RSD-TR-16-84

AUTOMATIC VISUAL SOLDER JOINT INSPECTION[1]

Paul Besl

Edward Delp[2]

Ramesh Jain

Electrical Engineering and Computer Science

The University of Michigan

Ann Arbor, Michigan 48109-1109

October 1984

CENTER FOR ROBOTICS AND INTEGRATED MANUFACTURING

Robot Systems Division

COLLEGE OF ENGINEERING

THE UNIVERSITY OF MICHIGAN

ANN ARBOR, MICHIGAN 48109-1109

---

[1]This work was supported by IBM Corporation, Data Systems Division, Kingston, New York, under the monitorship of Dr. Jack Contino.

[2]Now with Electrical Engineering, Purdue University, West Lafayette, Indiana

TABLE OF CONTENTS

1. INTRODUCTION 3

2. REVIEW OF PREVIOUS WORK 7

3. QUALITATIVE DESCRIPTION OF INSPECTION TECHNIQUE 8

3.1. Assumptions 9

3.2. Dataflow 11

3.3. Solder Joint Feature Selection 14

3.4. Solder Joint Class Selection 14

4. QUANTITATIVE DESCRIPTION OF FEATURES AND CLASSIFICATION 17

4.1. Brightness and Size Normalization 17

4.2. Basic Features 19

4.3. Inertia Features 20

4.4. Faceted Surface Area Feature 24

4.5. Gaussian Curvature Related Features 26

4.6. Minimum-Distance Classification Algorithm 31

4.7. Performance Measurement 34

5. EXPERIMENTAL RESULTS 36

6. FUTURE RESEARCH DIRECTIONS 39

6.1. Better Feature Selection 39

6.2. Better Classification Algorithms 40

6.3. Multiple View Classification 40

6.4. Sensitivity Analysis 41

6.5. Registration and Calibration Analysis 41

7. REFERENCES 42

Abstract

This paper describes an approach for the automatic inspection of solder joints on printed circuit boards. We identify common defects in solder joints and classify a joint as being good or belonging to one of the defective classes. This classification, rather than just detection of defective joints, is motivated by our desire to automatically take corrective action on the assembly line. The features used for classification are based on characteristics of *intensity surfaces*. It is shown that features derived from *facets* and Gaussian curvature are effective in the classification of solder joints using a minimum-distance classification algorithm. We also demonstrate the use of class separation plots to quickly study individual effectiveness of a feature or pair of features in classification. Our results show the efficacy of our approach.

1. INTRODUCTION

Computer vision and image processing techniques have been applied to many industrial inspection tasks [4]. In most applications, it is only necessary to determine whether a part is acceptable or not. This decision is usually performed based on the presence or absence of sub-parts, size of sub-parts, or surface finish. It is commonly believed that the most difficult inspection tasks are those requiring inspection of surface finish. Most early systems depended on binary images and were incapable of addressing this problem; later systems started using gray images but still could not address this problem successfully. The major problem in inspection of surface finish is that one should either have depth information at close points, or should exploit subtle variations in intensity values. Photometric stereo has been applied to study surface defects [19].

In most applications, attention has been given to inspection as a problem separate from the manufacturing process. It appears that just the detection of a fault is not enough if we want to develop a completely automatic manufacturing system. It may be required to find the type of defect and to decide the reason for this defect. If it is found that there is a particular stage in manufacturing that may be responsible for the defect, then a corrective action is in order. Clearly, for such a system, just detection of fault is not enough. The research described in this paper, although it addresses only the multiple unacceptable class inspection problem, is related to automated soldering of electronic components onto printed circuit boards.

The solder joint inspection problem is more challenging than many other visual inspection problems. It is a difficult area to apply computer vision techniques for

automatic inspection because of the variability in the appearance of acceptable solder joints. We have found that approaches based on edge detection or template matching are not adequate for our goals. The structured light and binary vision approaches used by others have provided only short-term narrowly-applicable solutions to this problem. Moreover, in some applications it is not enough to say that a joint is no good; a bad joint should be classified in one of several fault classes. Visual appearence of many classes of defects is quite similar and poses a challenging problem even to human inspectors. Laser heating inspection techniques have been implemented [17], but are often not desired because of possible damage to solder joints. A general approach of the type we are pursuing, which is based on gray-scale images, will hopefully provide a long-term widely-applicable solution to the inspection problem. If it is possible to duplicate the inspection capabilities of human inspectors, gray-scale computer vision will also provide *quantitatively consistent* and analyzable performance. It may even be possible to inspect solder joints faster than human inspectors. Figure 1 provides a table of all the solder joint defects which we would like to be able to detect automatically. Of course, it would be a significant result if we could reliably discriminate between ten or so of these defects.

Faulty solder joints are very costly defects for electronic equipment vendors and customers both. Woodgate [18] estimates that even a solder joint failure rate as low as 0.05% can cost a large manufacturing organization over three million dollars per year. Any effective techniques to improve solder joint inspection will be welcomed by industry and should have widespread beneficial results.

The ultimate goal of this research project is to demonstrate the feasibility of an automatic visual solder joint inspection system. To show feasibility without building an entire inspection system, we have designed, constructed, and tested a preliminary image analysis and classification section of the overall system which is diagrammed in Figure 2. A simple minimum-distance classifier has been implemented to discriminate between good and bad solder joints and among defect classes. We have used a variety of solder joint classes and a large collection of different features in our experiments. A person who was not familiar with the classification algorithm classified 1418 solder joint subimages for our experiments. The training and classification programs were executed using the input data generated by this person. The probability of missing a bad solder joint averaged approximately 6% while the probability of a false alarm on a good solder joint was in the neighborhood of 7% during these experiments. Both error probabilities have been 0% on several particular images. The average probability of a correct good/bad decision is then approximately 87% and has run as high as 98.4% for particular images. Better average results (around 94%) were obtained when one of the authors performed the preliminary human classification, but we wanted to verify the robustness of the algorithm. The detailed results will be described in Section 5 whereas the computational details are described in Sections 3 and 4 of this report. It should be emphasized that these numerical results apply to *only one* test printed circuit board which has many different kinds of defective solder joints. See Figures 4 and 5. Also, the test board is only somewhat like the real PC boards which will be encountered in practice, but we expect that the same technique would work as well or even better on a real board. We hope to test this in the near future. We did not determine a receiver

operating characteristic (ROC) curve for this defect detector because no maximum likelihood threshold was being used. We also wish to note that we are using one of the *simplest* possible classification schemes to establish a base level of performance for *evaluating* more *advanced* classification techniques. We have attempted to make the system as modular and as flexible as possible so that different feature lists, different class lists, and different classification algorithms can be tried in a short amount of time with only minor effort. The results we have obtained with our preliminary system are very encouraging.

The features used in our system are based on the characteristics of the *intensity surfaces* of solder joints. Some of the feature models are very close to the *facet model* proposed by Haralick [8]; other features, which are related to Gaussian curvature, are based on differential geometry of surfaces. Thus, we consider intensity as the third dimension (combined with x and y spatial directions) and study the characteristics of the surfaces in a solder joint image to classify them. The motivation for this approach is simple. Intensity values in an image are a function of the scene illumination, the visible surface geometry, and the surface reflectance. In an industrial inspection situation, we are relatively free to demand a fixed orientation of parts and a constant scene illumination. In the solder joint inspection problem, we know that the surface reflectance of the solder alloy will be fairly uniform. Hence, in each solder joint subimage taken from a particular fixed point of view, the only free variable in the intensity function is the solder joint surface geometry. This means that, under these constraints, the *variations* in the *intensity surface* of a solder joint are functionally related to the *variations* in the *physical solder joint surface*. Rather than worry about the details of this

functional dependence, we empirically note that all solder joints of a given class have similar intensity surface structure which is related to the fact that they have similar physical surfaces. For example, compare the different solder joint gray level image surfaces shown in Figure 3. One joint is definitely good and the other is definitely bad in this case. Therefore, we look for features of the intensity surface which characterize its shape and yield the most effective class discrimination.

2. REVIEW OF PREVIOUS WORK

Although a great deal of research has been done in the area of printed wiring board inspection [6] [10] [15] [16] [20], very little work has addressed the inspection of solder joints. This is probably due to the fact that relatively simple two-dimensional processing techniques can be used to inspect printed wiring patterns whereas solder joint inspection involves an extra degree of freedom. At least two other investigators have explored the use of automatic visual processing to inspect solder joints.

Nakagawa [13] at the Hitachi Production Engineering Research Lab has reported on a structured lighting approach. The advantages of this approach are the following: 1) The image processing/signal processing is very fast, 2) Results are not influenced by the high gloss of the solder joints, 3) Range information is produced, 4) Only two features are needed to discriminate among four classes, 5) Perfect performance is claimed for 483 solder joints. The disadvantages of this approach are also listed: 1) Only profiles of the solder joint are used and it seems possible that small but important defects can be missed; 2) Only good/bad decisions are made - the results indicate a lack of separation between the unacceptable classes, 3) Waveform cleanup operations are

required which could distort critical information, 4) Potential speedup of the inspection process is limited because different profiles are digitized by x-y table motion. This approach seems practical and worthwhile but also limited. His processing algorithms are different for IC and discrete part solder joints, and it seems that this approach would not generalize easily to handle other shapes which will be encountered when inspecting PC boards using surface mounting.

McIntosh [12] at the Honeywell Production Technology Lab reports on a special lighting/bounding box/peround/hole technique. The advantages of this technique are the following: 1) It is fast because it uses a binary vision approach to feature computation, 2) It is relatively inexpensive, 3) 99.8% flaw detection is claimed with less than 1% false alarm rate. The disadvantages are listed: 1) Binary vision may not be robust enough for all kinds of defects (see Figure 1), and 2) Post-processing is required to distinguish between regular and oversize/irregular-shape joints. The use of a reference file with the size and x-y locations of solder joints (similar to what we are using) is described in the paper. This practical approach also seems worthwhile but limited. The binary vision features seem likely to overlook some important details of the solder joint surface. We also expect that this type of approach will have difficulty generalizing to handle the new surface mounting technology that may replace solder joints in the near future.

3. QUALITATIVE DESCRIPTION OF INSPECTION TECHNIQUE

In this section, we present a qualitative discussion of our currently implemented inspection algorithm and how various human decisions influence it. The next section

will cover the quantitative formulas used to compute feature vectors and classification. First, we will point out the assumptions necessary for our technique to solve the solder joint inspection problem as opposed to the general computer vision/image understanding problem.

3.1. Assumptions

Any automated solder joint inpection system must be able to relate regions of particular images to individual solder joints on a printed circuit board or else it will be incapable of determining which solder joint on a given PC board is defective. This "bookkeeping" mechanism must know the x-y location and size of every solder joint on a PC board and where the subimage of that solder joint will occur in a digitized image. We assume that the joint size and location information in board coordinates will be available from a PC board design/manufacturing database for each type of board to be inspected. (If this information is not available, it is possible to determine joint size and location automatically using image processing algorithms to segment the PC board into joints and background. However, this computed information would not be as accurate as the manufacturing information and is, therefore, not preferred.) The distance to the camera and other camera parameters will determine the size in pixels of various joints. It is also assumed that a precision x-y table will tell us which solder joints are contained in a given image using the current x-y readout and the initial x-y readout. With the appropriate registration and calibration algorithms, we can segment every image into solder joint subimages and background PC board. This is the fundamental assumption upon which all our algorithms are based. This assumption is

certainly not valid in the general computer vision problem but can be justified for industrial inspection purposes.

We also assume that lighting will be sufficiently diffuse, intense, and uniform during inspection so that saturating bright specular reflections are avoided and all solder joint surfaces are adequately illuminated. Our laboratory experience and the images in Figures 4 and 5 indicate that this is a reasonable assumption. We do not require structured, colored, or other special lighting.

The classification method which we have been using classifies registered subimages via the computed characteristics of the gray level surface. These subimages could be pictures of anything in principle as long as we assume that subimages from different classes are enough different and subimages from the same class are similar enough that reliable classification decisions can be made. The notions of similarity and difference are with respect to the representative features which we compute from the subimages which characterize the gray level surface. This assumption may or may not be reasonable depending upon the list of classes of solder joints, the list of features to be extracted from the subimages, and the mathematical details of the classification algorithm. Stated differently, inspection performance depends upon the class list, the feature list, and the classification algorithm as well as the actual data. No matter how reliable an inspection algorithm is, there is always the possibility that it can be made more reliable by the appropriate changes in the feature list, the class list, or the algorithm (unless of course it is already perfect).

3.2. Dataflow

Given an image of solder joints on a PC board, the first step in our processing is to segment the image into solder joint subimages and background. In practice, we expect to have a program which would read the PC board manufacturing/design database and generate the appropriate data we need for classification. Since we do not have access to such a database, we currently generate the data needed for classification *and* training manually. This is done using the solder joint segmentation (SJS) program which we have implemented. This program operates as follows: 1) The user selects the appropriate size box so that one solder joint exactly fits the box. 2) The user interactively moves the box cursor so that it surrounds each solder joint, one at a time. 3) The user gives each solder joint a classification based on his or her own *opinion* while the box cursor surrounds it. 4) The user exits after all appropriate solder joints have been classified. The SJS program labels each joint and draws a box around it as the classification/segmentation proceeds so that the user knows which joints he or she has already classified. All information is logged in the SJS file for that image. Each solder joint has an identification number, an x-y pixel location, a size in pixels, and the presumably correct classification of the joint. See Figure 6 for an example of an SJS file. Figures 7 and 8 show how the SJS file information is used to label solder joints in the image. In the working inspection system, this file would be written automatically from the design/manufacturing database using another program and would not contain human inspector classifications.

A subset of representative solder joints is selected from each image's SJS file and set aside as a *training* SJS file. The training SJS file and the associated image are

presented as input to the training program. Each class is handled one at a time. The SJS file is scanned for all joints of a given class. For each joint of the given class, we extract the subimage from the original image using the segmentation data and compute all candidate features. The mean, standard deviation, maximum, and minimum of each feature are computed for each class as the training program executes. The training program outputs a mean feature vector for each class in the form of a mean feature/class (MFC) matrix. It also outputs the mean, standard deviation, maximum, and minimum information for each class to another file so that class separation graphs can be plotted. In summary, for each image of a section of a PC board, we are currently associating it with a SJS file, a SJS training file, a mean feature/class matrix, and a mean/std.dev./max/min file.

The class separation graphs mentioned above allow the user to get a quick idea of how good a particular feature or pair of features are after the training phase. It will also allow us to apply heuristic decision rules concerning the extracted features in future tests. Examples of class separation plots are shown in Figures 9 and 10. In Figure 9, the box height for each class is determined by the reciprocal variance of the feature for that class. The width is determined by the maximum and minimum feature value of the class. The mean value is denoted by a black line, and the half-height box shows the standard deviation about the mean. The ideal feature is one where the class boxes are tall and do not overlap (only one feature would be needed in this case). In Figure 10, the maximum and minimum of each feature determine the size of the box. The small class-labeled box is the mean-mean point whereas the intermediate size boxes show the standard deviations. The ideal pair of features has small

non-overlapping class boxes (only one pair would be needed in this case).

After the training phase, the classification processing can begin. The classification program accepts the original image, the associated SJS file, and the mean feature/class matrix as input. A feature weight file must also be specified as input, but it does not directly correspond to any particular image. It allows us to scale feature values so that we can emphasize or ignore particular features quickly and easily. The classification program processes each solder joint in the SJS file by extracting the given solder joint from the original image, computing all features, weighting all features, and computing a distance metric to all class mean feature vectors. The class with the minimum distance metric is assigned as the classification of the given solder joint. We compare the computed classification with the previously assigned human inspector classification and note all incorrect and correct classifications, all missed bad joints, all false alarms on good joints. After each classification run, a summary of class correctness, false alarm rate, miss rate, and good/bad decision correctness results is written out along with the name of the mean feature/class matrix file and the feature weighting file which were used during program execution. If the results are not satisfactory, the feature weighting file can be modified and another classification run can be executed. The flow of data between the segmentation program, the training program, and the classification program is summarized in the dataflow diagram in Figure 11.

3.3. Solder Joint Feature Selection

The selection of good numerical features is critical to the success of a classification algorithm. A good solder joint inspection feature is a quantitative function with the following qualities: 1) It can be easily and quickly computed from a solder joint subimage, 2) The values which the function takes on should cluster close together for all subimages of a given class, and 3) The clusters of values for different classes should be spaced far apart. Usually, it is very difficult to find such features.

Classification is a relatively simple process when the dimensionality of the classification information is small. Therefore, we try to compute a few numbers which best represent the images from which they were computed. There is no general existing theory which tells one which features to compute given the data one wishes to classify. The feature list is usually determined by the person designing the classification algorithm. However, given a large set of existing features, there are ways to decide which features are better than others. These methods are discussed at length in the statistical pattern recognition literature [5] [7]. We have attempted to utilize some of the covariance matrix techniques, but we have not yet assembled a large enough database to yield reliable statistical results.

3.4. Solder Joint Class Selection

The selection of good classes is also very important to the success of a classification algorithm. For instance, we could decide to only use two classes: acceptable and unacceptable. With our minimum-distance classifier, we would average all the features over all members of the two classes and compute the closest mean vector to

decide the classification. We have not done this because most of the features we are currently computing exhibit the following characteristic: some joint classes such as holes, which are unacceptable, have feature values which generally lie below the acceptable feature values while other joint classes such as fills and excess solder have feature values which generally lie above the acceptable feature values. The overall means for acceptable and unacceptable joints, therefore, lie fairly close together with the distribution of values spread out over a wide range. This was easily seen using the class separation graphs. The twelve member class list given below and the training SJS file of a particular image were used for these graphs. In this set of plots, the height of the main box for a given class is determined by the reciprocal standard deviation of the feature for that class. The width of the box is determined by the maximum and minimum values of the feature taken on by solder joint subimages of that class. A smaller box which has half the height of the main box shows the range of the standard deviation of the feature for the given class. Each class is labeled above the main box at the mean point of the class. Individual features are described in detail in section 4.

We have generally followed the philosophy that joints which look similar from a given view should be classified together. Therefore, we used the following 12 member class list:

A = Acceptable (dark to medium)

B = Acceptable (medium bright to bright)

C = Cup-shaped fill

D = Disturbed or Deformed solder

E = Excess solder (bulbous shape)

F = Fill

H = Hole (no lead)

I = Insufficient solder

N = No solder (with lead)

O = Outgassed joint

P = Pitted joint (small holes in fillet)

U = Unknown miscellaneous unacceptable class

The performance of the classifier was not bad with these 12 joint classes as will be discussed in section 5. Figures 9 and 10 demonstrate the separation and overlap of classes for two different features and two different pairs.

Better classification has been obtained for some images using a smaller modified six member class list which combines the above 12 classes:

A' $\Leftarrow$ { A, B }

C' $\Leftarrow$ { C }

E' $\Leftarrow$ { E, D }

F' $\Leftarrow$ { F }

H' $\Leftarrow$ { H }

N' $\Leftarrow$ { N, I, O, P, U }

The advantages of a smaller class list are the following: 1) Less computation to determine joint class during classification, and 2) Better separation of class means. The disadvantages of a smaller class list are 1) Less information is available concerning what is wrong with a given joint, and 2) Class variances tend to increase. Trade-off

decisions must be made concerning the advantages and disadvantages.

4. QUANTITATIVE DESCRIPTION OF FEATURES AND CLASSIFICATION

In this section, we will mathematically describe the features we are using to classify the solder joint subimages and the classification process itself. The features are grouped into four categories: 1) basic features involving sums of gray levels, 2) inertia features of the volume defined by the subimage, 3) the faceted surface area feature, 4) Gaussian curvature related features. Before we treat the individual features, we must first discuss some preliminary topics.

4.1. Brightness and Size Normalization

We have attempted 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. It is assumed that we are given discrete image data in the form of an n_x x 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 8 bits yielding the usual 256 gray levels. The number of numerically different images that can be represented with this image data is $2^{8 n_x n_y}$ which is quite large. A 6 x 6 window of an 8-bit image can represent over 10^{86} different window subimages. Attempts to classify these subimages must somehow reduce the dimensionality of the original data while preserving the relevant "information" content.

For computational purposes, a gray scale image will be defined as a set of discrete ordered triples:

$$Image = \left\{ (i,j,k) : k = f(i,j); \quad i \in \left\{0,1,...,n_x - 1\right\}, j \in \left\{0,1,...,n_y - 1\right\}, k \in \left\{0,1,...,2^{nbits} - 1\right\} \right\}$$

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

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

This normalization maps all gray levels in an image to the interval [0,1]. We will denote the normalized gray level function as

$$z(i,j) = \frac{f(i,j)}{f_{\max}} \qquad \text{for } all\ (i,j).$$

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

$$x(i) = \frac{(1 + 2i - n_x)}{2n_x}$$

$$y(j) = \frac{(1 + 2j - n_y)}{2n_y}$$

This combination of size and brightness normalization maps all subimages into surfaces which lie within the unit cube. If we were to interpolate continuous surface information from this representation, it is our hope that the interpolated surface data would be relatively invariant to light intensity changes and changes in the size of the image.

4.2. Basic Features

We map solder joint subimages into a normalized form so that we can compute representative features which characterize the visual appearance of the solder joints. Normalized volume is one feature that immediately comes to mind. For example, acceptable solder joints usually have large central peaks yielding a small normalized volume whereas a hole filled with solder with no component lead (referred to as "fills") usually has a flat gray level surface yielding a large normalized volume. This feature is defined as

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

The normalized volume is the mean normalized gray level. We also use a normalized standard deviation feature σ which is a measure of the variance of the normalized gray level surface about its mean value:

$$\sigma^2 = \frac{1}{n_x \; n_y \; - 1} \sum_{i=0}^{n_x - 1} \sum_{j=0}^{n_y - 1} \left(\frac{f(i,j)}{f_{\max}} - V_{tot} \right)^2$$

Note that these features are independent of the ordering or spatial distribution of pixels in the image. A simple but useful method to introduce the effects of the spatial distribution of gray levels is to compute volumes in fixed regions of the solder joint subimage. We introduce two features, the central subwindow volume and the outer frame region volume:

$$V_{csw} = \frac{1}{n_x \; n_y \; f_{\max}} \sum_{|x(i)| < 0.2} \sum_{|y(j)| < 0.2} f(i,j)$$

$$V_{ofr} = \frac{1}{n_x \; n_y \; f_{\max}} \sum_{|x(i)| > 0.4} \sum_{|y(j)| > 0.4} f(i,j)$$

Plated-through holes with no component leads and little or no solder (referred to as just "holes") have a significantly smaller central subwindow volume than other classes of solder joints. The region parameters, 0.2 and 0.4, were chosen to isolate the plated-through solder-pad metal and the component lead areas of a typical solder joint subimage. These features seem to be quite good for discriminating among various joint classes. It may be worthwhile to experiment with different regions' volumes for different shaped solder joints (e.g., rectangular vs. circular).

4.3. Inertia Features

The next set of features are based on the three-dimensional inertia properties of the uniform-density solid volume bounded by the four planes $x = \pm 0.5$, $y = \pm 0.5$

and the two surfaces defined by $z\,(i,j)$ and the upside down surface $-z\,(i,j)$. Note that the volume of this object is just twice the normalized volume computed above. We use the surface 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

$$[\,I_{CS}\,] = \int\int\int \rho\,(\,\underline{r}\,)\,(\,\underline{r}^{\,T}\,\underline{r}\,[\,1\,] - \underline{r}\,\underline{r}^{\,T}\,)\,dx\,dy\,dz$$

where $\underline{r} = (x,y,z)$ is the 3-D coordinate vector relative to the CS coordinate sytem, $[\,1\,] = 3 \times 3$ identity matrix, superscript T denotes transpose, underbars denote vectors, brackets denote matrices, and $\rho(\,\underline{r}\,) =$ Object density function.

The center of mass vector for a mass distribution $\rho\,(\,\underline{r}\,)$ is given by

$$\underline{r}_{cm} = \int\int\int \underline{r}\,\rho\,(\,\underline{r}\,)\,dx\,dy\,dz.$$

We represent each pixel in our normalized image representation as a rectangular block uniform-density mass distribution where the block's volume is directly proportional to the intensity at that pixel. The (i,j) pixel block has dimensions $\Delta x \times \Delta y \times 2z\,(i,j)$ where $\Delta x = \dfrac{1}{n_x}$ and $\Delta y = \dfrac{1}{n_y}$. The height of the block is twice the normalized gray level with the center of the (i,j) pixel block located at $(x(i),y(j),0)$. We compute the center of mass coordinates of the image volume assuming unit density as

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

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

$$z_{cm} = 0$$

where the mass function $m\,(i,j)$ is given by

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

The parallel axis theorem, which is proven directly from the definition of the inertia 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:

$$[\ I_{tr}\ ] = [\ I_{cm}\ ] + M_{tot}\ (\underline{v}^T\,\underline{v}\,[\ 1\ ] - \underline{v}\,\underline{v}^T\ )$$

where $\underline{v}$ = the translation vector from the old 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 dimensions $a \times b \times c$ and mass M_{tot} is well-known:

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

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:

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

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

This total inertia tensor for the image volume is then diagonalized to obtain the principal moment of inertia features which we refer to as as I_{aa} , I_{bb} , I_{cc} where I_{cc} is the moment of inertia around the z-axis. Since the z-component of $\underline{v}\ (i,j)$ is always zero, the matrix can be diagonalized using analytic formulas for 2×2 matrices. The trace of the inertia tensor is computed as a feature since it is invariant to orthogonal coordinate transformations, and it combines the principal moment values into one number. The sum of diagonal minors is also computed as a feature since it shares the same invariance properties as the trace. The trace and diagonal minor sum (dms) are computed as follows:

$$tr\ [\ I_{tot}\ ] = I_{aa} + I_{bb} + I_{cc}$$

$$dms\ [\ I_{tot}\ ] = I_{aa}\ I_{bb} + I_{bb}\ I_{cc} + I_{cc}\ I_{aa}$$

We do not use the determinant of the inertia tensor because this quantity was found to have very high variance and was not useful for classification. The ratio of the average of I_{aa} and I_{bb} to I_{cc} is used as a feature as a measure of the inertia along the intensity axis as compared to the inertia about axes in the x-y plane. This concludes the discussion of features based upon image volume inertia.

4.4. Faceted Surface Area Feature

The surface area of the gray level surface is an indication of the undulating quality of a surface. It is similar to the standard deviation feature except the spatial distribution of pixel gray levels is taken into account. Each set of adjacent four pixels are considered as the vertices of a non-planar quadrilateral. The quadrilateral is divided into two triangles in a fixed manner. The sum of the areas of the two triangles is computed as the area of the quadrilateral. The sum of all quadrilaterals in the image surface is the total faceted surface area of the image. For an $n \times n$ constant intensity image, the surface area is $(n-1)^2$. We divide the computed surface area by the constant intensity surface area to normalize this feature. The surface area computed using this method will be compared to the surface area computed using the first fundamental form of a surface, which is discussed in a later section.

Assume we are given four adjacent pixels in the image: $f(i,j)$, $f(i+1,j)$, $f(i,j+1)$, $f(i+1,j+1)$. We form the four 3-D points $\{ \underline{p}_1, \underline{p}_2, \underline{p}_3, \underline{p}_4 \}$ given by the following vectors:

$$\underline{p}_1 = [\ i \quad j \quad f(i,j)\ ]^T$$

$$\underline{p}_2 = [\ i+1 \quad j \quad f(i+1,j)\ ]^T$$

$$\underline{p}_3 = [\ i \quad j+1 \quad f(i,j+1)\ ]^T$$

$$\underline{p}_4 = [\ i+1 \quad j+1 \quad f(i+1,j+1)\ ]^T$$

We now consider the two triangles of the quadrilateral separately. Let $\underline{p}_1, \underline{p}_2, \underline{p}_3$

determine the first triangle. We compute two difference vectors first.

$$\underline{d}_1 = \underline{p}_2 - \underline{p}_1$$

$$\underline{d}_2 = \underline{p}_3 - \underline{p}_1$$

The unit normal to the triangle, which is used to define the z-direction, is computed as follows:

$$\underline{n}_z = \frac{\underline{d}_1 \times \underline{d}_2}{\|\underline{d}_1 \times \underline{d}_2\|}$$

We use $\underline{d}_1$ to define the x direction:

$$\underline{n}_x = \frac{\underline{d}_1}{\|\underline{d}_1\|}$$

The y-direction is then determined by $\underline{n}_x$ and $\underline{n}_z$ as follows:

$$\underline{n}_y = \underline{n}_z \times \underline{n}_x$$

These normal vectors now provide us with a rotation matrix with which we can rotate all three points into the x-y plane:

$$[\,R\,] = [\ \underline{n}_x \ | \ \underline{n}_y \ | \ \underline{n}_z \ ]$$

Since we have chosen $\underline{d}_1$ to define our x-direction, we can compute the area of the triangle, by computing two vector inner products:

$$area1 = \frac{1}{2} \left(\underline{n}_x{}^T \underline{d}_1\right) \left(\underline{n}_y{}^T \underline{d}_2\right)$$

Similarly, we compute *area2* using the same formulas applied to p_4, p_3, p_2. 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

$$a(i,j) = area1 + area2$$

and the total normalized surface area is given by

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

Note that $A = 1$ when the gray level surface is flat and increases as the surface contains more and more undulations. We note that there are other formulas for computing the area of a triangle which use only the lengths of the triangle sides [11] and do not consider surface normals.

4.5. Gaussian Curvature Related Features

Some preliminary research concerning the use of the Gaussian curvature function for characterizing the shape and visual appearance of a gray level surface has been completed. Several features related to Gaussian curvature are currently computed for our classification algorithm. They will be described in detail in this section, but first we must introduce some basic formulas from the differential geometry of surfaces. There are many good introductory textbooks on the subject (e.g. [14]).

We shall consider a continuous gray level surface S of the form:

$$S = \left\{ (x,y,z) : z = f(x,y) \right\}$$

The sampled gray level image surface I corresponding to S will be of the form introduced earlier:

$$I = \left\{ (i,j,k) : k = f(i,j) \right\}$$

The first fundamental form of a surface is determined by the following coefficients assuming. of course, that the first partial derivatives exist:

$$E = 1 + f_x^{\,2} \qquad F = f_x\, f_y \qquad G = 1 + f_y^{\,2} \qquad g = E*G - F^{\,2} = 1 + f_x^{\,2} + f_y^{\,2}$$

The second fundamental form of a surface is determined by the following coefficients assuming. of course, that the second partial derivatives exist:

$$L = \frac{f_{xx}}{\sqrt{g}} \qquad M = \frac{f_{xy}}{\sqrt{g}} \qquad N = \frac{f_{yy}}{\sqrt{g}} \qquad b = L*N - M^{\,2}$$

These two fundamental forms play an important role in many key theorems in the differential geometry of surfaces.

The gradient vector of first partial derivatives is given by

$$\nabla f = \begin{bmatrix} f_x & f_y & 1 \end{bmatrix}^T$$

The Hessian matrix of second partial derivatives is given by

$$Hess\ f = \begin{bmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{bmatrix}$$

The mean curvature function of a smooth surface is given by

$$H = \frac{(EN + GL - 2FM)}{2g}$$

The Gaussian curvature function of a smooth surface can be expressed in several different ways:

$$K = \frac{b}{g} = \frac{\det(Hess\ f)}{g^2} = \frac{\det(Hess\ f)}{\| \nabla f \|^4} = \frac{f_{xx} f_{yy} - f_{xy}^2}{(1 + f_x^2 + f_y^2)^2}$$

It can be shown that 1) $K>0$ for all points (x,y) where the local neighborhood of the surface has an ellipsoidal shape, 2) $K<0$ for all points (x,y) where the local neighborhood of the surface has a saddle shape, 3) $K=0$ for all points (x,y) where the local neighborhood of the surface is flat (planar) or cylindrical. This means that $K(x,y)$ does characterize the shape of a surface to a certain extent.

Now suppose that we wish to compute features based on the Gaussian curvature of the gray level surface. The first problem is that the image data is discrete and may or may not correspond to a smooth differentiable surface. The second problem is that even if we do compute a meaningful discrete version of the Gaussian curvature of an image's gray level surface, this discrete data still maintains the same dimensionality of the original image and is, therefore, not directly useful as a feature. Several features which are scalar variables have been chosen for computation. Most of the Gaussian

curvature related features that we are currently using are experimental in nature and generally need more work.

The computation of the partial derivatives themselves from discrete images is not a trivial problem. We have used window operators of the type utilized by Beaudet [2], Bolle and Cooper [3], and Haralick [9] for quadratic surfaces. These window operators provide a least squares estimate of the five partial derivatives required above using a best fit quadratic surface. The windows are convolved with an image to provide the numbers used for the partial derivatives in the expression for K.

As we noted before, the positiveness and the negativeness of the Gaussian curvature function does tell us something about the surface shape at individual points. Hence, we can obtain two size and brightness normalized features by computing the percentage of pixels with positive K values and the percentage of pixels with negative K values:

$$p_+ = \frac{card\left\{ (i,j) : K(i,j) > 0 \right\}}{n_x\, n_y}$$

$$p_- = \frac{card\left\{ (i,j) : K(i,j) < 0 \right\}}{n_x\, n_y}$$

where *card* is the cardinality set function which indicates the number of elements in a given set. Separate brightness normalization operations are not required for Gaussian curvature related features because all features are based on derivative information.

We also compute the average Gaussian curvature, the average mean curvature, the average second fundamental form determinant, and the average square root of the first fundamental form determinant as features:

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

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

$$b_{avg} = \frac{1}{n_x\, n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} b\,(i,j)$$

$$\sqrt{g}_{avg} = \frac{1}{n_x\, n_y} \sum_{i=0}^{n_x-1} \sum_{j=0}^{n_y-1} \sqrt{g\,(i,j)}$$

It turns out that the last feature here can be interpreted as an approximation of the surface area of the continuous image gray level surface. The integral of the square root of g over the surface parameter space gives an exact expression for the area of a smooth surface which is very similar to the expression for computing the arc length of curves:

$$Area = \int\int \sqrt{(\,1 + f_x{}^2 + f_y{}^2\,)}\; dx\; dy$$

We see that $n_x\, n_y \sqrt{g}_{avg}$ is a discrete approximation to this integral. In addition, we compute the ratio of the faceted surface area A and the differential geometry surface area $\sqrt{g}_{avg}$ which may be an indication of overall image surface smoothness. This concludes the discussion of Gaussian curvature related features.

4.8. Minimum-Distance Classification Algorithm

We have described each of the individual features in the preceding section. We now combine these features into a feature vector $\underline{\alpha}$ with the number of features $n_F = 20$. The order of the individual components within the vector is decided arbitrarily and is of no importance:

$$\underline{\alpha} = [\alpha_0 \quad \alpha_1 \quad \cdots \quad \alpha_{n_F-1}]^T$$

$$\alpha_0 = V_{tot} \qquad \alpha_1 = \sigma \qquad \alpha_2 = V_{csw} \qquad \alpha_3 = V_{tot} - (V_{csw} + V_{ofr})$$

$$\alpha_4 = V_{ofr} \qquad \alpha_5 = I_{aa} \qquad \alpha_6 = I_{bb} \qquad \alpha_7 = I_{cc}$$

$$\alpha_8 = \frac{(I_{aa} + I_{bb})}{2I_{cc}} \qquad \alpha_9 = tr [I_{tot}] \qquad \alpha_{10} = dms [I_{tot}] \qquad \alpha_{11} = \frac{\min_{i,j} f(i,j)}{f_{\max}}$$

$$\alpha_{12} = K_{avg} \qquad \alpha_{13} = H_{avg} \qquad \alpha_{14} = b_{avg}$$

$$\alpha_{15} = p_+ \qquad \alpha_{16} = p_-$$

$$\alpha_{17} = \sqrt{g}_{avg} \qquad \alpha_{18} = A \qquad \alpha_{19} = \frac{A}{\sqrt{g}_{avg}}$$

This defines the feature vector (20 numbers) which is used to represent each solder joint subimage ($n_x\ n_y$ numbers). All symbols have been defined in previous sections.

The first step in the classification algorithm is the training exercise. Using some strategy, a group of solder joint subimages are collected which best represent a given class. We will assume that we are working with n_C classes of solder joints. Each

class will be referred to as C_i where i is an integer between 0 and $n_C - 1$. For our experiments, n_C was either 6 or 12 and C_0 was always used as an acceptable class. We also assume that we have n_{TC_i} solder joint subimages of class C_i for training. The training phase computes a mean feature vector $\underline{\mu}_i$ for each class C_i:

$$\underline{\mu}_i = \frac{1}{n_{TC_i}} \sum_{k=0}^{n_{TC_i}-1} \underline{\alpha}_{ik}$$

where $\underline{\alpha}_{ik}$ is the k-th training sample of the i-th class. These mean vectors are combined together to form the mean feature/class matrix [MFC] which is needed by the classification program:

$$[\, MFC \,] = [\, \underline{\mu}_0 \mid \underline{\mu}_1 \mid \cdots \mid \underline{\mu}_{n_C-1} \,].$$

The training phase is complete when the mean feature/class matrix has been computed. The training program we have implemented also computes a standard deviation vector, a maximum vector, and a minimum vector for each class for subsequent analysis and class separation plots.

We now assume that we have n_{C_i} solder joint subimages of class C_i to be classified. The total number of solder joints to be classified is then

$$n_{tot} = \sum_{i=0}^{n_C-1} n_{C_i}$$

The classifier only knows that it has n_{tot} solder joint subimages to classify. For the j-th joint subimage, we compute a feature vector $\underline{\alpha}_j$ where $0 \leq j < n_{tot}$. The j-th

joint has the human inspector's classification $c_j^{\,h}$ which is unknown to the classifier. The classifier computes a distance metric d_{ij} from $\underline{\alpha}_j$ to each of the n_C class mean vectors $\underline{\mu}_i$:

$$d_{ij} = \| [\, diag(\, \underline{w}\,)\,] (\underline{\alpha}_j - \underline{\mu}_i\,) \|$$

where $\underline{w}$ is the feature weighting vector which contains a scale factor for each of the n_F features, $diag(\,\_\,)$ converts any vector to a diagonal matrix, and $\|(\,\_\,)\|$ denotes the vector norm. Different vector norms can be used to compute the distance metric. Let v_i represent the i-th component of the vector $\underline{v}$. We tried the following three vector norms to see if any one worked better than the other two:

$$\sum_i |\, v_i \,| \quad , \quad \sum_i v_i^{\,2} \quad , \quad \max_i |v_i|$$

We found empirically that performance did not vary much for the different norms. The Euclidean metric (sum of squares) performed slightly better for the few test cases we tried. We did find empirically that for misclassifications one of the metrics would compute the right class about half of the time. This means that we are fairly close to making the right decision for many of the mistakes. We have not yet decided how this behavior can be used to increase classification accuracy.

The j-th joint then receives the algorithmic classification $c_j^{\,a}$ which corresponds to the minimum distance obtained between $\underline{\alpha}_j$ and the mean vector of the particular class:

$$d_{kj} = \min_{0 \le i < n_C} d_{ij}$$

$$c_j{}^a = C_k$$

This concludes the description of the minimum-distance classification algorithm.

4.7. Performance Measurement

We can quantitatively measure the performance of the classification algorithm on solder joint images. Our error performance percentages were computed as follows:

$$p_{correct\_class} = \frac{1}{n_{tot}} \; card \left\{ \underline{\alpha_j} : c_j{}^a = c_j{}^h \right\}$$

$$p_{incorrect\_class} = \frac{1}{n_{tot}} \; card \left\{ \underline{\alpha_j} : c_j{}^a \neq c_j{}^h \right\}$$

$$p_{miss} = \frac{1}{n_{tot}} \; card \left\{ \underline{\alpha_j} : c_j{}^a = Acceptable \;\; \& \;\; c_j{}^h = Unacceptable \right\}$$

$$p_{false\_alarm} = \frac{1}{n_{tot}} \; card \left\{ \underline{\alpha_j} : c_j{}^a = Unacceptable \;\; \& \;\; c_j{}^h = Acceptable \right\}$$

$$p_{good/bad\_correct} = 1 - (p_{miss} + p_{false\_alarm})$$

We feel that the correctness percentage of good/bad decisions (acceptable vs. unacceptable solder joints) is the most important index of performance. The *miss rate* percentage is the next most important because a human being will probably be operating the touch-up station in the near future and can decide not to touch up a given joint if

it is a false alarm (it looks acceptable). The false alarm rate percentage is important though because too many false alarms would defeat the purpose of having an automatic inspector in the first place. The correct class and incorrect class percentages are not as important as the above performance indices, but they are very useful for two reasons:

(1) The type of unacceptable class can yield important information regarding the status of the wavesoldering machinery [1]. This would be very useful in automating the entire manufacturing system. If the unacceptable classifications are wrong, it may lead assembly line operators to look for the wrong problem in the soldering section of the assembly line.

(2) There are $\frac{n_C\ (n_C - 1)}{2}$ possible error transitions between classes. By monitoring these error transitions during the inspection algorithm development and testing cycles, we can formulate ways to improve performance.

Our experiments indicate that certain error transitions are much more likely than others; in fact, some error transitions never occur. Sometimes performance can be improved by modifying the weighting vector to deemphasize features which have too much overlap between the two classes of the most likely error transition.

The meaning of each error percentage should be kept in mind while perusing the experimental results shown in the next section.

5. EXPERIMENTAL RESULTS

A small database of thirteen (13) images was assembled for testing our preliminary inspection algorithm. Seven images of different sections of a test PC board were digitized with each image containing subimages of more than 190 solder joints each. These seven images were labeled A1, A2, A3, A4, A5, A6, A7. Image A1 is typical of these images and is shown in Figure 4. The total number of solder joint subimages in these seven images is 1418. Six more images were digitized with about 64 solder joints in each image containing a total of 378 solder joint subimages. These six higher spatial resolution images contain a subset of the 1418 solder joints in the first seven images. These six images were labeled B1, B2, B3, B4, B5, B6. Image B4 is typical of these images and is shown in Figure 5. All images were digitized with the camera directly over the PC board. The lighting and camera aperture were held fixed for all thirteen images. The PC board was moved manually under the camera and light to focus on different sections of the PC board. The experimental inspection results given below were obtained using the 13 different mean feature/class (MFC) matrices generated by the training program (one for each image), three different feature weighting vectors, and the two different class lists mentioned earlier in the report. We refer to the individual classification runs with some combination of the above items as an experiment. The three weight vectors are given in Figure 12. Weight Vector 1 was only partially optimized using the manual procedure mentioned previously for the six class list. Weight Vector 2 uses only five out of twenty features and can be considered as an abbreviated feature list of quickly computable features. It was optimized for five classes on an image similar to the ones shown in this report. Weight Vector 3 uses

nineteen out of twenty features and was optimized for the A1 image and the 12 member class list. None of the weight files have attempted to optimize performance over several images. We plan to replace this ad hoc approach with more rigorous statistical pattern recognition approaches.

Experiments 1-3 and 4-6 should be considered as a group; the results for these experiments are shown in Figures 13 and 14. In each of these experiments the mean feature/class (MFC) matrix created by the training algorithm from each individual image was used to classify its own corresponding image. (This is the mode of operation that the classification algorithm was designed to handle; other experiments were done to check the robustness of the algorithm by mixing MFC matrices with different images.) We used the twelve (12) member class list and weight file 1 for Experiment 1, weight file 2 for Experiment 2, and weight file 3 for Experiment 3. We used the six (6) member class list and weight file 1 for Experiment 4, weight file 2 for Experiment 5, and weight file 3 for Experiment 6. For Experiments 1-3, note that the average and the A1 image performance tend to improve as the weight file changes from 1 to 2 to 3 while the performance for the other images changes in unpredictable ways. On the other hand, A1 image performance gets worse in Experiments 4-6. Experiment 6 had particularly bad performance which may be due to the fact that weight file was optimized for the twelve class list as opposed to the six class list used in this experiment. Note that the A5 image does poorly with the twelve member class list for all weight files and quite well with six member class list for weight files 1 and 2. A complex multidimensional distribution of data is responsible for this behavior. The best performance occurred for the B4 image in Experiments 1 and 3. The worst performance in this

group of experiments occurred for B4 in Experiment 6.

In Experiments 7-9, we used the six member class list with weight file 1 and a fixed mean feature/class matrix for each group. The mean feature/class matrix was different in each experiment to see how performance changes. The results for these experiments are shown in Figure 15. The average performance does not change significantly with different MFC matrices indicating that the class features for a given set of images are relatively constant across images. In the future, we will compute class mean vectors across several different images to see if multiple image performance improves with a mean MFC matrix.

The results of all the above experiments are included in the report to demonstrate how performance can vary with class list, feature list, feature weighting vector, and training data. There is clearly much room for improvement, but the results are definitely encouraging, especially for particular images. We have not yet addressed the multidimensional problems in any sort of optimal way.

We suspect that performance will increase for real PC boards for the following reasons:

(1) Real PC boards generally use round, circular solder pads yielding smoothly rounded solder joints. The appearance of the acceptable joint surface should be much more consistent than the appearance created by joints with square solder pads.

(2) Most of the joints on real boards will be acceptable joints. In our experiments, acceptable joints are in the minority, and we are currently trying to distinguish

among many different types of subimages. When inspecting real PC boards, one is trying to sift out solder joints which look unusual, different from the acceptable majority of joints. Algorithms might be tailored to take advantage of this fact.

6. FUTURE RESEARCH DIRECTIONS

The performance of our preliminary inspection algorithm is encouraging. We expect that many improvements can be made to our current techniques.

6.1. Better Feature Selection

Better feature selection refers both to computing new individual features which are more representative of the solder joints classes and to computing combinations of features which provide even better class discrimination. As we noted earlier, individual feature selection is not based on an exact science. By concentrating on the errors of our current algorithm, we may find clues which will lead us to better features. In addition, our planned multiple view research may require introduction of some new features to handle the fact that we will get different appearances of solder joints from different viewpoints.

Combinations of features may provide better performance than what we can get with our simple weighting scheme. The statistical pattern recognition literature will be able to help us in this area after we have gathered enough data. In addition, many heuristic approaches may also be of assistance. We hope to experiment with several different ways of combining features to get better features and compare their performance to our current algorithm.

We also plan to improve our mean features by writing some new software which will make it easy to compute the mean feature/class matrix of many different images. The variance of the mean features across many images will also be analyzed when this is done. We currently compute statistics for only one image at a time.

6.2. Better Classification Algorithms

As we have mentioned, the minimum-distance classifier is a very simple classifier. There are many different kinds of classifiers, statistical and non-statistical, which could be applied to this problem. Expert systems, for example, make reliable decisions through the interaction of the input data with various production rules. Distance metrics applied to feature vectors are not used at all. Multi-level decision processing may also be helpful to us. If a decision on an n x n solder joint subimage is doubtful, it is possible to redigitize the subimage at double the spatial resolution and reclassify. There are currently no explicit decision confidence factors in our current approach which can be used for this purpose. We hope to survey several different decision-making techniques, select a few of the more promising methods for implementation, and test them against our current algorithm.

6.3. Multiple View Classification

It seems likely that classification performance could be improved if information from multiple views were combined. Also, some joints have defects which are not easily observed from directly over the PC board and, therefore, cause only small perturbations to the gray level surfaces which we classify. We plan to evaluate the use of multiple views of solder joints for more reliable inspection.

6.4. Sensitivity Analysis

Inspection algorithms will usually have certain parameters involved which change the performance. In our current algorithm, for example, we are planning to investigate the effects of the following parameter changes:

(1) Alter solder joint sizes and/or locations in the solder joint segmentation file in random or uniform ways and check inspection performance to compute sensitivity to size and location errors.

(2) Check classification performance for different lens f-stops and different lighting configurations to compute sensitivity to illumination changes.

(3) Check classification accuracy for the same solder joints at different spatial resolutions to compute sensitivity to size changes (changes in camera distance from PC board).

6.5. Registration and Calibration Analysis

Our inspection algorithm depends upon the ability to use pre-segmented images using information about the PC board, the camera and lens, their separation distance, and x-y table positions. It is important that we determine how accurately and repeatably we can predict the exact location of a solder joint in a digitized image. This will mainly depend on the precision of the x-y table and the manner in which the PC board is gripped by it. Judging from what others have done, it seems that this will not be much of a problem. If this analysis is in agreement with the size/location sensitivity analysis and if classification performance is improved to acceptable levels (99%+), it may be appropriate to construct a physical prototype automatic inspection

system.

7. REFERENCES

(1) Bernard, Casimir D., "Wave Soldered Solder Joint Quality Troubleshooting Guide," IBM Technical Report TR00.2761, pgs. 1-18, June 1976

(2) Beaudet, Paul R., "Rotationally Invariant Image Operators," *Proc. 4th Int'l. Joint Conf. Pattern Recognition,* Kyoto, Japan, pgs. 579-583, November 1978

(3) Bolle, Ruud M. and Cooper, David B., "Bayesian Recognition of Local 3-D Shape by Approximating Image Intensity Functions with Quadric Polynomials," *IEEE Trans. Pattern Analysis & Machine Intelligence,* Vol. PAMI-6, No. 4, pgs. 418-429, July 1984

(4) Chin, Roland T. and Harlow, Charles A., "Automated Visual Inspection: A Survey," *IEEE Trans. Pattern Analysis & Machine Intelligence,* Vol. PAMI-4, No. 6, pgs. 557-573, November 1982

(5) Duda, Richard O. and Hart, Peter E., *Pattern Classification and Scene Analysis,* New York: John Wiley & Sons, 1973

(6) Ejiri, M., Uno T., Mese, M., Ikeda, S., "A Process for Detecting Defects in Complicated Patterns," *Computer Graphics and Image Processing,* Vol. 2, pgs. 326-339, 1973

(7) Fukanaga, Keinosuke, *Introduction to Statistical Pattern Recognition,* New York: Academic Press, 1972

(8) Haralick, Robert M. and Watson, Layne, "A Facet Model for Image Data," *Proc. IEEE Conf. Pattern Recognition and Image Processing,* pgs. 489-497, August 1979

(9) Haralick, Robert M., "Digital Step Edges from Zero Crossing of Second Directional Derivatives," *IEEE Trans. Pattern Analysis & Machine Intelligence,* Vol. PAMI-6, No. 1, pgs. 58-68, January 1984

(10) Jarvis, John F., "A Method of Automating Visual Inspection of Printed Wiring Boards," *IEEE Trans. Pattern Analysis & Machine Intelligence,* Vol. PAMI-2, No. 1, pgs. 77-82, January 1980

(11) Lin, C. and Perry, M.J., "Shape Description using Surface Triangularization," *Proc. IEEE Workshop of Computer Vision: Representation and Control,* pgs. 38-43, August 1982

(12) McIntosh, William E., "Automating the Inpsection of Printed Circuit Boards", *Robotics Today,* pgs. 75-78, June 1983

(13) Nakagawa, Yasuo, "Automatic Visual Inspection of Solder Joints on Printed Circuit Boards," *Proc. SPIE,* Vol. 336 Robot Vision, pgs. 121-127, May 1982

(14) O'Neill, Barrett, *Differential Geometry,* New York: Academic Press, 1966

(15) Sterling, Warren M., "Automatic Non-reference Inspection of Printed Wiring Boards," *Proc. IEEE Conf. Pattern Recognition and Image Processing,* pgs. 93-100, August 1979

(16) Thibadeau, Robert, Friedman, Mark, and Seto, John, "Automatic Inspection for Printed Wiring," Carnegie-Mellon Technical Report, CMU-RI-TR-82-16, October 1982

(17) Vanzetti, Riccardo, Traub, Alan C., and Ele, John S., "Hidden Solder Joint Defects detected by Laser Infrared System," *Proc. IPC 24th Annual Meeting,* pgs. 1-15, April 1981

(18) Woodgate, Ralph W., *The Handbook of Machine Soldering,* New York: John Wiley & Sons, 1983

(19) Woodham, Robert J., "Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings," TR-457, MIT AI Laboratory, 1978

(20) Zeller, H. and Doemens, G., "Applications of Pattern Recognition in Semiconductor and Printed Circuit Board Production," *Signal Processing,* Vol. 5, pgs. 399-412, September 1983

Solder Joint Defects	
Defect Name	Brief Description of Defect
No Solder	No Solder at all on Joint
Cold Solder	Solder not sufficiently heated to form joint
Disturbed Solder	Solder Joint disturbed during solidification
Grainy Solder	Contaminants in solder cause graininess
Excess Solder	Too much solder for joint inspectability
Insufficient Solder	Not enough solder for good joint structure
Dewetted Pad	Solder retracts from solder pad surface
Dewetted Lead	Solder retracts from component lead surface
Non-wet Pad	Solder wets lead but not pad
Non-wet Lead	Solder wets pad but not lead
Icicling	Solder forms sharp peaks on joints
Webbing	Solder strands sticks to surrounding insulation
Pinholes	Small holes form in joint fillet surface
Blowholes	Large holes form in joint fillet surface
Solder Pits	Hole or depression in solder where bottom is visible
Oil Entrapment	Oil droplets trapped with solder joint
Solder Bridging	Solder makes unwanted connections with other leads
Rosin Joint	Rosin Flux surrounds lead rather than solder
Solder Balls	Solder adheres to board in balls (not at joints)
Spatter	Spattered solder adheres to board (not at joints)
Internal Voids	Open pocket within joint with no external signs
Bubbles	Open pocket within joint causing bulge in fillet
Outgassed Joint	Severe explosion disrupts joint surface
Grit & Dirt	Surface of joint dirty limiting inspectability
Flux Residue	Residual Flux remains on PCB causing problems
Bad Leads	Component leads are too short, too long, bent over

Figure 1. Solder Joint Defect Table[1]

[1] This table of solder joint defects was compiled from information obtained from the following sources: IPC Workshop Handbook, Handbook of Machine Soldering by Woodgate, Solders and Soldering by Manko, AWS Soldering Manual, IBM Wavesoldered Solder Joint Quality Troubleshooting Guide.

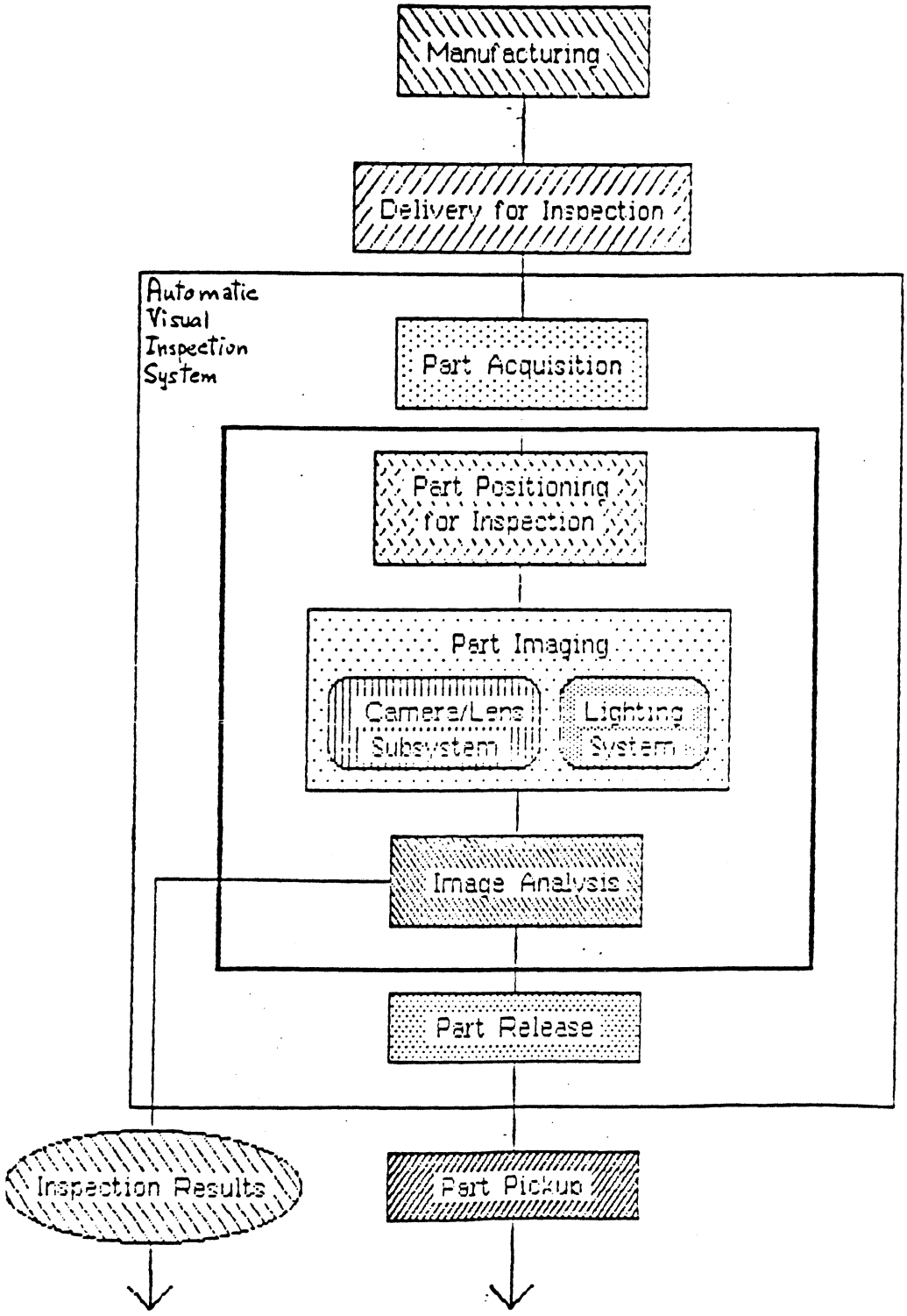


Figure 2. Block Diagram of Automatic Visual Inspection System

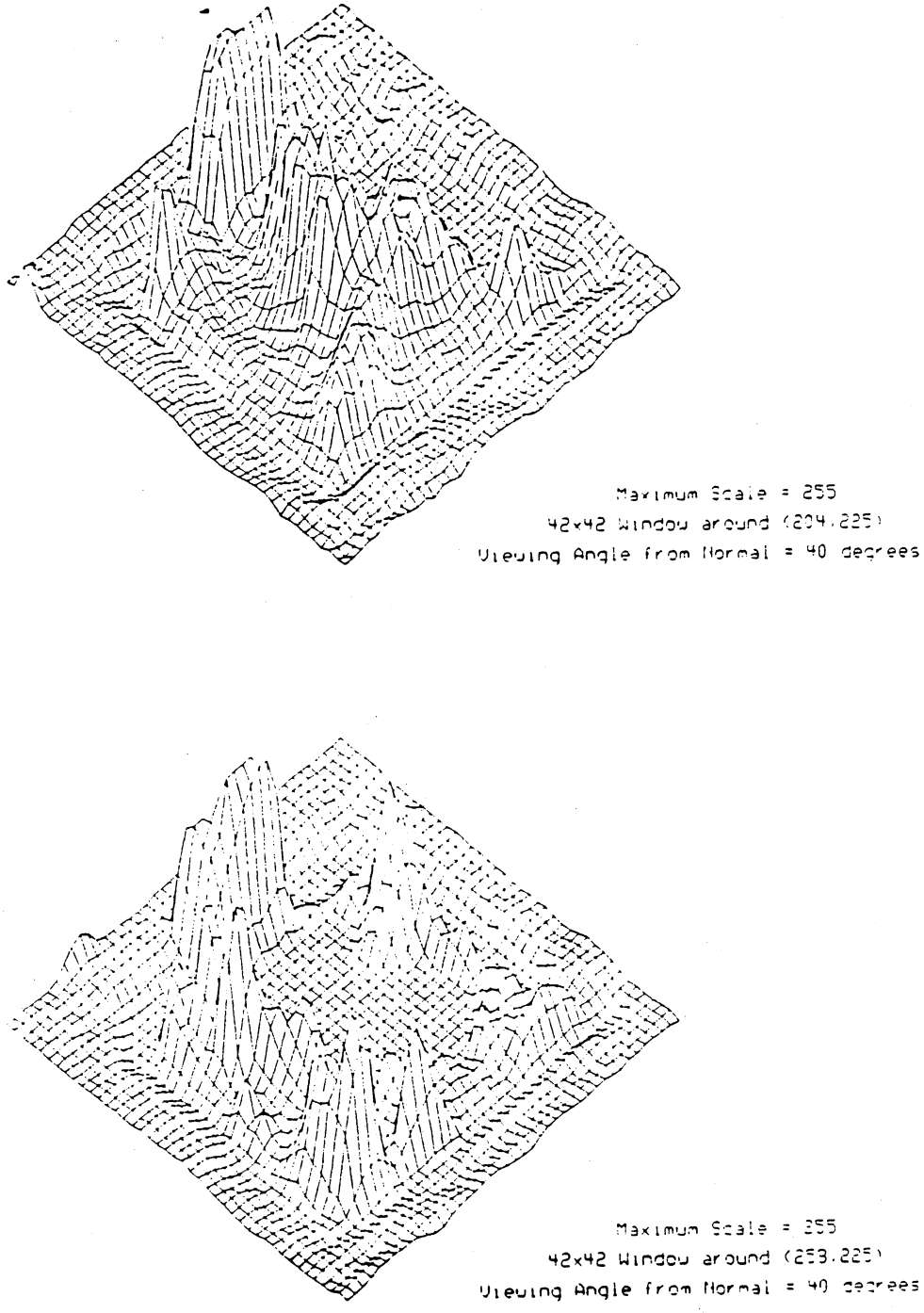


Figure 3. Gray Level Surface Plots (Good Joint-Top, Bad Joint-Bottom)

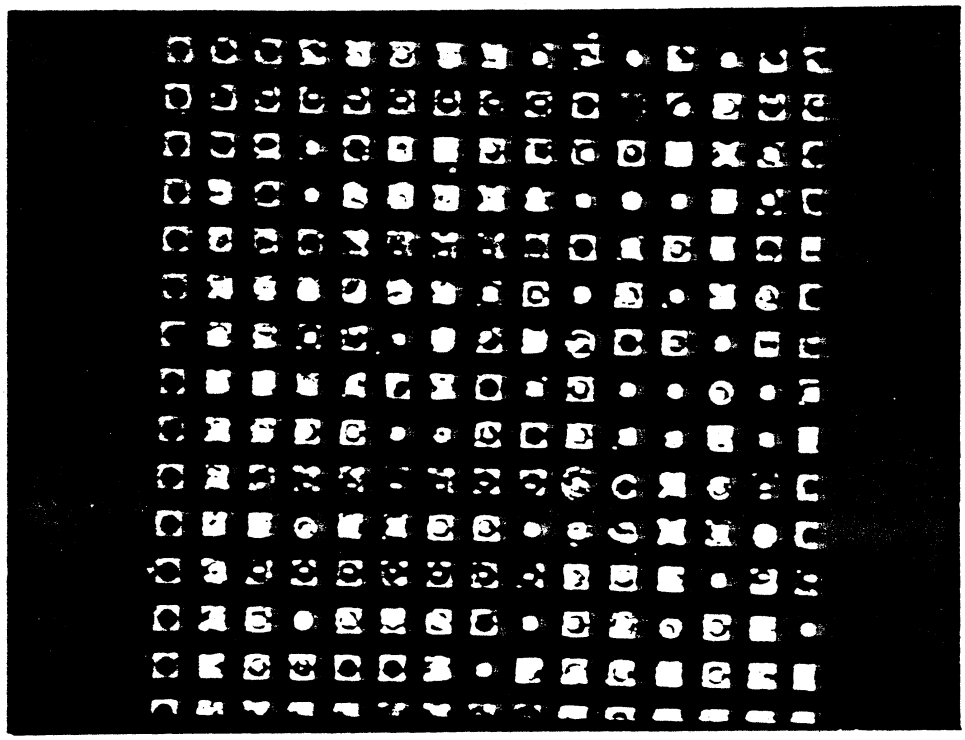


Figure 4. Solder Joint Image - A1

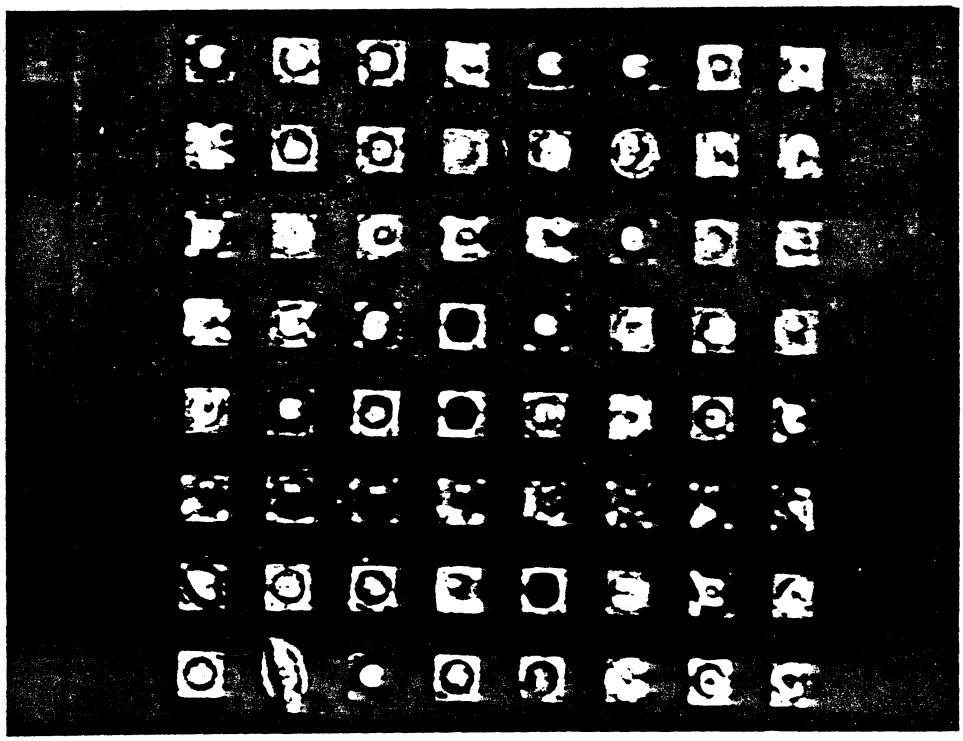


Figure 5. Solder Joint Image - B4

```
File Creator= ang
Creation_Time= Fri Jul 27 15:30:01 1984
Image_Filename= A1
Number_of_Joints_in_Image= 196
Number_of_Joints_in_File=  77
Size_of_Solder_Joint_Subimages= 24 x 24
-----------------------------------------------
id=   1   ix= 176   iy= 326   ws=  24   class= d
id=   2   ix= 177   iy= 228   ws=  24   class= d
id=   3   ix= 143   iy= 361   ws=  24   class= d
id=   4   ix= 142   iy= 326   ws=  24   class= o
id=   5   ix=  39   iy= 195   ws=  24   class= d
id=   6   ix=  38   iy= 293   ws=  24   class= e
id=   7   ix=  37   iy= 261   ws=  24   class= e
id=   8   ix=  72   iy= 226   ws=  24   class= e
id=   9   ix= 107   iy= 260   ws=  24   class= e
id=  10   ix= 142   iy= 260   ws=  24   class= e
id=  11   ix= 141   iy= 126   ws=  24   class= e
id=  12   ix= 175   iy= 126   ws=  24   class= e
id=  13   ix= 107   iy= 126   ws=  24   class= e
id=  14   ix= 314   iy= 227   ws=  24   class= e
id=  15   ix= 382   iy= 224   ws=  24   class= e
id=  16   ix= 246   iy= 424   ws=  24   class= h
id=  17   ix= 313   iy= 456   ws=  24   class= h
id=  18   ix= 348   iy= 456   ws=  24   class= h
id=  19   ix= 175   iy= 160   ws=  24   class= h
id=  20   ix= 143   iy=  60   ws=  24   class= h
id=  21   ix= 176   iy=  61   ws=  24   class= h
id=  22   ix= 244   iy= 258   ws=  24   class= h
id=  23   ix= 210   iy= 291   ws=  24   class= h
id=  24   ix= 347   iy=  93   ws=  24   class= h
id=  25   ix= 416   iy= 126   ws=  24   class= h
id=  26   ix= 416   iy=  26   ws=  24   class= h
id=  27   ix= 450   iy=  28   ws=  24   class= h
id=  28   ix= 483   iy=  28   ws=  24   class= h
id=  29   ix= 484   iy=  61   ws=  24   class= h
id=  30   ix= 485   iy=  93   ws=  24   class= h
id=  31   ix= 484   iy= 126   ws=  24   class= h
id=  32   ix= 484   iy= 158   ws=  24   class= h
id=  33   ix= 485   iy= 192   ws=  24   class= h
```

Figure 6. Solder Joint Segmentation File Example

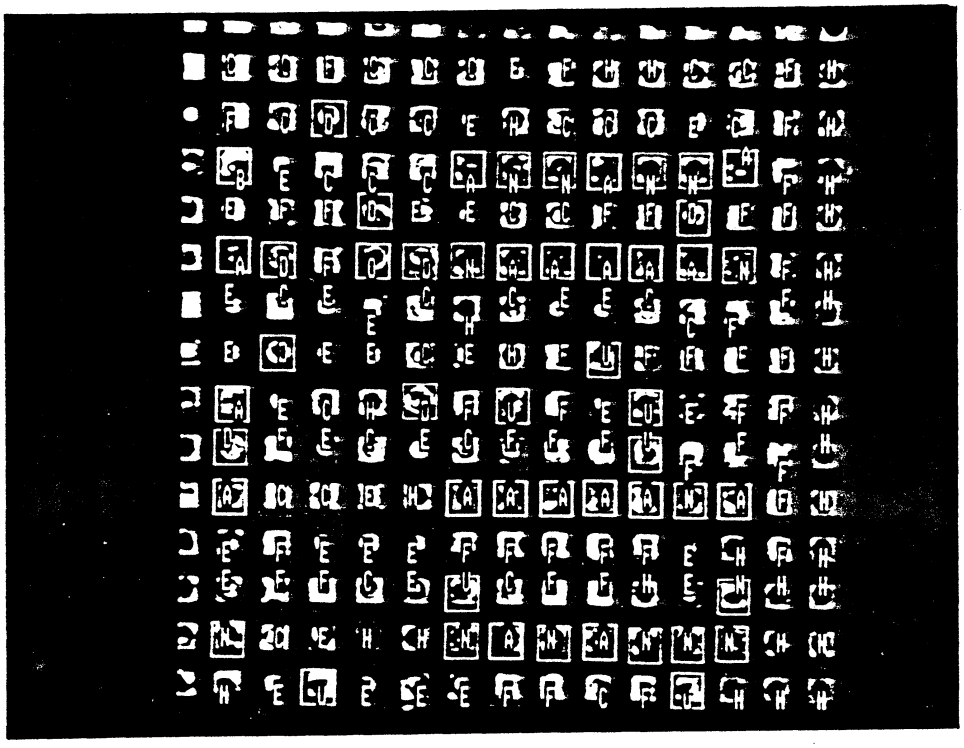


Figure 7. Class Labeled Solder Joint Image - A1

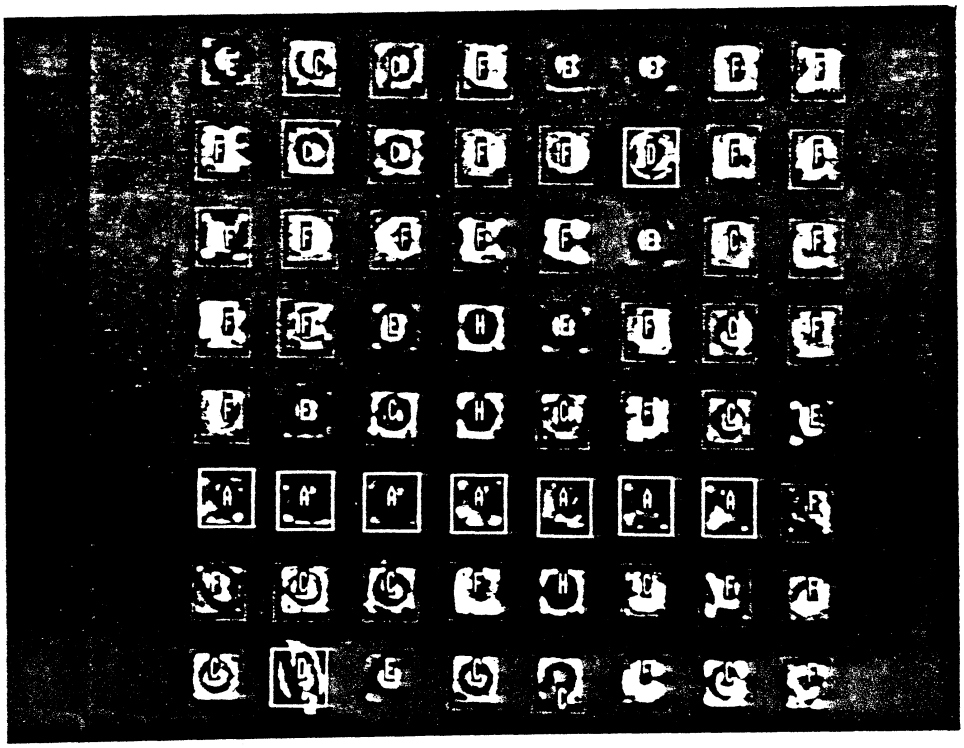
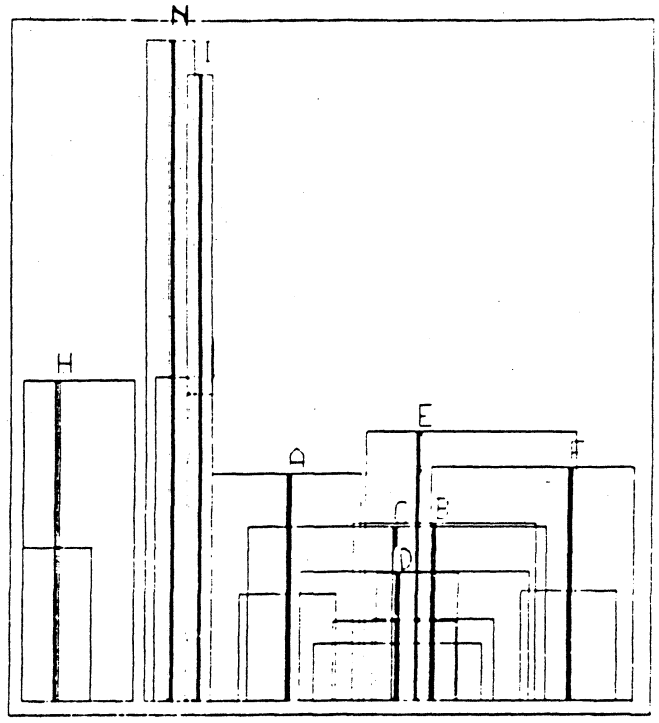


Figure 8. Class Labeled Solder Joint Image - B4



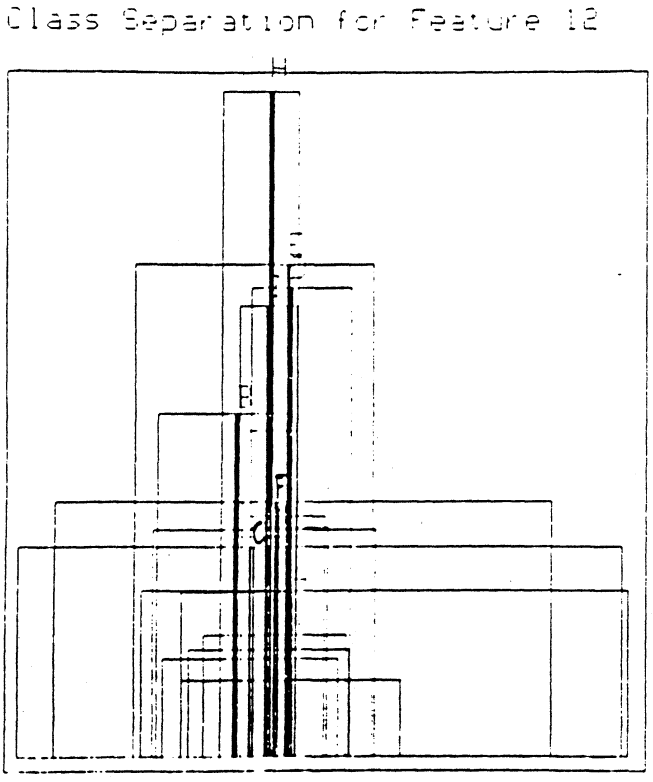


Figure 9. Single Feature Class Separation Plots (Top-Good, Bottom-Bad)

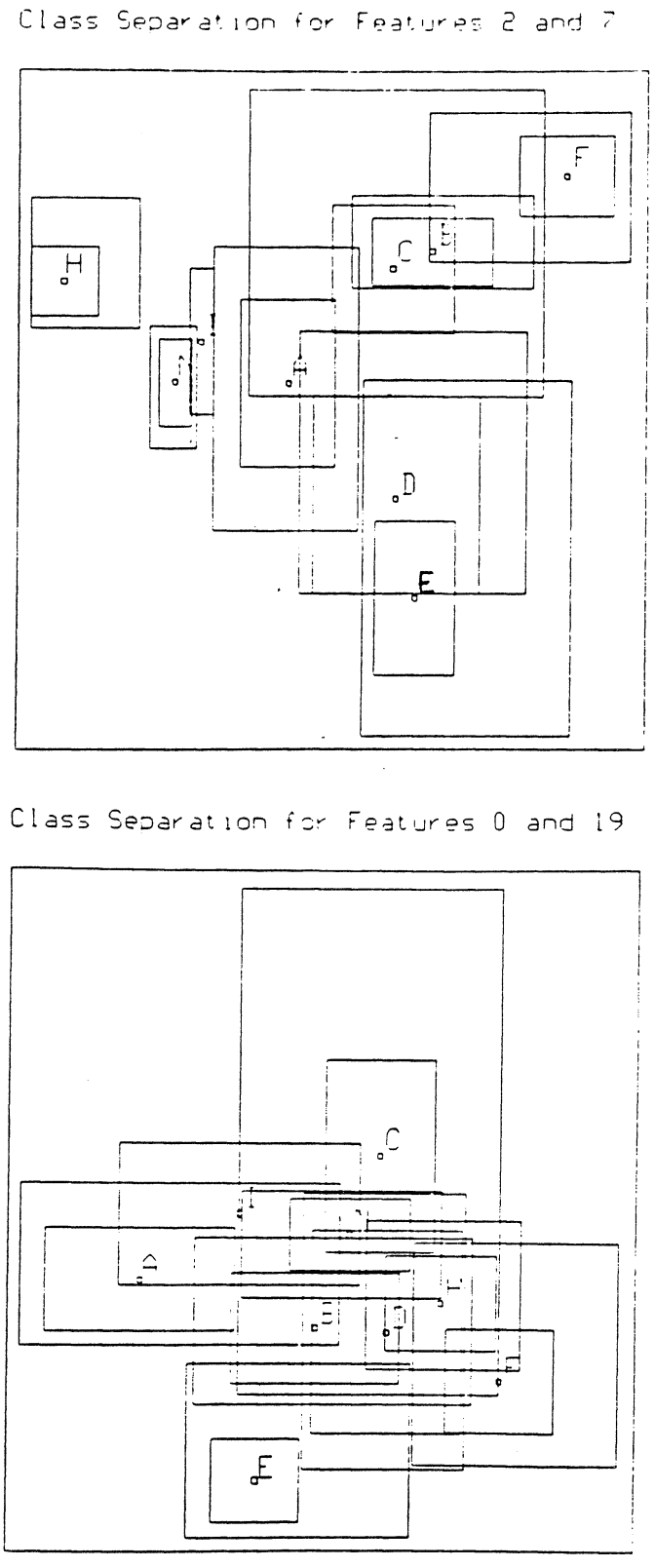


Figure 10. Feature Pair Class Separation Plots (Top-Good, Bottom-Bad)

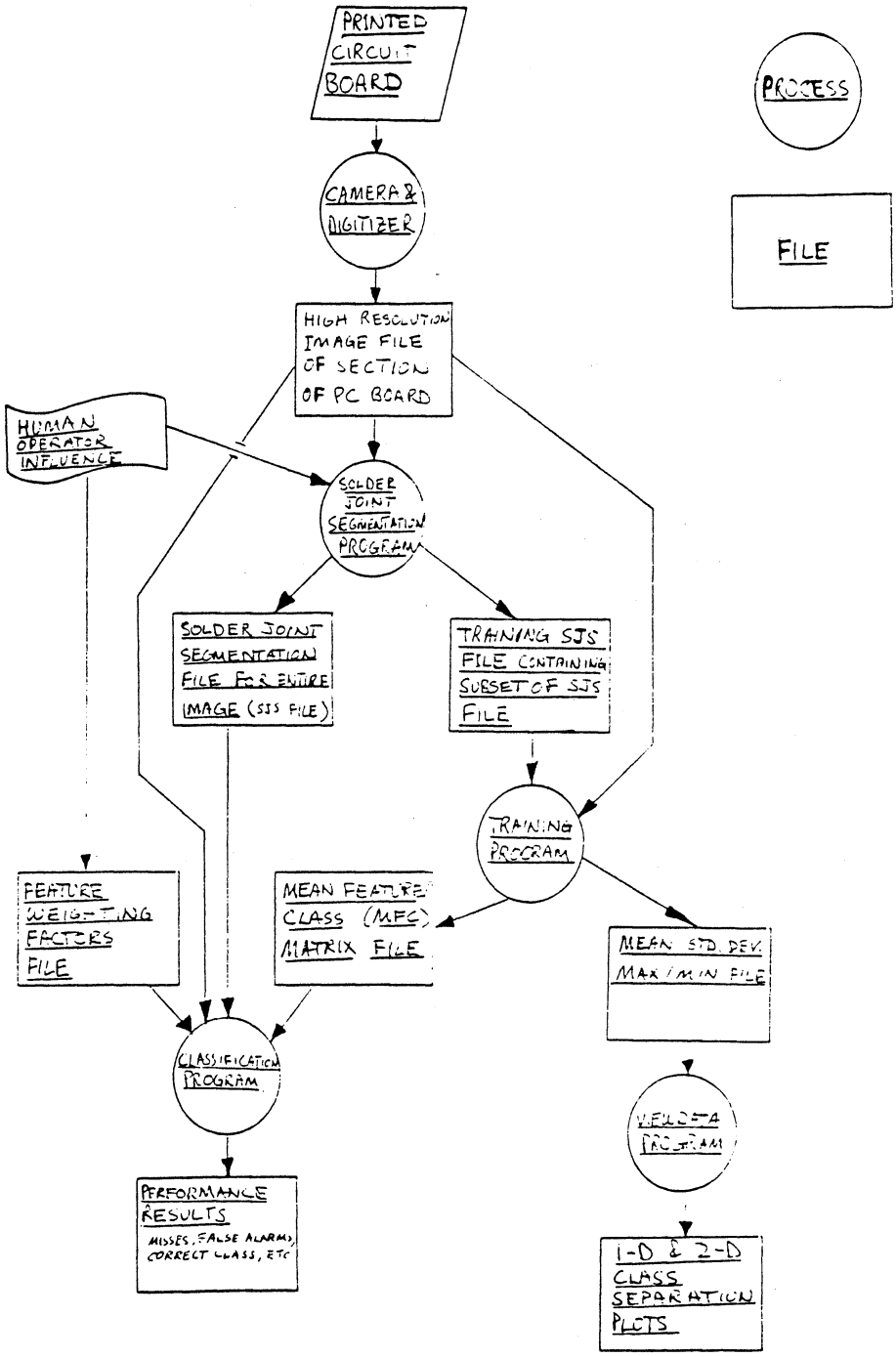


Figure 11. Dataflow Diagram for Preliminary Inspection Algorithm

Weight Vector

$$\underline{w} = [\, w_0 \quad w_1 \quad \cdots \quad w_{19} \,]^T$$

Weight Vector Table				
Related Feature	**Component**	**Vector 1**	**Vector 2**	**Vector 3**
V_{tot}	w_0	100.0	320.0	100.0
σ	w_1	10.0	0.0	10.0
V_{cew}	w_2	600.0	600.0	600.0
V_{mid}	w_3	200.0	0.0	200.0
V_{ofr}	w_4	1000.0	0.0	1000.0
I_{aa}	w_5	200.0	100.0	200.0
I_{bb}	w_6	200.0	0.0	200.0
I_{cc}	w_7	600.0	600.0	600.0
I *ratio*	w_8	0.0	80.0	10.0
$tr\ I_{tot}$	w_9	100.0	0.0	100.0
$dms\ I_{tot}$	w_{10}	1.0	0.0	1.0
$\min f$	w_{11}	1.0	0.0	100.0
K_{avg}	w_{12}	1.0	0.0	0.0
H_{avg}	w_{13}	1.0	0.0	1.0
b_{avg}	w_{14}	1.0	0.0	1.0
p_+	w_{15}	1.0	0.0	60.0
p_-	w_{16}	1.0	0.0	100.0
$\sqrt{g}_{avg}$	w_{17}	1.0	0.0	1.0
A	w_{18}	1.0	0.0	1.0
A *ratio*	w_{19}	1.0	0.0	20.0

Figure 12. Weight Vector Table

Experiment 1: Twelve Member Class List and Weight File 1					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	94.4	3.6	2.0	61.7
A2	A2	82.7	9.6	7.6	61.9
A3	A3	94.9	3.6	1.5	70.6
A4	A4	85.7	5.6	8.7	58.7
A5	A5	75.4	17.5	7.1	60.7
A6	A6	86.2	8.6	5.2	55.2
A7	A7	84.4	2.4	13.3	53.1
B1	B1	89.3	8.9	1.8	69.6
B2	B2	87.5	1.6	10.9	73.4
B3	B3	77.3	9.1	13.6	65.2
B4	B4	98.4	0.0	1.6	60.9
B5	B5	84.4	12.5	3.1	68.8
B6	B6	84.4	3.1	12.5	68.8
AVG	---	86.5	6.6	6.8	63.7

Experiment 2: Twelve Member Class List and Weight File 2					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	95.4	3.1	1.5	57.7
A2	A2	80.2	13.7	6.1	60.4
A3	A3	91.9	5.6	2.5	69.0
A4	A4	85.7	4.1	10.2	61.2
A5	A5	75.4	16.6	8.1	54.0
A6	A6	87.6	9.0	3.3	52.4
A7	A7	83.4	1.9	14.7	54.0
B1	B1	87.5	10.7	1.8	69.6
B2	B2	95.3	0.0	4.7	79.7
B3	B3	81.8	6.1	12.1	69.7
B4	B4	95.3	0.0	4.7	59.4
B5	B5	82.8	9.4	7.8	68.8
B6	B6	89.1	3.1	7.8	68.8
AVG	---	87.0	6.4	6.6	63.4

Experiment 3: Twelve Member Class List and Weight File 3					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	96.4	2.6	1.0	74.5
A2	A2	83.2	9.6	7.1	67.0
A3	A3	95.9	2.0	2.0	76.6
A4	A4	91.8	2.6	5.6	66.8
A5	A5	81.0	12.3	6.6	65.9
A6	A6	89.0	5.7	5.2	61.9
A7	A7	88.2	0.9	10.9	57.8
B1	B1	91.1	7.1	1.8	71.4
B2	B2	92.2	0.0	7.8	79.7
B3	B3	77.3	10.6	12.1	66.7
B4	B4	98.4	0.0	1.6	60.9
B5	B5	84.4	12.5	3.1	71.9
B6	B6	82.8	3.1	14.1	73.4
AVG	---	88.6	5.3	6.1	68.8

Figure 13. Experimental Results for Twelve Classes

Experiment 4: Six Member Class List and Weight File 1					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	94.9	4.1	1.0	70.9
A2	A2	83.8	8.6	7.6	72.1
A3	A3	95.4	1.5	3.0	75.6
A4	A4	81.1	5.1	13.8	64.3
A5	A5	91.9	3.3	4.7	74.4
A6	A6	88.6	3.8	7.6	71.0
A7	A7	82.9	3.8	13.3	67.8
B1	B1	91.1	8.9	0.0	73.2
B2	B2	87.5	1.6	10.9	73.4
B3	B3	77.3	9.1	13.6	65.2
B4	B4	96.9	1.6	1.6	65.6
B5	B5	84.4	12.5	3.1	68.8
B6	B6	84.4	3.1	12.5	68.8
AVG	---	87.7	5.2	7.1	70.1

Experiment 5: Six Member Class List and Weight File 2					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	93.4	5.1	1.5	64.3
A2	A2	85.8	7.6	6.6	71.6
A3	A3	93.9	2.5	3.6	73.6
A4	A4	82.7	3.1	14.3	62.2
A5	A5	89.6	3.3	7.1	70.1
A6	A6	89.5	3.8	6.7	72.9
A7	A7	81.0	2.4	16.6	65.4
B1	B1	87.5	10.7	1.8	78.6
B2	B2	95.3	0.0	4.7	78.1
B3	B3	81.8	6.1	12.1	69.7
B4	B4	95.3	0.0	4.7	65.6
B5	B5	82.8	9.4	7.8	68.8
B6	B6	89.1	3.1	7.8	68.8
AVG	---	88.3	4.4	7.3	70.0

Experiment 6: Six Member Class List and Weight File 3					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	76.5	23.0	0.5	9.2
A2	A2	73.1	21.8	5.1	15.2
A3	A3	83.8	15.7	0.5	13.2
A4	A4	77.0	19.4	3.6	13.8
A5	A5	68.2	30.3	1.4	10.9
A6	A6	65.2	32.9	1.9	19.5
A7	A7	67.8	25.1	7.1	13.7
B1	B1	92.9	7.1	0.0	12.5
B2	B2	67.2	29.7	3.1	12.5
B3	B3	72.7	16.7	10.6	13.6
B4	B4	53.1	46.9	0.0	9.4
B5	B5	65.6	34.4	0.0	28.1
B6	B6	67.2	26.6	6.3	28.1
AVG	---	71.6	25.3	3.1	15.4

Figure 14. Experimental Results for Six Classes

Experiment 7: Six Member Class List and Weight File 1					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A1	94.9	4.1	1.0	70.9
A2	A1	88.8	5.6	5.6	75.6
A3	A1	94.9	1.5	3.6	78.7
A4	A1	87.2	3.1	9.7	67.9
A5	A1	92.4	4.3	3.3	75.8
A6	A1	88.6	4.8	6.7	74.3
A7	A1	86.7	4.3	9.0	70.6
B1	B1	91.1	8.9	0.0	73.2
B2	B1	89.1	6.3	4.7	54.7
B3	B1	75.8	7.6	16.7	47.0
B4	B1	82.8	12.5	4.7	60.9
B5	B1	78.1	14.1	7.8	62.5
B6	B1	73.4	14.1	12.5	53.1
AVG	---	86.5	7.0	6.6	66.6

Experiment 8: Six Member Class List and Weight File 1					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A4	85.2	9.2	5.6	55.6
A2	A4	85.3	7.1	7.6	68.0
A3	A4	91.9	2.0	6.1	70.6
A4	A4	81.1	5.1	13.8	64.3
A5	A4	89.1	5.2	5.7	68.7
A6	A4	83.3	4.3	12.4	64.8
A7	A4	79.1	8.1	12.8	64.9
B1	B4	89.3	10.7	0.0	66.1
B2	B4	89.1	9.4	1.6	65.6
B3	B4	81.8	6.1	12.1	56.1
B4	B4	96.9	1.6	1.6	65.6
B5	B4	87.5	6.3	6.3	73.4
B6	B4	87.5	4.7	7.8	65.6
AVG	---	86.7	6.1	7.2	65.3

Experiment 9: Six Member Class List and Weight File 1					
Image	MFC	Good/Bad_%	Miss_%	False_Alarm_%	Correct_Class_%
A1	A6	85.2	13.3	1.5	60.2
A2	A6	88.3	5.1	6.6	75.1
A3	A6	93.9	1.0	5.1	69.0
A4	A6	87.2	2.0	10.7	55.1
A5	A6	86.7	8.5	4.7	66.8
A6	A6	88.6	3.8	7.6	71.0
A7	A6	86.7	2.8	10.4	69.2
B1	B6	87.5	10.7	1.8	67.9
B2	B6	84.4	7.8	7.8	51.6
B3	B6	80.3	4.5	15.2	42.4
B4	B6	98.4	0.0	1.6	67.2
B5	B6	85.9	0.0	14.1	68.8
B6	B6	84.4	3.1	12.5	68.8
AVG	---	87.5	4.8	7.7	64.1

Figure 15. Experimental Results for Fixed MFC Matrices