Jurnal Teknologi, 39(D) Dis. 2003: 87–104

NEURAL NETWORK PARADIGM FOR CLASSIFICATION OF DEFECTS ON PCB

RUDI HERIANSYAH[1], SYED ABDUL RAHMAN AL-ATTAS[2], & MUHAMMAD MUN'IM AHMAD ZABIDI[3]

Abstract. A new technique is proposed to classify the defects that could occur on the PCB using neural network paradigm. The algorithms to segment the image into basic primitive patterns, enclosing the primitive patterns, patterns assignment, patterns normalization, and classification have been developed based on binary morphological image processing and Learning Vector Quantization (LVQ) neural network. Thousands of defective patterns have been used for training, and the neural network is tested for evaluating its performance. A defective PCB image is used to ensure the function of the proposed technique.

Keywords: PCB, defects classification, morphological image processing, LVQ

Abstrak. Satu teknik baru dicadangkan untuk mengkelaskan kerosakan yang boleh terjadi pada PCB menggunakan paradigma rangkaian neural. Algoritma untuk membahagi-bahagikan imej menjadi corak primitif, melingkupi corak primitif berkenaan, penandaan corak, normalisasi corak, dan pengkelasan telah dibangunkan berdasarkan pemprosesan imej morfologi penduaan dan rangkaian neural Learning Vector Quantization (LVQ). Ribuan corak rosak telah digunakan untuk tujuan latihan, dan rangkaian neural diuji untuk menilai prestasinya. Satu imej PCB yang rosak digunakan untuk memastikan teknik yang dicadangkan berfungsi.

Kata kunci: PCB, pengkelasan kerosakan, pemprosesan imej morfologi, LVQ

1.0 INTRODUCTION

A printed circuit board (PCB) is a basic component of many electronic devices. The quality of PCBs will have a significant effect on the performance of many electronic products. Conventionally, visual inspection of PCBs is done manually by inspectors. It is known that humans are subject to make mistakes, and they are slow and less consistent, whereas automatic inspection systems remove the subjective aspects and provide fast quantitative dimensional assessments [1], [2]. When applied at each appropriate step of the assembly process, they can reduce rework costs. All of these mean better quality at a lower cost. Undoubtedly, the automation of visual inspection will increase productivity and improve product quality [3].

[1, 2&3] Dept. of Microelectronis & Computer Engineering (MiCE), Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia. Tel.: 07-5535274, Fax: 07-5566272. e-mail: rudi_hn@ieee.org

The process in PCB inspection (in term of the defects identification) could be separated into two main stages: (1) The defects detection, and (2) The defects classification. In defect detection, it is not important to know the type of defects. But in defect classification, it is desired to know the type of the detected or identified defects. Usually, the defects classification will take place after the defect detection mechanism has been carried out [1]. Presently, there has been a lot of work concentrating on the detection of defects on PCB. In general, one can group the approaches of these defects detection techniques into three major classes: (1) Reference based approaches, (2) Non-reference based approaches, and (3) Hybrid based approaches (combination of the first two groups) [1], [3]. In a PCB, the defects could be classified as either visual defects or functional defects. Unlike the visual defects, the functional defects could seriously affect the function of a PCB. Based on this reason, it is essential to classify the type of defects that exist on a PCB. Hence the development of algorithms for PCB defects classification is crucial. In addition, this work is also motivated by the fact that there are not many algorithms or systems, which have been developed for PCB defects classification. Many existing or developed algorithms just focus on PCB defects detection [1], [3]. This work then, is undertaken to develop an algorithm for defects classification in order to identify correctly the detected defects on a PCB.

Organization of this paper will be as follows. The design of training set and test set data for neural network is briefly discussed, and followed by neural network design discussion. Next section discusses on segmentation of the PCB image into basic primitive patterns, and the windowing technique to enclose these primitive patterns is briefly described. The patterns assignment is employed to determine the position of the defects relative to the test image and the defect patterns with its position are called then as the candidate patterns, and it is discussed in the next section. After that the discussion is focused on the candidate patterns normalization before they are fed into the neural network. The experimental result of the algorithm is given in the next section. Discussion and summary of the experimentation are presented in the final section.

2.0 NEURAL NETWORK DATA DESIGN

Based on a study in [3] and [4], there are about 14 defects that could occur on the PCB *i.e.* short, excessive short, missing conductor, conductor too close, missing hole, breakout (c-pattern), wrong size hole, pinhole, open, mousebite, spur, underetch, overetch, and spurious copper. In this work, short and excessive short defect will be treated as the same as short defect. Underetched and overetched will also be treated as the same as the etching problem defect. Hence, there will be only 12 defects that will be classified.

In this approach for the purpose of training and testing the neural network, 11 defective patterns have been designed. The designed pattern is in 8 x 8 pixels size, with binary format. Using this technique, spurious copper defect could be identified during pattern assignment operation (discussed in Section 6.0), therefore training

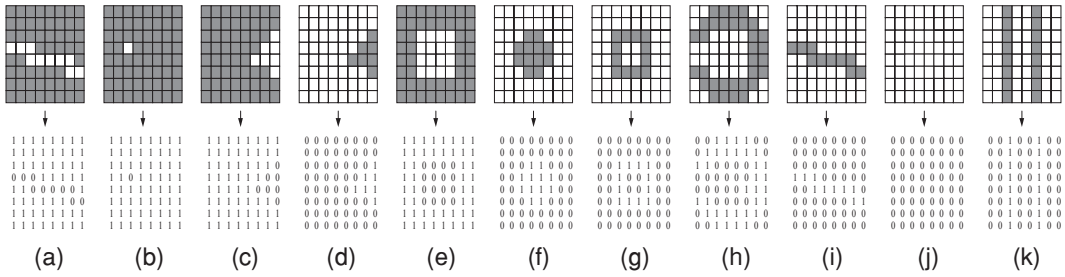

Figure 1 Designed defective patterns for NN training and testing. (a) Open. (b) Pinhole. (c) Mousebite. (d) Spur. (e) Etching problem. (f) Missing hole. (g) Wrong size hole. (h) Breakout. (i) Short. (j) Missing conductor. (k) Conductor too close.

patterns for this type of defect is not needed. Figure 1 shows some of these designed patterns.

3.0 NEURAL NETWORK DESIGN

The PCB defects could be formed into three groups: the defects on the foreground only, the defects on the background only, and the defects on both foreground and background (the defect is caused by interaction with other object). Figure 2 shows these defects groupings.

To classify the defects, LVQ neural network has been selected as the classifier. The designed patterns are trained and tested using this neural network. For the neural network implementation, only two groups of defects will be used for training (*i.e.* the foreground, and the background and foreground). Defects on background only (spurious copper) could be classified directly without any need to pass it to the neural network.

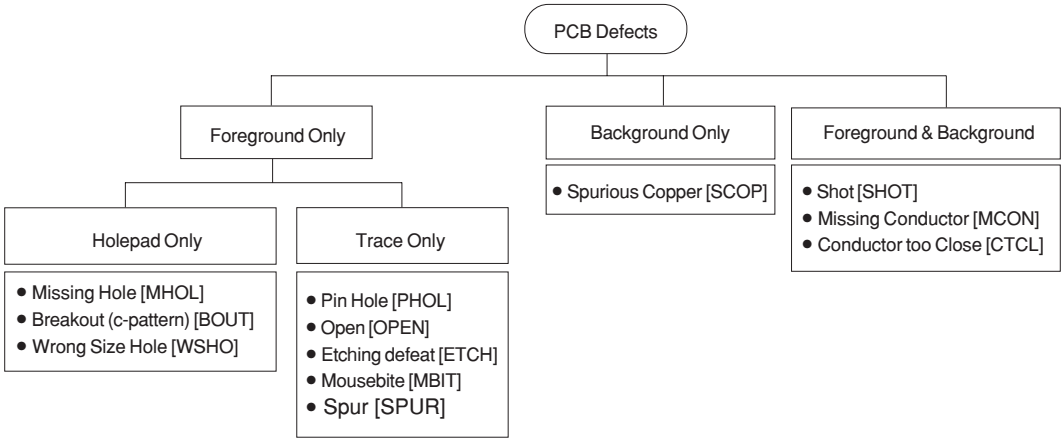

Figure 2 The defects grouping

The foreground only defects consist of two types of defects based on the defects location: the defects on hole pad and the defects on trace only. Hence, there will be three groups of defects: the defects on foreground and background (Group #1), the defects on hole pad only (Group #2), and the defects on trace only (Group #3), which will be the basis for designing the neural network. In the implementation of the neural network, three neural network designs are used for detecting three defects groups. The *first network* (LVQ #1) is for the foreground and background group which detects the short, missing conductor, and conductor too close. The *second network* (LVQ #2) is for the hole pad only group which identifies missing hole, wrong size hole, and breakout. Finally, the *third network* (LVQ #3) is for the pads and line group only (including rectangular pads, square pads, and lines), which classifies pinhole, open, mousebite, spur, and etching problem defects.

4.0 SEGMENTATION INTO PRIMITIVE PATTERNS

Segmentation is carried out using binary morphological image processing [5]. To simplify the operation, the primitive patterns are grouped into four main types: hole pad, rectangular pad, square pad, and line. In segmentation operation, morphological operators such as dilation, opening, and closing are used. This segmentation involves a series of image processing operations (morphological image processing based segmentation and subtraction operation). A detailed algorithm for this segmentation could be found in [6].

5.0 PRIMITIVE PATTERNS WINDOWING

After segmenting the PCB image into basic primitive patterns, the next step is to enclose each pattern so that only pixels under this window will be processed. Windowing operation is also employed to the detected defective patterns. Further details could be found in [6].

6.0 PATTERNS ASSIGNMENT

The flowchart for the defect classification based on the neural network is shown in Figure 3. The segmentation into basic primitive patterns is applied only onto the PCB reference image. The defects detection applied in this work is based on the image subtraction operation [1]. The segmented primitive patterns of the reference image will be enclosed using the windowing technique, and these window coordinates will be mapped onto the test image to generate the same window coordinates for the test image. At the same time, the detected defects from previous subtraction operation will also be enclosed using the same windowing technique. At this stage there will be three windows coordinates or three enclosed patterns: enclosed primitive patterns of the reference image *rpat[i]* (with their window coordinates *rowStartR, rowEndR,*

colStartR, and *colEndR*), enclosed primitive patterns of the test image *tpat[i]* (with their window coordinates *rowStartT, rowEndT, colStartT*, and *colEndT*), and enclosed detected defects *dpat[i]* (with their window coordinates *rowStartD, rowEndD, colStartD,* and *colEndD*), with *i* is the *i-th* pattern. See Figure 4 for the positions of these coordinates in each of their patterns. Note that window coordinates of the test image *tpat[i]* (*rowStartT, rowEndT, colStartT*, and *colEndT*) are equal to the window coordinates of the reference image *rpat[i] (rowStartR, rowEndR, colStartR*, and *colEndR*).

The next step is to do *the assignment operation*. The aim of this assignment operation is to define the position of the enclosed defect patterns *dpat[i]* (based on its window coordinates) relative to the enclosed test image patterns *tpat[i]* and *tpat[j]*, with *i* and *j* are the *i-th* and the *j-th* pattern respectively. Therefore based on this, the *dpat[i]* position can be either *match, inside, outside, a part of, between*, or *close* relative to *tpat[i]* and *tpat[j]*. Remember that the active patterns in this assignment operation are *dpat[i], tpat[i]*, and *tpat[j]*. The need in using two enclosed test patterns *tpat[i]* and *tpat[j]* is in order to check *dpat[i]* for the *between* case. Figure 5 shows active patterns for the assignment operation.

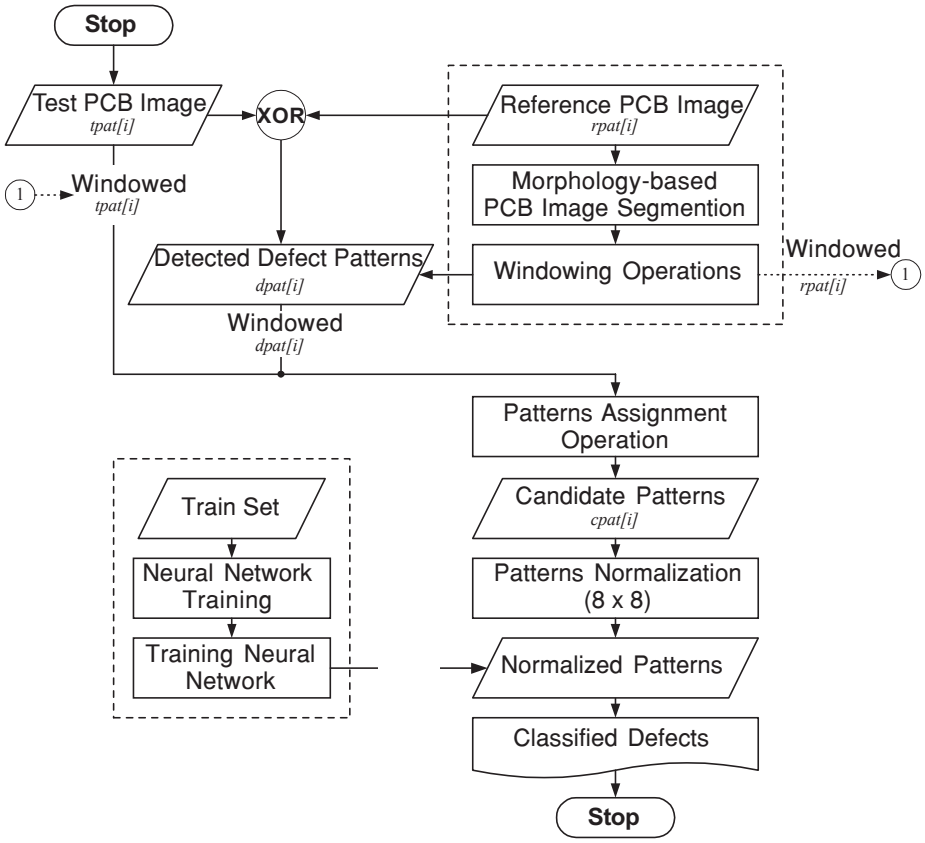

Figure 3 Neural network based PCB defects classification

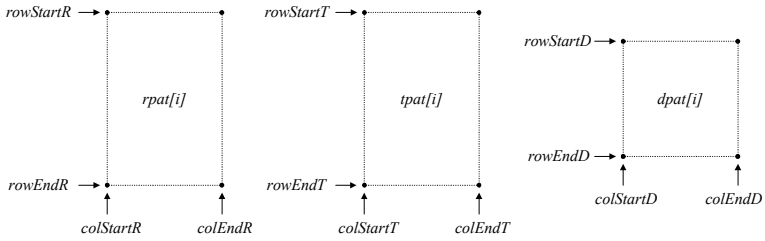

Figure 4 Window coordinates for each type of pattern

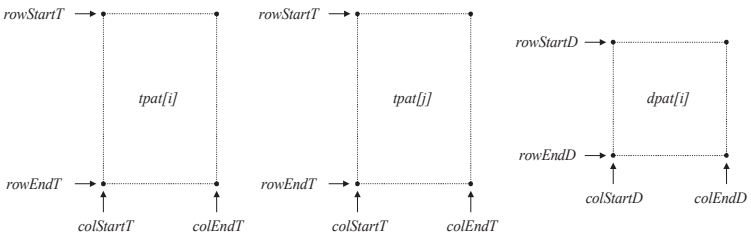

Figure 5 Active patterns for assignment operation

The position of *dpat[i]* whether it is *match*, *inside*, *outside*, *a part of*, *between*, or *close* to *tpat[i]* is illustrated in Figure 6 to 11.

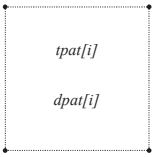

Figure 6 The case for *dpat[i] match tpat[i]*

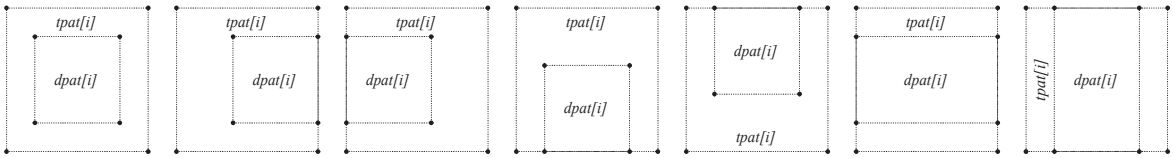

Figure 7 The case for *dpat[i] inside tpat[i]*

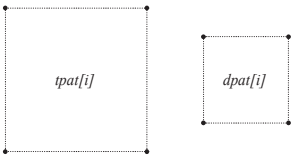

Figure 8 The case for *dpat[i] outside tpat[i]*

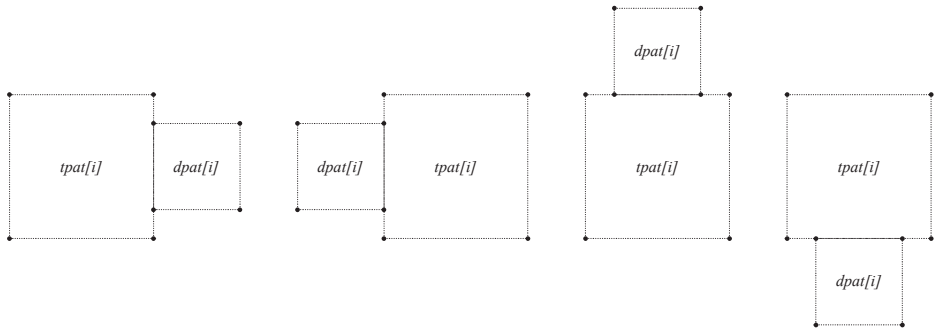

Figure 9 The case for *dpat[i] a part of tpat[i]*

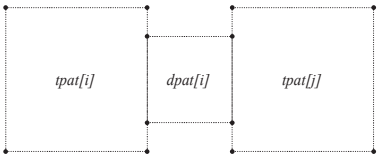

Figure 10 The case for *dpat[i] between tpat[i]* and *tpat[j]*

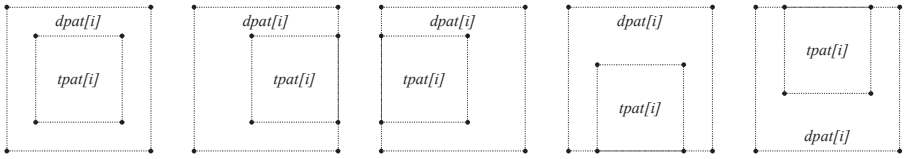

Figure 11 The case for *dpat[i] close* to *tpat[i]*

Note that in determining these cases, the process is applied to every defective pattern from *dpat[1]* to *dpat[n]*, relative to *tpat[1]* to *tpat[n]*. Once every *dpat[i]* pattern position has been determined with respect to *tpat[i]*, *dpat[i]* will be assigned to a new pattern name *cpat[i]*.

Based on *cpat[i]* pattern position and *tpat[i]* pattern's shape type (hole pad, pads, and lines) where *dpat[i]* is attached to, *cpat[i]* pattern will be assigned to a new window coordinates. These coordinates will either similar to its original *dpat[i]* coordinates (*rowStartD*, *rowEndD*, *colStartD*, *colEndD*) or to *tpat[i]* coordinates (*rowStartT*, *rowEndT*, *colStartT*, *colEndT*). In addition, the input image to the neural net, which will be called as the *source image*, will also be determined based on these two criteria. The other step in this stage is to decide to which neural network design (LVQ #1, #2, or #3), *cpat[i]* belongs to. The description for these steps (assignment operation) is listed in Table 1. Note from the table that the *outside* case is not included. The reason is, any *cpat[i]* pattern with this case, will be automatically identified as *spurious copper*

Table 1 Assignment operation

Position	Shape type of *tpat[i]*	*cpat[i]* New window coordinates	Source image	LVQ	To identify
o *Match*	Line	*dpat[i]*	Ref. Image	#1	MCON
o *Match, Inside*	Hole pad	*tpat[i]*	Sub. Image	#2	MHOL, WSHO, BOUT
o *Match*	Pad	*dpat[i]*	Sub. Image	#3	ETCH
o *Inside*	Pad, Line	*tpat[i]*	Test Image	#3	PHOL, OPEN, MBIT
o *A part of*	Any types	*dpat[i]*	Sub. Image	#3	SPUR
o *Between*	Any types	*dpat[i]*	Sub. Image	#1	SHOT
o *Close*	Line	*dpat[i]*	Sub. Image	#1	CTCL

defect. Hence it does not need further processing. Note that in Table 1, *Ref. Image* means reference image, and *Sub. Image* means subtracted image (or image after subtraction operation).

7.0 PATTERNS NORMALIZATION

The patterns are normalized into the size of 8×8 pixels. At this stage, any test pattern (primitive pattern or defective pattern), which is less than 8×8, will be enlarged, while the one that is larger will be reduced into 8×8 in size. The scaling algorithm is described below.

Suppose that the pattern before the normalization is named as old_pat_{ij} with i and j denote the row and $colu_{mn}$ respectively, where: $i = 1...k$ and $j = 1...l$. If both k and l are less than 8×8, the pattern will undergo the enlargement operation. Otherwise, if k and l are greater than 8×8, the pattern will undergo the minification operation.

Let the pattern after normalization due to *minification* or *enlargement* be represented by new_pat_{mn} and new_pat_{pq}, where: $m = 1...8$, $n = 1...l$, and $p = q = 1...8$. Note that is the pattern after minification or enlargement of old_pat_{ij} on the row (column) side only, and is the pattern after minification or enlargement of on the $colu_{mn}$ (or row) side. See Figure 12 for the illustration of old_pat_{ij} relationship with new_pat_{mn} and new_pat_{pq}.

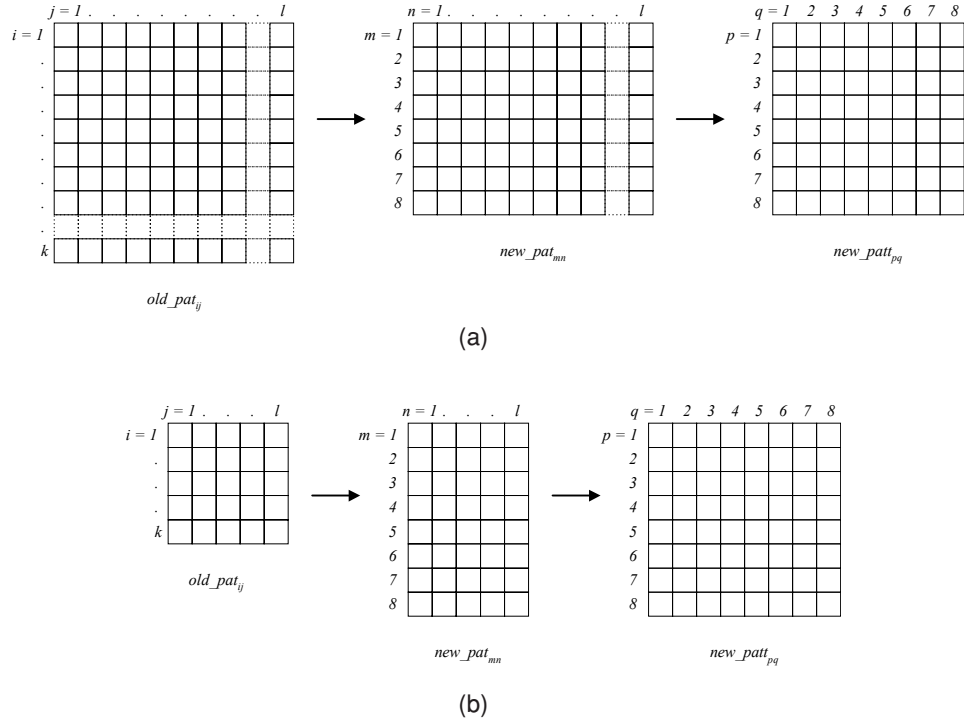

Figure 12 Relation between old_pat_{ij} with new_pat_{mn} and new_pat_{pq}. (a) Minification operation. (b) Enlargement operation.

7.1 Minification: Row

Let s_a be a *row combination factor*, *i.e.* it indicates how many rows from old_pat_{ij} need to be combined in order to be a row in new_pat_{mn}, and a indicates the row in new_pat_{mn}. In this situation, s_1 and s_8 will always equal to 1. In other words, this is to say that the row of new_pat_{mn} is equal to row 1 and row k of old_pat_{ij}. The reason for doing this is to preserve the border area.

If for example, $s_3 = 2$, this means that the 3rd row of new_pat_{mn} is composed of 2 rows of old_pat_{ij}. In row composition or combination process, a certain rule will be applied and this is discussed in Section 7.3. The following paragraphs discuss rules in determining combination factor s_a for $a = 2, 3, 4, 5, 6$, and 7.

As stated before s_1 will be set to the first row of old_pat_{ij} while s_8 will be set to the last row of old_pat_{ij}. For the 2nd to the 7th rows of new_pat_{mn}, they are constructed based on the combination factor $s_2 ..., 7$. Note that in minification of the row, for every old_pat_{ij} pattern, only the index i will be manipulated, the index j will be used in minification of the column, once the minification of the row has been done. Since the first row and the last row of new_pat_{mn} have already been assigned, there are 6 rows more that need to be filled up. Let then $restRow = i - 2$, indicates the remaining row of

old_pat_{ij} that need to be processed. Now, for every row in new_pat_{mn}, the combination factor, s_a, for $a = 2, ..., 7$, is given in Equation (1). The *ROUND* operator in the equation is to ensure that the result will always be an integer number.

$$rowNum = ROUND\left[\frac{restRow}{6}\right] \quad (1)$$

To ensure whether all i have been used for $s_2 ..., 7$, one of the combination factors, for example s_2 will be checked by using the following equation:

$$rowCheck = (restRow) - (5 \times s_2) \quad (2)$$

There could be three cases for *rowCheck*.

o ***rowCheck > 0***. If $rowCheck > 0$ then from Equation (1):

$$s_2...6 = rowNum \quad (3a)$$

$$s_7 = rowCheck \quad (3b)$$

To check whether the all-available rows i have been used correctly, the following equation is used:

$$restRow = \sum_{a=2}^{7} s_a \quad (4)$$

o ***rowCheck < 0***. There is only one case for $rowCheck < 0$ in which $i = 11$. Then the combination factor sa for $a = 2...4$ is as given in Equation (5a). The combination factor for the next three rows ($a = 5...7$), is given in Equation (5b):

$$s_2...4 = rowNum \quad (5a)$$

(5b)

o ***rowCheck = 0***. For this case, the following equations are applied:

$$s_2...5 = rowNum \quad (6a)$$

$$s_7 = 1 \quad (6b)$$

$$s_6 = s_2 - s_7 \quad (6c)$$

7.2 Minification: Column

Minification for the column side is carried out once minification for the rows is already completed. A new-reduced pattern new_patt_{pq} is constructed based on the reduced pattern new_pat_{mn} from the previous step. Basically, the rules are similar to the row minification combination operation rule.

7.3 Minification: Pixel Combinatiopn Operation

At this point, it is important to note that normalization process (either minification or enlargement) should preserve the property of the original image. In this project, it has been found that a simple logical AND operator could perform this task. Let *p1* and *p2* be the pixels in the 1st row (column) and 2nd row (column) respectively. Then the result of combining these two rows (columns) *p12* can be expressed as:

$$p12 = p1 \cdot \ p2 \tag{7}$$

7.4 Enlargement: Row

Enlargement operation is an opposite process from the minification operation. Enlargement operation will be applied only when $i < 8$ and $j < 8$ of old_pat_{ij}. If $i < 8$, then old_pat_{ij} will be enlarged row-wise to form a new pattern new_pat_{mn}, and if $n < 8$ then new_pat_{mn} will be enlarged column-wise to form another pattern new_patt_{pq}.

Let the *enlargement factor* be denoted by *sb*, where $b = 1,..., 7$ indicates the position of row of new_pat_{mn}. Unlike in the minification operation, enlargement operation uses these enlargement factors *sb* to expand one row of old_pat_{ij} to become several rows of new_pat_{mn}. Similarly, these factors are used to expand one column of new_pat_{mn} into several columns of new_patt_{pq}. In other words, the enlargement factor is to duplicate the numbers of the original row (column) accordingly.

In the event that old_pat_{ij} has only one row or $i = 1$ then the enlargement for both in row and column side will be enlarged 8 times as in Equation (8a).

$$rowCnt = 8 \tag{8a}$$

But if $i > 2$, then Equation (8a) has to be modified as:

$$rowCnt = ROUND\left[\frac{8}{1}\right] \tag{8b}$$

where the *ROUND* operator is to ensure that the result will always be an integer. From Equation (8b), the enlargement factor s_b, for $i = 1$ to 7, will be as follows:

- $i = 1$. For $i = 1$, s_1 is as in Equation (9).

$$s_1 = rowCnt \tag{9}$$

- $i = 2$. For $i = 2$, $s_{1,2}$ is as in Equation (10).

$$s_{1,2} = rowCnt \tag{10}$$

- $i = 3$. For $i = 3$, $s_{1...3}$ is as given in Equations (11a) and (11b).

$$s_{1,2} = rowCnt \tag{11a}$$

$$s_3 = 2 \tag{11b}$$

To check whether Equation (11a) and (11b) has distributed i correctly, Equation (12) could be used.

$$\sum_{b=1}^{i} sb = 8 \tag{12}$$

- ***i* = 4**. For $i = 4$, $s_{1...4}$ values are given in Equation (13).

$$s_{1...4} = rowCnt \tag{13}$$

- ***i* = 5**. For $i = 5$, $s_{1...5}$ values are given in Equations (14a) and (14b).

$$s_{1...3} = rowCnt \tag{14a}$$

$$s_{4...5} = 1 \tag{14b}$$

- ***i* = 6**. For $i = 6$, $s_{1...6}$ values are given in Equation (15a) and (15b).

$$s_{1...4} = rowCnt \tag{15a}$$

$$s_{5,\,6} = 2 \tag{15b}$$

- ***i* = 7**. For $i = 7$, $s_{1...7}$ values are given in Equation (16a) and (16b).

$$s_{1...6} = rowCnt \tag{16a}$$

$$s_7 = 2 \tag{16b}$$

7.5 Enlargement: Column

The enlargement for column wise has the same rule as with the enlargement for the row wise. All equations, Equation (9) to (16b), are employed for enlargement in column wise. After enlargement on the row wise is applied (if $i < 8$), and if , then the enlargement on column wise is carried out.

8.0 EXPERIMENTAL RESULTS

In PCB defects classification, neural network training is an off-line operation. Neural network was trained using LVQ net. There are three LVQ nets for three groups of defects. There are 64 inputs for each LVQ net, 3 outputs for LVQ #1 and #2, and 5 outputs for LVQ #3. Output was represented with [1 2 ... n], where n indicates type of *n-th* defect. Figure 13 shows neural network topology for LVQ #1, #2, and #3. In this approach, the numbers of first layer neurons (hidden layer neurons) for each LVQ net was selected by testing with various numbers of neurons, and by considering the most optimum one. By using the optimum parameters as already described in the previous section, each neural network was trained and tested by using various numbers of data.

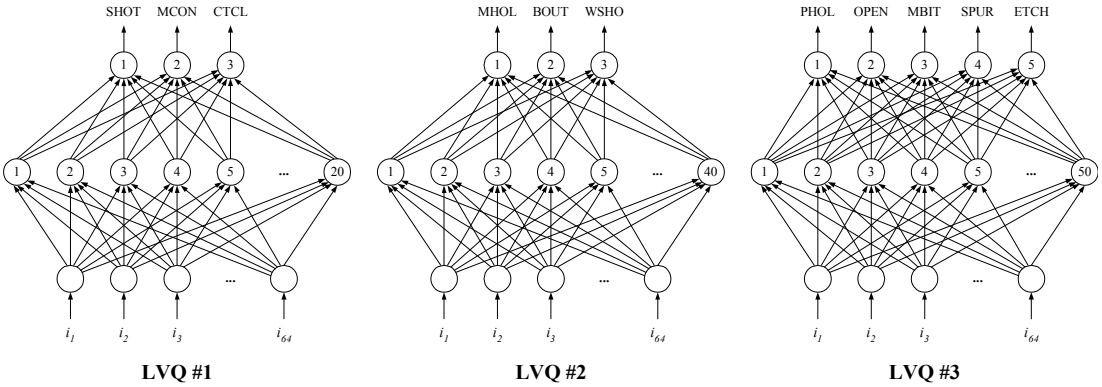

Figure 13 Neural network topology

There are 500 defective patterns for each type of defect as the training set (or a total of 5500 defective patterns for all defects for the training data set) and 100 defective patterns per set for the test data set for each type of defect. Classification result is shown in Table 2. The table shows an overall correct classification result as 97.55%.

Table 2 Classification result

Defects	**Classification (%)**
Foreground & Background	
o Short [SHOT]	**100**
o Missing Conductor [MCON]	**100**
o Conductor too Close [CTCL]	**100**
Holepad Only	
o Missing Hole [MHOL]	**96.00** (4.00 WSHO)
o Wrong Size Hole [WSHO]	**100**
o Breakout [BOUT]	**100**
Trace Only	
o Pinhole [PHOL]	**92.00** (8.00 OPEN)
o Open [OPEN]	**93.00** (7.00 PHOL)
o Mousebite [MBIT]	**97.00** (1.00 PHOL, 2.00 SPUR)
o Spur [SPUR]	**97.00** (3.00 ETCH)
o Etching Problem [ETCH]	**98.00** (1.00 OPEN, 1.00 SPUR)

Figure 14(a) shows a synthetic 256×256 PCB image as the reference, and Figure 14(b) shows a defective image, both are in binary format, for an example, to test the developed algorithm. The first step is to segment the reference image into basic primitive patterns, and then to generate the window coordinates for these individual segmented primitive patterns. The next step is to subtract the test image from the reference image

in order to detect the defects, and again the same windowing technique is employed to the defective patterns to generate the windows coordinates.

The third step is to define the position of the defective pattern *dpat[i]* relative to the segmented test primitive patterns *tpat[i]*, or to find out whether *dpat[i]* is in one of these cases: *inside*, *match*, *outside*, *a part of*, *between*, or *close* to *tpat[i]*. After defining the case for each defects patterns, it is to be decided to which neural network (LVQ #1, #2, or #3) *dpat[i]* belongs to. A new pattern notation *cpat[i]* is introduced to replace the previous notation of *dpat[i]*. In this stage also, the coordinates of *cpat[i]* are defined whether they are taken from *dpat[i]* or *tpat[i]* coordinates, and the source image of the pattern will also be determined in this step. From this result, the patterns under the coordinates are then normalized based on the source image (working area) for that pattern. After the normalization operation, all the normalized *cpat[i]* are fed to the neural network for classification process. Every *cpat[i]* will be passed to appropriate LVQ Net (#1, #2, #3) according to its **LVQ ID**.

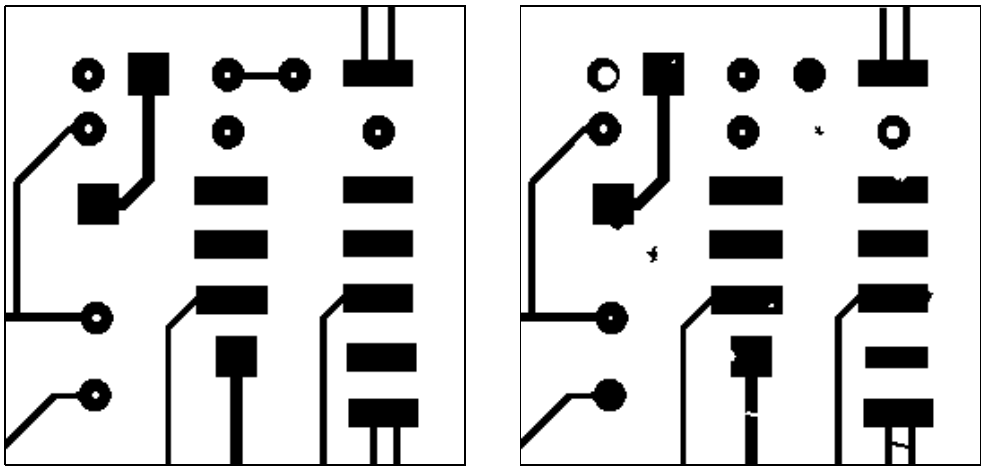

Figure 14 (a) Reference image. (b) Test image.

Figure 15 shows classification result for the test image. Table 3 is provided to appreciate the results obtained. Numbers under $colu_{mn}$ heading *cpat[i]* in Table 3 is associated to the numbers shown in Figure 15.

9.0 DISCUSSION

As shown in Figure 15 and Table 3, all defects could be successfully classified. For performance comparison, a *pixel-based* approach developed by Wu *et al.* [1] was used. At the time of writing this paper, this was the only algorithm designed for defect classification. The pixel-based approach could classify *seven defects* (short, missing hole, pinhole, open, mousebite, spur, and etching problem) [1], but the NN-based

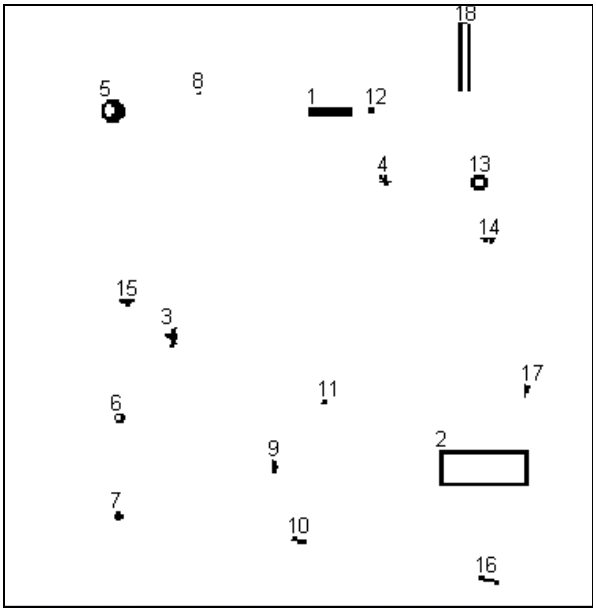

Figure 15 Classification result

Table 3 Defects classification

Defects Type	*cpat[i]*
Short [SHOT]	16
Missing Conductor [MCON]	1
Conductor too Close [CTCL]	18
Missing Hole [MHOL]	7, 12
Wrong Size Hole [WSHO]	6, 13
Breakout [BOUT]	5
Pinhole [PHOL]	8, 11
Open [OPEN]	10
Mousebite [MBIT]	9, 14
Spur [SPUR]	15,17
Etching Problem [ETCH]	2
Spurious Copper [SCOP]	3, 4

approach proposed in this work could classify *twelve defects* (short, missing conductor, conductor too close, missing hole, wrong size hole, breakout, pinhole, open, mousebite, spur, etching problem, and spurious copper).

Another issue, which is worth addressing, is the processing time in classifying each defect. In this approach there are few stages involved: *segmentation*, *windowing* (reference image and detected defects), *defects detection*, *pattern assignment*, *normalization*, and *classification*. For the neural network training part, since this process is done off-line, it does not affect the overall processing time. Among these five stages, the *windowing* task is the most time consuming operation (see Table 4), due to the operation working at pixel level, meaning that every individual pixel is analysed. In addition, the results of the windowing operation, *i.e.* window coordinates are saved into the hard disk, which adds more processing time.

Table 4 shows overall processing time for the sample image. The first two stages could be done separately and only once. By including segmentation and windowing of the reference image operation processing time, it takes 16.39s to classify all 18 defects on Pentium III 800 MHz with 256 MB RAM, while by excluding the segmentation and reference image windowing as in real implementation, it takes about 3.97s to classify all defects. The processing time for the classification stage is only 0.82s for 18 defects, or approximately 0.046s to classify each defects. For the pixel-based approach reported in [1], the processing time is 15.68s. It takes 5.26s for the defect detection and an average of 1.49s for classifying each defect.

Table 4 Classification time

Stages	Processing Time (s)
Segmentation	0.55
Windowing (reference image)	11.87
Defects detection	0.11
Windowing (test image)	2.50
Patterns assignment	0.16
Normalization	0.38
Defects classification	0.82

10.0 CONCLUSION

This paper proposes a new technique for PCB defects classification. It has been demonstrated that the proposed algorithm could classify correctly all possible type of defects that could occur on the PCB. It also has been shown that the defects classification technique based on neural network is better than pixel-based defects classification technique in term of numbers of classified defects and processing time.

REFERENCES

[1] Wu, W. Y., M. J. J. Wang, and C. M. Liu. 1996. "Automated Inspection of Printed Circuit Boards through Machine Vision". *Computers in Industry.* (29): 103-111.

[2] Wallace, A. M. 1988. "Industrial Application of Computer Vision Since 1982". *IEE Proc.* 135(3). 117-136.

[3] Moganti, M., F. Ercal, C. H. Dagli, and S. Tsunekawa. 1996. "Automatic PCB Inspection Algorithms: A Survey". *Computer Vision and Image Understanding.* 63(2).

[4] Charette, C., S. Park, R. Williams, B. Benhabib, and K. C. Smith. 1998. "Development and Integration of a Micro-computer Based Image Analysis System for Automatic PCB Inspection". Proceedings of International Conference on Computer Integrated Manufacturing. 129-135.

[5] Giardina, C. R. and E. R. Dougherty. 1988. *Morphological Methods in Image and Signal Processing*. Prentice-Hall, Inc.

[6] Heriansyah, R., S. A. R. Al-Attas, and M. M. A. Zabidi. 2002. "Segmentation of PCB Images into Simple Generic Patterns using Mathematical Morphology and Windowing Technique". Proceedings of National Conference on Computer Graphic & Multimedia (CoGRAMM 2002). 233-238.