INFORMATION TO USERS

This dissertation copy was prepared from a negative microfilm created and inspected by the school granting the degree. We are using this film without further inspection or change. If there are any questions about the content, please write directly to the school. The quality of this reproduction is heavily dependent upon the quality of the original material.

The following explanation of techniques is provided to help clarify notations which may appear on this reproduction.

1. Manuscripts may not always be complete. When it is not possible to obtain missing pages, a note appears to indicate this.

2. When copyrighted materials are removed from the manuscript, a note appears to indicate this.

3. Oversize materials (maps, drawings and charts are photographed by sectioning the original, beginning at the upper left hand corner and continuing from left to right in equal sections with small overlaps.

UMI®
ProQuest Information and Learning
300 North Zeib Road, Ann Arbor, MI 48106-1348 USA
800-521-0600

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
AN AUTOMATED SOLDER JOINT INSPECTION SYSTEM
USING MACHINE VISION
JASWINDER SINGH CHADHA
Department of Mechanical and Industrial Engineering

APPROVED:

Dr. W.C. Johnson, Chairman
Dr. Vladik Kreinovich
Dr. Thomas J. McLean

Dean of Graduate School

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
To My Parents
AN AUTOMATED SOLDER JOINT INSPECTION SYSTEM
USING MACHINE VISION

by

JASWINDER SINGH CHADHA, B.Tech.M.E.

THESIS
Presented to the Faculty of the Graduate School of
The University of Texas at El Paso
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Mechanical and Industrial Engineering
THE UNIVERSITY OF TEXAS AT EL PASO
MAY 1992
ACKNOWLEDGEMENTS

I want to thank all my teachers who have helped me in my journey through school. I specially want to thank Dr. Carroll W. Johnson and Dr. Vladik Kreinovich for guiding me continuously through my work and giving me their valuable time and support whenever I needed it. I also want to thank Dr. Tom McLean for being on my graduate committee. Special thanks also goes to Eulalio Rodriguez for his help in setting up the hardware and Navdeep Singh Chadha for his help in programming.

I am grateful to my parents for their continuous support, encouragement and guidance throughout my educational career, without which it would have been impossible for me to come this far.

This project was presented to the graduate committee on December 13, 1990.
Printed circuit (PC) boards are a vital part of modern electronic technology. The need for them is in millions and is constantly increasing. With the progress of manufacturing technology these boards have more and more components, and each of these components has to be soldered onto a board. So it becomes more and more difficult for human inspectors to check the quality of all the solder joints, and there is a great practical need for an automated solder joint inspection system. This system must not only to detect defective joints, but it must also classify the defects so that this information can be used as a feedback for the process control in manufacturing. At present there exist several automated systems for that, the majority of them use expensive methodologies like X-ray or infrared vision to test the boards that will be used in vitally important systems, for example, in space and military applications. However, they are too expensive to be used in mass production. For mass production machine vision systems have been developed. The best of them are reasonably good in distinguishing good joints from bad ones, but their ability to tell what precisely is wrong with a joint is still far behind the ability of experienced human inspectors. So, it is necessary to improve these systems.
In the present thesis several improvements of the current methodology for automated solder joint inspection are designed and implemented. First, an orthogonal illumination source is used, that allows us to diminish the noise caused by the normally used diffused light. Second, changes are made in the algorithms that extract features from the pixel-by-pixel data: before extracting we locate the actual center of the joint (that can be shifted from its ideal position), a circular window is used instead of a rectangular one (so that the features become more physically meaningful), three zones of an image (instead of two) are used to extract the features, and then statistical feature reduction methods are applied to delete irrelevant and redundant features and thus diminish the computation time that is necessary for feature extraction.

In addition to that theoretical foundations are laid for choosing additional features and for improving the existing decision-making methodologies.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Machine vision is feasible</td>
<td>1</td>
</tr>
<tr>
<td>1.1.2 Printed circuit boards inspection: an area where automation is a must</td>
<td>1</td>
</tr>
<tr>
<td>1.1.3 Two types of systems: advanced systems for critical applications and mass production systems. Current state of automated inspection</td>
<td>4</td>
</tr>
<tr>
<td>1.1.4 Main objective of present research</td>
<td>6</td>
</tr>
<tr>
<td>1.1.5 How visual solder joint inspection is performed now (brief description)</td>
<td>7</td>
</tr>
<tr>
<td>1.1.6 Why the existing methods often err?</td>
<td>8</td>
</tr>
<tr>
<td>1.1.7 How we are going to overcome these drawbacks</td>
<td>10</td>
</tr>
<tr>
<td>1.2 LITERATURE REVIEW</td>
<td>11</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
2. SOLDER joint inspection

2.1 Current state of solder joint inspection

2.2 Objective and research approach

2.3 soldering and solder joint technology

2.4 Illumination

2.4.1 Fundamentals of illumination

2.4.2 Solder joint illumination

3. inspection techniques and feature selection

3.1 Qualitative description of inspection technique

3.2 Quantitative description of features and classification

3.2.1 Brightness and size normalization

3.2.2 Circular sub-area feature set

3.2.3 Moment of inertia feature set

3.2.4 The faceted surface area feature

3.2.5 Surface curvature feature set

3.2.6 Gaussian curvature related features

viii
4. THEORETICAL BACKGROUND FOR FEATURE REDUCTION AND CLASSIFICATION

4.1 EXPERIMENTAL FEATURE REDUCTION

4.1.1 Stability test

4.1.2 Separability analysis

4.1.3 Correlation analysis

4.2 CLASSIFICATION

4.2.1 Classification algorithm

4.2.2 Subjective weighing factors

4.2.3 Fuzzy membership factors

5. CONCLUSIONS AND RECOMMENDATIONS

5.1 What has been done?

5.2 Suggestions for future work

REFERENCES

APPENDIX

CURRICULUM VITAE
LIST OF FIGURES

1. An ideal soldering cell with automated solder joint inspection..................... 19
2. Solder joint inspection setting........................................................................ 26
3. Gray level surface plots.................................................................................. 29
4. An ideal solder joint image............................................................................. 38
5. Solder joint image sub-areas........................................................................... 39
1.1.1 Machine vision is feasible.

Recent technological breakthroughs in machine vision systems have made them relatively cheap, sufficiently reliable and very powerful. They are therefore now actively used not only in research, but in manufacturing as well, with many applications in mass production. Their main application is in inspection and related process control. These systems are easily implemented in the situations where the desired shape of the inspected part is consistent (and therefore easy to formalize). But in many cases parts which are good and those which are bad are not are not well defined visually. In these cases the existing technologies rely on the ability of the human inspector. The manual inspector is a weak and subjective character of this judgement who causes errors.

1.1.2 Printed circuit boards inspection: an area where automation is a must.

An area in which the necessity to automate visual inspection processes is most urgent is in printed circuit (PC) board inspection. PC boards are typically
inspected extensively before and during the insertion of components and the soldering process. A great deal of attention has been spent on this inspection application in the past ten years [1]. The problem of automated inspection of "bare" PC boards (before the insertion of the components) has already been solved. Such automated systems are now widely used in manufacturing. However, as far as assembled PC boards are concerned, there are only few systems available that are not yet ready for manufacturing applications. When we inspect assembled PC board, we want to know two things: first, whether all electronic components are there or some of them are missing, and, second, whether all the solder joints are of sufficient quality. The first question is easier to answer, because the images with and without a component are clearly quite different, and there exist efficient automated systems that check the presence of these components. The main problem is to figure out whether the solder joints are good or not.

At present this problem is mainly solved by means of visual inspection. But this is a dead end, because the abilities of human inspectors are limited, and the number of elements placed on a printed board can be in thousands, and their population per board increases rapidly. The manual visual inspection of printed circuit board (PCB) already accounts for approximately fifteen percent of the labor cost of manufacturing a printed circuit board. As PCB component population
density increases, these problems increase exponentially.

Another problem that appears when human inspection is used is that the reliability of an average human inspector is far from ideal. Moreover, the inspectors get tired during the workday and the quality of their decision decreases drastically (to up to 20% of errors). These errors lead to three kinds of problems:

1) if we accept a bad solder joint as a good one, this defective board can cause the failure of the system, in which it will be used; this failure can sometimes lead to catastrophic consequences, and even if it simply causes a system to stop, it is usually extremely expensive to figure out what exactly is wrong with the system,

2) if we classify many good joints as defective, then we waste time and money spent on producing them and thus we further increase the total production cost. Moreover, erroneous classification of good solder joints as defective ones can lead to the unnecessary and costly inspections of the whole technological process,

3) if we correctly classify a solder joint as defective, but err in the nature of this defect, then the resulting correction of the technological process will lead us in the wrong direction. This will not only not solve the problem, but even add new defects to the existing ones.
Because of all that, there is an obvious necessity to develop an Automated Inspection and Diagnosis System for PC Board manufacturing. The roles of this system are:

(1) automating visual inspection processes which are usually performed by human inspectors in most industries,

(2) utilizing the inspection results generated from the automated inspection systems in an on-line diagnostic system to automatically control the manufacturing processes.

1.1.3 Two types of systems: advanced systems for critical applications and mass production systems. Current state of automated inspection.

Crudely speaking, one can divide all possible applications of printed circuit boards into two main groups:

1) applications in which reliability is so critical that essential expenses are justified if they increase the reliability of the system. These applications include many military applications, applications to space systems, to critical communications,

2) applications to mass production, where the production cost is a serious limitation. For example, there is no sense in decreasing the failure rate of TV
sets 10 times if this will lead to a drastic increase of their cost.

For critical situations there exist methods that allow us to tell good solder joints from defective ones. These methods include X-ray \cite{9,10,11} or infra-red screening \cite{13}, laser scanning \cite{12}, etc. All these methods unveil the whole 3-dimensional structure of the solder joint and thus give us an absolutely reliable information on whether it is good and if not what exactly is wrong. But these methods are extremely expensive, so there is no way how one can use them in usual manufacturing. Moreover, they include using health-hazard devices, so special care (and therefore additional cost) emerges from the necessity to make them safe for the workers.

Because of all that the only information about the solder joint that we can use in the mass manufacturing is the visual image of that joint. There exist several automated systems for solder joint inspection that are based on this visual information. The best of them is a system developed by Barlett \cite{18}. The percentage of errors that this system makes when it tells a good joint from a defective one is the same as with the best human inspectors and much better than that of an average human inspector. However, this system is not that good in classifying the defects. It often errs in deciding what exactly is wrong, and we still
need a human inspector to tell what to change in a technological chain. Therefore these systems are not widely used in manufacturing, and the need for automated solder joint inspection systems is still very urgent.

Additional drawbacks of the existing systems are as follows:

1) they are based on some rigid classification of possible defects; for example, it classifies a defective joint to one of 10 classes depending on which of the 10 defects is there; but actual joints can have more than one defect, and for such joints the existing systems do not work.

2) the existing systems are mainly based on brute-force methods. They include lots of computations and are therefore very slow (even on the faster computers).

Based on these drawbacks, we can formulate the main objective of the present research:

1.1.4 Main objective of the present research.

To develop a method of automated solder joint visual inspection, that will:
1) essentially increase the percentage of correct decisions,
2) correctly classify the joints with several defects,
3) essentially decrease the computation time.

In order to figure out how this can be achieved, let's briefly describe how visual solder joints inspection is performed now (detailed description will follow in Chapter 2).

1.1.5 How visual solder joint inspection is performed now (brief description).

At present the solder joint inspection is performed in 3 steps:
1) first, we illuminate the board and use a machine vision system to translate the optical image into a pixel-by-pixel (discrete) electronic one. After this step we get megabytes of data, and of course it is impossible to process all of them. So we need to compress the resulting image, i.e., compute some characteristics that will be used in future computations. These characteristics are called features. So we:
2) extract features from the image,
3) and then we train some classification algorithm so that it will allow us to tell
good joints from defective ones and tell what exactly is wrong in case of a
defective joint.

1.1.6 Why the existing methods often err?

To our viewpoint, all three stages of the existing methodology contribute to the
far-from-ideal overall performance of the existing systems. We’ll explain them on
the example of the best of these systems [18];

1) in order to simplify the processing algorithms, the authors of this systems use
the diffused light source that gives a uniform illumination. However, this kind
of illumination leads to the fact that the observed brightness in any point is
actually cause not only by this very point, but by the neighboring points as
well. In practical terms it means that when we analyze some area on the
board, we add a noise (caused by the neighboring points) to the resulting
brightness distribution. Additional noise is caused by the ambient light.

2) for the same reason: to simplify the processing algorithm, the features that
are extracted are based on choosing a rectangular region surrounding a joint.
From the physical viewpoint a rectangular region makes no sense, because the
solder joint is usually circular. So what we obtain is that to every feature we
add a physically senseless term corresponding to the difference between the
rectangle and the circle. This additional term also acts as a noise. Another thing that is not done properly on the second stage (of feature extracting) is that no attempt is made to locate the actual center of the solder joint. It is supposed that it is precisely in the place in which it should be according to the design of a board. So when there is a small deviation of the center and in all other senses the joint is perfect (and this is a rather frequent situation), all the features that are computed from the wrong point of reference are far from ideal and this solder is claimed to be defective.

3) There are three reasons why the processing stage adds to the imperfect performance of the system:

a) first, this system uses too many features, the majority of them have no physical meaning or are practically duplicating each other, and this leads to unnecessary computations,

b) on the other hand, in some cases evidently defective or good features are not properly classified because the existing features do not allow to distinguish between them,

c) additional errors are added on the training stage. The authors use the solders that are produced in the laboratory for training. However, these lab-generated solders give a simplified set. Therefore the resulting system does not always work for real joints, that can be much more complicated.
1.1.7 How we are going to overcome these drawbacks.

In the present thesis we explain how to modify the existing system so that it should overcome these difficulties and thus show the better performance.

We’ll formulate the proposals about all the 3 stages:

1) in order to diminish the noise caused by the diffused light we propose to use a source of light that is orthogonal to the surface of the PC board,

2) to diminish the noise that is caused by a rectangular window we use circular window for extracting features. To take care of the possible center-shift we first find an actual center of the image and then compute the features using this actual center as a reference point,

3) a) in order to diminish the running time we apply feature reduction, that is, we analyze all the existing features and choose only those that are relevant and independent (this allows to reduce the set of features from about 60 to about 15),

   b) we add a new physically meaningful features that correspond to the division of the image into 3 (and not 2 as usual) circular zones: the central, brightest zone that corresponds to the top of the lead, the intermediate (black) zone where the height of the joint is rapidly decreasing, and the
external region that is again almost flat,

c) we use actual solder joints for training instead of the lab-generated ones.

These ideas are already implemented. In addition there is an idea that is still in the stage of theoretical analysis and experiments: in order to take care of the cases when one joint can have several defects, and to diminish the computation time, we can apply fuzzy decision methods instead of statistical ones.

1.2 LITERATURE REVIEW

In order to reduce the cost and to increase the product reliability, automatic techniques for printed circuit boards have been introduced to bare board inspection [2,3], pad and track inspection [4,5], component inspection, chip bonding alignment inspection[6], and solder joint inspection[7].

Currently, four types of automatic inspection systems are available [7]:

(1) optical machine vision uses precision video techniques and sophisticated software to inspect through hole and leaded components and surface mount PC boards [6,8].

(2) X-ray image systems inspecting multi-layer board inner layer registration and
solder joints [9,10,11].

(3) laser scanning systems differentiating between PCB copper and substrate and laser/thermal profile solder joint inspection[12].

(4) thermal imaging systems detecting over-stressed components on PCB’s[13].

Of these technologies, most of academic work has been accomplished using the first technology because of their lack of need for sophisticated illumination sources and sensors such as X-ray or lasers. Generally, optical vision systems have benefits because they are relatively inexpensive, cause no damage to the inspected object, and are without any danger to the operator. However, they do not currently have the capability to detect structural defects within the solder.

Nakagawa[14,15] uses a structured lighting system with a projected light beam controlled by a slit plate and applies a simple waveform processing technique to extracted shape for judging the topology of a solder joint. The advantages of this method are:

(1) the technique is fast, 10 joints per second are possible for dual in-line IC’s.

(2) only two features are used to classify the joints into four classes.

The disadvantages of this method are:

(1) only gross profiles of solder are covered, which could miss other defects.
(2) the inspection results only indicate if a solder joint is acceptable or not.

The inspection lacks detailed information about the type of defect. Thus the method is practical but limited potential for process control.

McIntosh[16] used an inspection system with one specific lighting system with a robot indexing the board. The method is based upon a binary image that consists of up to 40 solder joints per scene. The advantages of this technique are:

(1) it is fast because of binary feature computation.
(2) high performance, 99.8% flaw detection is claimed to be achieved.
(3) the inspection board is automatically indexed by the robot.

The disadvantages are:

(1) since only the defective joints with hole and irregular shapes are detected, other defects will be missed by the algorithm.
(2) post-processing is required to distinguish between regular and oversize/irregular-shaped joints.
(3) lacks the potential ability to set up a process monitoring capability.

Besl[17], Barlett[18] developed an inspection algorithm employing 20 features to identify solder joint defects. The advantages of this method are:

(1) a large amount of information is obtained by various features and used to
describe the characteristics of the solder surface.

(2) weighing factors for the features are applied to increase the discrimination performances.

The disadvantages of this approach are:

(1) sufficiently diffuse and uniform lighting is not used, which is critical to the image of solder surface and may seriously decrease the performance of identifying features.

(2) the square-sized samples of solder joints were made in the laboratory, which may lack the real characteristics common in manufacturing processes.

(3) the classification standard for solder joints does not give much information about either the possible causes for the defects, or the suggestion of adjustments to improve quality.

(4) there is no information about what the high performance features are, so a feature-reduction algorithm for fast inspection is not available.

(5) the minimum distance classification algorithm used is not suitable for most acceptable joints in a real production situation. The algorithm used is worthwhile but not practical, and more effort is necessary to build up a good lighting system, to cut down the number of features, to set up a better classification standard and an integrated information database, and to design a more effective classifier to speed up the automatic inspection task.
Other attempts include Sanderson's structured highlight approach using 3-D array of illumination sources[19] and Ray's solder bump inspection system using dark-field and bright-field illumination scheme[20]. Performance of these academic systems are not completely satisfactory in real applications and most of commercial systems are quite expensive and complex.

Many of the commercial systems are developed using sophisticated illumination sources, sensors and mechanisms. The X-ray inspection technology is employed by IRT Corporation[21]. The IRT's system utilizes radiographic imaging techniques for obtaining X-ray images of solder joints. This system uses shape and volume properties of the X-ray images as a means of classification.

The 3-D laser measurement technology is used in Robotic Vision System's laser-based solder joint measurement system[22]. The system utilizes a structured-lighting method to obtain thousands of 3-D data points of solder joint surfaces.

The thermal testing technology employed in the Vanzetti system[23] utilizes infrared signatures of a solder joints for inspection. These signatures are produced when a laser heats an individual solder joint and a nearby infrared sensor records its thermal radiation curve. The system suffers from some speed limitations due
to the mechanical positioning required to study each joint as opposed to an imaging system which can obtain images of several joints within a single field of view. Also, the heating of solder joints for inspection purposes is a somewhat questionable practice since joint quality may be adversely effected.
CHAPTER 2
SOLDER JOINT INSPECTION

2.1 CURRENT STATE OF SOLDER JOINT INSPECTION

Most of the plated-through-hole (PTH) solder joints on printed circuit boards are formed using automatic wavesoldering machinery. There are various process parameters such as solder bath temperature, wave size, and conveyor speed. Even small defect rates can typically cost manufacturers millions of dollars per year. To reduce these large losses, the defective solder joints should be detected as early in the assembly as possible.

Traditionally, solder joint inspection has been performed manually and indirectly via electrical testing. Neither method is entirely reliable. Manual inspection suffers from several serious drawbacks including inconsistency and poor performance due to fatigue and boredom. Electrical testing can verify the correct operation of a circuit. But with more complex circuit boards the task of testing becomes more difficult. Moreover, joints with insufficient or missing solder which provide limited contact may go undetected during electrical testing, only to fail in the field after exposure to mechanical and thermal stresses.
Automated solder joint inspection has been considered to be one of the most difficult tasks for the application of automated visual inspection techniques. The great variability of solder joint defects and their appearances in acquired images make correct classification of those defects even more difficult. The identification of the defect as a pin-hole, for example, depends on the size as well as the location of the hole. Even in the class of acceptable solder joints, variations in appearance come from lead size, lead position, bent lead condition, and amount of solder. A similar problem in classification of defects is that multiple defects could be occurring in one joint, which may bias the result of the inspection. Another difficulty for solder joint inspection is to design a good illumination environment which can avoid the saturated reflection on the solder surface and provide the joint image with the stable geometrical information about the solder surface.

Unlike the bare board, the characteristics of solder joint are concerned with three dimensional geometry. Therefore, the most difficult task for solder inspection with a vision system is to select good features to describe the exact common geometrical properties of all the solder joints in one given class. These features are critical to the performance of inspection systems, both with regard to accuracy and speed. An ideal automatic solder joint inspection system should be able to decide much more than if joints are just good or bad, but to identify the
Figure 1. An ideal soldering cell with automated solder joint inspection.
class of the defect, and to suggest the cause of the defect both from the design and manufacturing point of view as shown in Figure 1. Then, specific adjustments can be sent to the process stations from a host computer to improve the quality as quickly as possible. Therefore, the ultimate goal of a perfect solder joint inspection system is not to correct the flaws but to provide a process control capability.

2.2 OBJECTIVE AND RESEARCH APPROACH

The objective of this research is to select good features under a specific lighting environment to distinguish various kinds of solder joints. In order to avoid the noise due to ambient light an intense light has been used which is orthogonal to the plane of the circuit board. Using this lighting system, the solder image is more stable and independent of background lighting. To make the system practical, PC boards from Alcatel Corporation were made available for the tests, and a classification standard developed by Davy[24] for wave solder joints is used to discriminate the defect types. In addition, a circular window and three sub-areas of the solder image are implemented in the study. This is because it is easier to detect some defects as they occur in certain sub-areas of the joint corresponding
to the windows. Also, circular windows come more closer to the solder joints than the rectangular windows because of the circular nature of the solder joints.

To start with, a number of typical solder joints from each class of defect are selected by manual inspection. Thereafter experiments in feature training and performance evaluation are based on these samples. The feature used in the system are based on the characteristics of the intensity surfaces of solder joints. The features are based on circular sub-areas, second moment of inertia, faceted surface area and surface curvature. They are employed in the analysis. Before the experiment, a simple test is executed to check the stability of these features. To delete bad features and to combine redundant ones, a separability factor and a correlation factor are defined and used to measure the performance of each feature and the similarities among them. Then, the features with low performance may be removed, and those with high correlations can be combined in order to reduce the dimensions of feature vector space. Therefore, a minimum number of effective features will be selected for the inspection task. Then, a simple experimental test is done to estimate the performance of the reduced feature set. Finally, fuzzy set theory [25,26,27,28] is employed to develop a classification algorithm which can categorize underlying solder joint defects with maximum accuracy.
2.3 SOLDERING AND SOLDER JOINT TECHNOLOGY

A wave solder joint classification scheme is followed in this research, which was developed by Davy [24]. The characteristic of this standard is to categorize the solder joint according to the probable cause of the defects, rather than only on superficial appearance. In general, defects come from design faults, materials, or processes.

By this method, only isolated joints on the bottom (joint) side are of interest. Some other defects associated with physical failures on the top (component) side are not taken into account either, such as incomplete fill plated through hole and incomplete flow through a plated through hole. Although a number of joints were made available, most of them were good joints. The number of joints with inclusions, disturbed joints, and dull joints were small. In addition, the defect of a joint with an exposed end can only be detected by the copper color at the end, so it can not be identified by a black/white camera. Hence, these types of defects are neglected by this system since they appear to occur only very early and have detection difficulties. Finally, therefore eight major types of joints are chosen to be classified,
(1) Good joint

(2) Joint with excess solder

(3) Joint with pin-hole

(4) Joint with too little solder

(5) Joint with missing lead

(6) Joint with no solder

(7) Joint with poor wetting on lead

(8) Joint with poor wetting on land (pad).

Although the classes of too little solder and missing lead fall into the acceptable type the classification scheme, the images of these two kinds of joints look obviously different from those of other classes of joints due to the inappropriateness of the process. So it is reasonable to categorize these two types of joints as defects in order to differentiate these joints from other classes. Since the machine wave solder joints are of interest in this research, the defect of cold solder typical of hand soldering is not taken into account. These eight classes cover the majority of the joints in machine soldering.
2.4 ILLUMINATION

2.4.1 Fundamentals of Illumination

Illumination is an essential part of a computer vision system, and is very critical to the performance of the inspection task. However, it is often under-evaluated and usually undergoes considerable redesign until a good image is acquired [29]. An ideal illumination system should provide the best possible images that can yield the most contrast between the features of interest and the background. Background lighting in the vicinity of the system should be avoided, since this will interfere with the planned illumination effect on the inspected objects. In spite of several typical lighting types, specific lighting design is dependent upon the characteristics of the objects being reviewed. Generally, there is no global rule for a good design. One way to achieve this, however, is to define the characteristics of the inspected object which need to be extracted and build an ideal image for that object. Then, the differences between the ideal image and the one under current lighting environment may be checked for correlation. After analyzing the differences, the objective is to try various methods to make up the differences.

In machine vision, four kinds of factors usually affect the performance of the
illuminatation [30].

(1) The light reflection from shiny surfaces
(2) The light absorption of dark objects
(3) The contrast between the inspected objects and the background
(4) The interference of ambient lighting from the surroundings.

2.4.2 Solder Joint Illumination

Before setting up the lighting system, some features about the solder joint inspection must be known:

• Whether the reflectance of the solder surface will saturate the image
• If the lighting will provide three-dimensional characteristics of the solder joint onto a two-dimensional image plane
• Local shaded areas of the solder should be avoided
• The lighting system should be isolated from the surroundings, and not interfered with by background lighting
• The illumination system must be inexpensive, easy to set up, simple to maintain, and otherwise physically robust.
Figure 2. Solder joint inspection setting.
In order to build a good lighting system, and to meet the requirements of an ideal solder joint image including high contrast of the joint surface and wide dynamic range, it is necessary to acquire a uniform and intense lighting source. To obtain this kind of lighting for this project, a 360° fibre optic fluorescent ring tube was selected. Figure 2 shows the ring light tube and the tested PCB's. In addition, the ring lamp is relatively close to the joint, thus a surrounding invariant illumination environment is established. This illumination system has the advantages of meeting all the preceding requirements.
CHAPTER 3

INSPECTION TECHNIQUES AND FEATURE SELECTION

3.1 QUALITATIVE DESCRIPTION OF INSPECTION TECHNIQUE

A specific illumination system has been established to represent the characteristics of three-dimensional solder joints as the intensities of two-dimensional joint images. Generally, the intensity of a solder image is a function of the surface reflectance, the scene illumination, and the surface geometry. The intensity function can be expressed by

\[ I = F(r, i, g) \]  \hspace{1cm} [3-1]

\( r \): surface reflectance
\( i \): scene illumination
\( g \): surface geometry

Under most conditions, the properties of solder alloy used for soldering are reasonably stable, so that the reflectance capabilities of solder surfaces are highly consistent. The scene illumination state will also be uniform with the same lighting system under control. Hence, the variables \( r \) and \( i \) can be treated as constants in the previous equation. The only free variable left in the function is
Figure 3. Gray level surface plots. (a) Good solder (b) Bad solder
the surface geometry, $g$, i.e.

$$I = F^*(g)$$

[3-2]

Empirically, the intensities of the solder joint images in one given class have certain common characteristics due to the similarities of the geometric shapes. Consequently, the image intensities of the solder joints need to be understood for the classification process. An inspected joint can be associated with right class based on the characteristics of its image.

Compare the different solder joint gray level images shown in Figure 3. One joint is definitely good and the other is definitely bad, in this case. Since we are using an illumination system in which the incident light is orthogonal to the plane of the PC board, each class of solder joints can be very well defined mathematically. If we study the change in the pixel gray level values with the curvature of the solder joint, we find that those areas whose curvature has a large gradient are darker than the ones which are relatively flat. This is due to the fact that in case of the curves with larger gradient, the incident light is reflected in a direction such that no or very few dispersed reflected rays enter the camera.

The features used in our system are based on the characteristics of the intensity
surfaces of the solder joints described above. Some of the feature models are very close to the facet model proposed by Haralick [31], Gaussian curvature based on differential geometry of surfaces by Besl [17]; other features, which are related to two dimensional fast fourier transform, fractals and entropy are based on the irregularities of the three dimensional geometry of the model described above. Thus, intensity is considered as the third dimension (combined with x and y spatial directions) and uses the characteristics of the surfaces in a solder joint image to classify them.

The selection of good features are critical to the performance of classification. Any irrelevant or redundant features will decrease the performance as well as increase the processing time. In general, one good feature set requires four qualifications [32]:

- **Discrimination**; features should present significantly different values for all the objects belonging to different classes
- **Reliability**; features should take on similar values for all the objects in the same class
- **Independence**; various features should be uncorrelated with each other
Small number; a small number of features will make the classification process simple and fast.

In this research, 49 features from 4 different sets are investigated. From this, a minimum number of good features are selected with a series of analyses which are based on the concepts listed above to provide variant characteristic measurements for the image intensities. The circular sub-area feature set presents the information of average and variation of image area, middle solder area, and lead area, where some specific defects tend to occur and can be easily detected.

The vision system used for this research is an INTELLEDEX 386P, which provides 64 gray level intensities for images. The size in the pixel of various joints is dependent on the distance of the camera and other camera parameters. A standard Pulnix CCD camera with a 50mm lens is used. An extension tube of 20 mm is used for magnification. A very high magnification is not desirable because it limits the number of solder joints which can be inspected in one picture frame and thus makes the system very slow. To avoid noise from the background image (board, track, etc.) and to reduce the processing time, circular windows are generated to fit the circular area of the joint image.
An automated solder joint inspection system should be capable of relating regions of particular images to individual solder joints on a PCB or else it will be incapable of determining which solder joint on a given PC board is defective. For this the exact x-y location of a solder joint on a PC board and where the subimage of that solder joint will occur in a digitized image should be known. This data of size and location of each solder joint can be available from PC board design/manufacturing database. Since no CAD data about joint location and size of the pads are available for these PC boards, manual registrations of solder joints are performed using a user interface program. However, this manual registration cannot be as accurate as the manufacturing information and is therefore not preferred.

3.2 QUANTITATIVE DESCRIPTION OF FEATURES AND CLASSIFICATION

In this section, the features which have been used to classify the solder joint subimages and the classification process itself, are defined mathematically. The features are grouped into 5 categories:

(1) basic features involving circular sub-areas

(2) inertia features of the volume defined by the subimage
(3) the faceted surface area feature
(4) Gaussian curvature related features
(5) fast fourier transform, entropy and fractals related features.

Only the first four are discussed here.

3.2.1 Brightness and Size normalization

An attempt has been made to choose features of solder joint subimages which can be made invariant to solder joint size and brightness levels and are dependent only upon the structure of the gray level surface, which is responsible for the visual appearance of the solder joint. If we are given discrete image data in the form of an $n_x \times n_y$ rectangular array of digitized gray values which range between 0 and $2^{nbits}-1$ where nbits is the number of bits used in digitization. All of our gray-scale images are digitized to 6 bits yielding the usual 64 gray levels.

Computationally, a gray scale image will be defined as a set of discrete ordered triple:

$$Image = \{(i,j,k) : k=f(i,j)\};$$

$$i \in \{0,1,...,n_x-1\},$$
\[ j \in \{0,1,\ldots,n_y-1\}, \]
\[ k \in \{0,1,\ldots,2^n_{bis}-1\} \]

where \( f \) is a function of two variables. Brightness normalization is accomplished by dividing all gray levels by \( f_{\text{max}} \), the maximum gray level in the image, which is defined as

\[ f_{\text{max}} = \max_{i,j} f(i,j). \]  \[3-4\]

This normalization maps all gray levels in an image to the interval \([0,1]\). Normalized gray level function will be denoted as

\[ z(i,j) = \frac{f(i,j)}{f_{\text{max}}} \]  \[3-5\]

for all \((i,j)\)

Size normalization is accomplished by mapping all pixel locations into the unit square centered at the origin \([-0.5, +0.5] \times [-0.5, +0.5]\). The pixel size depends upon \( n_x \) and \( n_y \). The coordinate functions \( x(i) \) and \( y(j) \) are defined so that the symmetry conditions \(|x(0)| = |x(n_x-1)|\) and \(|y(0)| = |y(n_y-1)|\) are met:

\[ x(i) = \frac{(1+2i-n_x)}{2n_x} \]  \[3-6\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
This combination of size and brightness normalization maps all subimages into surfaces which lie within the unit cube.

3.2.2 Circular Sub-Area Feature Set

In this feature group, basic information about the intensity distribution is of interest because of simplicity and speed consideration. The first two features to be defined are the mean and standard deviation of the intensity distribution, which provide the average intensities and the variations of an image. These are the same as the normalized volume. For example, acceptable solder joints usually have large central peaks yielding a small normalized volume, whereas a hole filled with no solder usually has a flat surface yielding a large normalized volume.

\[
y(i) = \frac{(1+2j-n_y)}{2n_y}
\]  \[3-7\]

\[\sigma = \sqrt{\frac{1}{n_x n_y f_{\text{max}}} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} f(i,j)}\]  \[3-8\]

Normalized standard deviation feature \(\sigma\) which is a measure of the variance of the normalized gray level surface about its mean value is the second feature.

These features are independent of the ordering or spatial distribution of pixels.
in the image. From a typical solder joint images shown in Figure 4, it is reasonable to divide the image into three sub-areas: outer-ring, middle solder area, and lead area as shown in Figure 5. Not only are the intensities within these areas of a joint different but various defects tend to occur within or near one of these areas. For example, the outer ring area of excess solder joint images are significantly darker than those of other classes of joints for the illumination system used. The defect of a joint with poor wetting on a pad may be identified either within out-ring area or the middle area. In order to get isolated information within the different areas, the mean and standard deviations of the three sub-areas established as features

\[ a_1 = \sigma^2 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \left( \frac{f(i,j)}{f_{\text{max}}} - V_{\text{opt}} \right)^2 \]  

in the image. From a typical solder joint images shown in Figure 4, it is reasonable to divide the image into three sub-areas: outer-ring, middle solder area, and lead area as shown in Figure 5. Not only are the intensities within these areas of a joint different but various defects tend to occur within or near one of these areas. For example, the outer ring area of excess solder joint images are significantly darker than those of other classes of joints for the illumination system used. The defect of a joint with poor wetting on a pad may be identified either within out-ring area or the middle area. In order to get isolated information within the different areas, the mean and standard deviations of the three sub-areas established as features

\[ a_2 = \mu_1 \]  
\[ a_3 = \sigma_1^2 \]  
\[ a_4 = \mu_2 \]  
\[ a_5 = \sigma_2^2 \]  
\[ a_6 = \mu_3 \]  
\[ a_7 = \sigma_3^2 \]  

In these features, the subscripts of 0,1,2,3 represent the whole circular joint.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4. An ideal solder joint image
Figure 5. Solder joint image sub-areas
area, out-ring area, middle solder area, and lead area respectively. One technical problem is to automatically segment the lead area exactly. This is a very difficult task which involves variations in lead sizes, lead locations, and lead area. An algorithm has been developed to accomplish this task. In this program, a scanning method is used to segment three sub-areas based on the binary images.

3.2.3 Moment of Inertia Feature Set

The second set of features are the moment of inertia of the image. The intensity of a image is taken as a three-dimensional surface is symmetrically reflected about the plane of the circuit board. Each pixel of the image represents a unit vertical block in which its height is twice the intensity of the pixel. Surface is used twice instead of only once to bound the volume in an attempt to increase the effect of the surface shape upon the inertia values and to simplify a few computations. The general expression for the inertia tensor of a mass distribution relative to a given coordinate system (CS) is given conveniently by

$$\mathcal{I}_{cs} = \int \int \int \rho(r) (r^2 r^T[I] - rr^T) \, dx \, dy \, dz \quad [3-16]$$

where $r = (x,y,z)$ is the 3-D coordinate vector relative to the CS coordinate system, $[I] = 3\times3$ identity matrix, superscript T denotes transpose, boldface
denotes vectors, brackets denote matrices, and \( \rho(r) \) = object density function.

The center of mass vector for a mass distribution \( \rho(r) \) is given by

\[
\mathbf{r}_{cm} = \iiint \mathbf{r} \rho(r) \, dx \, dy \, dz.
\]

[3-17]

The volume of each block on the pixel is \( \Delta x \Delta y \Delta z \) where \( \Delta x = 1/n_x \), \( \Delta y = 1/n_y \). The center of the block is at \( z = 0 \), on the x-y plane. We compute the center of mass coordinates of the image volume assuming unit density as

\[
x_{cm} = \frac{1}{2V_{tot}} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} m(i,j)x(i)
\]

[3-18]

\[
y_{cm} = \frac{1}{2V_{tot}} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} m(i,j)y(j).
\]

[3-19]

\[
z_{cm} = 0
\]

[3-20]

where \( m(i,j) = 2\Delta x \Delta y z(i,j) \).

The first step is to calculate the center of mass for the intensity volume with unit density, \( x_{cm}, y_{cm}, z_{cm} \). Since the surface is symmetrical along the z axis, \( z_{cm} = 0 \). According to parallel axis theorem, which is proven directly from the definition of the intensity tensor, tells us that if we know the inertia tensor \( [I_{cm}] \) relative to a coordinate system with its origin at the center of mass, we may compute the inertia tensor relative to any other translated coordinate system as follows:
\[ [I_{n}] = [I_{cm}] + M_{tot}(v^Tv[I]-vv^T) \]

where \( v \) = the translation vector from the center-of-mass coordinate system to the new coordinate system, and \( M_{tot} \) is the total mass of the mass distribution. The inertia tensor for a uniform density rectangular block of a dimensions \( a \times b \times c \) and mass \( M_{tot} \) is well-known:

\[
[I_{\text{block}}(a,b,c,M_{\text{tot}})]
= \frac{M_{\text{tot}}}{12} \begin{vmatrix}
  b^2+c^2 & 0 & 0 \\
  0 & c^2+a^2 & 0 \\
  0 & 0 & a^2+b^2 \\
\end{vmatrix}
\]

The total inertia tensor about the center of mass of the image volume is then computed by summing the appropriate inertia tensors of the individual pixels:

\[
[I_{\text{tot}}] = \sum \sum [I_{\text{block}}(\Delta x, \Delta y, 2z(i,j), m(i,j))] \\
+ m(i,j)(v(i,j)^Tv(i,j)[I]-v(i,j)v(i,j)^T)
\]

where \( v(i,j) = (x_{cm}-x(i), y_{cm}-y(j), 0) \).

Then, the three dimensional moment of inertia properties are the moments about the center of mass.

\[
m_0 = I_{xx} = \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} (i_x(i,j)+m(i,j)(y[i]-y_{cm})^2)
\]
where \( i_{xx}(i,j), i_{yy}(i,j), i_{zz}(i,j) \) are the moment of the block on pixel \((i,j)\) along \(x, y, z\) axes. They are given by,

\[
i_{xx}(i,j) = \frac{m(i,j)}{12} \left( \frac{1}{n_y} \right)^2 + (2z(i,j))^2
\]

\[
i_{yy}(i,j) = \frac{m(i,j)}{12} \left( \frac{1}{n_x} \right)^2 + (2z(i,j))^2
\]

Also,

\[
m(i,j) = \frac{1}{n_x} \left( \frac{1}{n_y} \right) (2z(i,j))
\]

Since \( i_{zz} \) is relatively small, it is neglected.

Other features based upon the combinations of these moments are introduced [17]
\[ m_4 = |m_3| \] \[ m_5 = \frac{(m_0 + m_1 + m_2)}{3.0} \] \[ m_6 = \frac{m_0 + m_1}{2m_2} \] \[ m_7 = m_0m_1 + m_1m_2 + m_2m_0 \] \[ m_8 = \sqrt{\frac{m_0^2 + m_1^2}{2}} \] \[ m_9 = \sqrt{\frac{m_0^2 + m_1^2 + m_2^2}{3}} \] \[ m_{10} = \frac{m_0m_1}{m_2} \] \[ m_{11} = \left| \frac{m_3}{m_2} \right| \] \[ m_{12} = \frac{m_0 - m_1}{m_2} \]
The two-dimensional moment features are given by

\[ m_{13} = \frac{m_0 - m_1}{m_3} \]  \[ [3-40] \]

\[ m_{14} = \left| \frac{m_3}{m_0 - m_1} \right| \]  \[ [3-41] \]

\[ m_{15} = \left| m_0 - m_1 \right| \]  \[ [3-42] \]

\[ m_{16} = \frac{(m_0^2 + m_1^2)}{2m_0 m_1} \]  \[ [3-43] \]

\[ m_{17} = \frac{2m_2}{\sqrt{m_0^2 + m_1^2}} \]  \[ [3-44] \]

\[ m_{18} = \left| (m_0 - m_1)(m_1 - m_2)(m_2 - m_0) \right| \]  \[ [3-45] \]

\[ m_{19} = \left| \frac{m_0^2 - m_1^2}{m_2^2} \right| \]  \[ [3-46] \]

The two-dimensional moment features are given by

\[ m_{20} = I_x \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \frac{1}{n_x n_y} z(i,j)(y[i] - y_{cm})^2 \]  \[ [3-47] \]
The maximum and minimum moments along the principal axes are rotationally invariant, and are treated as effective features,

\[
m_{21} = I_y = \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \frac{1}{n_x n_y} z(i,j)(x[i] - x_{cm})^2
\]

[3-48]

\[
m_{22} = \frac{I_x + I_y + \sqrt{(I_x - I_y)^2 + 4I_{xy}^2}}{2}
\]

[3-49]

\[
m_{23} = \frac{I_x + I_y - \sqrt{(I_x - I_y)^2 + 4I_{xy}^2}}{2}
\]

[3-50]

3.2.4 The Faceted Surface Area Feature

This feature is an indication of the changes in the surfcae of the gray level surface [Besl]. It is similar to the standard deviation except that the spatial distribution of pixel gray levels is also taken into account. A window of 2x2 pixels is taken. The gray level pixel values of these are taken comprise the vertices of a nonplanar quadrilateral. The area of the quadrilateral is found by dividing it into two triangles and finding the area of these triangles. The sum of all quadrilateral formed by taking sets of adjacent pixels in the image surface, gives
the total faceted surface area of the image. For an \( nxm \) constant intensity image, the surface area is \((n-1)x(m-1)\). The ratio of the computed surface area by the constant intensity surface area gives the normalized feature. This feature is compared to the surface area computed using the first fundamental form of a surface, which is discussed in the next section.

For four adjacent pixels in the image: \( f(i,j), f(i+1,j), f(i,j+1), f(i+1,j+1) \), the four 3-D points \( \{p_1, p_2, p_3, p_4\} \) is given by the following vectors:

\[
p_1 = [i \ j \ f(i,j)]^T \tag{3-51}
\]
\[
p_2 = [i+1 \ j \ f(i+1,j)]^T \tag{3-52}
\]
\[
p_3 = [i \ j+1 \ f(i,j+1)]^T \tag{3-53}
\]
\[
p_4 = [i+1 \ j+1 \ f(i+1,j+1)]^T \tag{3-54}
\]

Now, consider the two triangles of the quadrilateral separately. Let \( p_1, p_2, p_3 \) determine the first triangle, then, two difference vectors are given by,

\[
d_1 = p_2 - p_1 \tag{3-55}
\]
\[
d_2 = p_3 - p_1. \tag{3-56}
\]

The unit normal to the triangle, which is used to define the direction, is computed as follows:
We use $d_1$ to define the $x$ direction:

$$n_x = \frac{d_x d_2}{\|d_x d_2\|} \quad [3-58]$$

The $y$ direction is determined by $n_x$ and $n_z$ as follows:

$$n_y = n_z \times n_x, \quad [3-59]$$

These normals now provide us with a rotation matrix with which we can rotate all three points into the $x$-$y$ plane,

$$[R] = [n_x | n_y | n_z]. \quad [3-60]$$

Since $d_1$ is used to define our $x$ direction, we can compute the area of the triangle by computing two vector inner products:

$$A_{123} = \frac{1}{2} (n_x^T d_1)(n_y^T d_2). \quad [3-61]$$

Similarly, area 2 can be computed using the same formulas applied to $p_4$, $p_5$, $p_2$. So, the area associated with the four adjacent pixels $f(i,j)$, $f(i+1,j)$, $f(i,j+1)$, $f(i+1,j+1)$ is given by,

$$A = \frac{1}{(n_x-1)(n_y-1)} \sum_{i=0}^{n_x-2} \sum_{j=0}^{n_y-2} a(i,j). \quad [3-62]$$

The value of $A=1$, when the gray level surface is flat and increases as the surface
contains more and more undulations. There are other formulas for computing the area of a triangle which use only the lengths of the triangle sides [42] and do not consider surface normals.

3.2.5 Surface Curvature Feature Set

In this feature set, the transitions of the intensities are of primary interest. The differential variations of the pixel intensities can be interpreted as the surface curvatures of the intensity shape. This concept arises from differential geometry which is introduced to describe the surface geometry conditions [44]. For the gray level surface, the shape function is given by

\[ S = S(x, y, z) \]  \hspace{2cm} [3-63]

and the height \( z \) of each pixel corresponds to its intensity

\[ z = f(x, y) \]  \hspace{2cm} [3-64]

Any point on the surface is given by its position vector

\[ R(x, y) = xf + yf + z(x, y)k \]  \hspace{2cm} [3-65]

To determine this function in the polynomial form so that the partial derivatives can be taken, the image is divided into adjacent windows of \( N \times N \) pixels. Then the local 3-D image intensity shape function is approximated with quadric polynomials
[43]. The picture function $f(x,y)$ is assumed to be smoothly varying function, but are not quadric polynomials. An arbitrarily close quadric approximation to such a function can be made within a window if the window size is small enough. In this section, the assumption is made that only one surface is seen within a window.

First consider the 1-D approximation problem. The function $f(x)$, $x=-(N-1)/2, \ldots, 0, 1, \ldots, (N-1)/2$, is to be approximated by a quadric polynomial to minimize the sum of the squared errors. The polynomial is $a_0 \varphi_0(x) + a_1 \varphi_1(x) + a_2 \varphi_2(x)$, with $\varphi_i(x)$ orthogonal polynomials over the interval and given by

\[
\varphi_0(x) = [1/N]^{1/2}
\]

\[
\varphi_1(x) = [3/M(M+1)(2M+1)]^{1/2}x
\]

\[
\varphi_2(x) = \alpha^{1/2}[x^2-M(M+1)/3]
\]

where $\alpha = (28M^2/45) + (14M^4/9) + (10M^2/9) + (2M^2/9) - (2M/30)$, and $M=(N-1)/2$.

An orthogonal basis for quadric polynomials over 2-D $N \times N$ windows is the set of functions given by

\[
\psi_{ij}(x,y) = \varphi_i(x) \varphi_j(y),
\]

$(i,j) \in \{(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)\}$.

Then, a quadric approximation to $f(x,y)$ is obtained as

\[
\sum a_{ij} \psi_{ij}(x,y)
\]
with \( a_y \) is found by substituting the values of \( f(x,y) \) at each pixel in the window to the function and solving a set of linear equations. In summary, least squares second polynomial approximations can be made to \( f(x,y) \) over a window.

### 3.2.6 Gaussian Curvature Related Features

The fundamental form of the surface [44] represents the element arc length of curves on the surface which pass through the point under consideration. It is given by,

\[
I = ds^2 = dR.dR = Edx^2 + 2Fdxdy + Gdy^2 \tag{3-71}
\]

where

\[
E = 1 + f_x^2 \tag{3-72}
\]

\[
F = f_x f_y \tag{3-73}
\]

\[
G = 1 + f_y^2 \tag{3-74}
\]

and

\[
g = EG-F^2 = 1+f_x^2+f_y^2 \tag{3-75}
\]

The second fundamental form of a surface is given by [44],

\[
II = Ldx^2 + 2Mdxdy + Ndy^2 \tag{3-76}
\]

where,
In a similar way to \( g \), \( b \) is defined as

\[
b = LN - M^2 = \frac{f_{xx}f_{yy} - f_{xy}^2}{g} \quad [3-80]
\]

The second fundamental form is from the expression for the curvatures of the curves which are on the surfaces that pass through the point in a given direction. These two fundamental forms play an important role in the surfaces differential geometry.

For a smooth surface, the normal curvature of the surface in direction \( dy/dx \) is given by

\[
k = \frac{1}{L} = \frac{L+2M\lambda+N\lambda^2}{E+2F\lambda+G\lambda^2} = k(\lambda) \quad [3-81]
\]

where \( \lambda = dy/dx \). The maximum and minimum curvatures arise from the extreme value pair of \( \lambda \).
\[ k_{\text{min}} = \frac{(EN+GL-2FM) - [(EN+GL-2FM)^2 - 4gb]^2}{2g} \]  
\[ k_{\text{max}} = \frac{(EN+GL-2FM) + [(EN-GL-2FM)^2 - 4gb]^2}{2g} \]

The values of \( k_{\text{min}} \) and \( k_{\text{max}} \) are rotationally invariant, which represent the principal local surface curvatures.

The Gaussian curvature is a very important factor to indicate the surface type, which is given \[44\] by

\[ K = k_{\text{min}} k_{\text{max}} = \frac{b}{g} \frac{f_{xx} f_{yy} - f_{xy}^2}{f_x^2 + f_y^2} \]

The sign of Gaussian curvature implies three kinds of surfaces:

1. \( K > 0 \): indicates the neighbouring local points around \((x,y)\) are on elliptical shape.
2. \( K < 0 \): indicates the neighbouring local points around \((x,y)\) are on hyperbolic (saddle) shape,
3. \( K = 0 \): indicates the neighbouring local points around \((x,y)\) are on parabolic or flat shape.

Another related factor is the mean curvature which is given by

\[ H = \frac{k_{\text{min}} + k_{\text{max}}}{2} = \frac{EN+GL-2FM}{2g} \]
Obviously, Gaussian curvature $K$ and mean curvature $H$ are also independent of the co-ordinate rotation, which are helpful in the surface topology measurement.

Hence, the first two features based on this concept are the average of Gaussian and mean curvatures

$$c_0 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} K(i,j)$$

$$c_1 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} H(i,j)$$

The next two features are the number of pixels with positive and negative Gaussian curvatures on the surface,

$$c_2 = \frac{\text{card}[(i,j): K(i,j) > 0]}{n_x n_y}$$

$$c_3 = \frac{\text{card}[(i,j): K(i,j) < 0]}{n_x n_y}$$

where $\text{card}$ is the cardinality function which indicates the number of elements in a given set.

From the results of Peet [45] and Besl [46], other related features for measuring surface topology are also employed.
\[ c_4 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} b(i,j) \] [3-90]

\[ c_5 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \sqrt{g(i,j)} \] [3-91]

\[ c_6 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} (f_{xx}^2 + 2f_{xy}^2 + f_{xy}) \] [3-92]

\[ c_7 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} k_{\text{min}}(i,j) \] [3-93]

\[ c_8 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} k_{\text{max}}(i,j) \] [3-94]

\[ c_9 = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \sqrt{M^2 - K} \] [3-95]

\[ c_{10} = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} |k_{\text{min}}| \] [3-96]
\[ c_{11} = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} |k_{\max}| \quad [3-97] \]

\[ c_{12} = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \frac{|k_{\max}| + |k_{\min}|}{2} \quad [3-98] \]

\[ c_{13} = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \frac{|k_{\max}| - |k_{\min}|}{2} \quad [3-99] \]

\[ c_{14} = \frac{1}{n_x n_y} \sum \sum \max_j[|k_{\max}|, |k_{\min}|] \quad [3-100] \]

\[ c_{15} = \frac{1}{n_x n_y} \sum \sum \min_j[|k_{\max}|, |k_{\min}|] \quad [3-101] \]

A total of 16 features are chosen in this set. In total, there are 49 features from the four sets chosen for this research.
CHAPTER 4
THEORETICAL BACKGROUND FOR FEATURE REDUCTION AND CLASSIFICATION

4.1 EXPERIMENTAL FEATURE REDUCTION

In this section three steps for features reduction can be performed. In the first, a stability test, features which give unstable results will be detected. In the next, a separability test, the features with poor capabilities for identifying the class of joints will be removed. In the last, a correlation analysis, the redundant features can be deleted. Finally, the stable, high performance, and uncorrelated features can be evaluated [47].

4.1.1 Stability test

The purpose of this stability test is to delete the features which have significantly unstable performance. Theoretically, an ideal feature will present identical values for the same joint even though it is tested many times. Based on this concept, the experiment is to test the stability of all the features with an ideal solder joint by repeatedly evaluating the feature. During the test, lighting condition and joint
position should be kept fixed in order to make the test close to an ideal case. Under these circumstances, any fluctuation of the feature value is assumed to be normally distributed. The mean and standard of these values are computed to evaluate the stabilities of features.

In order to compare the stability of all features, it is necessary to normalize the feature values and their distributions. For a normal distribution, the mean and standard deviation are given by

\[ 
\mu = \frac{1}{n} \sum_{i=1}^{n} x_i \quad [4-1] 
\]

\[ 
\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \mu)^2} \quad [4-2] 
\]

If a constant factor \( \alpha \) is used as a multiplier of the data belonging to the same distribution, it becomes
\[ \mu^* = \frac{1}{n} \sum_{i=1}^{n} \alpha x_i = \alpha \mu \]  

\[ \sigma^* = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\alpha x_i - \mu^*)^2} = |\alpha| \sigma \]  

where \( \alpha \) cannot be zero. It is obvious that the scaled mean \( (\mu^*) \) and deviation \( (\sigma^*) \) values are scaled uniformly. Hence, the characteristics of the distributions with different scaling are the same. During the feature value computation in a classification process, it is often necessary to scale the feature data by multiplying certain coefficients to reduce the rounding off errors of the computer. Based on this concept, the degrees of the deviations of distinct distributions can be compared by different scaling of the same range.

In this test, the data of each feature are scaled by dividing their mean values so that the normalized mean values of all the features after scaling are equal to unity. Then the stability of the features can be compared by their normalized deviation values \( \sigma^* \). From a statistical point of view, a normalized feature value distribution with \( \sigma^* = 0.1 \) indicates that there is a probability of one third that the feature value has more than 10% deviation in comparison with the mean value. As expected, the feature with this value \( \sigma^* = 0.1 \) will cause more percentage error for
different joint images. Therefore, instability is defined when the normalized deviation value $\sigma^* \geq 0.1 (10\%)$. In contrast, the features with $\sigma^* < 0.01 (1\%)$ are defined to be highly stable. However, among the $\sigma^*$ values for the unstable features are adjusted to 0.09. All the features are divided into three groups according to their normalized sample standard deviation values:

1. **Group I**, high stable features of which $\sigma^* < 0.01$,
2. **Group II**, stable features of which $0.01 < \sigma^* < 0.09$,
3. **Group III**, unstable features of which $\sigma^* > 0.09$.

It is apparent that all the parameters in group III are relatively unstable, so that the unstable features in group III are deleted. The reasons for the instabilities of the features are due to slight systematic fluctuation effects, including those of fluorescent light intensity, image sensor sensitivity, and signal processing of the system.

**4.1.2 Separability Analysis**

The classification capabilities of the retained features will be examined by
training experiment. The purpose of this experiment are to measure the performance of these features as well as to train the system to recognize them. In this experiment, large number of samples of each class of joint are categorized by manual inspection. Since the solder joints of each given class have similar geometrical shapes, it is assumed that the intensity properties of the joints in each class are normally (Gaussian) distributed. Based upon this concept, the feature values may be analyzed using basic statistical methods.

In order to measure the classification performance of a feature, a separability factor is defined. For one feature in two different classes, a separability factor is defined as the measurement ability of this feature to distinguish between two classes by the variance normalization distance between the class means. That is, for feature $x$ in class $i$ and $j$, the separability is given by

$$Sp_{ij}^x = \frac{|\mu_i^x - \mu_j^x|}{\sqrt{\sigma_i^x + \sigma_j^x}}$$

If the value of this factor is large, it indicates two distributions are significantly apart, that is, this feature has good ability to tell one from the other. Similarly, the overall performance of this feature $x$ can be evaluated by the average separability factor $Sp_{avg}^x$ of all the class pairs. In this feature set, say the features with $SP_{avg} < 0.95$ can be deleted.
4.1.3 Correlation Analysis

After separability analysis, the candidate features need correlation analysis to remove the redundant ones. To do this, a correlation factor can be used to measure the similarities among these features. The correlation factor $C_{xy}$ of features $x$ and $y$ in class $p$ is defined as

$$C_{xy}^p = \frac{1}{n_p} \frac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y} \quad [4-6]$$

The value $C_{xy}^p$ is between -1 and +1. If it is near zero, it indicates two features $x$ and $y$ are highly uncorrelated, i.e. they are independent of each other. If it is near either +1 or -1, it implies that these two features are highly correlated with each other. If the average correlation factor of two features is close to 1.0, the performance of these two features is very similar. Then either one can be deleted or combined in the order to reduce the dimensions of feature vector space.
4.2 CLASSIFICATION

Fuzzy set theory [25] is employed to develop a classification algorithm which can categorize underlying solder joint defects with maximum accuracy [36]. The advantage provided by fuzzy sets is that the degree of membership in a set can be specified, rather than just the binary 'to be or not to be' a member. This can be especially advantageous in a pattern recognition problem, where frequently objects are not clearly members of one class or another as the solder joint recognition problem. Using crisp techniques, an ambiguous object will be assigned to one class only under any situation. On the other hand, fuzzy techniques will specify to what degree the object belongs to each class, which is useful information in many cases [26] [27].

4.2.1 Classification algorithm

Let $X$ be a solder joint described by $n$ features, then $X = \{F_1, \ldots, F_n\}$. For each solder joint class $W_i$, let $U_i(F_j) \in [0,1]$ be a membership function for $j^{th}$ feature of $i^{th}$ class. Then, for each feature $F_j$, $U_i(F_j)$ measures the membership of $X$ in $W_i$ from the standpoint of a single feature value $F_j$. This partial evaluation of the
degree of membership is combined with the subjective weighing factors $g_{ij}$ which expresses the extent to which a viewpoint of feature $F_j$ is important in evaluating a solder joint $X$ from the class $W_i$. The following measures $E_i$ gives an evaluation of the degree to which a solder joint $X$ belongs to class $W_i$:

$$E_i = \sum_{j=1}^{n} U_{ij}(F_j)g_{ij},$$

where, $n$ and $m$ denote the number of features and the number of defect classes, respectively. The combination of these two number of defect classes, respectively. Final decision for the classification of solder joint defects is made in such a way that a solder joint belongs to class $W_i$ if and only if $E_i$ is greater than $E_k$ for all $k \neq i$.

### 4.2.2 Subjective weighing factors

The subjective weighing factors $g_{ij}$ is determined heuristically using the following constraints and provided into (1):

$$\sum_{j=1}^{n} g_{ij} = 1$$  \hspace{1cm} [4-8]

$$g_{ij} \in [0,1]$$  \hspace{1cm} [4-9]
These constraints are used to impose uniform weightings between each different class. The procedures for determining the weighing factors are:

1. determine initially each value of the factors by intuition
2. check classification results and whether there is any "bias" for being classified to specific classes,
3. adjust the "bias" by changing the weighing factors for the features and the classes involved in the "bias", and
4. continue the steps (2) and (3) until arrive successful results.

4.2.3 Fuzzy membership factors

In order to evaluate (1), we also need to define the membership functions, $U_y(F_j)$. Two standard functions called S-function and π-function are used to define the membership functions. They are mathematically defined as [28]:

$$S(q;a,b,c) = \begin{cases} 
0 & \text{, } q \leq a \\
2((q-a)/(c-a))^2 & \text{, } a < q \leq b \\
1-2((q-c)/(c-a))^2 & \text{, } b < q \leq c \\
1 & \text{, } c < q 
\end{cases}$$

[4-10]
\[ \pi(q;a,b,c) = S(q;a,b,c) , \quad q < c \]
\[ = 1 - S(q;a,b,2c-b,3c-2b) , \quad q \geq c \]  
\[ \text{[4-11]} \]

where \( b \) denotes a cross-over point at which \( S(b;a,b,c) = 0.5 \), and \( b = (a+c)/2 \). In \( \pi(q;a,b,c) \), \( c \) denotes the central point at which \( \pi = 1 \), and \( 2(c-b) \) is the bandwidth, i.e., the separation between the cross-over points of a \( \pi \)-function. (4-10) and (4-11) define the membership functions corresponding to fuzzy sets 'q is large' and 'q is c', respectively [28]. Using these functions, several (say 3) types of the membership functions are defined:

**type 1** ; 
\[ U_y(F_j;1) = S(F_j,a_y,b_y,c_y) \]  
\[ \text{[4-12]} \]

**type 2** ; 
\[ U_y(F_j;2) = 1 - S(F_j,a_y,b_y,c_y) \]  
\[ \text{[4-13]} \]

**type 3** ; 
\[ U_y(F_j;3) = \pi(F_j,a_y,b_y,c_y) \]  
\[ \text{[4-14]} \]

The type of a membership function is essentially determined by characteristics and features in each pattern class. As illustrated in Figure ?, type-1 (type-2) membership function is provided to work as a window for the features \( F_j \) which give more certainty of being categorized into the class \( W_i \) (\( W_j \)) for greater (smaller) value of the features. Type-3 membership function is provided for the features which give more certainty for being categorized into a class when there values are closer to their statistical mean.
After the type of membership function is determined, the parameters $a_y, b_y, \text{ and } c_y$ in (4-12, 4-13, 4-14) which determine the shape of the membership functions are adjusted as follows using statistical data provided by training phase. These data are denoted by $m_y$ and $s_y$ which indicate statistical mean and standard deviation, respectively, of a feature $F_j$ extracted from a known $ith$ class of solder joints.

\[ c_y = m_y \]  \hspace{1cm} [4-15] \\
\[ b_y = c_y \cdot z \cdot s_y \]  \hspace{1cm} [4-16] \\
\[ a_y = 2b_y - c_y \]  \hspace{1cm} [4-17]

where, $z$ denotes a fuzzifier which determined the fuzziness of a membership function. When $z=0$, the membership functions become crisp characteristic functions whose boundary is $c$ [28].

The rationale that the parameters in membership functions are adjusted using the statistical data as in (4-15, 4-16, 4-17) is originated from the following observations:

(1) A statistical mean features implies a "dc-offset" of the samples in the feature space. Therefore, it can be utilized to determine a reference point of the corresponding membership function. The reference point is $c_y$ and employed (4-15) to coincide the peak point of the membership function with the statistical mean.
(2) A standard deviation of features denotes how widely the samples spread in the feature space. Therefore, it can be used to determine the fuzziness of a membership function. The cross-over point \( b_y \) is selected to determine the fuzziness in the membership function by incorporating the standard deviation with the fuzzifier \( z \) using (4-16).

In conclusion, the membership function for \( jth \) feature of \( ith \) class, \( U_{ij}(F_j; t_{ij}, m_{ij}, s_{ij}) \) can be determined by three factors: One is the type \( (t_{ij}) \) of the function and the others are the statistical data \( (m_{ij}, s_{ij}) \) measured from training sets of solder joints whose classes are known as a priori information. The value of the membership function measured using a feature \( F_j \) is applied to (4-7) in conjunction with the subjective weighing factors. The measure \( E_i \) in (4-7) gives a combinational degree of membership of a solder joint belonging to a defect class \( W_i \).
CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

5.1 What has been done?

A number of existing methods have been improved. First, a new illumination has been used, which can define a solder joint better in terms of surface intensities. Second, a number of noise reduction algorithms are used to reduce the unwanted noise which interferes with the actual image without tampering with the actual image. Now, by using this kind of lighting a solder joint can be well defined in terms of surface intensities, so that a direct correlation can be found between the actual three dimensional image of the solder joint and the one developed by plotting the surface intensities over the two dimensional plane. Once this is understood, more and better features can be defined which will be useful in making a better decision while classifying a solder joint.

For calculating the various features the solder joint image has been divided into three sub-areas and preliminary results show that this approach improves upon the existing methodology of using two sub-areas. An algorithm has been developed based upon the binary image, to detect the center of the lead and then divide the
areas accordingly.

For calculating the Gaussian curvature of the solder image, a mask of 3x3 pixels is moved at each pixel and it is approximated using quadratic polynomials which give a better result than other approximations.

5.2 Suggestions for future work

An algorithm (based upon many existing ones) for finding the optimum value of the threshold should be used to detect the center of lead of the solder joint. It should be integrated with the CAD data supplied by the designer of the board to determine the location of the solder joint. Currently it is necessary to input both the threshold and the ideal location of the solder joint manually.

The number of the features should be reduced based upon the theoretical background discussed in chapter 4. This is necessary to make the program more reliable and faster. Also the decision making for classifying the solder joint can be based upon the fuzzy logic also discussed in chapter 4. The main reason why it should be done is that a defective solder joint often has several different defects, but the currently used statistical classification schemes assign every joint to one of
the classes, thus disclosing only one of its defects. Fuzzy logic allows to express the possibility to be an element of several classes at a time and is thus preferable.

Rules for an expert system can be developed using these features. This can also improve our results because it gives us a flexibility of adding and subtracting decision rules without the necessity to rewrite the whole program.

Some more features should be tried based on the Fast Fourier Transform, entropy of the image and fractals.
REFERENCES


2. S.L. Spitz, "For higher yield: test or inspect bare PCBs?" *Electronic Packaging & Production*, vol. 27, no. 2, pp. 96-98, February 1987.


This program was developed by JASWINDER SINGH CHADHA at Machine Vision Applications Laboratory (MVAL) at The University of Texas at El Paso.

This program is written in Metaware High-C and can be run from an Intelledex 386 system. This program captures an image of the solder joint and calculates various features based on the surface intensities of the solder joint.

Include the various source files which are later used in the program.

```c
#include "stdio.h"
#include "vwindow.h"
#include "vhardwar.h"
#include "conio.h"
#include "vipl.h"
#include "vip2.h"
#include "vgraphic.h"
#include "vmemory.h"
#include "vblob.h"
#include "ctype.h"
#include "vcolors.h"
#include "math.h"
#include "vconfig.h"

VBLOB bloba;
FILE *fn1;

int p[75][75];
float z[75][75];
float p_norm[75][75];
int nx, ny;
int m = 0;
float x[75], y[75];
```
float V_csw, V_ofr, V_tot;
float sigma2;
float x_cm, y_cm, z_cm;
float Ixx, Iyy, Izz, Ixy;
float Ix, Iy;
int top = 125, lft = 75, rgt = 150, bot = 200;
int p_max;
int count, rowa, rowb, cola, colb, row_centre, col_centre, colplot, centre;
int win_x, win_y;
int r;
int thresh = 30;

main()
{
    int i, j;
    int xx, yy;
    int r1, r2, r3, r4, rr;
    int e_lft, e_rgt, e_top, e_bot;
    vactual();
    printf("\n Adjust the board (press any key to continue...)
");
    getch();
    vsnap(0);
    vsnapwait();
    printf("The snap has been shot, press any key to retrieve it\n");
    getch();
    vdig(0);
    printf("Press any key to bring the box for selecting the image\n");
    getch();
    printf("Adjust the window (Press Q to continue...) \n");
    vbox(&top, &bot, &lft, &rgt);
    vplotellipse(top, bot, lft, rgt, RED, 0);
    vwindow("WIND1", top, lft, bot-top+1, rgt-lft+1, "*ISEG");
    nx = bot-top;
    ny = rgt-lft;
    printf("top = %d, bot = %d, lft = %d, rgt = %d \n", top, bot, lft, rgt);
    for (i = 0; i <= nx-1; i++)
    {
        for (j = 0; j <= ny-1; j++)
        {
            xx = i + top;
        }
    }
yy=j+lft;
p[i][j] = vgetpixel(xx,yy);
if(p[i][j] > 63) {printf("ERROR ");
printf(\"%d \n\", p[i][j]);}
}

win_x=15;win_y=15;
vwindow("WIND2",top+win_x,lft+win_y,bot-top-2*win_x,rgt-lft-2*win_y,"*ISEG");
vi thresh("WIND2", thresh);
vconan("*ISEG",0,0,thresh);
centre=((lft+rgt)/2);
vgetmaxblob(&bloba);
rowa=(int)bloba.mini;
rowb=(int)bloba.maxi;
cola=(int)bloba.minj;
colb=(int)bloba.maxj;
row_centre=(int) ((rowa+rowb)/2) ;
col_centre=(int) ((cola+colb)/2);
printf("area= %d\n",bloba.area);
printf("rowa= %d\n",rowa);
printf("rowb= %d \n",rowb);
printf("cola= %d \n",cola);
printf("colb= %d \n",colb);
printf("rowcentre= %d \n",row_centre);
printf("colcentre= %d \n",col_centre);
printf("centre of the blob is %d,%d\n",row_centre,col_centre);
if(bloba.area > 35) printf("Excessive solder OR lead not found\n");
if(bloba.area < 15) printf("Lead missing\n");
vplotcross(row_centre-5,row_centre+5,col_centre-5,col_centre+5,RED,SET);
rl=row_centre-top;
r2=col_centre-lft;
r3=bot-row_centre;
r4=rgt-col_centre;
r=r1;
if (r>r2) r=r2;
if (r>r3) r=r3;
if (r>r4) r=r4;
printf("r1 = %d, r2 = %d, r3 = %d, r4 = %d, r = %d\n",r1,r2,r3,r4,r);
rr=(int)r/5;
printf("rr = %d\n",rr);
e_top = row centre - rr;
e_bot = row centre + rr;
e_lft = col centre - rr;
e_rgt = col centre + rr;
getch();
vplotellipse(e_top, e_bot, e_lft, e_rgt, RED, 0);
e_top = row centre - (int)r/4;
e_bot = row centre + (int)r/4;
e_lft = col centre - (int)r/4;
e_rgt = col centre + (int)r/4;
getch();
vplotellipse(e_top, e_bot, e_lft, e_rgt, RED, 0);
e_top = row centre - (int)3*r/5;
e_bot = row centre + (int)3*r/5;
e_lft = col centre - (int)3*r/5;
e_rgt = col centre + (int)3*r/5;
getch();
vplotellipse(e_top, e_bot, e_lft, e_rgt, RED, 0);
/*****************************/

Call the functions to calculate various features

circle();
pixel_max();
size_norm();
norm_gray();
cent_subwin();
outer_frame();
norm_vol();
norm_var();
cent_mass();
moment();
face_area();
gauss();

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

This function draws a circle based on the Breshemham’s circle drawing algorithm

/*****************************/
circle()
{
    int i=0,j=0,n=0;
    int x[300],y[300],pp[75];
    int row[300],col1[300],col2[300];
    int var;
    
    pp[0]=3-2*r;
    x[0]=0;
    y[0]=r;
    for (; ;)
    {
        if(x[i] <=y[i]){
            if(pp[i]<0){
                pp[i+1]=pp[i]+4*x[i]+6;
                x[i+1]=x[i]+1;
                y[i+1]=y[i];
            }
            else {
                pp[i+1]=pp[i]+4*(x[i]-y[i])+10;
                x[i+1]=x[i]+1;
                y[i+1]=y[i]-1;
            }
            i++;
        }
        else{i--;
            break;}
    }
    for(j =0; j <=i; j ++){
        x[i+j]=y[i-j];
        y[i+j]=-1+x[i-j];
    }
    n=2*i;
    for(j =0; j <=n; j ++){
        x[n+j]=x[n-j];
        y[n+j]=-1*y[n-j];
    }
    printf("Press any key to continue\n");
    getch();
    m=2*n;
    for(j =0; j <=m; j ++){
        x[m+j]=-1*x[m-j];
    }
}
y[m+j]=y[m-j];
row[j]=row_centre+y[m-j];
col1[j]=col_centre+x[m-j];
col2[j]=col_centre+x[m+j];
vplotpixel(row[j],col1[j],RED,0);
vplotpixel(row[j],col2[j],RED,0);
for(i=0;i<=nx-1;i++)
{
    if(top+i < row[0] | | top+i > row[m])
    {
        for(j=0;j<=ny-1;j++) p[i][j] = 0;
    }
}
for(i=0;i<=m-1;i++)
{
    var=row[i]-top;
    for(j=0;j<col2[i]-lft;j++) { p[var][j]=0; }
    for(j=col1[i]-lft+1;j<=ny-1;j++) { p[var][j]=0; }
}
getehO;
for(i=0;i<=nx-1;i++)
{
    for (j=0;j<=ny-1;j++)
    {
        vplotpixel(top+i,lft+j,p[i][j],0);
    }
}
for(j=0;j<=m;j++) { vplotpixel(row[j],col1[j],RED,0);
vplotpixel(row[j],col2[j],RED,0); }
}

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

function to draw ellipse given the normalised
co-ordinates

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
```c
int i;
int ttop, bbott, lift, rrgt;
i = xx * m;
ttop = row_centre - (int)(i);
bbot = row_centre + (int)(i);
i = yy * m;
lift = col_centre - (int)(i);
rrgt = col_centre + (int)(i);
vplotellipse(ttop, bbott, lift, rrgt, RED, 0);
}

function to calculate the maximum pixel value
**********************************************
pixel_max()
{
int i, j;
p_max = p[0][0];
for (i = 0; i <= nx - 1; i++)
{
    for (j = 0; j <= ny - 1; j++)
        if (p_max < p[i][j]) p_max = p[i][j];
}
printf("p_max = %d\n", p_max);
}

function for size normalization
**********************************************
size_norm()
{
    int i, j;
    float test_x, test_y;
    for(i = 0; i <= m - 1; i++)
    {
        test_x = 1 + 2*i - m;
        x[i] = test_x / (2*m);
    }
    for(j = 0; j <= m - 1; j++)
    {
        test_y = 1 + 2*j - m;
        y[j] = test_y / (2*m);
    }
}

/*****************************/
/* function to calculate normalized maps of gray scales */
/*****************************/

norm_gray()
{
    int i, j;
    float test;
    test = (float) 1 / p_max;
    printf("test = %f
", test);
    for(i = 0; i <= nx - 1; i++)
    {
        for(j = 0; j <= ny - 1; j++)
        {
            z[i][j] = p[i][j] * test;
        }
    }
}

/*****************************/
/* function for calculating normalized volume */
/*****************************/
norm_vol()  
{
    int i,j;
    float F=0;
    for (i=0;i<=m-1;i++)
    {
        for (j=0;j<=m-1;j++)
        {
            F+=p[row_centre-top-r+i][col_centre-lft-r+j];
        }
    }
    printf("F = %f\n",F);
    V_tot=F/(m*m*p_max);
    printf("V_tot = %f\n",V_tot);
}

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

        function to calculate variance of normalized gray level
        surface about its mean value

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

norm_var()
{
    int i,j;
    float sum=0;
    for (i=0;i<=m-1;i++)
    {
        for (j=0;j<=m-1;j++)
        {
            sum+= (z[row_centre-top-r+i][col_centre-lft-r+j]-V_tot)*
                   (z[row_centre-top-r+i][col_centre-lft-r+j]-V_tot);
        }
    }
    sigma2=sum/(m*m-1);
    printf("Sigma2 = %f\n",sigma2);
}
/***************

function for calculating the central subwindow volume
***************

cent_subwin()
{
    int i,j;
    float F=0;
    float area;
    float x_mod, y_mod;
    draw_ellipse(0.1,0.1);
    for(i=0;i<=m-1;i++)
    {
        for(j=0;j<=m-1;j++)
        {
            x_mod = x[i];
            y_mod = y[j];
            if(x_mod<0) x_mod=-1*x_mod;
            if(y_mod<0) y_mod=-1*y_mod;
            if(x_mod<0.1&&y_mod<0.1) F=F+p[row centre-top-r+i][col centre-lft-r+j];
        }
    }
    printf("F_cent= %f\n",F);
    area = 0.2*0.2*m*m;
    V_csw=F/(area*p_max);
    printf("V_csw = %f\n",V_csw);
}

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

function for calculating the outer frame region volume
***************

outer_frame()
{
    int i,j;
    float F=0;
    float area;
    float x_mod,y_mod;
    draw_ellipse(0.2,0.2);
draw_ellipse(0.3,0.3);
for(i=0;i<=m-1;i++)
{
    for(j=0;j<=m-1;j++)
    {
        x_mod=x[i];
        y_mod=y[j];
        if(x_mod < 0) x_mod=-1*x_mod;
        if(y_mod < 0) y_mod=-1*y_mod;
        if(x_mod > 0.2&&y_mod > 0.2&&x_mod < 0.3&&y_mod < 0.3)
            F=F+p[row_centre-top-r+i][col_centre-lft-r+j];
    }
}
printf("F_out= %f\n", F);
area = (0.3*0.3-0.2*0.2)*m*m;
V_offr=F/(area*p_max);
printf("V_offr = %f\n", V_offr);

 /******************************************************************************/
 function to calculate the center of mass coordinates
 /******************************************************************************/

cent_mass()
{
    int i,j;
    float X=0,Y=0;
    float test1,test2;
    test1=(float)1/m;
    test2=(float)1/m;
    for(i=0;i<=m-1;i++)
    {
        for(j=0;j<=m-1;j++)
        {
            X=X+2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]*x[i];
            Y=Y+2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]*y[j];
        }
    }

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
x_cm = X/(2*V_tot);
y_cm = Y/(2*V_tot);
z_cm = 0;
printf("x_cm = \%f \n", x_cm);
printf("y_cm = \%f\n", y_cm);

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

function to calculate various moments about the center of mass
***************************************************************************/

moment()
{
    int i,j;
    float buf1 = 0, buf2 = 0, buf3 = 0, buf4 = 0, buf5 = 0, buf6 = 0;
    float test1, test2;
    float m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11, m12, m13, m14, m15;
    float m16, m17, m18, m19, m20, m21, m22, m23;
    test1 = (float) 1/m;
    test2 = (float) 1/m;
    for(i = 0; i <= m-1; i++)
    {
        for(j = 0; j <= m-1; j++)
        {
            buf1 = buf1 + (1/12.0)*((test2)*(test2) + (2*z[row_centre-top-r+i][col_centre-lft-r+j])* 
                            (2*z[row_centre-top-r+i][col_centre-lft-r+j]) + 
                            2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]* 
                            (y[j]-y_cm)*(y[j]-y_cm));
            buf2 = buf2 + (1/12.0)*((test1)*(test1) + 
                            (2*z[row_centre-top-r+i][col_centre-lft-r+j]) + 
                            2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]* 
                            (x[i]-x_cm)*(x[i]-x_cm));
            buf3 = buf3 + 2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]* 
                            ((x[i]-x_cm)*(x[i]-x_cm) + (y[j]-y_cm)*(y[j]-y_cm));
            buf4 = buf4 + 2*(test1)*(test2)*z[row_centre-top-r+i][col_centre-lft-r+j]* 

            */
(x[i]-x_cm)*(y[j]-y_cm);

buf5 = buf5 + test1*test2*z[row_centre-top-r+i][col_centre-lft-r+j]*
      (y[j]-y_cm)*(y[j]-y_cm);

buf6 = buf6 + test1*test2*z[row_centre-top-r+i][col_centre-lft-r+j]*
      (x[i]-x_cm)*(x[i]-x_cm);

}

Ixx = buf1;
Iyy = buf2;
Izz = buf3;
Ixy = buf4;
Ix = buf5;
Iy = buf6;
m0 = Ixx;
m1 = Iyy;
m2 = Izz;
m3 = Ixy;
m4 = m3;
if(m4 < 0) m4 = -1*m4;
m5 = (m0 + m1 + m2)/3.0;
m6 = (m0 + m1)/(2.0*m2);
m7 = m0*m1 + m1*m2 + m2*m0;
m8 = sqrt((m0*m0 + m1*m1)/2.0);
m9 = sqrt((m0*m0 + m1*m1 + m2*m2)/3.0);
m10 = m0*m1/m2;
m11 = m3/m2;
if(m11 < 0) m11 = -1*m11;
m12 = (m0-m1)/m2;
m13 = (m0-m1)/m3;
m14 = m3/(m0-m1);
if(m14 < 0) m14 = -1*m14;
m15 = m0-m1;
if(m15 < 0) m15 = -1*m15;
m16 = (m0*m0 + m1*m1)/(2*m0*m1);
m17 = 2*m2/(sqrt(m0*m0 + m1 + m1));
m18 = (m0-m1)*(m1-m2)*(m2-m0);
if(m18 < 0) m18 = -1*m18;
m19 = (m0*m0-m1*m1)/(m2*m2);
if(m19 < 0) m19 = -1*m19;
m20 = Ix;
m21 = Iy;
m22 = (lx + ly + sqrt((lx-ly)*(lx-ly) + 4*ly*ly))/2.0;
m23 = (lx + ly - sqrt((lx-ly)*(lx-ly) + 4*ly*ly))/2.0;
printf("m0 = %f ", m0);
printf("m1 = %f ", m1);
printf("m2 = %f ", m2);
printf("m3 = %f ", m3);
printf("m4 = %f \n", m4);
printf("m5 = %f ", m5);
printf("m6 = %f ", m6);
printf("m7 = %f ", m7);
printf("m8 = %f ", m8);
printf("m9 = %f \n", m9);
printf("m10 = %f ", m10);
printf("m11 = %f ", m11);
printf("m12 = %f ", m12);
printf("m13 = %f ", m13);
printf("m14 = %f \n", m14);
printf("m15 = %f ", m15);
printf("m16 = %f ", m16);
printf("m17 = %f ", m17);
printf("m18 = %f ", m18);
printf("m19 = %f \n", m19);
printf("m20 = %f ", m20);
printf("m21 = %f ", m21);
printf("m22 = %f ", m22);
printf("m23 = %f \n", m23);
}

// Function to calculate the Facted surface Area feature

function face_area()
{
    int i,j;
    float d1, d2, d3, d4, d5;
    float D1, D2, A123 = 0, A234 = 0;
    float buf1, buf2, buf3, buf4, buf5;
    float A, AA;
float test1, test2;
test1 = (float) 1/(m-1);
test2 = (float) 1/(m-1);
for (i = 0; i <= m-2; i++)
{
    for (j = 0; j <= m-2; j++)
    {
        buf1 = z[row_centre-top-r+i+1][col_centre-lft-r+j] -
                z[row_centre-top-r+i][col_centre-lft-r+j];
        d1 = sqrt(1 + buf1*buf1);
        buf2 = z[row_centre-top-r+i][col_centre-lft-r+j+1] -
                z[row_centre-top-r+i][col_centre-lft-r+j];
        d2 = sqrt(1 + buf2*buf2);
        buf3 = z[row_centre-top-r+i+1][col_centre-lft-r+j] -
                z[row_centre-top-r+i+1][col_centre-lft-r+j+1];
        d3 = sqrt(1 + buf3*buf3);
        buf4 = z[row_centre-top-r+i+1][col_centre-lft-r+j+1] -
                z[row_centre-top-r+i+1][col_centre-lft-r+j];
        d4 = sqrt(1 + buf4*buf4);
        buf5 = z[row_centre-top-r+i][col_centre-lft-r+j+1] -
                z[row_centre-top-r+i+1][col_centre-lft-r+j+1];
        d5 = sqrt(1 + buf5*buf5);
        D1 = (d1 + d2 + d3)/2.0;
        D2 = (d3 + d4 + d5)/2.0;
        A123 = sqrt(D1*(D1-d1)*(D1-d2)*(D1-d3));
        A234 = sqrt(D2*(D2-d3)*(D2-d4)*(D2-d5));
        AA = AA + A123 + A234;
    }
}
A = test1 * test2 * AA;
printf("FACETED AREA = %f \
", A);
}

Gaussian features
First the discrete data is approximated as quadric polynomials.
A window of 3x3 pixels is used

 gauss();
int i,j;
int NN=3;
float MM;
float a00,a10,a01,a20,a11,a02;
float A1,A2,A3,A4;
float BX1,BX2,BX3,BY1,BY2,BY3;
float BXX,BYY,BXY;
float E,F,G,g;
float L,M,N,b,H,K;
float Fx,Fy;
float pplus=0,pminus=0,Kavg=0,Havg=0,bavg=0,gavg=0;
float test1,test2;
float sqrtg;
test1=(float)1/(m-1);
test2=(float)1/(m-1);

MM=(NN-1)/2.0;
A1=-1.732;
A2=sqrt((3/NN)*(NN+1)*(2*NN+1));
A3=(28*NN*NN*NN*NN*NN/45)+(14*NN*NN*NN*NN/9)+(10*NN*NN/9)-2*(NN/30);
A4=NN*(NN+1)/3;
for(i=1;i<=m-1;i++)
{
  for(j=1;j<=m-1;j++)
  {

    a10=(z[row_centre-top-r+i-1][col_centre-lft-r+j-1]*(-1)+z[row_centre-top-r+i-1][col_centre-lft-r+j]*(-1)+z[row_centre-top-r+i-1][col_centre-lft-r+j+1]*(-1)+z[row_centre-top-r+i][col_centre-lft-r+j-1]*0+z[row_centre-top-r+i][col_centre-lft-r+j]*0+z[row_centre-top-r+i][col_centre-lft-r+j+1]*0+z[row_centre-top-r+i+1][col_centre-lft-r+j]*0+z[row_centre-top-r+i+1][col_centre-lft-r+j+1]*0)*A1*A1;
  
}
\[ z[\text{row_centre-top-r+i}][\text{col_centre-lft-r+j+1}] * 0 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j+1}] * 1 + A1*A2; \\
\]

\[ a01 = (z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j-1}] * (-1) + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j}] * 0 + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j+1}] * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * 0 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j+1}] * 1 + A1*A2; \\
\]

\[ a02 = (z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j-1}] * (1-A4) + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j}] * A4 + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j+1}] * (1-A4) + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * 1 + A2*A2; \\
\]

\[ a11 = (z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j-1}] * (-1) * (-1) + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j}] * (-1) * 0 + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j+1}] * (-1) * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * 0 * (-1) + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * 0 * 0 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j+1}] * 0 * 1 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * 1 * (-1) + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * 1 * 0 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j+1}] * 1 * 1 + A2*A2; \\
\]

\[ a01 = (z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j-1}] * (1-A4) + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j}] * A4 + \\
\[ z[\text{row_centre-top-r+i-1}][\text{col_centre-lft-r+j+1}] * (1-A4) + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j-1}] * (1-A4) + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j}] * A4 + \\
\[ z[\text{row_centre-top-r+i+1}][\text{col_centre-lft-r+j+1}] * (1-A4) + \\
\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Now that the quadric functions have been approximated at each pixel, find the first derivative and the second derivatives.

\[
\begin{align*}
Fx &= a10*A1*A2 + (2*a20*A1*A3)*x[i] + (a11*A2*A2)*y[j]; \\
Fy &= a01*A1*A2 + (2*a02*A1*A3)*y[j] + (a11*A2*A2)*x[i]; \\
E &= 1 + Fx*Fx; \\
G &= 1 + Fy*Fy; \\
F &= Fx*Fy; \\
g &= 1 + Fx*Fx + Fy*Fy; \\
sqrtg &= sqrt(g); \\
L &= 2*a20*A1*A3/sqrtg; \\
N &= 2*a02*A1*A3/sqrtg; \\
M &= a11*A2*A2/sqrtg; \\
b &= L*N - M*M; \\
H &= (E*N + G*L - 2*F*M)/(2*g); \\
K &= b/g; \\
\text{if}(K > 0) \text{ pplus} &= \text{pplus} + 1; \\
\text{if}(K < 0) \text{ pminus} &= \text{pminus} + 1; \\
Kavg &= Kavg + test1*test2*K; \\
Havg &= Havg + test1*test2*H; \\
bavg &= bavg + test1*test2*b; \\
gavg &= gavg + test1 + test2*sqrt(g);
\end{align*}
\]

printf("pplus = \%f\n",pplus);
printf("pminus = \%f\n",pminus);
printf("Kavg = \%f\n",Kavg);
printf("Havg = \%f\n",Havg);
printf("bavg = \%f\n",bavg);
printf("gavg = \%f\n",gavg);
CURRICULUM VITAE

Jaswinder Singh Chadha was born on February 13, 1968 in Dehradun, U.P., India. The elder of the two sons of Major Harsaran Singh Chadha and Dr. Tarvinder Kaur Chadha, he graduated from D.A.V. College, Chandigarh, India, in the spring of 1986 and entered Indian Institute of Technology (I.I.T.), New Delhi, India, in the fall of 1986. While pursuing a bachelor's degree in mechanical engineering, he worked with I.I.T. Delhi as a research associate during the summer of 1988, with H.M.T. Ltd., Pinjore, Haryana, India, a machine tools manufacturing company, as a summer trainee engineer in the summer of 1989, and later worked with Tata Electric & Locomotive Co. (TELCO), Pune, India, an automobile manufacturer as a full time engineer after receiving his bachelor's of technology degree from Indian Institute of Technology, New Delhi, India, in spring of 1990. He was captain of the I.I.T. Delhi's basketball team and member of the institute's soccer and athletic's team. In the fall of 1990, he entered the Graduate School at The University of Texas at El Paso. He worked as a teaching assistant in fall'90 and spring'91 and as a research assistant in the Machine Vision Applications Laboratory at UTEP in fall'91. He is a member of ALPHA PI MU, Institute of Industrial Engineers and Society of Manufacturing Engineers.

Permanent address: C-3 , G.N.D.University
Amritsar, Punjab 143005, India.

This thesis was typed by Jaswinder Singh Chadha.