Segmentation of PCB Images into Simple Generic Patterns using Mathematical Morphology and Windowing Technique.

Conference Paper · October 2002

3 authors:

Rudi Heriansyah
Umm Al-Qura University

Syed Ab Rahman Abu Bakar
Universiti Teknologi Malaysia

Muhammad Zabidi
Universiti Teknologi Malaysia

SEGMENTATION OF PCB IMAGE INTO SIMPLE GENERIC PATTERNS USING MATHEMATICAL MORPHOLOGY AND WINDOWING TECHNIQUE

Rudi Heriansyah[1], Syed Abdul Rahman Al-Attas[2], and Muhammad Mun'im Ahmad Zabidi[3]

Department of Microelectronics and Computer Engineering
Faculty of Electrical Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor
Malaysia
Phone: 60 7 55 35274
Fax: 60 7 55 66272

[1]rudi_hn@ieee.org, [2]syed_rahman@ieee.org, [3]raden@suria.fke.utm.my

Abstract: Segmentation of PCB image into simple generic patterns enables us to transform the original problem of inspecting a complex PCB image into a simpler problem of inspecting only well-defined generic patterns. In this paper, the mathematical morphology operators will be used to segment PCB image pattern into its simple generic patterns such as pads and traces. For classification tasks, sometimes it is essential to enclose the specific pattern into its individual window. This technique is so called windowing will be developed also in this paper.

Keywords: *PCB, visual inspection, segmentation, mathematical morphology, windowing*

1. 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 electronics products [1].

In PCB inspection systems, segmentation of PCB image into simple generic patterns enables us to transform the original problem of inspecting a complex PCB image (composed of thousands of generic patterns) into a simpler problem of inspecting only well-defined generic patterns.

Segmentation comes with two free added advantages: (1) it helps in partitioning the inspection tasks among multiple processors for faster on-line processing, and (2) it helps in associating certain defect types with certain basic patterns and hence makes the inspection more modular and easier, i. e. different patterns have different types of defects [2].

2. PCB image segmentation approaches

Segmentation can be done directly on PCB image pattern or based on PCB CAD data. The former approach technically easier than the latter approach.

Based on PCB image, we can directly develop algorithm to segment that PCB image. But as the common problem in a digital image, sometimes we need to do pre-processing to the image before the segmentation process applied.

Moganti and Ercal [3], has developed a segmentation algorithm based on PCB image using image-processing approach. PCB image is binary image. Firstly, the PCB image is divided into a mesh of seed windows. The segmentation algorithm processes these windows to find basic sub-pattern in their locality by adjusting the sides of the window. The segmentation algorithm continues to work on the remaining seed windows. This approach although they claimed working on pattern level, but actually it is time consuming because to judge a seed window valid or not, it is needed to test by shifting operation in many possible locations of the window. Beside that this approach is very PCB type dependence, because in order to decide whether the valid window contains a valid sub-pattern or not, it must supply many initial sub-pattern definitions.

F. Ercal et al. [2] and O. Silvén et al. [4] proposed PCB image segmentation based on CAD data. Segmentation is performed on the artwork. The algorithm is developed to segment the PCB image into primitive patterns; line segments (nets), junctions, and apertures (special shapes), and information related to the location and identification of each segment is stored in a large image database to be used later for real-time inspection. In certain case, this CAD approach seems to be better than image processing approach, since CAD directly represents the artwork in vector format and no further image processing is required, so it is expected to increase efficiency and decrease error [2]. But in the implementation level, this technique is not as simple as we think. Because to develop the segmentation algorithm based on this CAD data, at least we must have any technical knowledges about the CAD data structure itself, this is one thing. If the PCB's CAD data is not available from the CAD designer, so for the purpose of testing our algorithm sometimes we need to design ourselves the PCB in order to create the CAD data, and it means that we must have a skill in designing the PCB using any PCB designer software, and this is the other thing. But however, we still could use this approach for PCB image segmentation as an alternative beside using bitmap image approach.

In this paper, we will use image-processing approach because of its simplicity. Mathematical morphology will be used extensively to decompose PCB image pattern into its main parts such as pads and traces.

3. Morphological operations

From a general scientific perspective, the word morphology refers to the study of form and structure. The term is used in this sense in biology, geography, and linguistics. In image processing, morphology is the name of a specific methodology originated by G. Matheron in his study of porous material.

The morphological approach is generally based upon the analysis of a two-valued image in terms of some predetermined geometric shape known as a structuring element [5].

An algebraic operators system of operators, such as those of mathematical morphology, is useful for computer vision because compositions of its operators can be formed which, when acting on complex shapes, are able to decompose them into their meaningful parts and separate the meaningful parts from their extraneous parts.

Morphological operations can simplify image data preserving their essential shape characteristics and eliminate irrelevancies. As the identification and decomposition of objects, object features, object surface defects, and assembly defects correlate directly with shape, it is only natural that mathematical morphology has an essential structural role to play in machine vision [6, 7].

Morphological operation can be divided into two main forms, binary morphology and gray-scale morphology [5, 6, 7, 8]. In this paper, we will use binary morphology to segment PCB image pattern into simple generic patterns.

3.1 Dilation

Dilation is one of the two basic operators in the area of mathematical morphology, the other being erosion. It is typically applied to binary images, but there are versions that work on grayscale images. The basic effect of the operator on a binary image is to gradually enlarge the boundaries of regions of foreground pixels (i.e. white pixels, typically).

The dilation operator takes two pieces of data as inputs. The first is the image, which is to be dilated. The second is a (usually small) set of coordinate pints known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the dilation on the input image.

The dilation of A by B is denoted by $A \oplus B$ and is defined by:

$$A \oplus B = \{c \in E^N \mid c = a + b\} \tag{1}$$

for some $a \in A$ and $b \in B$

The first set A of the dilation is associated with the image underlying morphologic processing and the second set B is referred to as the structuring element, that shape which acts on A through the dilation operation to produce the result $A \oplus B$. We will refer to A as a set or as an image.

Dilation by structuring elements correspond to isotropic or expansion algorithms common to binary image processing. Dilation by small squares (3 x 3) is a neighborhood operation easily implemented by adjacency connected array architectures and is the one many image processing people know by the name "fill", "expand", or "grow".

The dilation operation can be represented as a union of translates of the structuring element:

$$A \oplus B = \bigcup_{a \in A} B_a \ . \tag{2}$$

This union of translates of the structuring element can be thought of like a neighborhood operator. The structuring element B is swept over the image. Each time the origin of the structuring element B touches a binary 1-pixel; the entire translated structuring element shape is OR-ed to the output image.

3.2 Erosion

The same with the dilation, the erosion is also typically applied to binary images, but there are versions that work on grayscale images. The basic effect of the operator on a binary image is to erode away the boundaries of regions of foreground pixels (i.e. white pixels, typically). Thus areas of foreground pixels shrink in size, and holes within those areas become larger.

The erosion of A by B is denoted and is defined by:

$$A \Theta B = \{x \in E^N \mid x = x + b \in A\} \tag{3}$$

for every $b \in B$

The erosion of an image A by a structuring element B is the set of all elements x of E^N for which B translated to x is contained in A.

Whereas dilation can be represented as a union of translates, erosion can be represented as an intersection of the negative translates:

$$A \Theta B = \bigcap_{b \in B} A_{-b} \tag{4}$$

This means that the same architecture which accomplishes dilation can accomplish erosion by changing the OR function to a AND function and using the image translated by the negated points of the structuring element instead of using the image translated by the points of the structuring element.

3.3 Opening and closing

Dilations and erosions are usually employed in pairs, either dilation of an image followed by the erosion of the dilated result, or erosion of the image followed by the dilation of the eroded result. In either case, the result of successively applied dilations and erosions is an elimination of specific image detail smaller than the structuring element without the global geometric distortion of unsuppressed features. For example, opening an image with a disk-structuring element smoothes the contour, breaks narrow isthmuses, and eliminates small islands and sharp peaks or capes. Closing an image with a disk structuring element smoothes the contours, fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps on the contours.

The opening of image B by structuring element K is denoted by $B \circ K$ and is defined by $B \circ K = (B \Theta K) \oplus K$. The closing of image B by structuring element K is denoted by $B \bullet K$ and is defined by $B \bullet K = (B \oplus K) \Theta K$. If B is unchanged by opening it with K, we say that B is open with respect to K or B is *open*ed under K, while if B us unchanged by closing it with K, then B is *clos*ed with respect to K or B is closed under K.

4. Morphology based PCB image segmentation

In this section, it will be showed the implementation of the algorithm to segment the PCB image pattern into simple generic patterns [9]. The segmentation algorithm will be employed using mathematical morphology operators as have been described in the previous section. The binary morphology will be used for segmentation process.

Fig. 1 shows a sample 64 x 64 PCB image pattern that will be decomposed into simple generic patterns.

The segmented patterns will be grouped into four main parts or generic patterns, i.e. hole pad, square pad (box), rectangular pad, and trace (line), and the segmentation process will be done based on these four types of pattern.

Flowchart for this PCB segmentation is depicted in Fig. 2 and Fig. 3 shows the binary image of Fig. 1, and segmented PCB image is shown in Fig. 4.

From Fig. 4, numeral 1 is the label for square pads, 2 is the label for hole pads, 3 is the label for rectangular pads, 4 is the label for thick lines, and 5 is the label for thin lines. The detail explanation on this labeling technique will be given in section 5.1.

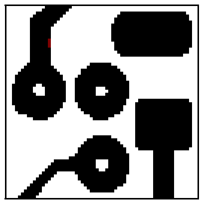

Fig. 1. A sample PCB image

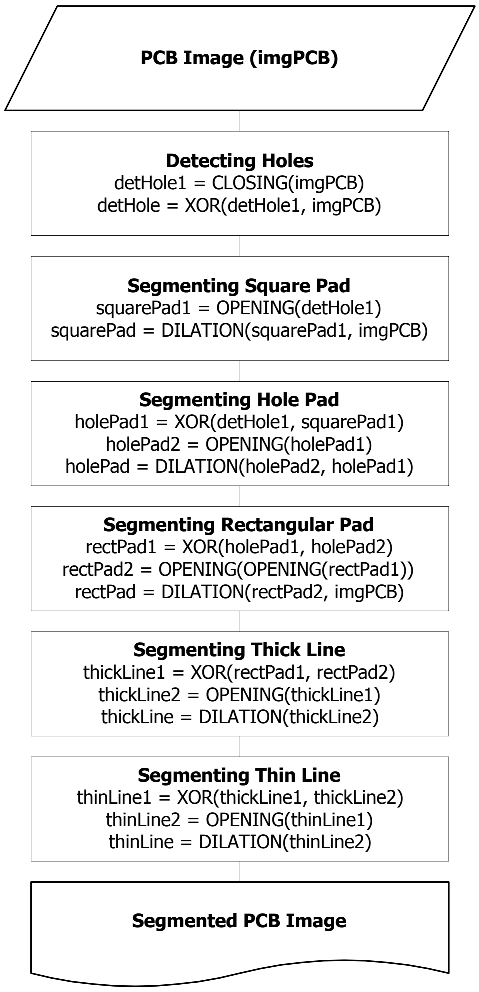

Fig. 2. Flowchart for PCB segmentation

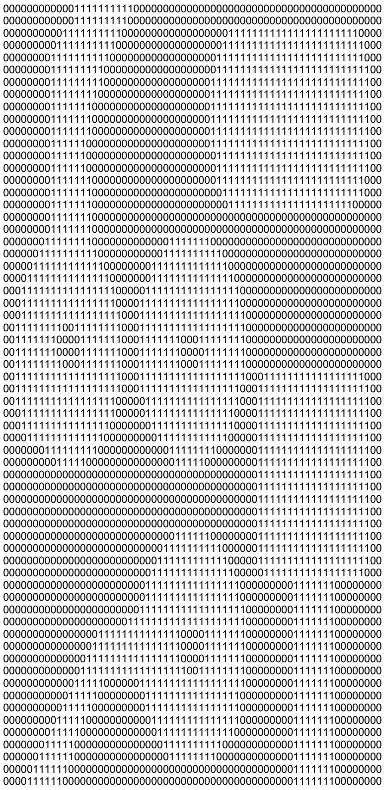

Fig. 3. Binary PCB image

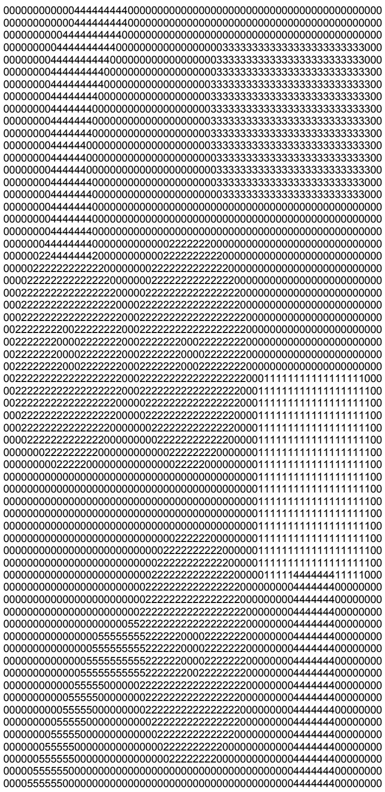

Fig. 4. Segmented PCB image

5. PCB generic patterns windowing

For further process, sometimes we need to enclose the segmented PCB image or pattern windowing, for instance as an input for the classifier in the pattern classification tasks. Based on the previous result i.e. segmented image, we have developed a simple algorithm for image windowing on basis of binary image.

5.1 Generic patterns labeling

The first step in image windowing is image labeling or in more technical sense, so called connected component labeling.

Connected component labeling scans an image and groups its pixels into components based on pixel connectivity (4- or 8-connectivity). Once all groups have been determined, each pixel is labeled with a certain character or numeral according to the component it was assigned to.

This section will describe a binary connected component labeling.

Connected component labeling works by scanning an image, pixel-by-pixel (from top to bottom and left to right) in order to identify connected pixel regions, i.e. regions of adjacent pixels which share the same set of value `V = {1}`.

The connected components labeling operator scans the image by moving along a row until it comes to a point `p` (where `p` denotes the pixel to be labeled at any stage in the scanning process) for which `V = {1}`. When this is true, it examines the four neighbors of `p` which have already been encountered in the scan (i.e. the neighbors (i) to the left of `p`, (ii) above it, and (iii and iv) the two upper diagonal terms). The pseudo-code labeling of `p` is shown in Fig. 5 [10].

```
//** LABELING **//

if (all four neighbors are 0)
   assign a new label to p;
elseif (only one neighbor has V={1})
   assign its label to p;
elseif (one or more of the neighbors
       have V={1}) {
           assign one of the labels to p
           and make a note of
           the equivalences;
   }
```

Fig. 5 Labeling pseudo-code

Fig. 6 shows an example of image labeling using 4-connectivity.

For the implementation of image labeling on the segmented PCB image, firstly the label is made on the individual pattern on the basis of previous algorithm for each type of segmented pattern. After that based on the individual labeled image, labeling for overall image is generated.

Pseudo-code for PCB image labeling is given in Fig. 7. A sample of labeled PCB image is already shown in Fig. 4.

1	1	1	0	0	0	0	0
1	1	1	0	1	1	0	0
1	1	1	0	1	1	0	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	0	1	0
1	1	1	0	0	1	1	0
1	1	1	0	0	0	0	0

(a)

1	1	1	0	0	0	0	0
1	1	1	0	2	2	0	0
1	1	1	0	2	2	0	0
1	1	1	0	0	0	3	0
1	1	1	0	0	0	3	0
1	1	1	0	0	0	3	0
1	1	1	0	0	3	3	0
1	1	1	0	0	0	0	0

(b)

Fig. 6 Image labeling. (a) Original. (b) Labeled image

```
//** Segmented PCB image labeling **//

//-- Refer to Fig. 2 for used notations

     Label assignator:    1 – square pad
                          2 – hole pad
                          3 – rectangular pad
                          4 – thick line
                          5 – thin line
--//

// Labeling for each type of pattern
square_lbl = LABELING(squarePad);
hole_lbl = LABELING(holePad);
rect_lbl = LABELING(rectPad);
thick_lbl = LABELING(thickLine);
thin_lbl = LABELING(thinLine);

// Overall labeling
for (each row of image) {
    for (each column of image) {
        if square_lbl != 0 then all_lbl = 1;
        if hole_lbl != 0 then all_lbl = 2;
        if rect_lbl != 0 then all_lbl = 3;
        if thick_lbl != 0 then all_lbl = 4;
        if thin_lbl != 0 then all_lbl = 5;
        }
}

// Labeled PCB image
all_lbl;
```

Fig. 7 PCB image labeling pseudo-code

5.2 Generic patterns windowing

Based on the labeled image in the previous section, we could develop an algorithm for image windowing. This section will discuss the algorithm.

From Fig. 6, label for individual pattern is stored in variables `SQUARE_LBL`, `HOLE_LBL`, `RECT_LBL`, `THICK_LBL`, and `THIN_LBL`. These variables give us its position coordinates in the image. By searching the pixel-by-pixel from left to right and top to bottom, the most top, bottom, left, and right pixel will be found, then we can create a window to enclose the pattern based on these coordinates.

Pseudo-code for windowing is shown in Fig. 8. In Fig. 9 is shown the searching mechanism of the most top, bottom, left, and right pixel coordinates.

```
//** WINDOWING **//

// Input image
square_lbl;
hole_lbl;
rect_lbl;
thick_lbl;
thin_lbl;
all_lbl;

// Get the coordinates
for (each label type) {
    for (each row of image){
        for (each column of image){

            mostTop = mostBottom = 0;
            mostLeft = mostRight = 0;

            if (curentPos > mostTop ||
                   mostBottom || mostLeft ||
                   mostRight){

                (mostTop || mostBottom ||
                   mostLeft || mostRight)
                   = currentPos;}
        }
    }
}

// Generate window
for (mostTop to mostBottom of  each lbl){
    for (mostLeft to mostRight of  each lbl){
        window = current coordinate
                   of all_lbl;
    }
}
```

Fig. 8 Windowing pseudo-code

In Fig. 10 it is shown a windowed pattern based on algorithm in Fig. 8. This windowed image is used for other processing, usually for input to the classifier in a pattern classification tasks, for example as an input to Neural Network in the application of PCB defects classification.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 9 Coordinates searching example

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 10 Windowed pattern

6. Summary

Mathematical morphology approach is demonstrated in this paper to segment the PCB image into simple generic patterns.

Sample PCB image is successfully segmented into common type of patterns, i.e. hole pad, square and rectangular pad, and line. Morphology approach with its simplicity is appropriate for PCB image segmentation. It is also shown both algorithms for segmented image labeling and windowing. It is done as a pre-processing usually for classification tasks.

7. References

[1] W. Y. Wu, M. J. J. Wang, and C. M. Liu, “Automated Inspection of Printed Circuit Boards through Machine Vision.” Computers in Industry 28.103-111 (1996).

[2] F. Ercal, F. Bunyak, F. Hao, and L. Zheng, “A Fast Modular RLE-Based Inspection Scheme for PCBs.” Proc. SPIE Conf. On Architectures, Networks, and Intelligent Systems for Manufacturing Integration. 3203-06 (1997).

[3] M. Moganti and F. Ercal. “Segmentation of Printed Circuit Board Images into Basic Patterns.” Computer Vision and Image Understanding. 70. No. 1 (1998).

[4] O. Silvén, I. Virtanen, T. Westman, T. Piiroonen, and M. Pietikäinen, “A Design Data-Based Visual Inspection System for Printed Wiring.” in Advances in Machine Vision, Jorge L. C. Semz, ed. (Springer-Verlag New York Inc, 1989).

[5] C. R. Giardina and E. R. Dougherty, Morphological Methods in Image and Signal Processing (New Jersey: Prentice-Hall, Inc, 1988).

[6] R. M. Haralick, “Mathematical Morphology and Computer Vision,” (Institute of Electrical and Electronics Engineers, New York, 1988), pp. 468-479.

[7] H. Joo and R. M. Haralick, “Understanding the Application of Mathematical Morphology to Machine Vision,” (Institute of Electrical and Electronics Engineers), pp. 977-982.

[8] J. Serra, Image Analysis and Mathematical Morphology, Vol. 1 (Academic Press, Inc., 1990).

[9] SDC Information Systems, “SDC Morphology Toolbox for MATLAB.” (http://www.mmorph.com/)

[10] R. Fisher et al., “Image Processing Learning Resources.” (http://www.dai.ed.ac.uk/HIPR2/)

Acknowledgment

RH is grateful to Universiti Teknologi Malaysia for UTM Fellowship Award.