5-31-1991

Automatic infrared and visual inspection of solder joints on PCBs

Jiang Huang
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Title of Thesis: Automatic Infrared and Visual Inspection of Solder Joints on PCBs

Author: Jiang Huang
Master of Science in
Electrical Engineering, 1990

Thesis directed by: Dr. Y. Q. Shi
Assistant Professor of Electrical and Computer Engineering

Department : Electrical and Computer Engineering

This thesis reports the research results of automatic inspection of solder joints on printed circuit boards (PCBs). The previous work on this regard has been advanced significantly in the following three aspects. With the application of some microlens and the principle of geometric optics solder joints of real size are inspected in this work instead of larger simulation solder joints in the previous work. A new set of features is formulated and used for infrared inspection. Consequently, better results have been achieved in this work.

Reasonably good results demonstrate the capability of infrared inspection of solder joints on PCBs. Once again, the fusion of infrared and visual imaging techniques for the inspection is proved to bring out better results than that obtained by using each technique alone.

AUTOMATIC INFRARED AND VISUAL INSPECTION OF SOLDER JOINTS ON PCBS

by

Jiang Huang

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1991

APPROVAL SHEET

Title of Thesis: AUTOMATIC INFRARED AND VISUAL INSPECTION OF SOLDER JOINTS ON PCBS

Candidate: Jiang Huang
Master of Science in Electrical Engineering, 1991

Thesis and Abstract Approved by the Examining Committee:

Dr. Y. Q. Shi, Thesis Advisor — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. N. Ansari — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. E. Hou — Date
Assistant Professor
Department of Electrical and Computer Engineering

Dr. C. Q. Shu — Date
Research Associate
Department of Electrical and Computer Engineering

New Jersey Institute of Technology, Newark, New Jersey.

VITA

Jiang Huang

Date of Birth

Place of Birth

Education

1989-1991	New Jersey Institute of Technology, Newark, New Jersey, U.S.A. MSEE
1982-1986	Guang Zhou College, Guang Zhou, P.R.C. BS
1978-1980	Guang Zhou Mechanical & Electrical Institute, Guang Zhou, P.R.C.

Dedicated to

My Wife and My Parents

ACKNOWLEDGMENT

Many people have assisted me during the course of this work. I am very grateful to my thesis advisor Dr. Y. Q. Shi whose inspiration and guidance benefitted me significantly. Without his support this work could not have been completed.

I would like to express my appreciation to Dr. C. Q. Shu, manager of the Electronic Imaging Laboratory at Electrical and Computer Engineering Department, NJIT. His help in regard of Datacube Image System, both the software application and hardware configuration, and his comments on the research work are valuable.

And I would like to express my appreciation to Dr. W. F. Kosonocky for his leadership as NJIT Foundation Chair for Optoelectronics and Solid-State Circuits.

I would also like to express my appreciation to many of my laboratory colleagues, in Room 410 of Faculty Hall, who gave their assistance in many ways, and to my family who have supported my study at NJIT.

Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose

This thesis is a part of a long term project on the application of electronic imaging techniques, which is one of the major tasks undertaken at the Electronic Imaging Laboratory, led by Dr. Kosonocky, at Electrical and Computer Engineering Department, NJIT.

The research work described in this thesis is automatic inspection of solder joints on printed circuit boards (PCBs), using a small scale image processing system and unsupervised statistical pattern recognition technique. The target is to classify solder joints on the PCB into distinct classes according to their different quality status. The subject falls into machine vision area.

Machine vision, technically composed of image processing and pattern recognition, is one of the fastest growing areas to which many scientists and engineers majoring in computer and signal processing devoted their great efforts in the past decade. Machine vision is one of the most important parts of artificial intelligence. Any significant improvement in this area will affect the performance of the next generation of computers. On the other hand, the application of machine vision in industry has significant economic profits.

Machine vision systems – computer systems that can see and recognize – have been applied in virtually every branch of manufacturing, in which most of the applications are in inspection and quality control. Automatic solder joint inspection is only one of the many applications in this regard. System development and research papers in this category are only a small part in the survey of automated visual inspection. The work included in this thesis is a part of new efforts in this attractive developing area.

1.2 A Description of the Previous Work

A fellow student, Mr. K. K. Zhou, did this research and finished his master thesis [14] in May 1990. In what follows a brief description of his work is given for the reference.

1.2.1 A small scale image processing system

A small scale image processing system had been set up. Zhou applied this system to

- 1. Acquire digital image frames from pictures taken by a camera;
- 2. Display digital image;
- 3. Process and store the digital image in the host computer;
- 4. Implement solder joint inspection using the Window Tool on the SUN workstation.

1.2.2 Visual light inspection experiment

Zhou [14] began his experiment with a printed circuit board. Solder joints were hand-made on the PCB. The joints on the PCB were of real size for dual

in-line ICS mounting. Four fundamental types of solder joints were made. They were the excess solder, the normal, the insufficient solder and the no-solder ones. A CCD visual light camera was used. The original lens had a focal length of 16 mm and a minimum photo distance of 0.3 m. Then a microlens having a focus of 75 mm and a minimum photo distance of 1 m was used to take pictures. The size of solder joint subimage on the picture was 10 pixels in X axis direction and 14 pixels in Y axis direction. By solder joint subimage, we mean a portion of the image entirely containing the joint. His best experiment results were that the good/bad rate was 78.7% or 85 in total 108 solder joints, the correct classified rate was 62.96% or 68 in total 108 solder joints. The good/bad rate means the rate of correctly classifying solder joints to the following two categories: the "good" and the "bad." The correct classified rate means the rate of correctly classifying solder joints to one of the four categories mentioned above.

1.2.3 Infrared inspection of simulation solder joints experiment

Zhou [14] tried another approach to realize automatic inspection of the solder joints on PCBs. An infrared camera which had a focal length of 1.5m was utilized. Because the original solder joints were too small to be inspected correctly, he made an enlarged simulation of solder joints on a board for experimental purpose. There were 20 joints, each in diameter of 15mm, in four different classes mentioned above. Each class had 5 joints.

The best results achieved were that the good/bad inspection rate was 85% and the correct classified rate 80% with solder joint subimage size of 34×46 pixels.

1.2.4 Fusion inspection experiment

Since the result was still not satisfied, the fusion inspection was conducted. A visual light image and an infrared image of the same simulation solder joints were taken and processed respectively. The individual results were then used with some appropriate weight vector to obtain better results.

The best result obtained by visual inspection with respect to the simulation solder joints was that the good/bad rate was 95% and the correct classified rate 90% with the solder joint size of 46×48. The best results in infrared inspection were that the good/bad rate was 85% and the correct classified rate 80% with the joint subimage size of 36×42. The best results of fusion inspection with respect to the simulation solder joints were 100% in both the good/bad inspection rate and the correct classified rate [14].

Chapter 2

System Construction

2.1 Hardware Construction

The host computer of the system in the Electronic Imaging Laboratory is a SUN 3/260 workstation. The MaxVideo modules, made by Datacube, Inc., are used to construct a special image processor to perform some function like video data acquisition and image displaying. Each module presents some individual functions of image processing. They can work compatibly with each other to construct different systems having its own purpose and performance, also for particular application requirement and investing scope.

In the Datacube real time image processing system, there are different operating functions for every MaxVideo module. There are some connector ports, as shown in Figure 2.1 which is a top view. Every MaxVideo module board has 14 pins. The connection between boards can be changed by the user. The Datacube image processing system provides a flexible function selector (hardware setting and the Hardware Access Tool) to fit the different operating requirement.

Only three MaxVideo modules are used in this work. They are MAX-SCAN, ROI-STORE, and MAX-GRAPH. The MaxVideo modules group uses

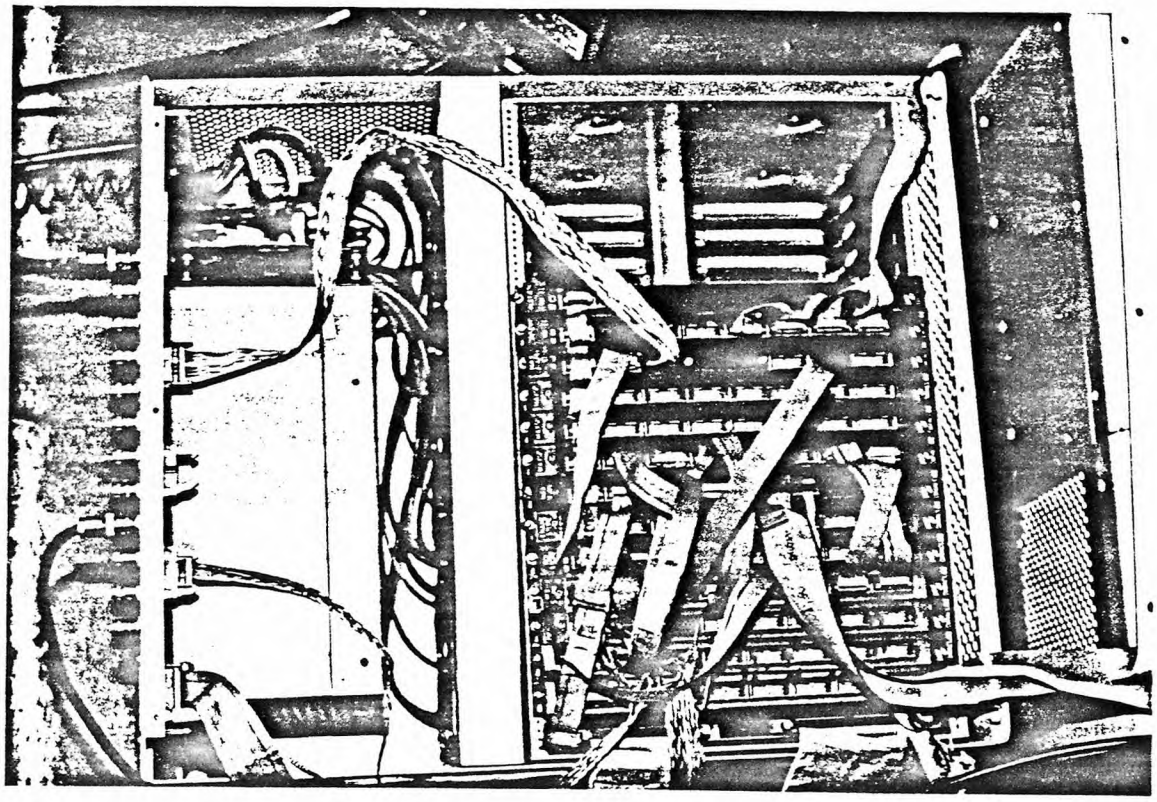

Figure 2.1: A Datacube Image System

an open modular structure to provide the real time image manipulation, acquisition, storage, processing and display functions. These are briefly described below.

(1) MAX-SCAN is a complex, programmable asynchronous input module which can acquire either analog or digital input data with a variety of resolution and frame rate. We only use its analog signal I/O port (P12) which can be operated at data rates up to 10 MHz. MAX-SCAN can digitalize the output of low resolution, very fast frame rate sensors, and also very high resolution slow frame rate sensors. It creates a 10 MHz stream of scan sequential digital video data, suitable for connection to a wide variety of the MaxVedio modules. The digitized video signal is downloaded from the MAXbus Digital Output (P5) to the Digital Input Port (P9) of ROI-STORE in our construction.

(2) ROI-STORE is a region-of-interest (ROI) memory and a MaxVideo compatible framestore module which supports user programmable video resolutions and processes the regions-of-interest image. ROI-STORE accommodates video signals from RS-170 and non-RS-170 MAXbus sources and transfers data in and out of its MAXbus ports at the standard MAXbus rates (10 MHz) and 7.16 MHz. ROI-STORE module can store images, or create regions of images in its memory, with any horizontal or vertical resolution up to 4K. The P4 of ROI-ROI-STORE connected with P4 of MAX-SCAN is for the ROI Timing.

(3) MAX-GRAPH is a graphics generation module. It displays graphics in interlaced RS-170 format as well as in non-interlaced format. It supports mouse and light pen interfaces and RGB color output. It is compatible with

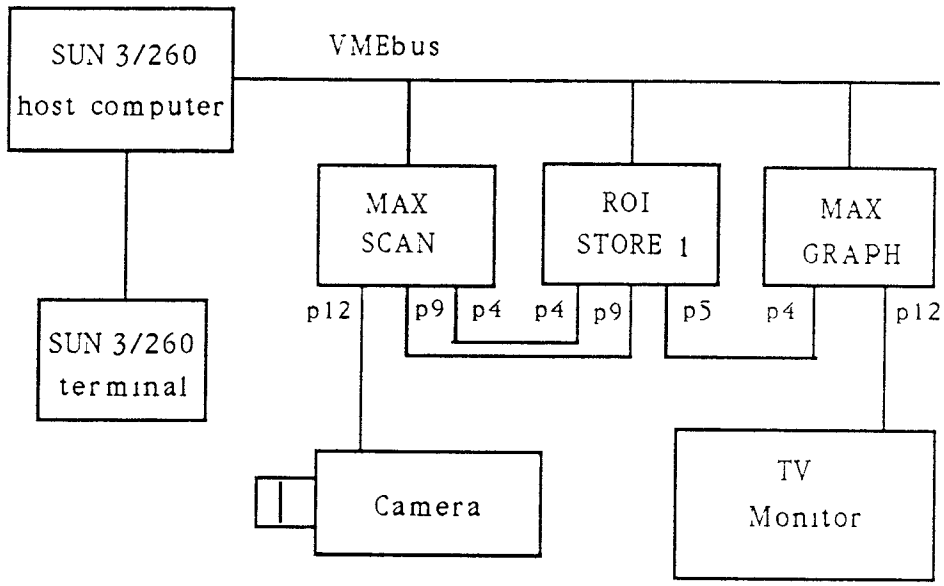

Figure 2.2: Construction of Image Processing System

10 MHz streams of digital video data (for 512×512 systems). The signal is from the Digital Output (P5) of ROI-STORE to the Digital Input (P4) of MAX-GRAPH. And then, the signal is output from the Analog Output (P12) of MAX-GRAPH to a TV monitor.

In the small system, the VMEbus is used as the data channel between MaxVideo modules and the host computer. B bus adapter and cable are connected to the VMEbus of host computer SUN 3/260. The hardware construction is shown in Figure 2.2.

The functions of the Datacube real time image processing system we applied are:

- Show thermal images from the IR-CCD camera on a TV monitor.

- Freeze what the camera is viewing.
- Store and load the digital image into and out of the host computer (SUN Workstation).

Driving program for individual module offered by Datacube can be mounted into the kernel of operating system of SUN workstation. The operating of modules could be made into standard library procedure. What user should do is programming in language C and call for these library procedures if certain module operating is wanted.

As seeing from above the system setting up includes the connecting of hardware system, testing, debugging and programming for the user's subroutine. The final target is making the system easy to use for any professors or graduate students who want to use it.

2.2 Pattern Recognition Method for Solder Joint Inspection

2.2.1 Basic concepts of statistical pattern recognition

Statistical pattern recognition method is utilized in this work for automatic solder joints inspection. A computer program is written in C to specifically implement the pattern recognition.

Recognition is regarded as a basic attribute of human being. A pattern is the description of an object. Pattern recognition means classifying a class of objects from any other classes by computer systems instead of human beings. In statistical pattern recognition a pattern is described by a series of mathematics quantities, termed as feature vector, or simply, features. An object or a pattern could be distinguished from others only if it has distinct features

from the others. In this thesis only the photographed object is concerned . In other words, the visual light and infrared phonographs of solder joints on PCB are needed to be recognized.

Only the concept of pattern classification by distance functions, the simplest and most intuitive approach to the statistical pattern classification, is discussed here briefly.

Prior to a more quantitative discussing some notations are given next.

Symbol Explanation

$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{pmatrix}$	vector; using lowercase bold
x_j	subscripted vector.
$\|\|x\|\| = \sqrt{\sum_{j=1}^{n} x_j^2}$	Euclidean norm or magnitude of vector x.
n	dimensionality of pattern vectors.
E^n	n-dimensional Euclidean space
w_i	i th pattern class
m_i	mean vector of class w_i
I	matrix; using uppercase bold

Consider the simplest situation where only two classes, w_i and w_j are concerned. Furthermore, each is described by a 2-dimensional feature vector. These two feature vectors can be positioned in a 2-dimensional Euclidean space, i.e., a planed as shown in Figure 2.3. Every point with the two contours is a feature vector, which is computed from certain object belonging to

w_i or w_j. Assuming that

$$o_1, o_2, ...o_s \in w_i$$

$$p_1, p_2, ...p_t \in w_j$$

The mean vectors $m_i = \{m_{1i}, m_{2i}\}$ and $m_j = \{m_{1j}, m_{2j}\}$ can be computed as

$$m_{1i} = \frac{1}{s}\sum_{i=1}^{s} o_{k1}$$

$$m_{2i} = \frac{1}{s}\sum_{i=1}^{s} o_{k2}$$

$$m_{3i} = \frac{1}{t}\sum_{i=1}^{t} p_{k1}$$

$$m_{4i} = \frac{1}{t}\sum_{i=1}^{t} p_{k2}$$

A distance between two vectors $a = \{a_1, a_2\}$ and $b = \{b1, b2\}$ is defined as

$$||a - b|| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

Now consider an unknown object X that needs to be classified. Its features can be computed, then it is positioned on the plane, as shown in Figure 2.3, noted as x. In Figure 2.3 the distance from x to w_i is obviously less than that of x to w_j. x is therefore classified to w_i according to the minimum-distance classification algorithm.

Now the space in Figure 2.3 can be imagined as an n-dimensional Euclidean space E^n, every point is an n-dimensional vector. Same algorithm can be used as a classifier. In the space E^n, distance between two vectors a and b has been defined previously in the table about symbols.

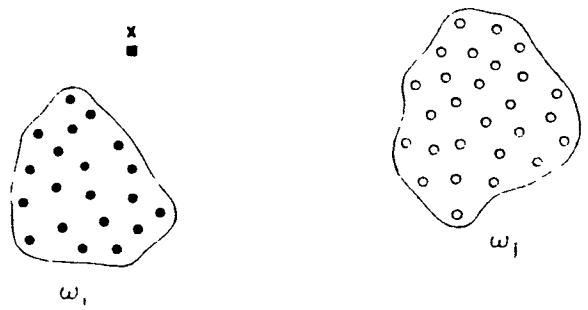

Figure 2.3: Patterns in Euclidean Space

If there are more than two classes in the E^n space, the same algorithm is still working as a classifier. That is the statistical pattern recognition algorithm used in this automatic solder joint inspection system.

2.2.2 Feature extraction for the solder joint inspection

The features used to classify the solder joint subimages are mathematically defined next:

1. Normalization: It is assumed that discrete solder joint subimage datum are digitized gray values of $f(i,j)$ in the rectangular form of $n_x \times n_y$, where $i = 1,2...n_x$, $j = 1,2,...n_y$. Define

$$f_{max} = max_{i,j} f(i,j)$$

The 2-D array data are normalized into the interval $\{0,1\}$, denoted as

$$z(i,j) = \frac{f(i,j)}{f_{max}}$$

for all possible (i,j).

2. Total volume or mean normalized gray level is defined as:

$$V_{tot} = \frac{1}{n_x n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} z(i,j)$$

This is the first feature used, i.e.,

$$Feature_1 = V_{tot}$$

3. Variance of normalized gray level defined below is the second feature:

$$\sigma^2 = \frac{1}{n_x n_y - 1} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} [z(i,j) - V_{tot}]^2$$

$$Feature_2 = \sigma^2$$

4. Central subwindow volume:

$$V_{csw} = \frac{1}{n_x n_y} \sum_{i \in x_{0\,3}} \sum_{j \in y_{0\,3}} z(i,j)$$

where $x_{0\,3}$ is the set of those index i, such that $|\frac{i-\frac{n_x}{2}}{n_x}| \leq 0.3$, $y_{0\,3}$ is similarly defined for j. And we define

$$Feature_3 = V_{csw}$$

5. Outer frame region volume:

$$V_{ofr} = \frac{1}{n_x n_y} \sum_{i \notin x_{0\,7}} \sum_{j \notin y_{0\,7}} z(i,j)$$

where $x_{0.7}$ and $y_{0.7}$ are defined similarly in the same way as that used for $x_{0.3}$ and $y_{0.3}$ except 0.3 is replaced by 0.7. And

$$Feature_4 = V_{ofr}$$

6. Inertia features: I_{aa}, I_{bb}, I_{cc}, where I_{cc} is the moment of inertia around the z axis. Note that $x(i) = i - \frac{1}{n_x}$ and $y(j) = j - \frac{1}{n_y}$. The computational procedure is

$$\Delta x = \frac{1}{n_x}, \quad \Delta y = \frac{1}{n_y}$$

$$m(i,j) = \frac{2z(i,j)}{n_x n_y} = 2\Delta x \Delta y z(i,j)$$

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

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

$$v'(i,j) = (x_{cm} - x(i), y_{cm} - y(j), 0)$$

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

$$[I_{tot}] = \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \{[I_{block}(\Delta x, \Delta y, 2z(i,j), m(i,j))] + m(i,j)\left[v(i,j)^T v(i,j)\,[1] - v(i,j)v(i,j)^T\right]\}$$

After $[I_{tot}]$ is diagonalized, I_{aa}, I_{bb}, I_{cc} are the elements on the main diagonal line, and $I_{ratio} = \frac{I_{aa}+I_{bb}}{2I_{cc}}$. Define

$$Feature_5 = I_{aa}$$

$$Feature_6 = I_{bb}$$

$$Feature_7 = I_{ratio}$$

We use seven features to classify the solder joints subimages. They work very well in classifying the visual light solder joints subimages, but in infrared image, we found some of these features did not work so well. The reason is infrared image has its own characteristics. The main difficulty we met was the temperature in the PCB was not well-distributed caused by the different quantity of the integrated chip holder mounted on the other side of PCB. So, we should find some other classifying features. We did find some other special features for the infrared image processing which will be discussed in Chapter 4 in detail.

Chapter 3

Visual Light Inspection Experiment

A raw printed circuit board is soldered manually. The joints on the PCB are real size for dual in-line ICs mounting. Four fundamental types of solder joints are made. They are excess solder, normal, less solder and no solder, which are shown on Figure 3.1.

The real size solder joints are so small that is very difficult for taking pictures of solder joints suitable for automatic inspection. The approach we adopt is utilizing a microlens to enlarge the original solder joints more than 6 times. The distance we made photographing is about 30 cm or less. The size of solder joint subimage on the picture is more than 18 pixels in X axis direction and more than 18 pixels in Y axis direction now. But there are some other problems for automatic inspection. They are the rosin and the illuminating light. So, before taking a picture of the PCB, we used some alcohol to remove the rosin on the surface of the solder joints. And a circular fluorescent lamp is used for illuminating instead of projector illumination to achieve better quality of pictures. Set the camera in front of the PCB, focus it well and take pictures, and store the images into the host computer by the

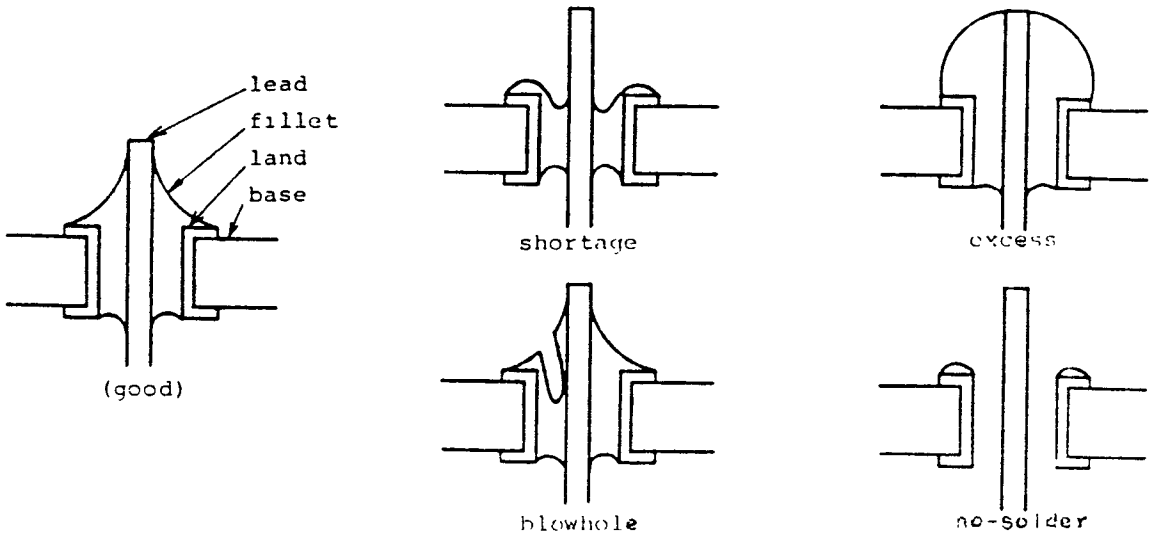

Figure 3.1: Solder Joint Defects[11]

software named "sample."

3.1 Experiment and Results

After we obtained images stored in host computer, we can run the program "pcb1" to process them. We have tried to process solder joint subimages with size varying from 16×16 to 46×46, but the output does not show much difference. So we choose the size about 22×20 in our work. The image in our work is shown on Figure 3.2.

When a solder joint is being classified, its different feature is multiplied by a different weight. A set of weights is called a weight vector. Various different weight vectors are tried to obtain the best classification result. These weight vector and the associated results are shown in Figure 3.3. There, Box X

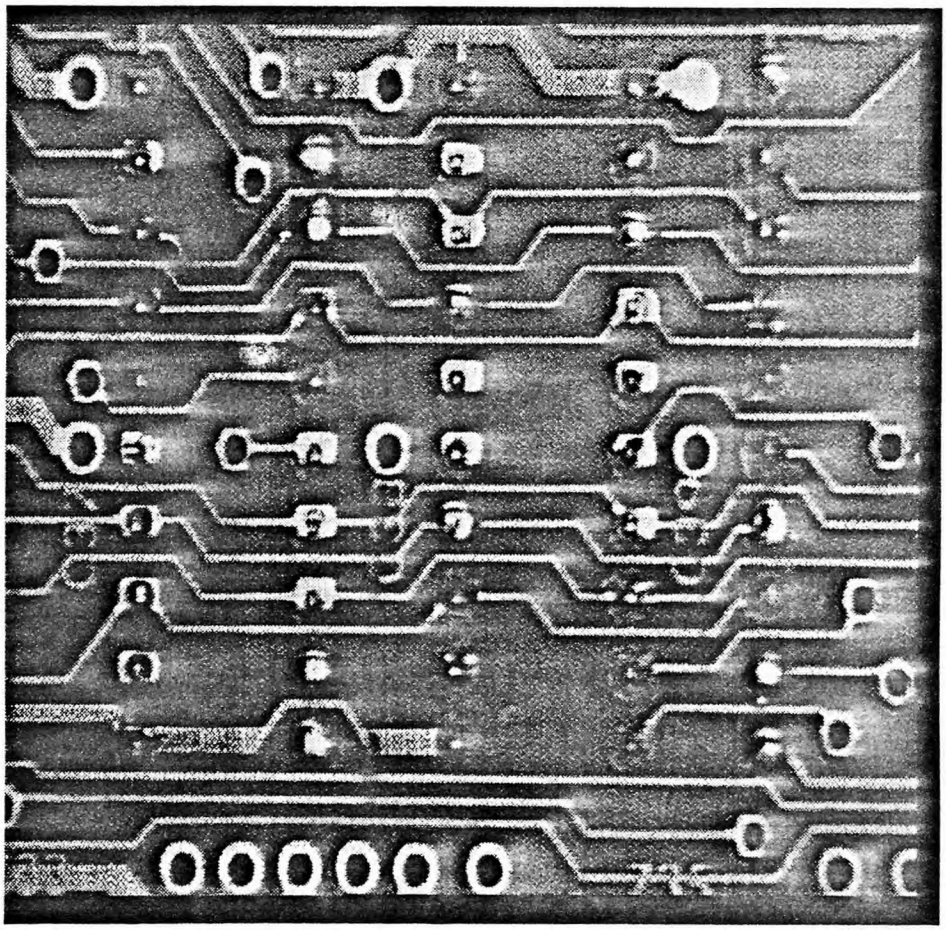

Figure 3.2: The Visual Light Image

——Total joints is 50——						
Box X	Box Y	Weight	CC	G/B	%CC	%G/B
20	16	1	49	49	98%	98%
20	18	1	48	49	96%	99%
20	20	1	48	50	96%	100%
22	20	1	48	50	96%	100%
24	20	1	45	47	90%	96%
24	22	1	47	50	96%	100%
22	22	1	48	50	98%	100%
22	22	2	49	50	98%	100%
22	22	3	47	48	94%	96%
22	22	4	46	49	92%	98%
22	22	5	46	49	92%	98%
22	22	6	48	50	96%	100%

Figure 3.3: The Results of the Visual Light Inspection

and Box Y indicate the size of the subimages of the joints being processed. The number in the column of Weight means which file among the six files, i.e., Weight(n).vec ($n = 1, 2, ..., 6$), is used for the experiment associated with that row. From Appendix D, it is clear that each file among Weight(n).vec ($n = 1, 2, ..., 6$) represents a different weight vector. CC means the correct classified rate. G/B means the good/bad rate. The best results are that, the correct classified rate is 98%, in other words, 49 among total 50 solder joints are correctly classified, and the good/bad rate is 100%.

There are some reasons that we can not reach the correct classified rate of 100%. Firstly, the solder joints manually made in our work look very different from those joints made by soldering machine. The uniformity even for those supposed to be good solder joints is poor. Some solder joints are hard to be

determined which class they should be classified to even by a human inspector. Secondly, the surface of the solder joints is covered by rosin. Though we have used alcohol to try to remove it, the remaining rosin still makes the image worse. Finally, the reflection of the fluorescent lamp also affect the quality of images of joints.

Chapter 4

Infrared Image Inspection

To overcome the effect caused by the remaining rosin on the surface of the joints and the reflection of the light, we use another approach to realize the automatic inspection of the solder joints. That is infrared inspection.

4.1 Introduction of the IR-CCD Camera

The IR-CCD camera used in this research work was built by graduate students of the E.E. Department, NJIT, at the David Sarnoff Research Center (DSRC), in Princeton, N.J., under the direction of Professor Walter F. Kosonocky, NJIT Foundation Chairman for Optoelectronics and Solid State Circuit [10]. There are two main sections in this IR-CCD camera, the camera head section (lens and sensor) and the electronic box (A/D converter for 12-bit digital output port and D/A converter for RS-170 display). They are shown in Figure 4.1.

A 320×244-PtSi IR CCD camera is utilized for the automatic inspection. It operates with a 100-mm ff/1.4 lens, f/1.3 cold shield, and a long-pass filter having a cut-off wavelength of 3.4 μm. The characteristics of the IR-CCD camera are summarized below:

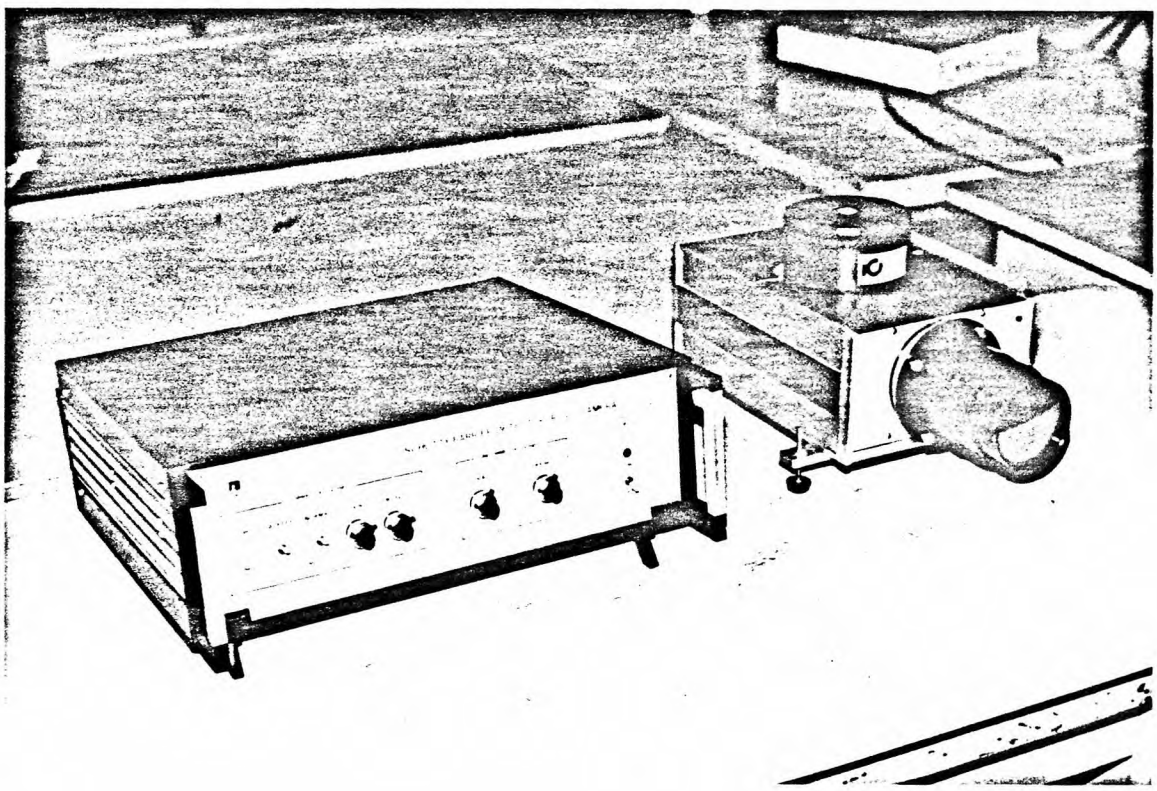

Figure 4.1: The IR-CCD Camera

*** FPA structure**

-Number of Pixels :	320 (H) x 244 (V)
-Pixel Size :	40 um (H) x 40 um (V)
-Chip Size :	40 um (H) x 40 um (V)
-Imager Architecture :	Monolithic Silicon ITCCD
-IR Sensor Element :	PtSi Schottky-barrier Detector

*** Camera Electronics**

-Frame Rate :	30 frames/s with 2 vertically interlaced 320 x 122 fields/frame
-Hor. Line Readout :	63.5 us/HL with imager signal on alternate horizontal TV lines
-Uniformity Correction :	one-point off-set-type averaging up to 16 reference frames

*** Optics**

-Lens :	100 mm , f/1.4
-Cold Shield :	f/1.3
-IR Filter :	3.4 μm long pass

*** System Performance**

-Operating Temperature :	77 K
-Imager Responsivity :	
—at FPA output :	57.9 mV/K
—at camera output :	34.5 mV/K
-NEΔT at FPA output :	0.034 K

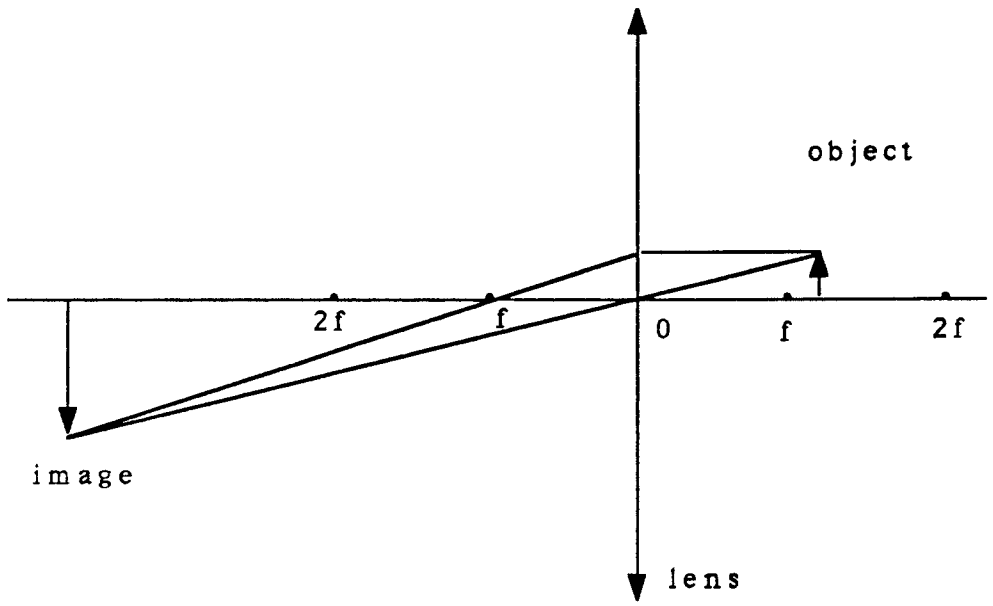

Figure 4.2: Ray Diagram of A Lens

-NEΔT at camera output : 0.0579 K

-Temporal Noise :

—at FPA output : 2 rms mV, 772 elec./pixel

—at camera output : 2.544 rms mV, 982 elec./pixel

4.2 Enlargement of the Pictures of Solder Joints

The first step is still to enlarge the pictures of solder joints. There are two methods to achieve this task. One is using a special camera microlens, which can be installed in front of the original camera lens. Another is utilizing the principle of the geometric optics that is shown in Figure 4.2, i.e,

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

where f is the focal length, u is the distance between the object and the center of the lens, v is that between the image and the center of the lens. What we obtain is an inverse. If the values of u and v are selected appropriately, we can enlarge the image.

Both methods enlarge the image of joints very well. The picture we shown on Figure 4.3 is taken by using the second method.

4.3 Image Enhancement

After we obtain the infrared (IR) joint pictures, we need to process the images before classification can be implemented. Firstly, on the IR picture we obtained, only every other line has data, the rest lines are all black (0 gray level). Interpolation is hence applied. At each column assign to each "empty" pixel the average gray value of its immediate upper and lower pixels, Figure 4.4 is the IR picture after the interpolation. Compared with Figure 4.3, the effect of improvement is obvious.

Secondly, the histogram of the IR images reveals the following fact. That is, the distribution of the gray levels in the IR images is generally concentrated in the range from gray level 120 to gray level 220, sometimes this range is from 60 to 180, depending on the temperature of the PCB. The full range of gray levels for an 8 bit digital image is from 0 to 255. The difference between the highest gray level and the lowest gray level is about 100 to 120. The features on the IR images are therefore not very apparent. So, we enhance the picture by applying histogram modification techniques.

There are several ways to modify the histogram of a digital image. Here, we apply the linear gray transform. The transform function is shown in Figure

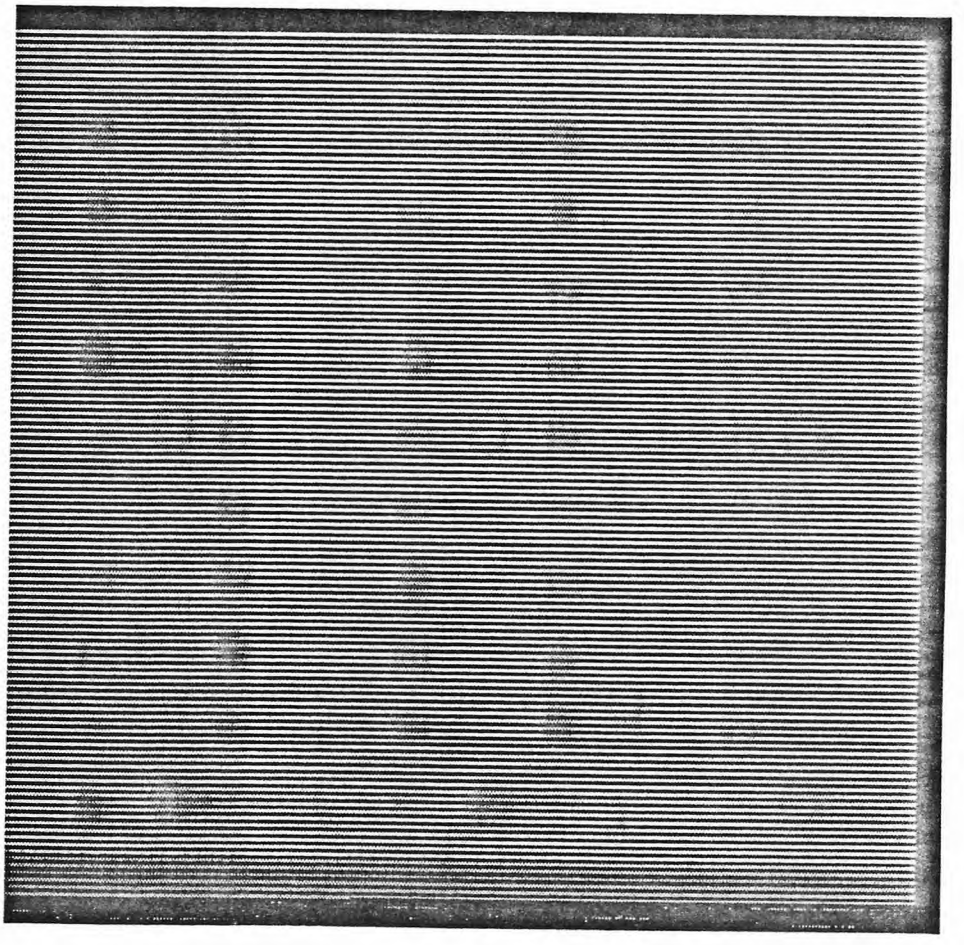

Figure 4.3: The Thermal Image Taken with IR-CCD

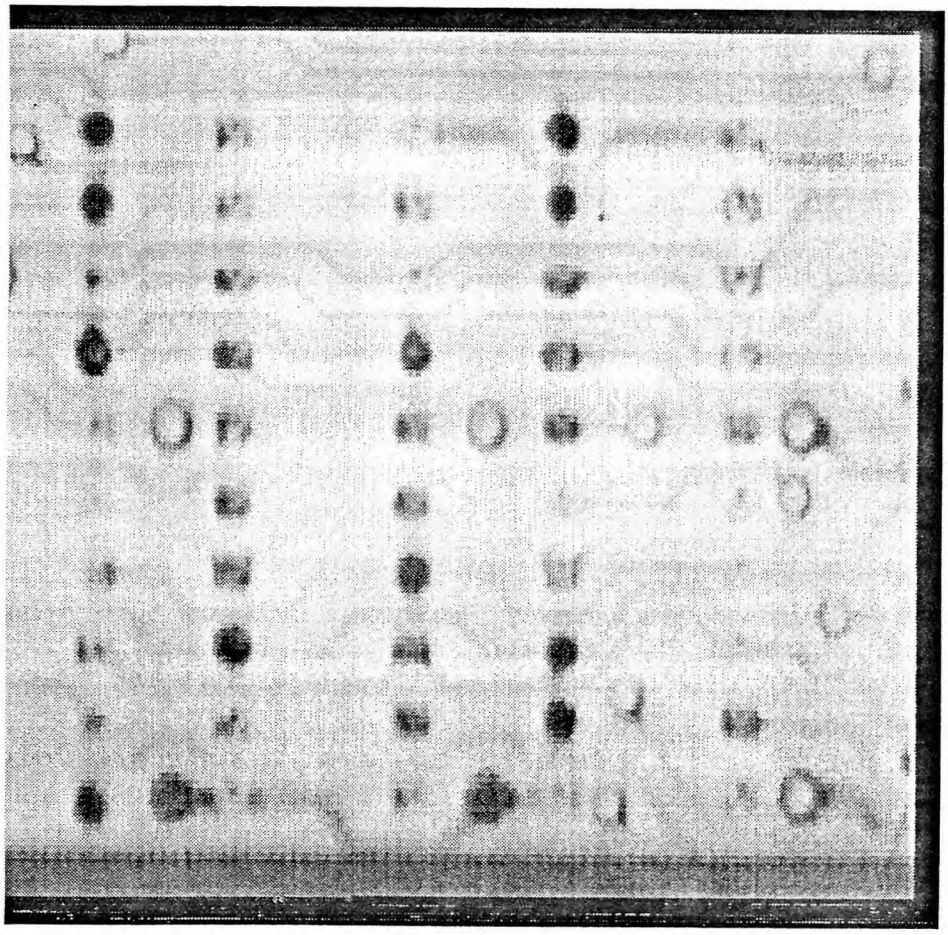

Figure 4.4: The Thermal Image after Interpolation

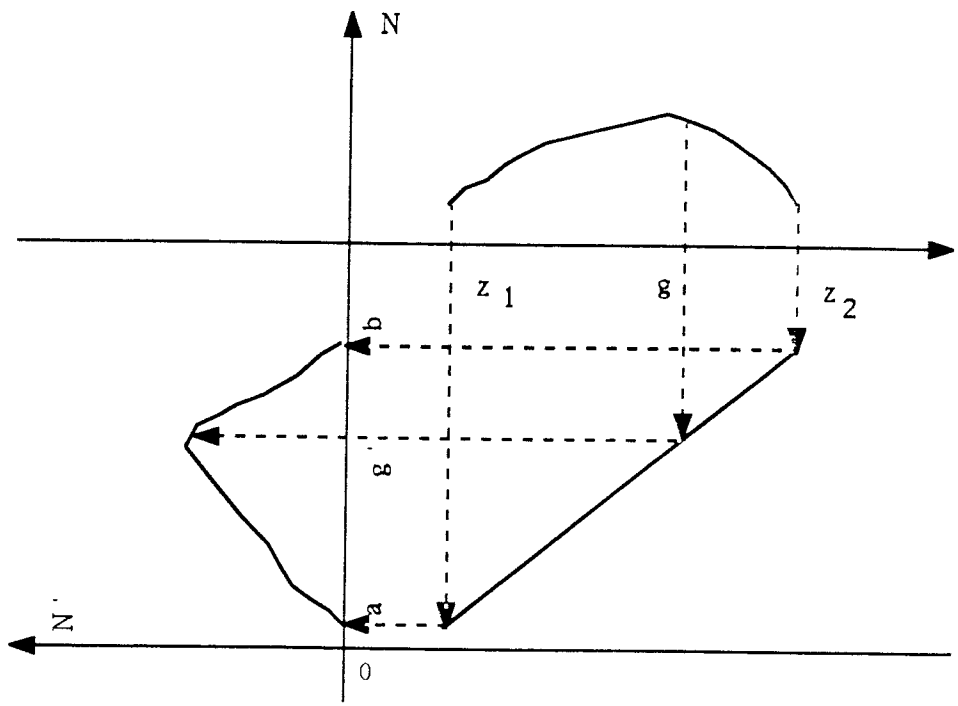

Figure 4.5: The Histogram Transformation

4.5.

where z_1 and z_2 represent the lower and upper bounds of gray levels in the original image, respectively, while a and b represent the corresponding bounds for the new image.

Mathematically, the transform can be represented by the following formula,

$$g' = a + \frac{(b-a)(g-z_1)}{(z_2-z_1)}$$

where g is the gray level for the original image, while g' is the new value after transform.

After this transform, the quality of the IR image becomes better, see Figure 4.6. The correct classified rate is raised from 60% to 80%.

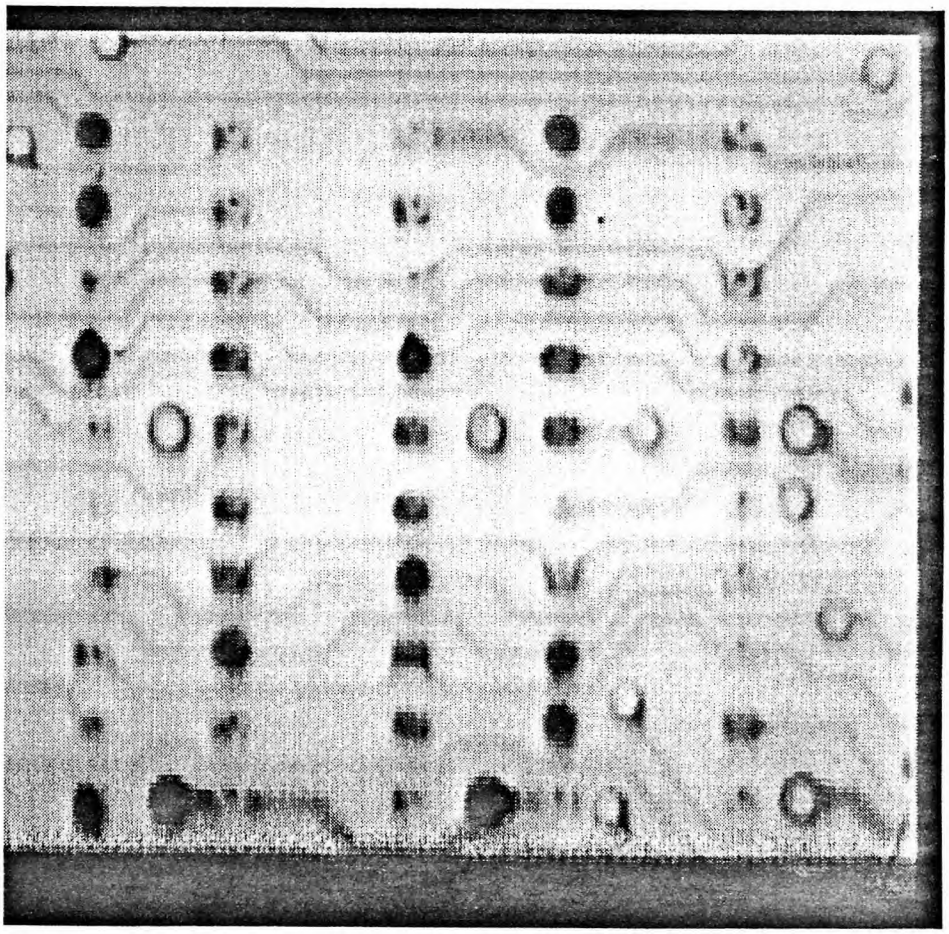

Figure 4.6: The Thermal Image after Histogram Transformation

4.4 Compensation of the Chip Holders Effect

After using the histogram transformation, the IR image is much clearer in the screen. Now, we find that it has some different background temperature. Why? Because there are some chip holders on another side of the PCB, so that the cooling processes of different parts of the PCB are not the same. In the center part of the holders, the temperature is higher than that on the edges of the holders. We can see this on the screen, some parts have a little bit darker background than other parts. These differences in temperature distort the useful information contained in the thermal behavior of joints and make the classification more difficult.

We use two ways to try to compensate the background temperature differences caused by chip holders. First, we try to average background temperature. The idea is that we sum each pixel gray level of the whole board except the joint pixel, then divide the total number of the pixels which have been counted. The result is the general average background gray level denoted by $\overline{AVER}$. Then, we compute a small part average background gray level $\overline{AVER_1}$, where a joint subimage is located. There is a difference Δ between $\overline{AVER}$ and the $\overline{AVER_1}$. Then we subtract this difference from the gray level of each pixel in IR image. Mathematically, what is mentioned above can be expressed below,

$$\overline{AVER} = \frac{1}{N}\sum_{i=1}^{N} g_i$$

$$\overline{AVER1} = \frac{1}{N_1}\sum_{i=1}^{N_1} g_i$$

$$\Delta = \overline{AVER} - \overline{AVER1}$$

$$g_i' = g_i - \Delta$$

where g_i is the gray level of the i th pixel, g_i' is the new gray level assigned to the i th pixel, N is the total number of pixels counted in calculating $\overline{AVER}$, N_1 is the total number of pixels counted in calculating $\overline{AVER1}$.

After this measure being taken, the correct classified rate is raised to near 90%.

But, we think that when we take this measure to eliminate the temperature difference in the cooling processes of different parts of the PCB, at the same time, some useful information of the solder joints may be also removed. So, we try to select new features suitable for IR images which are discussed in the next section.

4.5 A New Set of Features

Despite the holders' effect, we still could find some useful information from the solder joints subimages. Those are different from that contained in visual light images. By carefully analyses, we found that in a solder joint subimage, the difference between different areas was apparent. The subimages of the four different joint classes have their own distinct appearance for different areas. So, we should find some special features for the IR image. Obviously, something like mean value in the features becomes less useful, because the chip holders' temperature differences exist. But the variance, the second order moment, the differences of each area of the joint become much more meaningful. So, another set of features is extracted. They are described as follow.

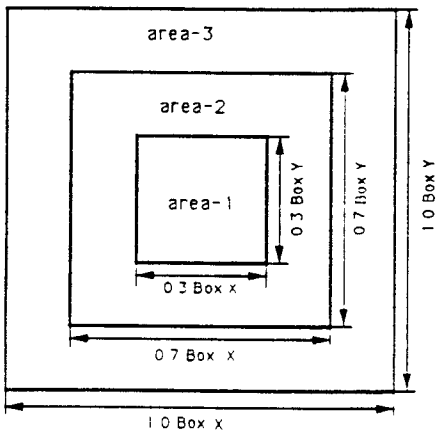

Figure 4.7: The Structure of a Joint Subimage

$$Mean_2 = \frac{1}{n_{x2}n_{y2}} \sum_{i=1}^{n_{x2}} \sum_{j=1}^{n_{y2}} z(i,j)$$

$$Feature_1 = \frac{1}{n_{x1}n_{y1}} \sum_{i=1}^{n_{x1}} \sum_{j=1}^{n_{y1}} [z(i,j) - Mean_2]^2$$

$$Feature_2 = \frac{1}{n_{x2}n_{y2}} \sum_{i=1}^{n_{x2}} \sum_{j=1}^{n_{y2}} [z(i,j) - Mean_2]^2$$

$$Feature_3 = \frac{1}{n_{x3}n_{y3}} \sum_{i=1}^{n_{x3}} \sum_{j=1}^{n_{y3}} [z(i,j) - Mean_2]^2$$

$$Feature_4 = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} [z(i,j) - Mean_2]^2$$

where $n_x n_y$ is the number of total pixels in a given solder joint subimage, $n_{x1}n_{y1}$, $n_{x2}n_{y2}$ and $n_{x3}n_{y3}$ are the numbers of total pixels in area 1, area 2 and area 3, respectively, which are shown in Figure 4.7. $Mean_2$ represents the mean value of *area* 2 of a given solder joint subimage.

We kept some features from the visual light inspection [4]. They are,

$$Feature_5 = \sigma^2$$

$$Feature_6 = I_{bb}$$

$$Feature_7 = I_{ratio}$$

Thus, a new set of features is formulated. With this new set of features, the result of the correct classified rate increases up to more than 90%.

4.6 Experiment of the Infrared Inspection

Before we take IR images, we should make an initial focusing of the camera with respect to the object position. That means we should fix the distance between the camera and the object. Because when the board is heated, at beginning, the picture in the monitor is all white. So, at that time we can not do any focusing. When the picture is becoming apparent, the temperature decreases fast, we do not have enough time to adjust the focus at that stage.

We use an oven to heat the PCBs. Using some aluminium paper makes the surrounding temperature same. Heating continues until the PCB reaches more than 75°C.

Put the PCB at the position we previously determined. Adjust a little bit focusing if it is still needed. When the temperature goes down, first, the

IR subimages of excess-solder joints and non-solder joints become apparent, next the less-solder joints. The subimages of normal joints become apparent last. When the temperature decreases to about 48°C to 40°C , the subimages of joints become clearest. One thermometer has been attached to another side of the PCB to monitor the temperature. IR pictures are taken as the temperature goes down every 2°C degree, i.e., at 50°C, 48°C, 46°C, 44°C, 42°C, 40°C and 38°C.

Apply the methods we described above to process the image in the SUN workstation. The best correct classified rate is about 94% and the good/Bad rate 96% at the 42°C. The size of subimages of solder joints is 22×22 or 20×16. There are total 50 joints in the PCB. Only are 3 joints wrongly classified. The correct classified rate for other experiments are about 90%. These results are shown in Figure 4.8. The w_i $(i = 1, 2, ..., 7)$ are the file names of the images; z_1, z_2 are the gray levels mentioned in Figure 4.5; The meanings of CC and G/B are the same as defined in Section 3.1.

(1) Total joints is 50 in 20×16

Temperature	z_1 z_2	CC	G/B	%CC	%G/B
w_1 50°C	250 140	39	41	78%	82%
w_2 48°C	250 140	42	45	84%	90%
w_3 46°C	250 140	43	45	86%	90%
w_4 44°C	250 140	45	47	90%	94%
w_5 42°C	250 140	47	48	94%	96%
w_6 40°C	160 60	41	47	82%	94%
w_7 38°C	180 100	40	45	80%	90%

Continue

— (2) Total joints is 50 in 22 × 18 —

Temperature	z_1 z_2	CC	G/B	%CC	%G/B
w_1 50°C	250 140	39	41	78%	82%
w_2 48°C	250 140	44	45	88%	90%
w_3 46°C	250 140	45	47	90%	94%
w_4 44°C	250 140	45	47	90%	94%
w_5 42°C	250 140	46	47	92%	94%
w_6 40°C	160 60	46	47	92%	94%
w_7 38°C	180 100	43	47	86%	94%

—(3) Total joints is 50 in 22 × 20 —

Temperature	z_1 z_2	CC	G/B	%CC	%G/B
w_1 50°C	250 140	33	35	66%	70%
w_2 48°C	250 140	43	44	86%	88%
w_3 46°C	250 140	46	48	92%	96%
w_4 44°C	250 140	43	45	86%	90%
w_5 42°C	250 140	47	48	94%	96%
w_6 40°C	160 60	46	48	92%	96%
w_7 38°C	180 100	38	45	76%	90%

—(4) Total joints is 50 in 22 × 22 —

Temperature	z_1 z_2	CC	G/B	%CC	%G/B
w_1 50°C	250 140	36	41	72%	82%
w_2 48°C	250 140	43	44	86%	88%
w_3 46°C	250 140	43	44	86%	88%
w_4 44°C	250 140	43	45	86%	90%
w_5 42°C	250 140	47	48	94%	96%
w_6 40°C	160 60	43	46	86%	92%
w_7 38°C	180 100	41	46	82%	92%

—(5) Total joints is 50 in 24 × 20 —

Temperature	z_1 z_2	CC	G/B	%CC	%G/B
w_1 50°C	250 140	37	40	74%	80%
w_2 48°C	250 140	45	46	90%	92%
w_3 46°C	250 140	45	47	90%	94%
w_4 44°C	250 140	42	43	84%	86%
w_5 42°C	250 140	46	48	92%	96%
w_6 40°C	160 60	45	48	90%	96%
w_7 38°C	180 100	41	48	82%	96%

Figure 4.8: The Results of the Infrared Inspection

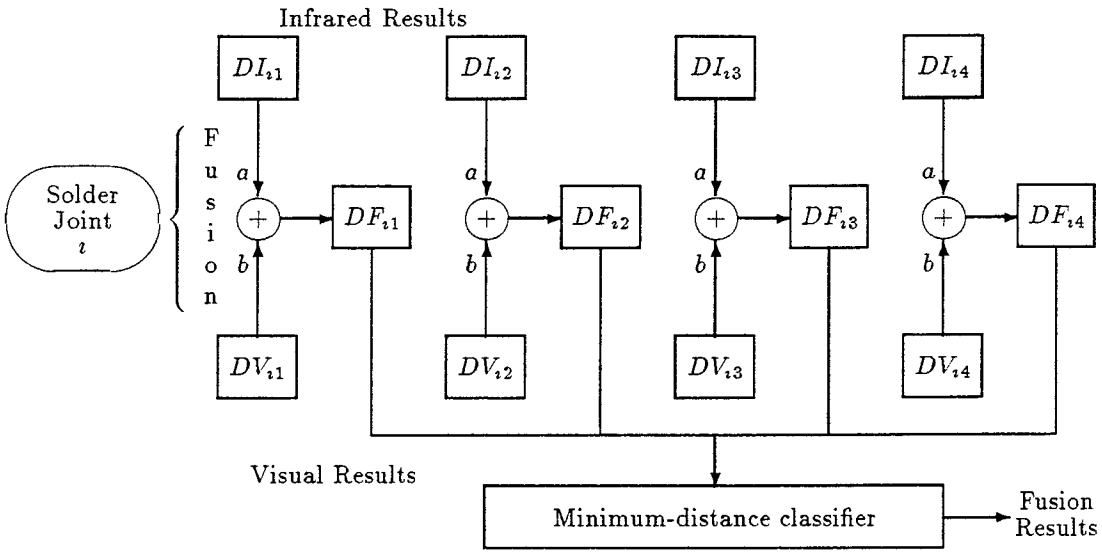

Figure 4.9: Fusion Inspection Approach

4.7 Experiment of Fusion Inspection

The fusion of infrared and visual light inspection data has been conducted to see if it can achieve any improvement. The block diagram of the fusion inspection is shown in Figure 4.9. The normalized feature distance means the normalized distance between the features of solder joint to the mean features of certain class, as defined in Chapter 2. If the solder joints on the PCB are denoted by S_i $i = 1, ..., 20$, DI_{ij} is the normalized feature distance from S_i to Class j in infrared inspection. DV_{ij} is the normalized feature distance from S_i to Class j in visual light inspection. A normalized fusion distance DF_{ij} is defined as

$$DF_{ij} = aDI_{ij} + bDV_{ij}$$

where $i = 1, ..., 20$, $j = 1, ..., 4$, and $a + b = 1$. A set of new normalized fusion distances are used to classify the solder joints into certain classes according to the minimum distance principle.

A group of experiment results is shown in Figure 4.10. The original results are that the correct classified rate is 98% and the good/bad rate 98% with the joint subimage size of 20×16 in visual technique, and that the correct classified rate is 94% and the good/bad rate is 96% with the joint subimage size of 20×16 in IR technique. Then, the fusion results show that it reaches to the best 100% in both the correct classified rate and the good/bad classified rate. The a and b are weights mentioned in Figure 4.9. The CC and G/B are the same definitions as mentioned in Section 3.1.

—Total joints is 50 —					
a	b	CC	G/B	%CC	%G/B
0.7	0.3	48	49	96%	98%
0.5	0.5	49	50	98%	100%
0.3	0.7	49	50	98%	100%
0.2	0.8	50	50	100%	100%

Figure 4.10: The Results of the Fusion Inspection

Chapter 5

Window Tool for Solder Joint Inspection

This tool is a complete application software package written in Sunview window. Each step of unsupervised statistical pattern recognition is done by pushing certain software button using the mouse of the SUN workstation. The C source code of the whole software is shown in Appendix B.

While totally four kinds of windows are offered by Sunview, only two different kinds of windows, Canvas and Panel are used in this tool. Canvas is an environment for graph and image displaying. Panel is one of the main methods for man-machine communication, such as keying-in file names, selecting functions, setting parameters etc.. This tool, when working on the SUN workstation screen, is that shown in Figure 5-1, where Image 't' is showing on the Canvas. Whole procedure is described next step by step.

A. Solder joint segmentation:

1. Typing-in image data file name: move the arrow cursor near by the line prompted with "Draw data file1:", press the left button of the mouse, a small triangle cursor will flash next to the ":", type in the data file name, denoted

as filename1 below.

2. Typing-in position file name: Position file is used to tell the computer where a certain class of solder joints is. Do the same thing as Stop(1), let small triangle flash on the second line, prompted with "Position file1:", type in the initial name of position file, denoted as name2 below. An extension ".pos" is assumed.

3. To show the image: push the Panel button "New", that is done by moving arrow cursor onto the Panel button "New" and pressing the left button of the mouse. Through out this thesis this concrete operating procedure will not be repeated when a term *push Panel button "(specific function name)"* is mentioned.

4. To position the solder joints: When arrow cursor is moved onto Canvas, it becomes square with a cross tail, horizontal and vertical, which is helpful to position adjacent solder joints. The keys "W, Z, A, S" on the keyboard are used to move the square cursor one pixel "up, down, left, right", respectively. If the center of the cross seems accurately in the center of the solder joint which belongs to the class to be positioned, press the key "E" on the keyboard, a small frame box will appear surrounding the center of the solder joint and a pair of X-Y coordinates is recorded in memory. The size of the box , either X side or Y side, should be adjusted by changing the Panel items "Box X" and "Box Y:", that will be explained below in detail.

5. To erase an input position: If a box seems not accurately positioned on the center of a solder joint, move square cursor onto the box and press key "Q" to erase the old box together with the position record in memory. Use "W, Z, A, S" to adjust the square cursor and press "E" when the position seems satisfactory. Repeat this step until the box is located on the center of

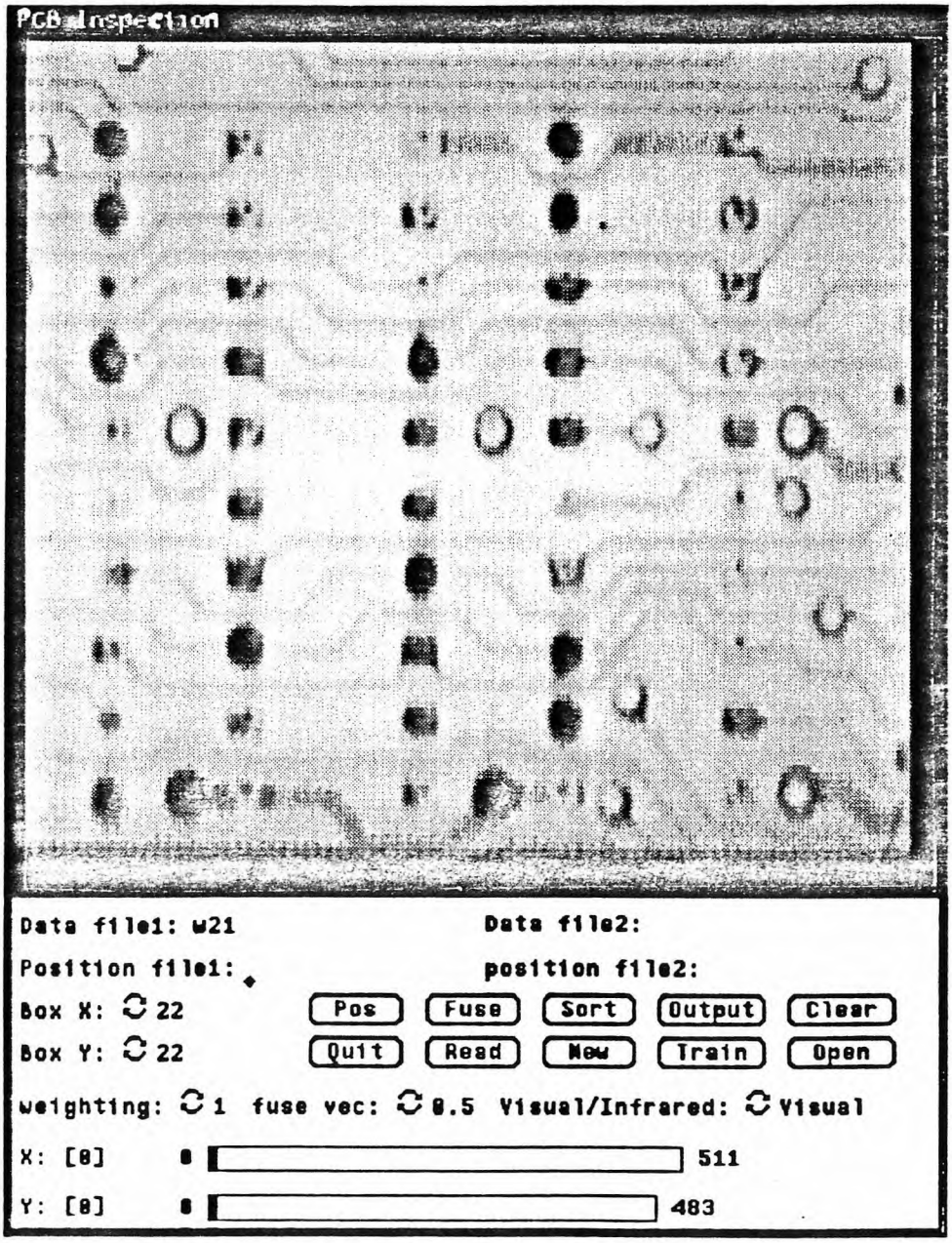

Figure 5.1: A Window Tool for Solder Joint Inspection

the solder joint.

6. To create a position file: When a certain class of solder joints are partially or totally positioned, push the Panel button "Output", which will save all records onto the position file name2.pos.

7. To modify a position file: After the Steps (1), (2), (3), push the Panel button "Read", every position defined in the old position file will be called back and boxes are shown on the solder joints. Boxes can be erased, added, modified. After everything is ready, push "Output" again. The old position file will be covered by a new on with the same name.

Repeat Steps (2)-(7) to create different position files for different classes of solder joints. This completes the segmentation phase of the inspection.

B. Mean features learning or training:

Before training an important preparation is to decide the processing size of the solder joint subimage. The size is set by Panel items "Box X:" and "Box Y:". One way to change them is moving the arrow cursor near the circle symbol and Pushing the left button of the mouse. Size will be changed to presetting value circularly for every click. Another way, a pull-down menu can be selected by pushing the right button of the mouse, moving mouse to highlight certain setting then releasing the button. Training steps are:

1. The first step here is the same as Step (1) of Phase A.

2. To begin a new round of training: push the Panel button "Open" for beginning a new round of training, name1.lnm and name1.fea will be created or renewed. Because the new data will be appended to these files while different classes of solder joints are used for training one by one.

3. Same as Step (2) of Phase A.

4. To train: push the Panel button "Train", the class in the name2.pos will be used for training and statistical mean features are recorded.

Repeat Steps (3), (4) until every position fill for the image name1 goes through. This finishes the training phase. It is recommended that always keep the same order of the position files for different rounds of training. Mean feature vectors are ready for the next phase "Classification:", in file name1.fea. Another output file name1.lnm is minimum and maximum values for every feature and every class. It is used to calculate the maximum value needed to normalize all features. That is also useful for analysis purpose. After normalization features are in the range of 0 to 1 and have equal weight to the distance decision. In the meantime a set of files name2.fea are output for analysis purpose. The standard deviation and relative errorfor every kind of feature are useful to evaluate the feature.

C. Classification:

Before starting to classify, a weighting vector should be ready. As the initial value, a vector has its each component being 1. Text editor could be used to create a file named as weight1.vec, where only 1s separated by space exist. The number of 1s is the same as the number of features to be trained. Other weighting vectors could be put on weight2.vec, weight3.vec,.... Using Panel circle selecting item "weighting :" to decide which vector should be used. Be sure that always keep the same box size in one round of training and classifying. Classifying steps are described in the following:

1. The same as Step (1) of Phase A.
2. Selecting weighting vector: use Panel circle item "weighting :".
3. The same as Step (2) of Phase A.

4. Classifying: push the Panel button "Sort".

Repeat Steps (3),(4) for every class of solder joint by changing position files in the same orders when they are used for training . In classification phase two kinds of files are generated. One is name1.sot. This is an appended file. Sorting every position file will produce one line message. Classes are named as 1 to 4, following the order in which they are trained and sorted. The number shown after each class number means how many solder joints have being classified into this class. because first line shows the results of the classification of Class 1, a largest number in the first line of the group is expected after the integer 1. The largest number in the second line of the group is expected after the integer 2, and so on. Two main error rates, Good/bad inspection rate and correct classified rate, are used to evaluate the performance of the classifier. Since only one class of solder joints is known as "good", Class 2 in the experiment here, the Good/bad inspection rate is computed by plus how many solder joints of Class 2 are classified into Class 2 and how many solder joints of other classes are not recognized as Class 2, then divided by total number of solder joints in this image. The correct classified rate is the number of solder joints which are classified into the right classes divided by total.

Another kind of output files after sorting is a set of filename2.out files. The X-Y coordinates of every solder joint, the normalized distances between the feature vector associated with the solder joint and the mean feature vector of each class are printed in one line. At the end of this line is the class number to which the solder joint is assumed to belong. That is, it has the least distance to this class comparing to others. These files give the information about why a certain solder joint is sorted to a wrong class, how far it is away from its

own class, or how far it is away from other classes when it is sorted correctly.

A group of experiment, training or sorting, means going through all position files associated with one image, when all parameters kept unchanged. A round of experiment means beginning from a group of training and ended by one or more groups sorting. For example, changing the weighting vector, more than one group of sorting can be implemented and different results could be obtained while the mean features are kept unchanged. Changing the size of the solder joint subimage, by changing the "Box X:" and "Box Y:" setting, and changing the typical sets of solder joints used for training, more rounds of training and sorting could be tested and more different sorting results could be obtained.

D. Classification of fusing visual light and infrared images.

There are two set of features for training and sorting. One is the original set, which can be used in visual light image processing. The other one is for thermal image processing. If you want to use these features to process the thermal image, you can push the Panal item "Visual/Infrared" to make your choice. After that you can proceed the process as described above. as reverse, you can push the Panal item again to choose the original set of the features

To make fusion experiment of visual light and infrared images, first, you should process one visual light image and one infrared image respectively to obtain the output files which have the extension ".out". Then, you should put one visual light image file name which has been processed beside the Panal item, "Data file1:", and its position file name beside the Panal item "Position file1:". Then put the processed infrared image file name beside the Panal item "Data file2", and its position file beside the "Position file2:". Then, choosing

the vector by push the Panal item "Fuse vec", this is the weighting vector of the position filename1, and the 1-vector will be the weighting vector of the position filename2. After that push the Panal button of "Fuse", the output will be stored in the files with extension ".fus". You can do fusion inspection for different position files by change the position filename1 and the position filename2. Repeat the process described above.

Chapter 6

Conclusion and Discussion

1. The capability of the infrared automatic inspection for solder joints on PCBs is proved in this research work. The results agree with the theoretical expectation which has been predicted in [9].

2. Very good results have been achieved via our work in the Electronic Imaging Laboratory. The automatic infrared inspection of solder joints on PCBs can be applied to PCB industry. In the next step, we should look for industrial partners to continue this work.

3. Fusion inspection is a useful measure to obtain better results.

4. More weighting vectors could be tried to improve the results. One way to select weighting vector is to average the relative errors for every feature in different classes which are listed in the files named as filename2.fea.

References

[1] P. J. Besl, E. J. Delp and R. Jain, "Automatic visual solder joint inspection," *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, pp. 42-56, Mar. 1985.

[2] S. L. Bartlett, P. J. Besl, C. L. Cole, R. Jain, D. Mukherjee and K. D. Skifstad, "Automatic solder joint inspection," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 1, pp. 31-43, Japan, 1988.

[3] P. D. Blume, "Uniformity corrector and signal processing for a 160×244 elements IR-CCD image with PtSi Schottky-barrier detectors," Master's thesis directed by Dr. W. F. Kosonocky, Electrical and Computer Engineering Department, New Jersey Institute of Technology, Dec. 1989.

[4] S. Blume, "Infrared TV camera with PtSi Schottky-barrier detectors," Master's thesis directed by Dr. W. F. Kosonocky, Electrical and Computer Engineering Department, New Jersey Institute of Technology, Dec. 1989.

[5] *Applications of Thermal Imaging,* Edited by S. G. Burnay, T. L. Williams and C. H. Jones, Adam Hilger, Bristol and Philadelphia, 1988.

[6] R. T. Chen, "Survey automated inspection: 1981 to 1987," *Computer vision, Graphics, and Image processing*, 41, pp. 346-381, 1988.

[7] *Datacube User's Manual,* Datacube Inc., Oct. 1988.

[8] R. C. Gonzalez and P. Wintz, *Digital Image Processing* (Second Edition), Addison-Wesley Publishing Company, Reading, Massachusetts, Nov. 1987.

[9] W. F. Kosonocky, A. N. Akansu, C. H. Lu and Y. Q. Shi, "Infrared imaging for machine vision," *Proposal to New Jersey Commission of Science and Technology*, Nov. 1987.

[10] C. M. Leng, "Uniformity correction for infrared IR-CCD camera," Master's thesis directed by Dr. W. F. Kosonocky, Electrical and Computer Engineering Department, New Jersey Institute of Technology, Dec. 1990.

[11] Y. Nakagawa, "Automatic visual inspection of solder joints on printed circuit boards," *Proc. SPIE, Robot Vision*, Vol. 336, pp. 121-127, May 1982.

[12] *Sunview Programming User's Instruction,* SUN Microsystem Inc., 1988.

[13] J. T. Tou and R. C. Genzalez, *Pattern Recognition Principles,* Addision-Wesley Publishing Company, Reading, Massachusetts, 1974.

[14] K. K. Zhou, "Automatic infrared inspection of solder joints on printed circuit boards," Master's thesis directed by Dr. Y. Q. Shi, Electrical and Computer Engineering Department, New Jersey Institute of Technology, May 1990.

Appendices

A Solder Joint Inspection Window Tool C Source Code

```
#include <stdio.h>
#include <math.h>
#include <suntool/sunview.h>
#include <suntool/panel.h>
#include <suntool/canvas.h>
#include <suntool/tty.h>

#define   CMS_SIZE  256

static    void help_proc(), quit_proc(), pos_proc(),
fuse_proc();
static    void output_proc(), sort_proc(), train_proc(),
draw();
static    void handle_event(), read_proc(), new_proc(),
features();
static    void clear_proc(), open_proc(), size_proc(),
rpos_proc();

static    int  value_to_op();
static    char value_to_c(), value_to_d(), value_to_e();

static    char    cmdstring[256], *fv;
static    short   posvect[512]=NULL;
static    int     pcbpix[64][64], xsize, ysize;
static    int     bx2, by2, cswx, cswy, ofrx, ofry;
static    float   feavect[20], fea[120][10], data[512],
data1[512], mean[6][20];
static    float   deltax, deltay, delta, deltx2, delty2,
pos, vec;
static    float   sjpix[64][64], ll[5][20], mm[5][20]; ;

static    Panel_item     fname_item;
static    Panel_item     f1name_item;
static    Panel_item     f2name_item;
static    Panel_item     f3name_item;
static    Panel_item     box_xsize;
```

```
static    Panel_item     box_ysize;
static    Panel_item     weight_num;
static    Panel_item     visual_infrared;
static    Panel_item     fuse_vec;
static    Panel_item     cursor_x;
static    Panel_item     cursor_y;
Pixfont   *bold;
struct  rasterfile rh;

/* static    short    icon_image[]={
#include </usr/mesunb/users/kxz4467/squr>
};

DEFINE_ICON_FROM_IMAGE(my_icon, icon_image); */

static Frame        frame;
static Panel        panel;
static Canvas       canvas;
static Tty          tty;
static Cursor       cursor;
static Menu         menu;



main(argc, argv)
int  argc;
char **argv;
{
     Rect             *r;
     register Pixwin  *pw;
     register int     i, k ;
     un_char  red[CMS_SIZE], green[CMS_SIZE],
blue[CMS_SIZE];
     struct pixrect   *orig_pr, *new_pr;

     pos=0;
     bold=pf_open("/usr/lib/fonts/fixedwidthfonts/screen.b.1
2");
     if (bold==NULL) exit(1);

     frame=window_create(0, FRAME, FRAME_LABEL, "PCB
Inspection",
                         FRAME_CMDLINE_HELP_PROC,
help_proc,
                         FRAME_ARGS, argc, argv,
                       /* FRAME_ICON, &my_icon,*/
                         FRAME_NO_CONFIRM,  TRUE,
                         WIN_ERROR_MSG, "Can't create
window.",
                         WIN_HEIGHT, 484,
                         WIN_WIDTH, 512,
```

```
                        0);


     menu=menu_create(MENU_ACTION_ITEM, "Quit",  quit_proc,
0);

     canvas=window_create(frame, CANVAS,
                        WIN_CONSUME_KBD_EVENT,
                        WIN_ASCII_EVENTS,
                        WIN_CONSUME_PICK_EVENT, LOC_DRAG,
                        WIN_HEIGHT, 484,
                        WIN_WIDTH, 512,
                        WIN_EVENT_PROC, handle_event,
                        0);
     window_fit(canvas);



     /* copy the orignal cursor image */

     orig_pr = (struct pixrect *) (LINT_CAST(
         cursor_get(window_get(canvas, WIN_CURSOR),
CURSOR_IMAGE)));
     new_pr = (struct pixrect *)(LINT_CAST(mem_create(
          orig_pr->pr_width, orig_pr->pr_height, orig_pr-
>pr_depth)));
          (void)pr_rop(new_pr, 0, 0, new_pr->pr_width,
new_pr->pr_height,
                   PIX_SRC, orig_pr, 0, 0);

     cursor = cursor_create(CURSOR_IMAGE, new_pr,
                        CURSOR_OP,    PIX_SRC ^ PIX_DST,
                        0);


    cursor_set(cursor,
             CURSOR_SHOW_CURSOR,           1,
             CURSOR_XHOT,                  7,
             CURSOR_YHOT,                  7,
             CURSOR_OP,                    PIX_NOT(PIX_DST),
             CURSOR_CROSSHAIR_LENGTH,      64,
             CURSOR_CROSSHAIR_GAP,         16,
             CURSOR_CROSSHAIR_COLOR,       255,
             CURSOR_CROSSHAIR_OP,          PIX_SRC
             CURSOR_SHOW_CROSSHAIRS,       TRUE,
             0);
     window_set(canvas, WIN_CURSOR, cursor, 0);

     panel=window_create(frame, PANEL, WIN_X, 0,
                          WIN_BELOW, canvas,
                          WIN_FONT, bold,
```

```
                          0);
     create_panel_item();
     window_fit_height(panel);


/*   tty=window_create(frame, TTY, WIN_BELOW, panel,
                          WIN_X, 0,
                          WIN_WIDTH, 512,
                          WIN_ROWS, 18,
                          WIN_FONT, bold,
                          0);

     window_fit(tty);       */
     window_fit(frame);

     pw=canvas_pixwin(canvas);

     for (i=0; i<CMS_SIZE; i++)
          red[i]=green[i]=blue[i]=i;
     pw_setcmsname(pw, "showcolor" );
     pw_putcolormap(pw, 0, CMS_SIZE, red, green, blue );

     size_proc();

     window_main_loop(frame);
     exit(0);
}


create_panel_item()

{
     fname_item = panel_create_item(panel, PANEL_TEXT,
                     PANEL_LABEL_STRING, "Data file1:",
                     0);
     finame_item = panel_create_item(panel, PANEL_TEXT,
                     PANEL_LABEL_X, ATTR_COL(0),
                     PANEL_LABEL_Y, ATTR_ROW(1)+9,
                     PANEL_LABEL_STRING, "Position file1:",
                     0);
     f2name_item = panel_create_item(panel, PANEL_TEXT,
                     PANEL_LABEL_X, ATTR_COL(32),
                     PANEL_LABEL_Y,ATTR_ROW(0)+4,
                     PANEL_LABEL_STRING, "Data file2:",
                     0);
     f3name_item = panel_create_item(panel, PANEL_TEXT,
                     PANEL_LABEL_X, ATTR_COL(32),
                     PANEL_LABEL_Y, ATTR_ROW(1)+9,
                     PANEL_LABEL_STRING, "Position file2:",
                     0);
```

```
    box_xsize = panel_create_item(panel, PANEL_CYCLE,
                    PANEL_LABEL_X, ATTR_COL(0),
                    PANEL_LABEL_Y, ATTR_ROW(2)+10,
                    PANEL_LABEL_STRING, "Box X:",
                    PANEL_VALUE, 6,
                    PANEL_CHOICE_STRINGS,
                    "8", "12", "14", "16", "18", "20",
                    "22", "24", "30", "34", "40", 0,
                    PANEL_NOTIFY_PROC, size_proc,
                    0);

    box_ysize = panel_create_item(panel, PANEL_CYCLE,
                    PANEL_LABEL_X, ATTR_COL(0),
                    PANEL_LABEL_Y, ATTR_ROW(4)-6,
                    PANEL_LABEL_STRING, "Box Y:",
                    PANEL_VALUE, 6,
                    PANEL_CHOICE_STRINGS,
                    "8", "12", "14", "16", "18", "20",
                    "22", "24", "30", "34", "40", 0,
                    PANEL_NOTIFY_PROC, size_proc,
                    0);


    weight_num = panel_create_item(panel, PANEL_CYCLE,
                    PANEL_LABEL_X, ATTR_COL(0),
                    PANEL_LABEL_Y, ATTR_ROW(5)+4,
                    PANEL_LABEL_STRING, "Weighting:",
                    PANEL_VALUE, 0,
                    PANEL_CHOICE_STRINGS,
                    "1", "2", "3", "4", "5", "6", "7", "8",
0,
                    0);


    visual_infrared = panel_create_item(panel, PANEL_CYCLE,
                    PANEL_LABEL_X, ATTR_COL(33),
                    PANEL_LABEL_Y, ATTR_ROW(5)+4,
                    PANEL_LABEL_STRING, "Visual/Infrared:",
                    PANEL_VALUE, 3,
                    PANEL_CHOICE_STRINGS,
                    "Visual", "Infrared",  0,
                    0);

    fuse_vec = panel_create_item(panel, PANEL_CYCLE,
                    PANEL_LABEL_X, ATTR_COL(16),
                    PANEL_LABEL_Y, ATTR_ROW(5)+4,
                    PANEL_LABEL_STRING, "Fuse vec:",
                    PANEL_VALUE, 4,
                    PANEL_CHOICE_STRINGS,
                    "0.1", "0.2", "0.3", "0.4", "0.5",
                    "0.6", "0.7", "0.8", "0.9", 0,
                    0);
```

```
    cursor_x = panel_create_item(panel, PANEL_SLIDER,
                    PANEL_LABEL_X, ATTR_COL(0),
                    PANEL_LABEL_Y, ATTR_ROW(7)-4,
                    PANEL_LABEL_STRING, "X:",
                    PANEL_SLIDER_WIDTH, 256,
                    PANEL_VALUE, 0,
                    PANEL_MIN_VALUE, 0,
                    PANEL_MAX_VALUE, 511,
                    0);

    cursor_y = panel_create_item(panel, PANEL_SLIDER,
                    PANEL_LABEL_X, ATTR_COL(0),
                    PANEL_LABEL_Y, ATTR_ROW(8)+4,
                    PANEL_LABEL_STRING, "Y:",
                    PANEL_SLIDER_WIDTH, 242,
                    PANEL_VALUE, 0,
                    PANEL_MIN_VALUE, 0,
                    PANEL_MAX_VALUE, 483,
                    0);



    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Pos", 5, 0),
                    PANEL_ITEM_X, ATTR_COL(20),
                    PANEL_ITEM_Y, ATTR_ROW(2)+10,
                    PANEL_NOTIFY_PROC, pos_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Fuse", 5,
0),
                    PANEL_ITEM_X, ATTR_COL(28),
                    PANEL_ITEM_Y, ATTR_ROW(2)+10,
                    PANEL_NOTIFY_PROC, fuse_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Sort", 5,
0),
                    PANEL_ITEM_X, ATTR_COL(36),
                    PANEL_ITEM_Y, ATTR_ROW(2)+10,
                    PANEL_NOTIFY_PROC, sort_proc, 0);


    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Output", 6,
0),
```

```
                    PANEL_ITEM_X, ATTR_COL(44),
                    PANEL_ITEM_Y, ATTR_ROW(2)+10,
                    PANEL_NOTIFY_PROC, output_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Clear", 6,
0),
                    PANEL_ITEM_X, ATTR_COL(53),
                    PANEL_ITEM_Y, ATTR_ROW(2)+10,
                    PANEL_NOTIFY_PROC, clear_proc, 0);


    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Quit", 5,
0),
                    PANEL_ITEM_X, ATTR_COL(20),
                    PANEL_ITEM_Y, ATTR_ROW(4)-6,
                    PANEL_NOTIFY_PROC, quit_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Read", 5,
0),
                    PANEL_ITEM_X, ATTR_COL(28),
                    PANEL_ITEM_Y, ATTR_ROW(4)-6,
                    PANEL_NOTIFY_PROC, read_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "New", 5, 0),
                    PANEL_ITEM_X, ATTR_COL(36),
                    PANEL_ITEM_Y, ATTR_ROW(4)-6,
                    PANEL_NOTIFY_PROC, new_proc, 0);

    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Train", 6,
0),
                    PANEL_ITEM_X, ATTR_COL(44),
                    PANEL_ITEM_Y, ATTR_ROW(4)-6,
                    PANEL_NOTIFY_PROC, train_proc, 0);


    panel_create_item(panel, PANEL_BUTTON,
PANEL_LABEL_IMAGE,
                    panel_button_image(panel, "Open", 6,
0),
                    PANEL_ITEM_X, ATTR_COL(53),
                    PANEL_ITEM_Y, ATTR_ROW(4)-6,
```

```
                    PANEL_NOTIFY_PROC, open_proc, 0);

}



static    void
train_proc()     /* input: file, file.pos. output: file.fea
*/
{
     Pixrect      *image;
     FILE         *fpi, *fopen();
     register int i=0, j=0, k=0;
     int          x, y;
     short        l;
     float        fea[128][10];
     Pixwin       *pw=canvas_pixwin(canvas);

     new_proc();

     sprintf(cmdstring, "%s.pos",
panel_get_value(finame_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     while(i<512)
          fscanf(fpi,"%hi", &posvect[i++]);
     fclose(fpi);


     for(i=0;i<20;i++)   {
          mean[0][i]=0;
          mean[5][i]=0;
          }
     i=0;
     while((x=posvect[i++])!=0)    {
          y=posvect[i++];
          pw_batch_on(pw);
          for(j=0;j<xsize+1;j++)
               for(k=0;k<ysize+1;k++){
                    pcbpix[j][k]=pw_get(pw, x-bx2+j, y-
by2+k);
               }
          pw_batch_off(pw);
          features(i,x,y);
          k=i/2;
          for(j=0;j<7;j++)    {
               if(ll[0][j]>feavect[j])  ll[0][j]=feavect[j];
               if(mm[0][j]<feavect[j])  mm[0][j]=feavect[j];
               mean[0][j]=mean[0][j]+feavect[j];
               fea[k][j]=feavect[j];
               }
```

```
          pw_batch_on(pw);
          for(k=0;k<ysize+1;k++)   {
               for(j=0;j<xsize+1;j++)   {
                    pw_put(pw, x-bx2+j, y-by2+k, 0);
                    }
               }
          pw_batch_off(pw);
     }


     k=i/2;
     for(i=0;i<7;i++)    {
          mean[0][i]=mean[0][i]/k;
          for(j=1;j<=k;j++)
               mean[5][i]=mean[5][i]+(fea[j][i]-
mean[0][i])*(fea[j][i]-mean[0][i]);
          mean[5][i]=(float)sqrt((double)mean[5][i]/(k-1));
          }
     sprintf(cmdstring, "%s.fea",
panel_get_value(f1name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");
     for(i=0;i<7;i++)
          fprintf(fpi,"%f ", mean[0][i]);
          fprintf(fpi,"   for %3d  samples\n", k);

     for(i=0;i<7;i++)
          fprintf(fpi,"%f ", mean[5][i]);
          fprintf(fpi,"\n");

     for(i=0;i<7;i++)
          fprintf(fpi,"%.4f%% ", mean[5][i]/mean[0][i]*100);
          fprintf(fpi,"\n");

     for(i=0;i<7;i++)
          fprintf(fpi,"%.4f ", mean[0][i]/mean[5][i]);
          fprintf(fpi,"\n");
     fclose(fpi);


     sprintf(cmdstring, "%s.fea",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "a");
     for(i=0;i<7;i++)
          fprintf(fpi,"%f ", mean[0][i]);
          fprintf(fpi,"\n");
     fclose(fpi);

     sprintf(cmdstring, "%s.lnm",
panel_get_value(fname_item));
```

```
     fv=&cmdstring[0];
     fpi = fopen(fv, "a");
     for(i=0;i<7;i++)    {
          fprintf(fpi,"%f ", ll[0][i]);
          ll[0][i]=1000;
          }
          fprintf(fpi,"\n");
     for(i=0;i<7;i++)    {
          fprintf(fpi,"%f ", mm[0][i]);
          mm[0][i]=0;
          }
          fprintf(fpi,"\n");
     fclose(fpi);
}



static    void
features(oi, ox, oy)     /* output: file.fea */
int  oi, ox, oy;
{
     FILE         *fpi, *fopen();
     register int i=0, j=0, k=0;
     short        l, m;
     float        midd,vtot, sigma2, vcsw, vofr, x, y, x2,
y2;
     float        sum;
     float        xcm, ycm, iaa, ibb, icc, iratio;
     float        ia, ib, ic, id, mij, zz;

     sprintf(cmdstring, "%c",
value_to_c(panel_get_value(weight_num)));
     m=cmdstring[0];

     sprintf(cmdstring, "%c",
value_to_d(panel_get_value(visual_infrared)));
     m=cmdstring[0];

if(m==105){


/* normalization */
     l=255;
     m=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)   {
               if(l>pcbpix[j][i])  l=pcbpix[j][i];
               if(m<pcbpix[j][i])  m=pcbpix[j][i];
               }
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
```

```
                sjpix[j][i]=(float)pcbpix[j][ı]/m;

/* Mean grey level, Normalized volume */
     sum=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
               sum=sum+sjpix[j][i];
     vtot=sum*delta;
     iaa=vtot;

/* Central subwindow mean */
     sum=0;
     for(j=bx2-cswx;j<by2+cswy+1;j++)
          for(i=bx2-cswx;i<by2+cswy+1;i++)
               sum=sum+sjpix[j][i];
     vcsw=sum*delta;


/* Outer frame region volume */
     sum=0;
     for(j=bx2-ofrx;j<by2+ofry+1;j++)
          for(ı=bx2-ofrx;i<by2+ofry+1;i++)
               sum=sum+sjpix[j][i];
     vofr=sum*delta;
     vofr=vtot-vofr;
     iratio=vofr;

/*middle frame region volume */
     midd=vtot-vcsw-vofr;
     ıcc=mıdd;

/* Variance of normalized grey level */
     sum=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
               sum=sum+(sjpix[j][i]-midd)*(sjpıx[j][i]-
mıdd);
     vtot=sum/((xsize+1)*(ysize+1)-1);

/* Central subwindow variance  */
     sum=0;
     for(j=bx2-cswx;j<by2+cswy+1;j++)
          for(i=bx2-cswx;i<by2+cswy+1;i++)
               sum=sum+(sjpix[j][i]-midd)*(sjpix[j][i]-
midd);
     vcsw=sum*delta;


/* Outer frame region variance */
     sum=0;
     for(j=bx2-ofrx;j<by2+ofry+1;j++)
```

```
          for(i=bx2-ofrx;i<by2+ofry+1;i++)
               sum=sum+(sjpix[j][i]-midd)*(sjpix[j][i]-
midd);
     vofr=sum*delta;
     sigma2=vofr-vcsw;
     vofr=vtot-vofr;

/* Inertia Features */

}


else{

/* normalization */
     l=255;
     m=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)   {
               if(l>pcbpix[j][i])  l=pcbpix[j][i];
               if(m<pcbpix[j][i])  m=pcbpix[j][i];
               }
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
               sjpix[j][i]=(float)pcbpix[j][i]/m;

/* Mean grey level, Normalized volume */
     sum=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
               sum=sum+sjpix[j][i];
     vtot=sum*delta;


/* Variance of normalized grey level */
     sum=0;
     for(j=0;j<xsize+1;j++)
          for(i=0;i<ysize+1;i++)
               sum=sum+(sjpix[j][i]-vtot)*(sjpix[j][i]-
vtot);
     sigma2=sum/((xsize+1)*(ysize+1)-1);

/* Central subwindow volume */
     sum=0;
     for(j=bx2-cswx;j<by2+cswy+1;j++)
          for(i=bx2-cswx;i<by2+cswy+1;i++)
               sum=sum+sjpix[j][i];
     vcsw=sum*delta;


/* Outer frame region volume */
```

```
     sum=0;
     for(j=bx2-ofrx;j<by2+ofry+1;j++)
          for(i=bx2-ofrx;i<by2+ofry+1;i++)
               sum=sum+sjpix[j][i];
     vofr=sum*delta;
     vofr=vtot-vofr;

/* Inertia Features */
     xcm=0;
     for(i=0;i<xsize+1;i++)
          for(j=0;j<ysize+1;j++)
               xcm=xcm+sjpix[i][j]*(i-bx2);
     xcm=xcm/vtot*delta;


     ycm=0;
     for(i=0;i<xsize+1;i++)
          for(j=0;j<ysize+1;j++)
               ycm=ycm+sjpix[i][j]*(j-bx2);
     ycm=ycm/vtot*delta;

     iaa=0;
     ibb=0;
     icc=0;
     for(i=0;i<xsize+1;i++)
          for(j=0;j<ysize+1;j++)   {
               x=xcm+bx2-i;
               y=ycm+by2-j;
               x2=x*x;
               y2=y*y;
               mij=2*delta*sjpix[i][j];
               zz=sjpix[i][j]*sjpix[i][j];
               ia=((4*zz*zz+delty2)/12+y2)*mij;
               ib=x*y*mij;
               ic=((4*zz*zz+deltx2)/12+x2)*mij;
               id=((deltx2+delty2)/12+x2+y2)*mij;
               x=ia*ib*ib;
               iaa=iaa+x;

               ibb=ibb+y-x;
               icc=icc+id;
               }
     iratio=(iaa+ibb)/2/icc;



}
     feavect[0]=vtot;
     feavect[1]=sigma2;
     feavect[2]=vcsw;
     feavect[3]=vofr;
```

```
     feavect[4]=iaa;
     feavect[5]=icc;
     feavect[6]=iratio;

}



static    void
sort_proc()    /* input: f, f.pos, f1.fea-f5.fea. output:
f.cls */
{
     Pixrect      *image;
     FILE         *fpi, *fopen();
     register int i=0, j=0, k=0;
     int          x, y, c, port[5];
     short        l;
     float        dist[5 ], dd;
     char         cc;
     Pixwin       *pw=canvas_pixwin(canvas);

     new_proc();

     sprintf(cmdstring, "weight%c.vec",
value_to_c(panel_get_value(weight_num)));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     for(i=0;i<7;i++)    {
          fscanf(fpi,"%f", &mean[0][i]);
          ll[0][i]=1000;
          mm[0][i]=0;
          }
     fclose(fpi);


     sprintf(cmdstring, "%s.fea",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     for(i=1;i<5;i++)    {
          port[i]=0;
          for(j=0;j<7;j++)    {
               fscanf(fpi,"%f", &mean[i][j]);
               }
          }
     fclose(fpi);

     sprintf(cmdstring, "%s.lnm",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
```

```
     for(j=1;j<5;j++)    {
          for(i=0;i<7;i++)    fscanf(fpi,"%f", &ll[j][i]);
          for(i=0;i<7;i++)    fscanf(fpi,"%f", &mm[j][i]);
          }
     fclose(fpi);


     for(j=0;j<7;j++)
          for(i=1;i<5;i++)    {
               if(ll[0][j]>ll[i][j])    ll[0][j]=ll[i][j];
               if(mm[0][j]<mm[i][j])    mm[0][j]=mm[i][j];
               }

     for(i=1;i<5;i++)
          for(j=0;j<7;j++)
               mean[i][j]=mean[i][j]/mm[0][j];

     i=0;
     sprintf(cmdstring, "%s.pos",
panel_get_value(f1name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     while(i<512)
          fscanf(fpi,"%hi", &posvect[i++]);
     fclose(fpi);

     sprintf(cmdstring, "%s.out",
panel_get_value(f1name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");


     i=0;
     while((x=posvect[i++])!=0)    {
          y=posvect[i++];
          pw_batch_on(pw);
          for(j=0;j<xsize+1;j++)
               for(k=0;k<ysize+1;k++){
                    pcbpix[j][k]=pw_get(pw, x-bx2+j, y-
by2+k);
                    }
          pw_batch_off(pw);

          features(i,x,y);

          for(j=0;j<7;j++)
               feavect[j]=feavect[j]/mm[0][j];


          for(k=1;k<5;k++)    {
               dist[k]=0;
```

```
                for(j=0;j<7;j++)    {
                     dd=(mean[k][j]-feavect[j])*mean[0][j];
                     dist[k]=dist[k]+dd*dd;
                     }
                dist[k]=sqrt((double)dist[k]);
                }
           dd=1000;
           for(k=1;k<5;k++)    {
                if(dd>dist[k])      {
                     dd=dist[k];
                     c=k;
                     }
                }
           port[c]++;
           if(c==1)  cc='1';
           if(c==2)  cc='2';
           if(c==3)  cc='3';
           if(c==4)  cc='4';
           fprintf(fpi, "%3d %3d %.4f %.4f %.4f %.4f    %c
\n",
           x, y, dist[1], dist[2], dist[3], dist[4], cc);
           pw_batch_on(pw);
           for(j=0;j<xsize+1;j++)
               for(k=0;k<ysize+1;k++)
                   pw_put(pw, x-bx2+j, y-by2+k, 0);
           pw_batch_off(pw);
           }
     fclose(fpi);


     sprintf(cmdstring, "%s.sot",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "a");
     k=i/2;
     for(i=1;i<5;i++)
          fprintf(fpi, "    %d  %3.2f%%", i,
(float)port[i]/k*100);
     fprintf(fpi, "   for %3d  samples\n", k);
     fclose(fpi);

}



static    void
size_proc()
{
     xsize=value_to_op(panel_get_value(box_xsize));
     ysize=value_to_op(panel_get_value(box_ysize));
     bx2=xsize/2;
```

```
    by2=ysize/2;
    cswx=(int)((float)bx2*.3);
    cswy=(int)((float)by2*.3);
    ofrx=(int)((float)bx2*.7);
    ofry=(int)((float)by2*.7);

    deltax=(float)1/(float)(xsize+1);
    deltay=(float)1/(float)(ysize+1);
    delta=deltax*deltay;
    deltx2=deltax*deltax;
    delty2=deltay*deltay;

}


static int
value_to_op(value)
register int   value;
{
    switch (value) {
     case 0: return 8;
     case 1: return 12;
     case 2: return 14;
     case 3: return 16;
     case 4: return 18;
     case 5: return 20;
     case 6: return 22;
     case 7: return 24;
     case 8: return 30;
     case 9: return 36;
     case 10: return 40;
    }
}


static char
value_to_c(value)
register int   value;
{
    switch (value) {
     case 0: return '1';
     case 1: return '2';
     case 2: return '3';
     case 3: return '4';
     case 4: return '5';
     case 5: return '6';
     case 6: return '7';
     case 7: return '8';
     case 8: return '9';
    }
}
```

```
static char
value_to_d(value)
register int   value;
{
    switch (value) {
     case 0: return 'i';
     case 1: return 'v';
    }
}

static char
value_to_e(value)
register int    value;
{
    switch (value) {
        case 0: return 1;
        case 1: return 2;
        case 2: return 3;
        case 3: return 4;
        case 4: return 5;
        case 5: return 6;
        case 6: return 7;
        case 7: return 8;
        case 8: return 9;
    }
}


static    void
help_proc(name)
char *name;
{
     fprintf(stderr, "This program is %s", name);
     frame_cmdline_help(name);
}

static    void
quit_proc()
{
     window_set(frame, FRAME_NO_CONFIRM, TRUE, 0);
     window_destroy(frame);
}


static    void
ls_proc()
{
     char cmdstring[256];
     sprintf(cmdstring, "ls -l %s*\n",
panel_get_value(fname_item));
```

```
     ttysw_input(tty, cmdstring, strlen(cmdstring));
}

static    void
pos_proc()
{
     int  x, y;

     x=(int)(LINT_CAST(panel_get_value(cursor_x)));
     y=(int)(LINT_CAST(panel_get_value(cursor_y)));
     window_set(canvas, WIN_MOUSE_XY, x, y, 0);
}



static void
fuse_proc(item, event)
Panel_item     item;
Event          *event;
{
     FILE         *fpi, *fopen();
     register int i=0, j=0, k=0;
     int          x, y, c, port[5];
     short     l;
     float     dist[5], dd;
     char cc;
     Pixwin    *pw=canvas_pixwin(canvas);

     vec=value_to_e(panel_get_value(fuse_vec));
     vec=vec/10;
     for(j=0;j<512;j++)  {
        data[j]=0;
        data1[j]=0;
        }


     sprintf(cmdstring, "%s.out",
panel_get_value(f1name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     while(i<512)
          fscanf(fpi,"%f", &data[i++]);
     fclose(fpi);
     i=0;
     sprintf(cmdstring, "%s.out",
panel_get_value(f3name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     while(i<512)
          fscanf(fpi,"%f", &data1[i++]);
     fclose(fpi);
```

```
     sprintf(cmdstring, "%s.fus",
panel_get_value(f1name_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");

     i=0;
     while((dd=data[i])!=0)  {
          x=(int)data[i++];
          y=(int)data[i++];
          for(k=1;k<5;k++)   {
              dist[k]=vec*data[i]+(1-vec)*data1[i];
              i++;
          }
              i++;
              dd=1000;
              for(k=1;k<5;k++)     {
                 if(dd>dist[k])    {
                   dd=dist[k];
                   c=k;
                   }
              }
              port[c]++;
              if(c==1)   cc='1';
              if(c==2)   cc='2';
              if(c==3)   cc='3';
              if(c==4)   cc='4';


              fprintf(fpi, "%3d %3d %.4f %.4f %.4f %.4f
%c \n",
              x, y, dist[1], dist[2], dist[3], dist[4], cc);
}
              fclose(fpi);
}

static    void
draw(canvas_local)
Canvas    canvas_local;
{
     struct pixrect *image;
     FILE           *fpi,  *fopen();
     colormap_t     *colormap=NULL;
     Pixwin         *pw=canvas_pixwin(canvas_local);

     sprintf(cmdstring, "%s", panel_get_value(fname_item));
     fv=&cmdstring[0];
     rh.ras_magic = RAS_MAGIC;
     rh.ras_width = 512;
     rh.ras_height = 484;
     rh.ras_depth = 8;
```

```
     rh.ras_length = 512*484;
     rh.ras_type = RT_STANDARD;
     rh.ras_maptype = RMT_NONE;
     rh.ras_maplength = 512;
     fpi = fopen(fv, "r");
     image = pr_load_std_image(fpi, &rh, colormap);
     pw_write(pw, 0, 0, 512, 484, PIX_SRC^PIX_DST, image, 0,
0);

}


static    void
read_proc()
{
     FILE           *fpi, *fopen();
     register short i=0;
     short          x, y, k;
     Pixwin         *pw=canvas_pixwin(canvas);
     i=pos;
     sprintf(cmdstring,"%s.pos",panel_get_value(finame_item)
);
     fv=&cmdstring[0];
     fpi = fopen(fv, "r");
     while(i<512)   {
          fscanf(fpi,"%hi", &posvect[i++]);
          }
     fclose(fpi);
     i=0;
     while(posvect[i]!=0 || posvect[i+1]!=0) i+=2;
     pos=i;


     i=0;
     while(posvect[i]!=0 || posvect[i+1]!=0) {
          x=posvect[i++];
          y=posvect[i++];
          pw_batch_on(pw);
          pw_vector(pw, x-bx2, y-by2, x+bx2, y-by2, PIX_SRC,
255);
          pw_vector(pw, x+bx2, y-by2, x+bx2, y+by2, PIX_SRC,
255);
          pw_vector(pw, x-bx2, y-by2, x-bx2, y+by2, PIX_SRC,
255);
          pw_vector(pw, x-bx2, y+by2, x+bx2, y+by2, PIX_SRC,
255);
          pw_batch_off(pw);
          }
     window_set(canvas, WIN_MOUSE_XY, 256, 242, 0);
}
```

```
static    void
clear_proc()
{
     FILE           *fpi, *fopen();
     register short i=0;
     short          x, y, k;

     sprintf(cmdstring, "%s.err",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");
     fclose(fpi);

}


static    void
open_proc()
{
     FILE           *fpi, *fopen();
     register short i=0;
     short          x, y, k;

     sprintf(cmdstring, "%s.fea",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");
     fclose(fpi);

     sprintf(cmdstring, "%s.lnm",
panel_get_value(fname_item));
     fv=&cmdstring[0];
     fpi = fopen(fv, "w");
     fclose(fpi);

     for(i=0;i<20;i++)   {
          ll[0][i]=1000.0;
          mm[0][i]=0;
          }
}


static    void
new_proc()
{
     register int   i=0;
     Pixwin         *pw=canvas_pixwin(canvas);

     while(i<512)
          posvect[i++]=0;
```

```
     pos=0;
     pw_writebackground(pw, 0, 0, 512, 484, PIX_SRC);
     draw(canvas);
}



static void
handle_event(canvas_local, event, arg)
Canvas    canvas_local;
Event     *event;
caddr_t   arg;
{
     Pixwin       *pw=canvas_pixwin(canvas);
     register int i=0, j=0, k;
     short        x, y;

     if (event_is_up(event))
          return;
     x=event_x(event);
     y=event_y(event);
     k=event_id(event);


     switch(k){
          case 'w':
               y--;
               j=1;
               break;
          case 'a':
               x--;
               j=1;
               break;
          case 's':
               x++;
               j=1;
               break;
          case 'z':
               y++;
               j=1;
               break;
          case 'e':
               while(posvect[i]!=0 || posvect[i+1]!=0) i+=2;
          posvect[i++]=x;
          posvect[i++]=y;
          pw_batch_on(pw);
          pw_vector(pw, x-bx2, y-by2, x+bx2, y-by2, PIX_SRC,
255);
          pw_vector(pw, x+bx2, y-by2, x+bx2, y+by2, PIX_SRC,
255);
          pw_vector(pw, x-bx2, y-by2, x-bx2, y+by2, PIX_SRC,
```

```
255);
          pw_vector(pw, x-bx2, y+by2, x+bx2, y+by2, PIX_SRC,
255);
          pw_batch_off(pw);
          break;


          case 'q':
               while(posvect[i]!=0 || posvect[i+1]!=0) i+=2;
          i--; k=0;
          while(i>0)     {
               if(abs(y-posvect[i]) < bx2 && abs(x-
posvect[i-1]) < bx2) {
                    k=i; i=0;
               }
               else i-=2;
               }
          if(k!=0)  {
               while(k<254)   {
                    posvect[k-1]=posvect[k+1];
                    posvect[k]=posvect[k+2];
                    k+=2;
                    }
               pw_writebackground(pw, 0, 0, 512, 484,
PIX_SRC);
               draw(canvas);
               while(posvect[i]!=0 || posvect[i+1]!=0) {
                    x=posvect[i++];
                    y=posvect[i++];
                    pw_batch_on(pw);
                    pw_vector(pw, x-bx2, y-by2, x+bx2, y-
by2, PIX_SRC, 255);
                    pw_vector(pw, x+bx2, y-by2, x+bx2,
y+by2, PIX_SRC, 255);
                    pw_vector(pw, x-bx2, y-by2, x-bx2,
y+by2, PIX_SRC, 255);
                    pw_vector(pw, x-bx2, y+by2, x+bx2,
y+by2, PIX_SRC, 255);
                    pw_batch_off(pw);
                    }
                }
          break;


          default:
               break;
          }
     if(j==1)  {
          window_set(canvas, WIN_MOUSE_XY, x, y, 0);
          return;
          }
```

```
     switch (event_action(event)) {
        case MS_LEFT:
        case MS_MIDDLE:
        case MS_RIGHT:
         /* translate the event to window space,
          * then show the menu.
          */
         (void) menu_show(menu, canvas_local,
                    canvas_window_event(canvas_local,
event), 0);
         break;

         default:
         break;
     }

}


static    void
output_proc()
{
     FILE          *fpi, *fopen();
     register int  i=0, k;
     int           o, p, r;
     short         m, n, l, q;

     while(posvect[i]!=0 || posvect[i+1]!=0) i+=2;
     p=i-1;

     o=1;
     r=1;


while(o<p)     {
     l=posvect[o];
     for(i=o+2;i<=p;i+=2)
          if(abs(posvect[i]-l)>20) break;
     r=i;
     for(i=r+2;i<=p;i+=2)     {
          if(abs(posvect[i]-l)<=20)     {
               m=posvect[i-1];
               n=posvect[i];
               posvect[i-1]=posvect[r-1];
               posvect[i]=posvect[r];
               posvect[r-1]=m;
               posvect[r]=n;
               r+=2;
               }
```

```
        }
    for(i=o;i<r;i+=2)   {
        q=posvect[i-1];
        for(k=i+2;k<r;k+=2)
            if(posvect[k-1]<q)  {
                m=posvect[k-1];
                n=posvect[k];
                posvect[k-1]=posvect[i-1];
                posvect[k]=posvect[i];
                posvect[i-1]=m;
                posvect[i]=n;
                q=m;
                }
        }
    o=r;
    }


    i=0;
    sprintf(cmdstring, "%s.pos",
panel_get_value(finame_item));
    fv=&cmdstring[0];
    fpi = fopen(fv, "w");
    while(posvect[i]!=0 || posvect[i+1]!=0) {
        fprintf(fpi,"%d  ",posvect[i++]);
        fprintf(fpi,"%d\n",posvect[i++]);
        }
    fclose(fpi);
}
```

B Interpolation C Source Code

```
/* Filename $1 $2  */

#include <stdio.h>

int   mmax;
float x,y;
main (argc,argv)
int   argc;
char  **argv;
{
     int  i,j,k,l,tempi;
     unsigned char m[512][512],tempc;
     FILE *file1,*file2,*file3,*file,*fopen(),*fclose();

     if (argc != 3) {
          fprintf(stderr,"Usage: can't open file
%s\n",argv[0]);
          exit(1);
     }


     if ((file1 = fopen(argv[1],"r")) == NULL) {
          fprintf(stderr,"\007 can't open file
%s\n",argv[1]);
          exit(1);
     }

     if ((file2 = fopen(argv[2],"w")) == NULL) {
          fprintf(stderr,"\007 can't open file
%s\n",argv[2]);
          exit(1);
     }
     mmax=255;
     fread(m,sizeof(char),512*512,file1);
     for(i=4;i<510; i++){
        for(j=0; j<510; j++){
           if(m[i][j]>0)
              x=0;
           else if(m[i+1][j]>0)
                  x=0;
                else{
                   m[i][j]=(m[i-1][j]+m[i+2][j])/2;
                   m[i+1][j]=m[i][j];
                   }
           }
     }
     fwrite(m,sizeof(char),512*512,file2);
```

```
    fclose(file1);
    fclose(file2);
}
```

C Histogram Transformation C Source Code

```
/* Filename $1 $2  */

#include <stdio.h>

int    mmax;
float  x,y;
main ( argc,argv )
int    argc;
char   **argv;
{
     int     i,j,k,l,tempi;
     unsigned char m[512][512],tempc;
     FILE    *file1,*file2,*file3,*file,*fopen(),*fclose();

     if (argc != 3) {
          fprintf(stderr,"Usage: can't open file
%s\n",argv[0]);
          exit(1);
     }


     if ((file1 = fopen(argv[1],"r")) == NULL) {
          fprintf(stderr,"\007 can't open file
%s\n",argv[1]);
          exit(1);
     }

     if ((file2 = fopen(argv[2],"w")) == NULL) {
          fprintf(stderr,"\007 can't open file
%s\n",argv[2]);
         exit(1);
     }
     mmax=255;
     fread(m,sizeof(char),512*512,file1);
     for (i=0;i<512; i++)
        for (j=0; j<512;j++){
            if ( m[i][j] > 250) m[i][j] = 250;
            y=(float)m[i][j];
            x=250*(y-140)/(250-140);
            if(x<10)x=10;
            if(x>255)x=250;
            m[i][j]=(int)x;
            if(m[i][j]<0) m[i][j]=0;
            }
     fwrite(m,sizeof(char),512*512,file2);

     fclose(file1);
     fclose(file2);
}
```

D Weight.vec Files

```
weight1.vec
1 1 1 1 1 1 1

weight2.vec
0.7 0.7 0.7 0.7 0.3 0.3 0.3

weight3.vec
0.3 0.3 0.3 0.3 0.7 0.7 0.7


weight4.vec
8.8739 0 9.4205 11.6631 0 10.1209 0

weight5.vec
1 0 1 1 0 1 0

weight6.vec
1 1 1 1 0 1 0
```