
#define BUF1_SIZE       1
#define NO_LEVELS       256
#define NO_LINS         512
#define NO_COLS         512
#define L_FACTOR        2.4
#define N               512
#define cls             printf("\033[2J\033[;H")

bridges()

{

           int i, j, k, n, row, Threshold, g_step,
               start_row, end_row, flag, f_h;
           int line, l_start, g_start, col_var, index;
           int start_y, l_width, g_width, start_col,
               end_col;
           int l_end, g_end, lead_num, first_l_end, count;
           int l_scan_field, g_scan_field, step, counter;
           int gray_level, max_var_line;
           int g_center, l_center, l_length, l_numbers,
               g_numbers;
 unsigned long variance, var[NO_COLS], l_var[NO_COLS],
               offset;
 unsigned long no_pix[NO_LEVELS], variance_of[NO_LINS];
 unsigned long max_no_pix, mean_no_pix, max_var, mean_pix,
               sum, total, value;
 unsigned char buf[ BUF_SIZE ], buf1[ BUF1_SIZE ] ;
          char *file_name[20];

 *file_name = "smd.ima";
 if((f_h = open( *file_name, O_RDONLY | O_BINARY )) < 0 )
    {
     puts("\7\7ERROR file name TRY AGAIN");
     exit(0);
    }
 puts("\t\t\t\tWorking...");
 Threshold = 255;
 max_var = 0;
 for(line = 0; line < NO_LINS; line++)
     variance_of[line] = 0;

 for(line = 0 ; line < NO_LINS; line++)
    {
     read( f_h, &buf, BUF_SIZE);
     for(i= 0;i < NO_LEVELS; i++)
         no_pix[i] = 0 ;

     for(i = 0; i< NO_COLS; i++)
         no_pix[(int)( buf[i])]++;

     sum = 0;
     for(i = 0; i < NO_COLS; i++)
```

```
            {
             sum += buf[i];
             mean_no_pix = sum / N;
            }

        variance = 0;
        for(i = 0; i< NO_COLS; i++)
            variance += abs((buf[i] - mean_no_pix));

        variance_of[line] = variance;
       }

   for(line = 0; line < NO_LINS; line++)
       {
        if(max_var < variance_of[line])
          {
           max_var = variance_of[line];
           max_var_line = line;
          }
       }
   line = max_var_line; /* first  line contains leads */

   for(row = line; row < line + 1; row++)
       {
        lseek(f_h, 512L * row, 0);
        read( f_h, &buf, BUF_SIZE);

        for(i= 0;i < NO_LEVELS;i++)
            no_pix[i] = 0 ;

        for(i = 0; i< NO_COLS; i++)
           {
            buf[i] = (buf[i] >= Threshold) ? = 255 : 0;
            no_pix[(int)( buf[i])]++;
           }

        total = 0;
        for(i = 0; i < NO_COLS; i++)
            total += buf[i];

        mean_pix = total / N;
        variance = 0;
        for(i = 0; i< NO_COLS; i++)
            var[i] = 0;
        for(i = 0; i< NO_COLS; i++)
           {
            variance = (( buf[i] - mean_pix) * ( buf[i] -
                        mean_pix));
            if(variance > 9999)
              {
               variance = 1;
              }
            else variance = 0;
            var[i] = variance;
   }

l_width = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
        l_start = i; /* start of the lead */
    if(var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
        l_end = i; /* end of the lead */
   }
l_width = l_end - l_start + 1;
g_width = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 0 & var[i-1] == 1
             & var[i+l_width] == 1)
        g_start = i; /* start a gap */
    if(var[i] == 0 & var[i+1] == 1
                   & var[i-l_width] == 1)
        g_end = i; /* end a gap */ g_width = g_end - g_start;
   }
l_length = ceil((float)(l_width + g_width)
                                * L_FACTOR);

g_start = 0;
g_end = 0;
g_width = 0;
k = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 0 & var[i-1] == 1
                   & var[i+l_width] == 1)
        g_start = i; /* start a gap */
    if(var[i] == 0 & var[i+1] == 1
                   & var[i-l_width] == 1)
        g_end = i; /* end a gap */
    g_width = g_end - g_start;
    if(g_width < 10 & g_width > 1)
      {
        g_center = ceil((float)g_width /2);
        /* gap width / 2 */
        g_step = ceil((float)(l_length * 3 / 5));
        start_col = g_start + g_center;
        end_col = g_start + g_center;
        start_row = line + g_step+1; /* start line for scaning */
        end_row = line + l_length+3; /* last row to scan */
        fgmask(fg_overl); xorrect(start_col, start_row,
        2, (end_row - start_row),
        for(j = start_row; j <= end_row; j++)
           {
            for(k = start_col; k <= end_col; k++)
               {
                flag = 0 ;
                offset = (512L * j) + k;
```

```
                        lseek(f_h, offset, 0); read( f_h,
                        &buf1, BUF1_SIZE );
                        value = buf1[ 0 ];
                        flag = ( value >= 64) ? 1 : 0;
                        if(flag == 1)
                        printf("Solder Bridge Detected....\n");
                       }
                   }
               g_width = 0;
               g_start = 0;
               g_end = 0;
              }
           }
        }
  puts("Bridges detection Finished....");
}
/* End of Bridges Detection Routine */


/*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
              Excess Solder Detection Routine
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

#include <io.h>
#include <dos.h>
#include <bios.h>
#include <math.h>
#include "mavis.h"
#include <fcntl.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>


#define BUF_SIZE        512
#define BUF1_SIZE       1
#define NO_LEVELS       256
#define NO_LINS         512
#define NO_COLS         512
#define L_FACTOR        2.4
#define N               512
#define cls             printf("\033[2J\033[;H")

balleads()

{

           int i, j, n, k, line, row, l_start, g_start,
               k_step, f_h; int start_y, l_width, g_width,
           Threshold, flag, counter;
           int l_end, g_end, lead_num, first_l_end,
               start_col, end_col;
           int l_scan_field, g_scan_field, l_step,
                     start_row, end_row;
                 int gray_level, max_var_line;
                 int g_center, l_center, l_length, l_numbers,
                     g_numbers, jump;
unsigned long variance, max_var, mean_pix, sum, total,
                     var[NO_COLS];
unsigned long no_pix[NO_LEVELS], variance_of[NO_LINS],
                     max_pix_var;
unsigned long max_no_pix, mean_no_pix, ref[NO_COLS],
                     offset, value;
unsigned char buf[ BUF_SIZE ], buf1[ BUF1_SIZE ];
              char  *file_name[20];

*file_name = "smd.ima";
if ((f_h = open( *file_name, O_RDONLY | O_BINARY )) < 0 )
   {
    puts("\7\7ERROR file name TRY AGAIN");
    exit(0);
   }
puts("\t\t\t\tWorking...");
Threshold = 255;
start_y = 0;
max_var = 0;
for (line = start_y; line < NO_LINS; line++)
    variance_of[line] = 0;

for (line = start_y; line < NO_LINS; line++)
    {
     read( f_h, &buf, BUF_SIZE);
     for (i= 0;i < NO_LEVELS; i++)
         no_pix[i] = 0 ;

     for (i = 0; i< NO_COLS; i++)
         no_pix[(int)( buf[i])]++;

     sum = 0;
     for (i = 0; i < NO_COLS; i++)
         {
          sum += buf[i];
          mean_no_pix = sum / N;
         }

     variance = 0;
     for (i = 0; i< NO_COLS; i++)
         variance += abs((buf[i] - mean_no_pix));

     variance_of[line] = variance;
    }

for (line = 0; line < NO_LINS; line++)
    {
     if (max_var < variance_of[line])
        {
         max_var = variance_of[line];
```

```
            max_var_line = line;
           }
       }
   line = max_var_line; /* first line contains leads */

    for (row = line; row < line + 1; row++)
        {
         lseek(f_h, 512L * row, 0);
         read( f_h, &buf, BUF_SIZE);

         for(i= 0;i < NO_LEVELS;i++)
            no_pix[i] = 0 ;

         for(i = 0; i< NO_COLS; i++)
            {
             buf[i] = (buf[i] >= Threshold) ? = 255 : 0;
             no_pix[(int)( buf[i])]++;
            }

         total = 0;
         for(i = 0; i < NO_COLS; i++)
            total += buf[i];

         mean_pix = total / N;
         variance = 0;
         for(i = 0; i< NO_COLS; i++)
            var[i] = 0;
         for(i = 0; i< NO_COLS; i++)
            {
             variance = (( buf[i] - mean_pix)
                        * ( buf[i] -mean_pix));
                        if (variance > 9999)
                {
                 variance = 1;
                }
             else variance = 0;
             var[i] = variance;
            }

         l_width = 0;
         for(i = 0; i< NO_COLS; i++)
            {
             if (var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
                l_start = i; /* start of the lead */
             if (var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
                l_end = i; /* end of the lead */
             l_width = l_end - l_start + 1;
            }
         g_width = 0;
         for(i = 0; i< NO_COLS; i++)
            {
             if (var[i] == 0 & var[i-1] == 1 &
                                  var[i+l_width] == 1)
                g_start = i; /* start a gap */
            if (var[i] == 0 & var[i+1] == 1 &
                                  var[i-l_width] == 1)
               g_end = i; /* end a gap */
            g_width = g_end - g_start;
           }
        jump = l_width + g_width;
        l_length = ceil((float) jump * L_FACTOR);
        l_step = ceil((float)(l_length * 3 / 5));

        l_width = 0;
        l_start = 0;
        l_end = 0;
        for(i = 0; i< NO_COLS; i++)
           {
            if(var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
              l_start = i; /* start of the lead */
            if(var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
              l_end = i; /* end of the lead */
            l_width = l_end - l_start + 1;
            if(l_width <20 & l_width > 4)
              {
               start_row = line + l_step +1; /* start line for scaning */
               end_row = line + l_length +1; /* last row to scan */
               start_col = l_start - 2;
               end_col = l_end + 2;
               k_step = l_width + 4;
               fgmask(fg_overl);
               xorrect(start_col, start_row, 1, (end_row -
                                 start_row), fg_blue);
               xorrect(end_col, start_row, 1, (end_row -
                               start_row), fg_blue);
               for(j = start_row; j <= end_row; j++)
                   {
                   for(k = start_col; k <= end_col; k +=
                                          k_step)
                      {
                       flag = 0 ;
                       offset = (512L * j) + k;
                       lseek(f_h, offset, 0); read( f_h,
                       &buf1, BUF1_SIZE );
                       flag = (value >= 64) ? 1 : 0;
                       if(flag == 1)
                       printf("Excess Solder Detected\n");
                      }
                  }
               l_width = 0;
               l_start = 0;
               l_end = 0;
             }
          }
       }
    puts("Finished Excess Solder Detection....");
 }
/*  End of Excess Solder Detection  Routine */
```

```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                  Shift Leads Detection Routine
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++*/

#include <io.h>
#include <dos.h>
#include <bios.h>
#include <math.h>
#include "mavis.h"
#include <fcntl.h>
#include <stdio.h>
#include <alloc.h>
#include <stdlib.h>
#include <string.h>

#define BUF_SIZE        512
#define BUF1_SIZE       1
#define NO_LEVELS       256
#define NO_LINS         512
#define NO_COLS         512
#define L_FACTOR        2.4
#define N               512
#define cls             printf("\033[2J\033[;H")

shiftleg()

{

           int i, j, n, k, line, row, l_start, g_start,
               k_step, f_h;
           int start_y, l_width, g_width, Threshold, flag,
               counter;
           int l_end, g_end, lead_num, first_l_end,
               start_col, end_col;
           int l_scan_field, g_scan_field, l_step,
               start_row, end_row;
           int gray_level, max_var_line;
           int g_center, l_center, l_length, l_numbers,
               g_numbers;
 unsigned long variance, max_var, mean_pix, sum, total,
               var[NO_COLS];
 unsigned long no_pix[NO_LEVELS], variance_of[NO_LINS],
               max_pix_var;
 unsigned long max_no_pix, mean_no_pix, ref[NO_COLS],
               offset, value;
 unsigned char buf[ BUF_SIZE ], buf1[ BUF1_SIZE ];
          char *file_name[20];

 *file_name = "smd.ima";
 if((f_h = open( *file_name, O_RDONLY | O_BINARY )) < 0 )
    {
     puts("\7\7ERROR file name TRY AGAIN");
    exit(0);
   }
puts("\t\t\t\tWorking...");
Threshold = 255;
start_y = 0;
max_var = 0;
for(line = start_y; line < NO_LINS; line++)
    variance_of[line] = 0;

for(line = start_y; line < NO_LINS; line++)
   {
    read( f_h, &buf, BUF_SIZE);
    for(i= 0;i < NO_LEVELS; i++)
        no_pix[i] = 0 ;

    for(i = 0; i< NO_COLS; i++)
        no_pix[(int)( buf[i])]++;

    sum = 0;
    for(i = 0; i < NO_COLS; i++)
       {
        sum += buf[i];
        mean_no_pix = sum / N;
       }
    variance = 0;
    for(i = 0; i< NO_COLS; i++)
        variance += abs((buf[i] - mean_no_pix));

    variance_of[line] = variance;
   }

for(line = 0; line < NO_LINS; line++)
   {
    if(max_var < variance_of[line])
      {
       max_var = variance_of[line];
       max_var_line = line;
      }
   }
line = max_var_line; /* first line contains leads */

for(row = line; row < line + 1; row++)
   {
    lseek(f_h, 512L * row, 0);
    read( f_h, &buf, BUF_SIZE);

    for(i= 0;i < NO_LEVELS;i++)
        no_pix[i] = 0 ;

    for(i = 0; i< NO_COLS; i++)
       {
        buf[i] = (buf[i] >= Threshold) ? = 255 : 0;
        no_pix[(int)( buf[i])]++;
       }
```

```
total = 0;
for(i = 0; i < NO_COLS; i++)
    total += buf[i];

mean_pix = total / N;
variance = 0;
for(i = 0; i< NO_COLS; i++)
    var[i] = 0;
for(i = 0; i< NO_COLS; i++)
   {
    variance = (( buf[i] - mean_pix)
              * ( buf[i] -mean_pix));
              if(variance > 9999)
      {
       variance = 1;
      }
    else variance = 0;
    var[i] = variance;
   }

l_width = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
       l_start = i; /* start of the lead */
    if(var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
       l_end = i; /* end of the lead */
    l_width = l_end - l_start + 1;
   }
g_width = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 0 & var[i-1] == 1
             & var[i+l_width] == 1)
       g_start = i; /* start a gap */
    if(var[i] == 0 & var[i+1] == 1
             & var[i-l_width] == 1)
       g_end = i; /* end a  gap */
    g_width = g_end - g_start;
   }
l_length = ceil((float)(l_width + g_width)
                              * L_FACTOR);
                              l_step =
                              ceil((float)(l_length *
                              8 / 10));

l_width = 0;
l_start = 0;
l_end = 0;
for(i = 0; i< NO_COLS; i++)
   {
    if(var[i] == 1 & var[i-1] == 0 & var[i+4] == 1)
       l_start = i; /* start of the lead */
```

```
        if(var[i] == 1 & var[i-4] == 1 & var[i+1] == 0)
           l_end = i; /* end of the lead */
        l_width = l_end - l_start + 1;
        if(l_width <20 & l_width > 4)
          {
           start_row = line + l_step; /* start line for scaning */
           end_row = line + l_length + 2; /* last row to scan */
           start_col = l_start - 3;
           end_col = l_end + 3;
           k_step = l_width + 6;
           fgmask(fg_overl);
           xorrect(start_col, start_row, 1, (end_row -
                              start_row), fg_green);
           xorrect(end_col, start_row, 1, (end_row -
                            start_row), fg_green);
           for(j = start_row; j <= end_row; j++)
              {
               for(k = start_col; k <= end_col;
                                   k += k_step)
                  {
                   flag = 0 ;
                   offset = (512L * j) + k;
                   lseek(f_h, offset, 0); read( f_h,
                   &buf1, BUF1_SIZE );
                   value = buf1[0];
                   flag = (value >= 96) ? 1 : 0;
                   if(flag == 1)
                   printf("Shifted Leads Detected...\n");
                  }
              }
           l_width = 0;
           l_start = 0;
           l_end = 0;
          }
       }
   }
    puts("Finished Shifts Detection");
}
/* End of Shifts Detection Routine */
```

Appendix

B

Glossary

Glossary

Amplitude: The strength of the signal representing brightness value; in image processing, the analog video signal.

Analog-to-Digital converter (A/D): A hardware device that changes an analog video signal to a binary quantity; a process known as digitization.

Aperture: The hole in the lens that allows light to pass through. The hole is variable in size which is measured in *f* stops.

Artificial Intelligence (AI): A research deals with trying to understand intelligent activities performed by humans and attempting to emulate them in machines.

Assembly language: Symbolic form of machine language in which each statement represents one machine language instruction.

Back porch: That part of a composite video display signal between the trailing edges of a horizontal synchronization pulse and the corresponding blanking pulse.

Beam: A concentrated, undirectional flow of electrons or other energy.

Blanking: The process of changing the display signal level so that the return trace is not visible on the display screen.

Brightness: The value associated with a pixel representing its grey-level value; for example, in a monochrome 8-bit system, black = 0 and white = 255.

Byte: Unit of information or memory size equal to eight bits; memory size is normally measured in kilobytes (1024) or megabytes (one million bytes).

Charge-Injection device (CID): A charge transfer device used as an image sensor in which the image points are accessed by reference to their Cartesian coordinates. CIDs have low dark current, are resistant to blooming but are relatively noisy.

Charge-Coupled Device (CCD): A self-scanning semiconductor array that uses MOS technology, surface storage, and information transfer by digital shift register techniques.

Contrast: the amount of brightness or the clarity of an image. A high contrast indicates that the image contains largely black and white brightness values; medium contrast indicates that the image contains a wide range of grey level values; low contrast image indicates that the image contains a small range of grey level values.

Contrast enhancement: An operation that changes the contrast attributes of an image to more clearly define a portion of an image.

Digital image: An image composed of discrete pixels of digital brightness values.

Digitize: The process of converting an analog signal's amplitude to a digital (binary) value.

Display: The device in which an image is converted from electrical to optical signals, for example, a television monitor.

DIP (Dual In-line Package): An integrated circuit package that has a row of pins for through-hole mounting along each of its longer sides.

EPROM: Acronym for 'erasable programmable read only memory'; type of **ROM** which can be erased by exposure to ultraviolet and reprogrammed.

Fillet: The concave junction formed by the solder between the footprint pad and the SMD lead or pad.

Filter: In image processing, an operation that changes the spatial and intensity characteristics of an image.

Flicker: A perceived rapid periodic change. Flicker disappears when the frequency of the stimulus change exceeds a rate called the critical flicker frequency.

Footprint: A group of metal pads on the PCB arranged in the same pattern as the leads or pads on a SMD and to which the leads or pads are soldered.

Frame: One complete **TV** picture usually representing a snapshot of a moving scene with an effective exposure time of 1/25 or 1/30 of a second.

Frame buffer: A high-speed memory designed to store one or more images and allow simultaneous video display, pipeline processing, and CPU access; also known as a frame store.

Grey-scale: Variations in the luminance value of "white" light, from black to white. Shades of grey are defined as grey-scale gradations that differ by the square root of 2. In an 8-bit monochrome system, these values can range from 0 = black and white = 255.
Histogram: A graphic representation of the grey-scale values contained in an image, that effectively measures the colour/grey values in an image. The horizontal axis represents the grey levels and the vertical axis displays the number of pixels within each grey level.

Histogram equalization: This process changes the dynamic range of an image expanding the image to contain the entire range of grey-scale or colour indices.

High pass filter: During the spatial filtering process, selectively passing through only high-frequency components, while removing all other components from the image. A high pass filter tends to sharpen the specified area of interest. The process of filtering is based on either a convolution or a selection of appropriate information.

Image processing: The analysis and alteration of an image with the purposes of enhancement or feature recognition.

Intensity: The strength of light at a particular point in an image. Pixels in a visible spectrum image represent intensity values that are perceived as brightness by the eye.

Interlace: The means by which an image is scanned, where the odd and even fields are displayed alternately. Interlace is used to reduce flicker in an image.

IC (Integrated Circuit): A device in which all of the components of an electronic circuit are fabricated on a single piece of semiconductor material, often called a chip which is placed in a plastic or metal package with terminals for external connection.

Incident light: The light falling on to a subject.

Low Pass Filter: During a filtering process, selectively passing low-frequency components, while removing high-frequency components from the image. A low-pass filter tends to blur an image.

Monitor: Display device used for images (for example, an RS-170 monitor).

Monochrome: Any combination of colors of the same hue, but of different saturations and luminances. Generally, a monochrome image is known as black and white (B/W).

National Television System Committee (NTSC): Used to identify the colour-encoding method adopted by the committee in 1953. The NTSC standards was the first monochrome compatible, simultaneous colour system used for public broadcasting.
Pick-and-place Machine: A programmable machine usually having a robot arm which picks up components from an automatic feeder, moves to a specified location on a PCB, and places or inserts the component onto or into the correct location.

Pixel: The smallest unit of storage in a digital image or raster line that can be discretely controlled by the acquisition, storage, and display system. A pixel is addressed by its horizontal and vertical coordinates that determine its location in storage.

PCB (Printed Circuit Board): A substrate of epoxy glass, clad material or other material upon which a pattern of conductive traces is formed to interconnect the components which will be mounted upon it.

Real-time: An operation that can be completed in one frame time is said to be performed in real-time. For standard television (RS-170) equipment a frame time is 1/30 of a second.

Resolution: The number of bits of accuracy or number of grey levels that can be represented in a pixel; for example , 8 bits = 256 levels, 6 bits = 64 levels.

Spatial enhancement: These operations use of selecting and removing certain frequency components from a digital image, via high- and low-pass filters.

Spatial resolution: The number of pixels into which an image is divided: the precision or accuracy horizontally and vertically. For example, a spatial resolution of 512x512 means that the image has 512 lines, with 512 pixels each.

Specular reflection: Reflection of light in which the angle of incidence is equal to the angle of reflection.

SMD (Surface Mounted Device): A device (components) designed to be mounted and soldered to pads on the surface of a PCB rather than inserted into through-holes in a PCB.

Soldering Vapour Phase Reflow: A type of reflow soldering in which the PCB assembly is passed through a vaporized inert fluorocarbon. The latent heat given up when the fluorocarbon condenses causes the solder to reflow.

Voids: Cavities inside the solder joint formed by gases which are released during reflow or by flux residues which fail to escape from the solder before it hardens.