

# COLOR-BASED PRINTED CIRCUIT BOARD SOLDER SEGMENTATION

*Tz-Sheng Peng (彭志昇), Chiou-Shann Fuh (傅楸善)*

Dept. of Computer Science and Information Engineering, National Taiwan University  
E-mail: r96922118@csie.ntu.edu.tw

## ABSTRACT

Our method can be briefly divided into three phases: training phases, thresholding phase, and refining phase. We only have to train once. We do not have to train again until we have different color PCB (Printed Circuit Board) images. Thresholding and refining phases are used to segment every image.

The goal of training phase is to calculate the centroid of solder color. In thresholding phase, apply an automatic threshold finding algorithm to find the threshold recursively. After thresholding, we want to obtain better result. Consequently, we use connected component analysis to remove the component with too many or too few elements, because solder area is not too big or too small. Calculate the minimum color distance  $d$  from the centroid of solder color to connected component.

Our method is effective for solder image segmentation. Experiment data show us this method has high-speed and high-precision.

## 1. INTRODUCTION

As technology is much more advanced nowadays, electronic devices are ubiquitous in our daily life. PCB (Printed Circuit Board) plays an important role in almost every modern electronic device. In Fig. 1, we can see a PCB image. However, there still is not a perfect PCB manufacturing process. The PCB manufacturing company cannot afford the bad quality caused by PCB defects [3]. How to inspect PCB defects fast and precisely becomes a new challenge. Because of complexity of PCB, the high-speed and high-precision PCB inspection and testing is unfeasible and impossible by human vision. The inspection and testing automation is important stage for the PCB manufacturing process.

Computer vision is a developed technique in computer science, and its reliability and regularity are essential in PCB inspection and testing automation. Several parts of PCB inspection and testing automation can be solved by computer vision. AOI (Automatic Optical Inspection) is a more specialized word for the field of PCB inspection by computer vision.

Consequently, AOI becomes a key component in computer vision, and is widely used to detect the PCB defect and segment the PCB solder area.

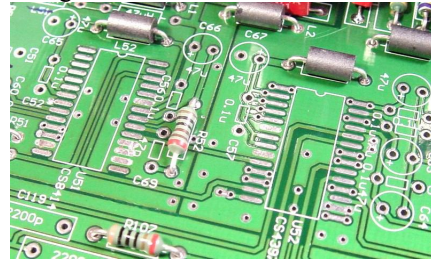


Fig. 1: Printed Circuit Board

AOI is automated visual inspection of large amount of devices, such as PCBs, LCD (Liquid Chrystal Display), and so on. We always use a camera to capture image from device, and then we can inspect the defect in the device by captured image. Fig. 2 is image used by AOI. Low costs and flexible property make AOI a necessary part of electronic device manufacture process.

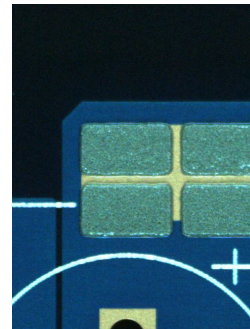


Fig. 2: Color Image of PCB

AXI (Automatic X-ray Inspection) is a technology based on the same principle as AOI. Rather than visible light used in AOI, AXI uses X-ray as light source to inspect the feature hidden in the ICs (Integrated Circuits) [9]. Some defect in ICs cannot be found by AOI, because those defects are invisible. AXI takes advantage of X-ray to penetrate object, and take several images from different directions. Then we can calculate the difference between retrieved images, we can produce the height image. Fig. 3 is height image produced by AXI.

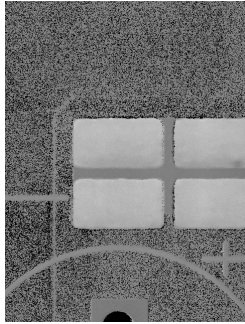


Fig. 3: Height Image of PCB

## 2. IMAGE SEGMENTATION [13]

Image segmentation is the partition of an image into a set of non-overlapping regions whose union is the entire image. The purpose of image segmentation is to decompose the image into parts that are meaningful with respect to a particular application.

The goal of image segmentation is to simplify and/or change the representation of an image into something that is typically used to locate objects and boundaries (lines, curves, etc) in images.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristics.

Some of practical applications of image segmentation are:

- ◆ Medical imaging
- ◆ Face recognition
- ◆ Fingerprint recognition
- ◆ Traffic control systems
- ◆ Brake light detection
- ◆ Machine vision

There are various kinds of segmentation methods and techniques based on different domain knowledge. The clustering methods are based on distributing pixels into cluster iteratively. The histogram-based methods are based on histogram analysis.

### 2.1 Clustering Method

Clustering is the assignment of pixels into clusters. The pixels in the same cluster have the similarity such as color information, spatial information, and so on. Clustering is a well-known method for statistical data analysis, and can be used in many fields.

We will introduce a popular clustering method,  $K$ -means algorithm. The  $K$ -means algorithm is an

iterative technique that is used to partition an image into  $K$  clusters. The basic algorithm is:

1. Pick  $K$  cluster centers, either randomly or based on some heuristic.
2. Assign each pixel in the image to the cluster that minimizes the variance between the pixel and cluster center.
3. Re-compute the cluster centers by averaging pixels in the cluster.
4. Repeat Steps 2 and 3 until convergence is attained.

The disadvantage of this method is how to choose an appropriate  $K$  and how to determine the initial seed value. There is some improvement to this algorithm such as randomly generating the initial seed value and setting  $K$  to eight because eight is big enough to cover all possible color clusters in the entire image.

### 2.2 Histogram-Based Method

Histogram-based methods are more efficient than other methods, because we usually need one pass through entire image. In histogram-based methods, a histogram is computed from every pixel in the image, and the peaks and valleys are used to segment the cluster in the image. A refinement of this technique is to recursively apply the histogram-based method to cluster in order to divide them into smaller clusters.

Histogram-based algorithms have low computational complexity, because we only need to compute the histogram once and locate the peaks and valleys. One disadvantage of the histogram-based method is that it may be difficult to identify the peaks and the valleys in the image, because the peaks and the valleys may not be obvious.

## 3. PREVIOUS WORK

In order to prevent solder from bridging between conductors [16], solder mask provides a permanent protective coating for the copper. Areas that should not be soldered to may be covered with a polymer solder mask coating. The solder resist prevents solder from bridging between conductors and thereby creating short circuits. Solder resist also provides some protection from the environment. Consequently, we have the ROI (Region Of Interest) to segment the solder area.

Because of the small image, we do not have to use a complex algorithm. We can only segment by calculating the histogram. We choose the color space by analyzing the histogram. After choosing the channel, we observe the histogram to find the upper and lower bound for segmentation. But this kind of method is only suitable for adequate small image.

The disadvantage is copper area and solder area cannot be discriminated. This is the disadvantage

we want to improve most. The hole in the solder area also makes result discontinuity.

#### 4. OUR METHOD

Our method can be briefly divided into three phases: training phases, thresholding phase, and refining phase. We only have to do training phase once. We do not have to train again until we have different color PCB images. Thresholding and refining phase are use to segment every image.

##### 4.1 Color Space

A color space is an abstract mathematic representation for different colors by tuples of numbers [11], typically as three or four channels, such as RGB (Red, Green, Blue) or CMYK (Cyan, Magenta, Yellow, black). In image processing, various kinds of color space can be calculated from the tristimuli  $R$ ,  $G$ , and  $B$ . How to choose an effective color space for image segmentation is important for the segmentation result.

Let  $S$  be the image to be segmented; and  $\Sigma$  be the covariance matrix of the distributions of  $R$ ,  $G$ , and  $B$  in  $S$ . Let  $\lambda_1$ ,  $\lambda_2$ , and  $\lambda_3$  be the eigenvalues of  $\Sigma$  and  $\lambda_1 \geq \lambda_2 \geq \lambda_3$ . Let  $W_i = (w_{iR}, w_{iG}, w_{iB})$  be the eigenvector of  $\Sigma$  corresponding to  $\lambda_i$ . The color feature  $X_i$  can be defined as follows:

$$X_i = w_{Ri} \cdot R + w_{Bi} \cdot B + w_{Gi} \cdot G$$

By analyzing large amount of image by above method, Ohta [7] found that a set of color feature,  $(R + G + B)/3$ ,  $R - B$ , and  $(2G - R - B)/2$ , derived from  $R$ ,  $G$ , and  $B$  are effective. These three features are significant in this order and in many cases a good segmentation can be achieved by using only the first two.

##### 4.2 Training

First, we apply adaptive median filter to reduce the noise in the image. The concept of adaptive median filter [4] is similar to median filter; the difference between these two filters is the changeable window size. Median filter use a constant window size, and find the median of all elements in the window. Adaptive median filter uses a variable window size, and window size of this filter is calculated recursively until the median does not equal the maximum or minimum value in the window. Table 1 is the pseudo-code of adaptive median filter:

Table 1: pseudocode of adaptive median filter

for each pixel on the image
set window center to current pixel
set window size $w$ to 3
L1: find the maximum value $max$ in the window
find the minimum value $min$ in the window
find the median $med$ in the window
if $med$ equals to $max$ or $min$
increase the window size $w$
go to L1

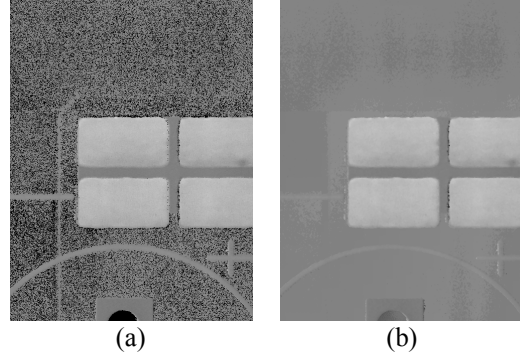


Fig. 4: (a) Original Image. (b) After applying adaptive median filter.

We can see the image processed by adaptive median filter in Fig. 4 and easily segment the image manually or by other threshold finding methods. We can see the result after segmentation in Fig. 5.

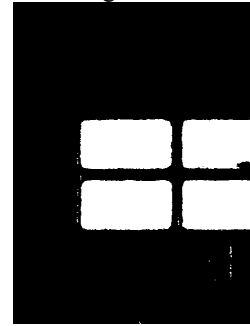


Fig. 5: Result Image.

We remove non-solder area manually, and then calculate the average color in the region of interest. We use this average color as a representative color for solder area for later phase. In this case, training result is  $(R, G, B) = (80, 140, 130)$ .

##### 4.3 Thresholding

In Section 1.3, we know that a good set of color feature,  $(R + G + B)/3$ ,  $R - B$ , and  $(2G - R - B)/2$ , derived from  $R$ ,  $G$ , and  $B$  are effective for color segmentation. As a result, we use the most important color feature  $(R + G + B)/3$ , which is brightness, to segment the solder area.

We want to segment the image roughly, so we use an automatic threshold selection method, Otsu method [8]. The basic concept is to minimize the within-group variance.

$$\sigma_{within}^2(t) = n_{below}(t)\sigma_{below}^2(t) + n_{over}(t)\sigma_{over}^2(t)$$

where  $n_{below}(t)$  is the ratio of pixels below  $t$ ;  $n_{over}(t)$  is the ratio of pixels over  $t$ ;  $\sigma_{below}^2(t)$  is the variance of pixels below  $t$ ;  $\sigma_{over}^2(t)$  is the variance of pixels over  $t$ ; and  $\sigma_{within}^2(t)$  is the within-group variance in  $t$ .

We can simplify the computation by maximizing the between-group variance, because

$$\sigma_{between}^2(t) = \sigma^2 - \sigma_{within}^2(t)$$

$$\sigma_{between}^2(t) = n_{below}(t)[\mu_{below}(t) - \mu]^2 + n_{over}(t)[\mu_{over}(t) - \mu]^2$$

$$\sigma_{between}^2(t) = n_{below}(t)n_{over}(t)[\mu_{below}(t) - \mu_{over}(t)]$$

where  $\sigma^2$  is the variance of entire image;  $\mu$  is the average of entire image;  $\mu_{below}(t)$  is the average intensity of pixels below  $t$ ; and  $\mu_{over}(t)$  is the average intensity of pixels over  $t$ .

Based on above equation developed, the basic Otsu method can be done by pseudocode in Table 2.

Table 2: pseudocode of Otsu method.

for each possible threshold  $t$   
 calculate the between-group variance  
 find the threshold  $t_{max}$  with maximum between-group variance

But if we cannot use only one threshold to segment result, how do we improve Otsu method to have better result? Ohta [7] proposed a recursive segmentation method with the flowchart shown in Fig. 6. The method can be briefly described as follows:

1. A mask corresponding to the whole image is placed at the bottom of the stack.
2. One mask is taken from the top of the stack. Let  $S$  be the region represented by the mask
3. Histograms of color features in the region  $S$  are computed.
4. Apply Otsu method to the region  $S$ .
5. Connected regions are extracted. For each connected region, a region mask is generated and pushed down on the stack.

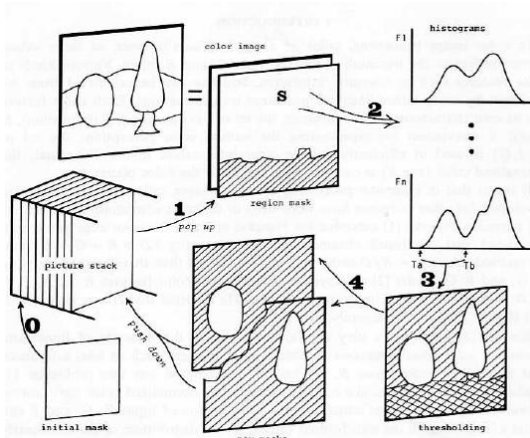


Fig. 6: Flowchart of recursive thresholding [7].

For every different color PCB image, we need to manually adjust the way to apply Otsu method. After finding the tight upper and lower bound, we can segment the solder area roughly.

#### 4.4 Refining

After previous step, we obtain the coarse result. There are many false alarms in the result. We need to refine the result, to reduce the false alarm. We apply

the connected component analysis, and the take advantage of the property of each connect component. We will show how we remove the false alarm by different aspects.

Calculate the average color  $c_{avg}$  for each connected component, and then compute the color distance  $colorDis(c_{avg}, c)$ , where  $c$  can be obtained from training data. We define the color distance as follows:

$$colorDis(a, b) = abs(a_R - b_R) + abs(a_G - b_G) + abs(a_B - b_B)$$

where  $a, b$  are pixels with  $R, G, B$  channels.

Check if the color distance from current connected component to training data is smaller than a threshold. If color distance is too large, we remove the connected component. Because the color of this connected component is different from training data.

We also calculate the pixel count and the bounding box for each connected component. Apply width, height, ratios of connected component pixels count to bounding box area, occlusion rate and ratio of bounding box width to bounding box height, and occlusion rate of connected component  $A$  can be defined as:

$$occ(A) = \frac{\#\{A\}}{w \times h}$$

where  $\#\{A\}$  is the pixel count of connected component  $A$ ; and  $w$  and  $h$  are width and height of bounding box of  $A$ .

Use the appropriate parameter to filter the result. Manually adjust the parameter to obtain the lower false alarm and misdetection. We will explain a segmentation example step by step in Fig. 7.

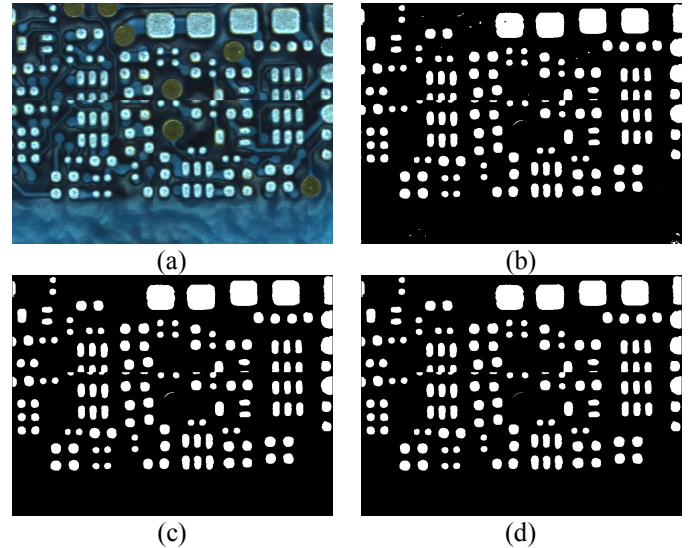


Fig. 7: A segmentation example. (a) Original image. (b) Image after recursive thresholding. (c) After removing the holes and weak connected component. (d) Remove false alarm by calculating color distance.

## 5. EXPERIMENT RESULT

### 5.1 Experiment Environment

- Operating System: Windows Vista SP1
- Central Processing Unit: Intel T5850@2.16GHz
- Memory: 3GB
- Integrated Development Environment: Visual Studio C++ 2005
- Image Size: 4800x2700 pixels
- Execution Time: less than 1 sec

Use our rough result and manually verify solder area. After previous verification, we can obtain our ground truth. We label the detected solder area as white pixel, the non-solder area as black pixel, false alarm area as red pixel, and misdetection area as blue pixel.

### 5.2 Experiment Result

We will first show the comparison of boundary by listing our result in Fig. 8 to Fig. 10, and then list the false alarm and misdetection rates in Table 3.

In Fig. 11 to Fig. 13, we segment the image by the following criterion: Training result is  $(R, G, B) = (141, 197, 206)$ . Connected component must be larger than 150 pixels. Color distance must be smaller than 50.

We apply false-alarm and misdetection rate to evaluate whether the segmentation method is better than others. We define false-alarm and misdetection as  $P_{fa}$  and  $P_{md}$ .

$$P_{fa} = \frac{\#\{\forall p \in R, p \notin S\}}{\#\{R\}}$$

$$P_{md} = \frac{\#\{\forall p \in S, p \notin R\}}{\#\{S\}}$$

where R is the set of segmentation result; S is the true solder area. If S is a set,  $\#\{S\}$  is the number of elements of set S.

In this case, we can say false-alarm rate is the ratio of our segmentation result but not in the true solder area to the area of our segmentation result. Misdetection rate is the ratio of true solder area but not in our segmentation result to the area of true solder area. Figure 1.4 represents false alarm and misdetection.

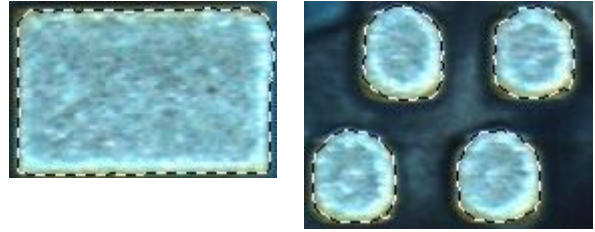


Fig. 8: Boundary of our result for blue image.

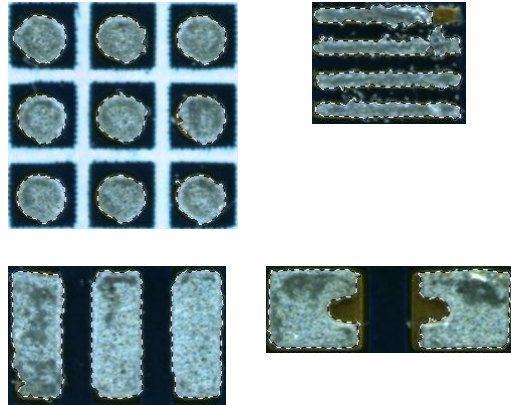


Fig. 9: Boundary of our result for deep blue image.

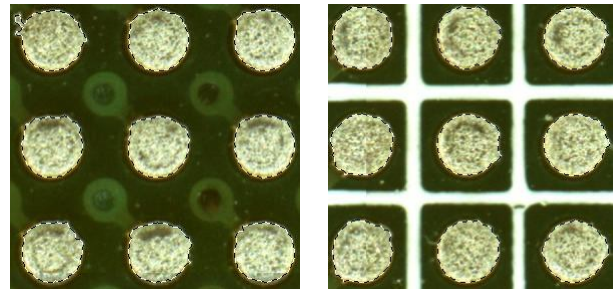
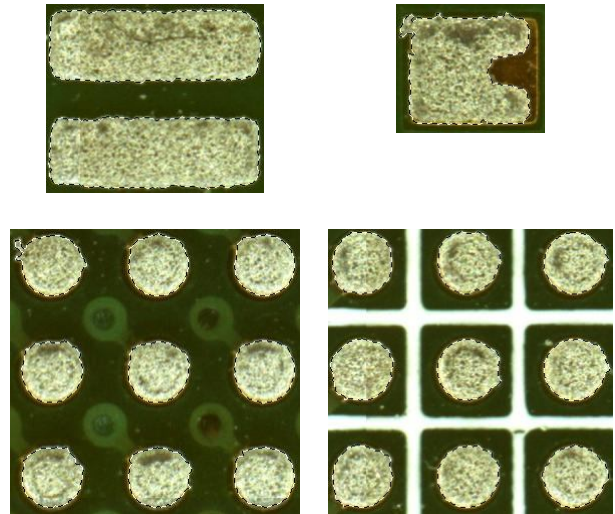
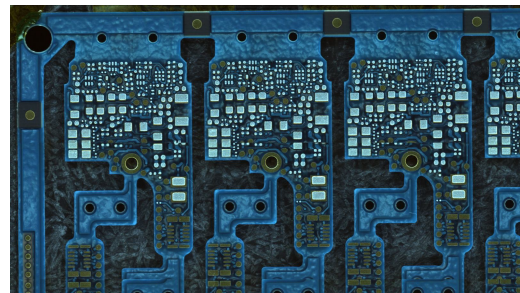
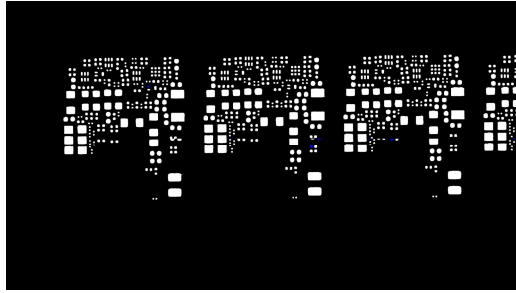


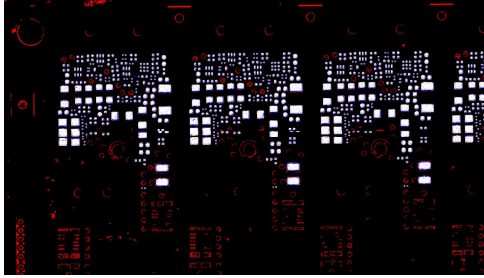
Fig. 10: Boundary of our result for green image.



(a)



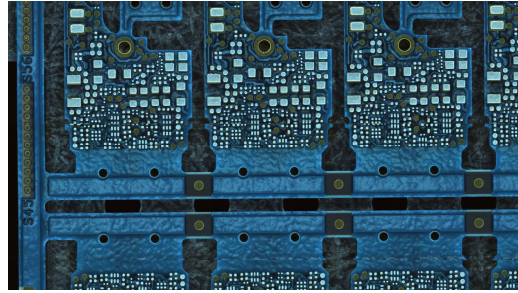
(b)



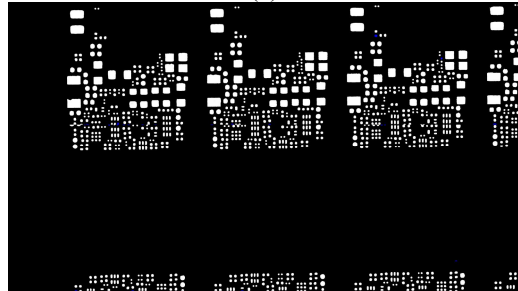
(c)

Fig. 11: (a) Original image. (b) Our result (false alarm 0% with red pixels and misdetection 0.2% with blue pixels). (c) Traditional method (false alarm 24.4% with red pixels and misdetection 20% with blue pixels).

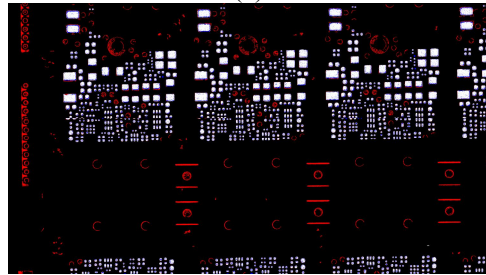
pixels). (c) Traditional method (false alarm 24.4% with red pixels and misdetection 20% with blue pixels).



(a)

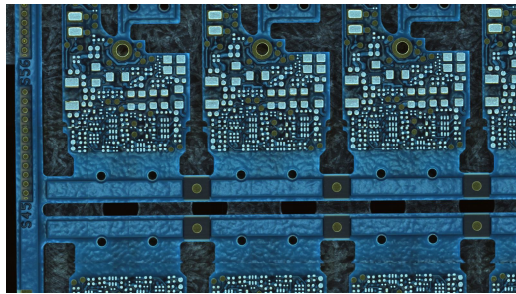


(b)

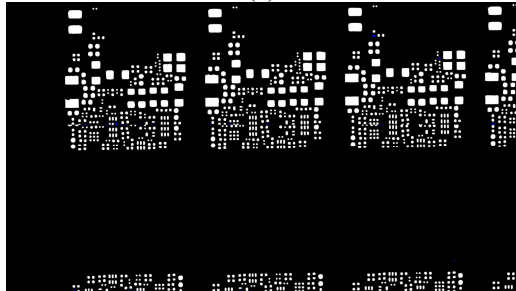


(c)

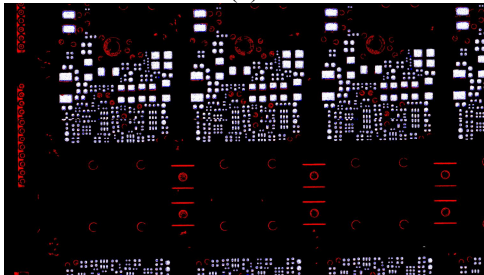
Fig. 13: (a) Original image. (b) Our result (false alarm 0% with red pixels and misdetection 0.3% with blue pixels). (c) Traditional method (false alarm 20.7% with red pixels and misdetection 23.3% with blue pixels).



(a)



(b)



(c)

Fig. 12: (a) Original image. (b) Our result (false alarm 0% with red pixels and misdetection 0.2% with blue pixels).

Table 3: False alarm and misdetection rate for above test image.

Our Method		Traditional Method	
False Alarm	Misdetection	False Alarm	Misdetection
0%	0.2%	24.4%	20%
0%	0.3%	20.7%	23.3%
0%	1%	18.7%	27.9%
6.2%	0%	40.3%	3%
4.4%	0%	31.7%	2%
4.3%	0%	35.3%	3%
0.1%	0%	10%	7%
0%	0%	10%	5%
3%	0%	33%	9%
0.3%	0.6%	36%	10%
0.9%	0%	36%	8%
0.03%	1.1%	24%	8%
5.6%	0%	14%	10%
0.1%	0%	23%	10%
0.4%	0.1%	13%	10%

## 6. CONCLUSION AND FUTURE WORK

We proposed an effective PCB solder area segmentation method. By this method, we can

discriminate solder area fast and precisely. Almost all false alarm and misdetection rates are less than five percent. Our segmentation is better than traditional segmentation.

But there is still some possible ways to improve this method. Because the image quality is closely related to segmentation result, how to calibrate the PCB image before segmentation is important. We can improve the thresholding phase to obtain better coarse result, and then we can improve the false alarm and misdetection rates. We can also use more complicated connected component analysis, because the solder and non-solder pixels may connect together after thresholding. If we can split them into different groups, we can reduce the false alarm rate.

#### REFERENCE

- [1] R. de Beer, "Histogram-Based Methods," <http://dutnsj2.tn.tudelft.nl:8080/main/node52.html>, 2009.
- [2] J. Garcia-Consuegra, G. Cisnero, and E. Navarro, "A Sequential ECHO Algorithm Based on the Integration of Clustering and Region Growing Techniques," Proceedings of International Geoscience and Remote Sensing Symposium, Honolulu, Hawaii, Vol. 2, pp. 648-650, 2000.
- [3] J. George, "Circuit Construction Techniques," [http://wiredworld.tripod.com/tronics/pcb\\_techniques.html](http://wiredworld.tripod.com/tronics/pcb_techniques.html), 2001.
- [4] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 2nd edition, Prentice Hall, Upper Saddle River, NJ, 2002.
- [5] R. M. Haralick and L. G. Shapiro, Computer and Robot Vision, Vol. I, Addison Wesley, Reading, MA, 1992.
- [6] O. J. Morris, M. de J. Lee, and A. G. Constantinides, "Graph Theory for Image Analysis: an Approach Based on the Shortest Spanning Tree," IEE Proceedings F: Communications, Radar and Signal Processing, Vol. 133, No. 2, pp. 146-152, 1986.
- [7] N. Ohta, T. Kanade, and T. Takai, "Color Information for Region Segmentation," Computer Graphics and Image Processing, Vol. 13, No. 3, pp. 222-241, 1980.
- [8] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, pp. 62-66, 1979.
- [9] TRI Innovation, "TR7600," [http://www.tri.com.tw/en/products\\_a\\_txt.aspx?id=P\\_0000019&cid=C\\_00000002&pname=TR7600&cname=AOI%2fAXI](http://www.tri.com.tw/en/products_a_txt.aspx?id=P_0000019&cid=C_00000002&pname=TR7600&cname=AOI%2fAXI), 2009.
- [10] Wikipedia, "Cluster Analysis," [http://en.wikipedia.org/wiki/Data\\_clustering](http://en.wikipedia.org/wiki/Data_clustering), 2009.
- [11] Wikipedia, "Color Space," [http://en.wikipedia.org/wiki/Color\\_space](http://en.wikipedia.org/wiki/Color_space), 2009.
- [12] Wikipedia, "Edge Detection," [http://en.wikipedia.org/wiki/Edge\\_detection](http://en.wikipedia.org/wiki/Edge_detection), 2009.
- [13] Wikipedia, "Image Segmentation," [http://en.wikipedia.org/wiki/Image\\_segmentation](http://en.wikipedia.org/wiki/Image_segmentation), 2009.
- [14] Wikipedia, "K-means Clustering," [http://en.wikipedia.org/wiki/K\\_means](http://en.wikipedia.org/wiki/K_means), 2009.
- [15] Wikipedia, "Minimum Spanning Tree," [http://en.wikipedia.org/wiki/Minimum\\_spanning\\_tree](http://en.wikipedia.org/wiki/Minimum_spanning_tree), 2009.
- [16] Wikipedia, "Solder Mask," [http://en.wikipedia.org/wiki/Solder\\_mask](http://en.wikipedia.org/wiki/Solder_mask), 2009.