PIN DEFECT INSPECTION WITH X-RAY IMAGES

[1] *Hsien-Pei Kao (高咸培), Tzu-Chia Tung (董子嘉), Hong-Yi Chen (陳弘毅)*,
[2]*Cheng-Shih Wong (翁丞世)* [1,2] *Chiou-Shann Fuh (傅楸善)*

[1] Dept. of Computer Science and Information Engineering,
[2] Graduate Institute of Biomedical Electronics and Bioinformatics,
National Taiwan University, Taiwan
E-mail: jack805201@gmail.com, fuh@csie.ntu.edu.tw

Abstract. A method of Printed Circuit Board (PCB) pin defect inspection is proposed in this paper. First, we input the pin location image. Then, we align images with Circle Hough Transform (CHT) [4]. Finally, we train cascade classifier with adaptive boosting [3] and Local Binary Pattern (LBP) [5], and detect pin defect to reduce false alarm rate and miss detection rate and thus enhance pin production yield rate.

Keywords: X-ray inspection, pin, Circle Hough Transform, adaptive boosting, cascade training

1 INTRODUCTION

In industry, factory automation is getting more important; including manufacturing, transporting, packaging, and defect inspection. Defective products lead to bad brand reputation and product recall and repair cost, so defect inspection plays an important role in gaining profits. Our goal is to decrease false alarm and miss detection rates. However, miss detection may cause defective final product and higher repair cost, and thus has higher weight than false alarm rate. Nowadays, Haar-like feature and Local Binary Pattern (LBP) are two main features used at cascade classifier training; we will compare their differences in next section.

We aim to implement the pin defect inspection with X-ray image. Generally, many factors affect this issue, including X-ray intensity, the distance between X-ray tube and board, the angle between X-ray tube and camera, the intensity of light source, and so on. These factors affect the quality of X-ray images.

In this paper, we will focus on defect inspection by using adaptive boosting with Local Binary Pattern (LBP) feature. Our approach consists of two parts, analysis and detection.

Since pins are physically on Printed Circuit Board (PCB), in different images, pins may not locate at the same location and have different values in *x*, *y* axes. Different light source intensities also generate different pin image intensities. This may cause displacement in X-ray image and also affect the image quality. Thus alignment is the first step to carry out, helping to align pin positions.

Pin X-ray images have many types, including different sizes and directions, and also background pixels. Thus, training has to consider different pin locations, hole sizes, and image intensities to improve performance. Therefore, we adopt Circle Hough Transform (CHT) to clip background pixels and resize to retain only hole and pin region to align our input images.

In defect inspection, we use adaptive boosting to collect a group of inferior classifiers. Each classifier's hit rate should be above 50 percent, so it is better than random guessing. The higher the hit rate, the more likely it will be chosen.

At last, we aim to improve final defect inspection performance. There are many methods to achieve lower false alarm rate, but we aim to lower miss detection rate due to final defective product and high repair cost. In this problem, we use intersection of two cascades (each cascade with 20 stages) to decrease miss detection rate, because it takes too much time to group two times of inferior classifiers into one cascade (with 40 stages which takes roughly 30 times more computation than 20 stages).

Some important techniques used in our approach will be introduced in Section 2. The detailed procedure of our approach will be described in Section 3.Our experimental results will be demonstrated in Section 4. Section 5 concludes this paper, and the final section is the list of references.

2 BACKGROUND

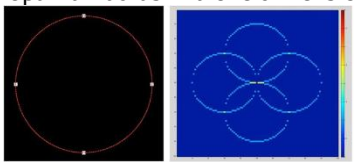
2.1 Circle Hough Transform

The Circle Hough Transform (CHT) [4] is a prime algorithm used in Digital Image Processing to find circular objects in a digital image, and also a feature extraction technique to detect circles. CHT is a special form of Hough Transform. CHT aims to find circles in imperfect image inputs. The circle candidates are generated by "voting" in the Hough parameter space and then select the local maxima in an accumulator matrix.

In a two-dimensional space, a circle can be described by:

$$(x-a)^2+(y-b)^2=r^2$$

where *a* and *b* is the center point of the circle, and radius is represented by *r*. The parameter space would be three-dimensional that consists of *a*, *b*, and *r*. Moreover, all the parameters that satisfy *(x, y)* would lie on the surface of an inverted right-angled cone whose apex is at *(x, y, 0)*. In the 3D space, the circle parameters can be identified by the intersection of many tapered appearances which are defined by points on the 2D circle. This process can separate into 2 main stages. One stage is to find the optimal center of circles in a 2D parameter space by fixing radius. Another stage is to find the optimal radius in a one dimensional parameter space.



(a) (b)

Fig. 2-1. Four points on a circle in (a) the original image [4]. The Circle Hough Transform's result is shown in the right side (b). The intersection of all the circles shows the center of the circle in (a).

An accumulator matrix is introduced to detect the cross point in the parameter space in practice. First, we need to partition the parameter space into "buckets" using a grid and generate an accumulator matrix according to the grid. The element in the accumulator matrix denotes the number of "circles" in the parameter space passing through the corresponding grid cell in the parameter space. The number is also called "voting number". At first, all elements in the matrix default to zero. For each "edge" point in the original space, we can formulate a circle in the parameter space and increase the voting number of the grid cell where the circle passes through. This process is called "voting". After voting, we can find local maxima in the accumulator matrix. The positions of the local maxima correspond to the circle centers in the original space.

2.2 Adaptive Boosting [3]

Adaptive Boosting is a machine learning meta-algorithm. Many other types of learning algorithms can use it to improve and enhance their capability. According to these learning algorithms, we can combine them into a weighted sum, which stands for the final result of boosted classifier.

Adaptive Boost is sensitive to outliers and noisy data. In some problems, it can be less susceptible to the overfitting problem than other learning algorithms. The single learners can be inferior, but as long as correct rate of each learner is briefly better than random guessing (e.g., correct rate of random guessing is 0.5, and correct rate of single learner is above 0.5 for binary classification), we can prove a superior learner be converged by previous inferior learners.

Adaptive Boost implies a specific method of boosted classifier training. A boost classifier is a classifier in the form

$$F_r(x) = \sum_{t=1}^{T} f_t\,(x)$$

where each f_t is an inferior learner that takes an object *x* as input and returns a binary value showing the category of the input object. Similarly, the Tth classifier will be correct if the sample has faith in being correct class and wrong otherwise.

For each case in the training set, each inferior learner generates an output, hypothesis $h(x_i)$. An inferior learner is chosen and given a coefficient α_t such that the sum training error E_t of the resulting *t*-stage boost classifier is minimized at each iteration *t*.

$$E_t = \sum_i E[F_{t-1}(x_i) + \alpha_t h(x_i)]$$

where F_{t-1} is the boosted classifier that has been composed of the previous classifier of training, $E(F)$ is a function that count error rate and $f_t(x) = \alpha_t h(x)$ is the inferior learner that is being believed for addition to final output. A weight ω_t is given to each sample in the training set according to the current error $E\big(F_{t-1}(x_i)\big)$ on that

sample at each iteration of the training process. We can use these weights to know the training property of inferior learners, for instance, inferior learner with high weights will split more branches than low weights in decision trees.

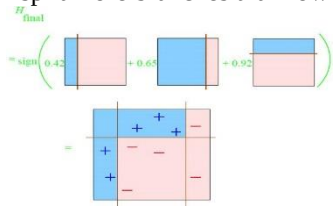


Fig. 2-2. The schematic diagram of adaptive boosting with three inferior learner [1].

2.3 Visual Descriptor [6]

Visual descriptors or image descriptors, in computer vision, are descriptions of the visual features of the contents in videos, applications, images or algorithms that generate these descriptions. They depict elementary characteristics such as texture, shape, motion or color, and so on.

There are two main groups in Visual Descriptor: General information descriptors: they contain low-level descriptors which give a description about the texture, the shape, the motion or the color, and so on. Specific domain information descriptors: they give information about objects and events in the scene.

2.3.1 Local Binary Pattern (LBP) [5]

Local Binary Patterns (LBP) is a type of visual descriptors used for classification in computer vision. LBP is the particular case of the Texture Spectrum model proposed in 1990.

In its purest way, LBP feature vector is produced by the following pattern: Partition the examined window into cells (e.g. 9 × 9 pixels for each cell). For each pixel in a cell, compare the center pixel with each of its 8 neighbors. Along the direction of the edge of the circle (i.e. clockwise or counter-clockwise). If the center pixel's value is less than the neighbor's value, assign "1". Else, assign "0".

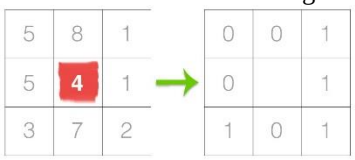


Fig. 2-5. Follow the pixels along a circle, and give value after comparing each pixel with center pixel [1].

This produces an 8-bit binary number (it is convenient to convert to decimal value). Calculate the histogram of the occurrence of each "number" occurring. This histogram can be seen as a 256-dimensional feature vector, which shows some feature points in the examined window.

Fig. 2-6. The result of Lena after running our own Local Binary Pattern [5].

3 METHODOLOGY

The flow of our proposed algorithm is as follows:

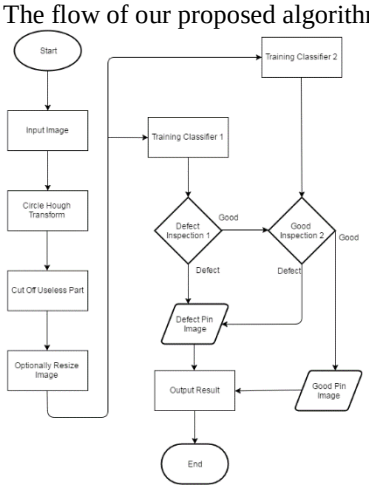
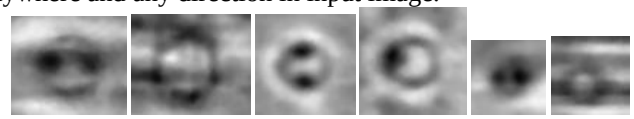


Fig. 3-1. The flow of our proposed algorithm.

In our approach, our input images have several types. The target we detect may be anywhere and any direction in input image.

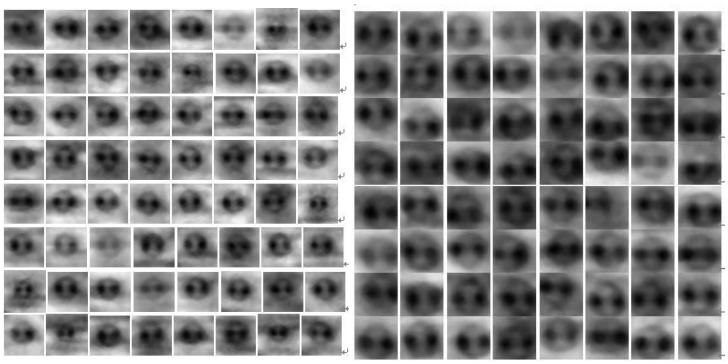


(a) (b) (c) (d) (e) (f)

Fig. 3-2. Different types of input images. (a) 82 × 70 pixels good image. (b) 82 × 69 pixels defect image. (c) 66 × 66 pixels good image. (d) 67 × 67 pixels defect image. (e) 52 × 52 pixels good image. (f) 52 × 52 pixels defect image.

We have to clip some useless region in order to enhance our performance. How do we know where is important and where is ignorable in our original image. Since the hole of the pin is circular, we use Circle Hough Transform and set a particular range to find this special circular hole.

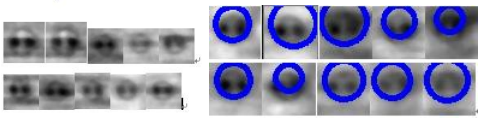
Our goal is to rotate image to make direction uniform. Now we use Microsoft Office Picture Manager to manually rotate 66 × 66 pixels good and defect images counterclockwise 90 degrees to make both pins horizontal.



(a) (b)

Fig. 3-3. The result of pin images (a) before and (b) after Circle Hough Transform and clipping.

In this step, we face serious problems. Due to weak intensity of X-ray images, Circle Hough Transform cannot find pins and hole in some images. We have about 200 images failing to find pins and hole. There are two problems: no pins and hole found and incorrect hole position.



(a) (b)

Fig. 3-4. Circle Hough Transform result. Blue circle shows the Circle Hough Transform output of detected hole and good pin center and radius. Circle Hough Transform cannot detect hole and good pin for defect image thus no center and radius output. (a) Part of 130 images no pins and hole found: 10 images. (b) Part of 60 images with incorrect hole position: 10 images.

After Circle Hough Transform, there may be several different sizes of image. We can optionally resize images.

Then, it is time to train classifier. At each stage, we choose some adjacent rectangles from pool, and select the highest correct rate from good and defect images.

$$F_r(x) = \sum_{t=1}^{T} f_t(x)$$

Because next stage's learner is related to previous learner, and we expect the increasing of correct rate when our stage gets larger. Under this condition, an inequality is applied as follows

$$F_r(x) \geq F_{r-1}(x)$$

to make sure our correct rate iterates higher.

Experimental Environment
1. OS: Windows 10 Enterprise
2. CPU: 3.1GHz Intel Core i5
3. Memory: 8 GB
4. **Platform: Visual Studio 2012 with OpenCV 3.0.0**

Step Name	Time Cost
Circle Hough Transform	0.1 second / 1200 images
Cascade Classifier 1 Training Time	52 minutes
Cascade Classifier 2 Training Time	37 minutes
Intersection of 2 Cascade Classifiers	0.001 second
Defect Inspection	0.2 second / 240 images

Fig. 3-5. Time table of each step.

In Section 1, we train 2 cascade classifiers (each with 20 stages) and take intersection to decrease miss detection rate.

	False Alarm Rate	Miss Detection Rate
Classifier 1	0%	2.5%
Classifier 2	0%	4.1%

Fig. 3-6. Two cascade classifier miss detection rates and false alarm rates.

Now, we take intersection of two cascade classifiers to enhance our performance.

False Alarm Rate	Miss Detection Rate
0%	0.8%

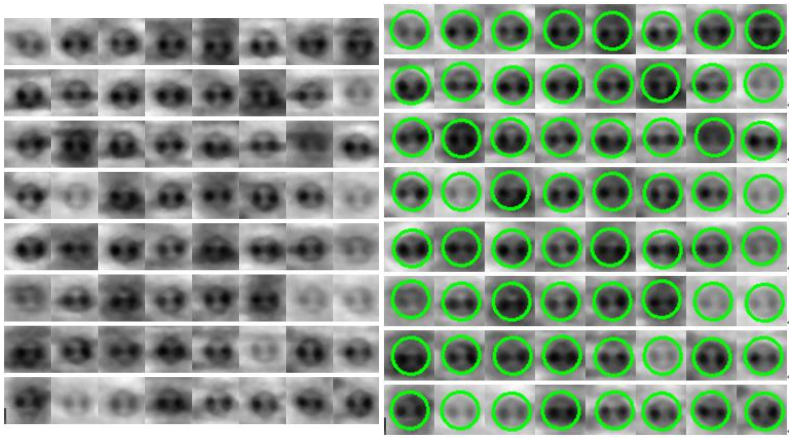
Fig. 3-7. Miss detection rate and false alarm rate of intersection of two cascade classifiers (originally each with 20 stages).

At last, we show images mis-classified as good, but it is physically defect. Thus manufacturer knows what kind of images will be identified as a good image even they are defective.

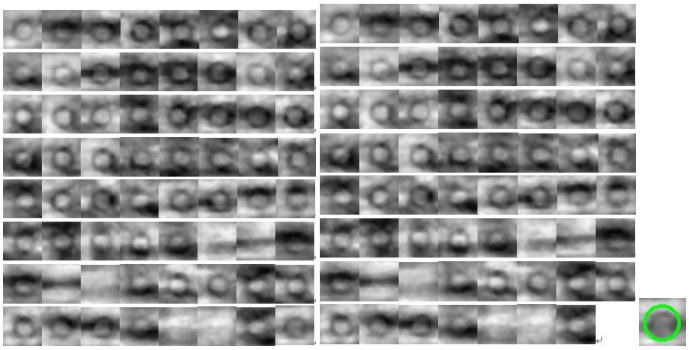
4. EXPERIMENTAL RESULT

We have 1430 good images and 1430 defect images. Their base solutions are between 44 × 44 pixels and 60 × 60pixels at 8-bit/pixel images.

We divide into two parts, training data and test data (training data have 1190 images and test data have 120 images for both good images and defect images, respectively).



(a) (b)



(c) (d) (e)

Fig. 4-1. Our final intersection of two cascade classifiers output. Green circle shows the classifier output of detected hole and good pin center and radius. Cascade classifier cannot detect hole and good pin for defect image thus no center and radius output. (a) Part of 120 good input images: 64 images. (b) Part of 120 correctly classified as good images: 64 images. 0 good image mis-classified as defect. (c) Part of 120 defect input images: 64 images. (d) Part of 119 correctly classified as defect images: 63 images. (e) 1 defect images mis-classified as good images (0.8%=1/120).

5. CONCLUSION

Defective products lead to bad brand reputation and product recall and repair cost. Miss detection rate is a critical element for the successful automation of manufacture. The pin defect inspection is a challenging problem due to the complex background. In our work, we propose a method for this purpose based on Circle Hough Transform to clip and resize to enhance images, and cascade classifier training and cascade classifier intersection to detect pin defect. Our experiment result shows that the proposed method can reduce miss detection rate to 0.8% for pin defect inspection.

ACKNOWLEDGEMENT

This research was supported by the Ministry of Science and Technology of Taiwan, R.O.C., under grants MOST 103-2221-e-002-188 and 104-2221-e-002-133-my2, and by Test Research (TRI), Liteon, Egistec, Delta Electronics, and Lumens Digital Optics.

REFERENCES

[1] A. Rosebrock, “Local Binary Patterns with Python & OpenCV”, http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/, 2016.

[2] Read01.com, “Machine Learning”, https://read01.com/Dngkd.html, 2016.

[3] Wikipedia, “AdaBoost”, https://en.wikipedia.org/wiki/AdaBoost, 2016.

[4] Wikipedia, “Circle Hough Transform”, https://en.wikipedia.org/wiki/Circle_Hough_Transform, 2016.

[5] Wikipedia, “Local Binary Pattern”, https://en.wikipedia.org/wiki/Local_binary_patterns, 2016.

[6] Wikipedia, “Visual Descriptor”, https://en.wikipedia.org/wiki/Visual_descriptor, 2016.