

# HsuKit: 3D Reconstructed Solder Ball Shape Improvement and Defect Inspection with X-Ray Images

<sup>1</sup>Ming-Chen Hsu (許銘真), <sup>2</sup>Chih-Kwang Li (李志洸), <sup>2</sup>Yu-Hsiang Shao (邵育翔), <sup>3</sup>Kai-Ren You (游凱任), <sup>3</sup>Chiou-Shann Fuh (傅楸善)

<sup>1</sup>Graduate Institute of Networking and Multimedia,

<sup>2</sup>Graduate Institute of Biomedical Electronics and Bioinformatics,

<sup>3</sup>Department of Computer Science and Information Engineering,

National Taiwan University, Taipei, Taiwan,

[R10944009@ntu.edu.tw](mailto:R10944009@ntu.edu.tw) [yaolin45040@gmail.com](mailto:yaolin45040@gmail.com) [alex0967100@gmail.com](mailto:alex0967100@gmail.com)

[z7852143z@gmail.com](mailto:z7852143z@gmail.com) [fuh@csie.ntu.edu.tw](mailto:fuh@csie.ntu.edu.tw)

## ABSTRACT

The PCB (Printed Circuit Board) defect detection plays an important role in the electronics industry. However, there are many overlapping electronic components on the multi-layer PCB, so the 3D reconstructed PCB image that is captured by X-ray and reconstructed by FBP (Filtered Back Projection) will lose some information, resulting in a rhombus shape of the solder ball and insufficient contrast, which will affect the defect inspection. The simplest way to solve this problem is to photograph X-ray images from a wider angle, but the photography angle of the equipment is limited. To solve the above problems, we propose to use the deep learning architecture HsuKit to generate larger angle images to improve the contrast of the images and the shape of the solder balls in the images.

**Keywords:** PCB defect inspection, Solder ball, Image contrast, X-ray images, Deep learning.

## 1. INTRODUCTION

The PCB (Printed Circuit Board) is one of the basic parts of electronic products. It can put the complex circuit between different parts on the same board after being designed. After the products are manufactured, all parts must be confirmed by the manufacturer to be correct. Therefore, PCB inspection is a very essential part of the electronic industry to find out the defects, such as voids, short circuits, open circuits, burr, and so on [1]. In the current PCB inspection, the common methods are Automated Optical Inspection (AOI), Solder Paste Inspection (SPI), and Automated X-ray Inspection (AXI). AOI uses optical instruments to obtain the surface state of the products and then uses image processing technology to detect defects [2]. SPI also uses optical images to check the products' quality, but the difference between AOI is that the solder paste inspection machine

adds a solder paste thickness measuring laser device [3]. AXI uses product density differences to analyze the defects [4]. Despite there being some precise ways to inspect the defects, PCB defect inspection is still a challenge due to the multilayer structure of the PCB where many electronic components overlap.

The PCB on electronic products may be designed as a multi-layer structure, and there are many electronic components on each layer. This situation will cause the components to overlap and lead to missing information when using X-rays to photograph PCB [5] in Figure 1(a). Therefore, if these 2D PCB images with overlapping and missing information are implemented in 3D reconstruction, the reconstructed 3D solder balls will have a shape like a rhombus in Figure 1(c). Moreover, the image contrast is not enough, resulting in inconspicuous voids and other defects. These shortcomings will lead to false positives in defect detection. The most intuitive and easiest way to solve this problem is to photograph the PCB with a bigger angle in Figure 1(b). Because when the photography angle is bigger, the less overlapping of electronic components. However, the photography angle of the equipment is limited. Taking our X-ray machine as an example, the maximum photography angle can only reach 43°. If there is a bigger photography angle, the missing information will be relatively reduced. Therefore, this is the main problem to solve: to generate bigger-angle images.

With the explosive development of deep learning technology, more and more researchers use this deep learning architecture to deal with the problems on the images. AutoEncoder (AE) is an unsupervised neural network consisting of an Encoder and Decoder, and these two parts are fully connected structures. It uses the Encoder to extract the representative features in the input, and the Decoder will use these features to reconstruct the input, and similarity to the original one will be better [6].

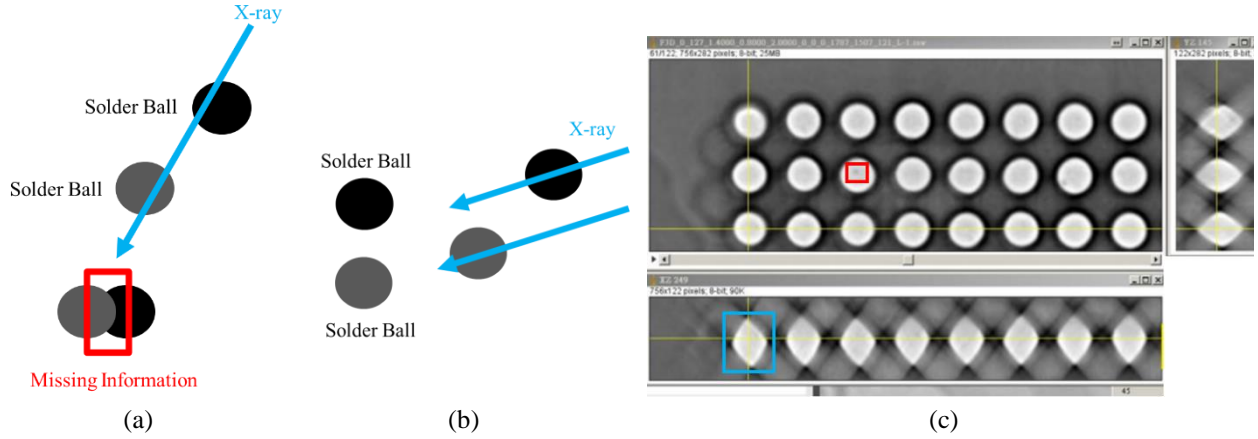


Fig. 1: (a) Diagram of overlapping and missing information caused by overlapping solder balls. (b) A bigger photograph angle diagram. (c) The reconstruction results of 3D solder balls with missing information. The red bounding box is void, and the blue bounding box is solder balls. The void is not obvious, and the shape of the ball is similar to the rhombus and incorrect and to be improved.

However, when the parameters of AE are complex to a certain extent, it is easy to have the risk of overfitting [7]. In order to solve the problem of overfitting, P. Vincent et al. proposed a deformation of AE, Denoising AutoEncoder (DAE), which enhances the stability of the model by randomly adding noise to the input layer [8].

Moreover, S. Rifai et al. also proposed the deep learning method of Contractive AutoEncoder (CAE). It uses the Jacobian matrix for weights of the model to perform punishment so that the Encoder can learn anti-interference features. Furthermore, to allow AutoEncoder to effectively extract better features in images, J. Masci et al. modify the original AutoEncoder architecture, changing the fully connected layer to a convolutional layer [10]. More and more researchers have made changes to the architecture of autoencoders as time goes on, and in 2013 a major improvement was achieved through Variational AutoEncoder (VAE) [11]. The model adds some constraints to the Encoder, forcing the generated vectors to follow the Gaussian distribution, so VAE theoretically allows us to control generated image.

When it comes to generating images, it is inevitable to mention the Generative Adversarial Network (GAN) [12]. This architecture is composed of a generator and discriminator. The Generator generates images that are similar to the ground truth images, and the discriminator distinguishes the image as fake or real. Therefore, scholars combined the idea of GAN with AE to create the architecture of Adversarial AutoEncoder (AAE) [13, 14], and also obtained good results. Adversarial AE uses the architecture of AE to generate images and uses the discriminator of GAN to distinguish whether the Latent Space extracted by the encoder of AE is similar to our expected distribution.

The aforementioned various modified AEs aim to improve solution results. Besides, these modified AEs have been applied in different fields [15-19], so we also want to use our modified AE algorithm to learn the difference in the angle in the images photographed at

different angles to generate a bigger-angle image, in order to solve the problem of photography angle limitation on the machine.

We aim to use HsuKit, a modified AutoEncoder, to learn the polar angle information in the image and to generate a bigger-angle X-ray PCB image. Thus, we can obtain a better solder ball image with less overlapping and missing information. In addition, there will be many circuits on the PCB board leading to complex backgrounds, increasing the difficulty of this task. The HsuKit architecture is in Figure 2.

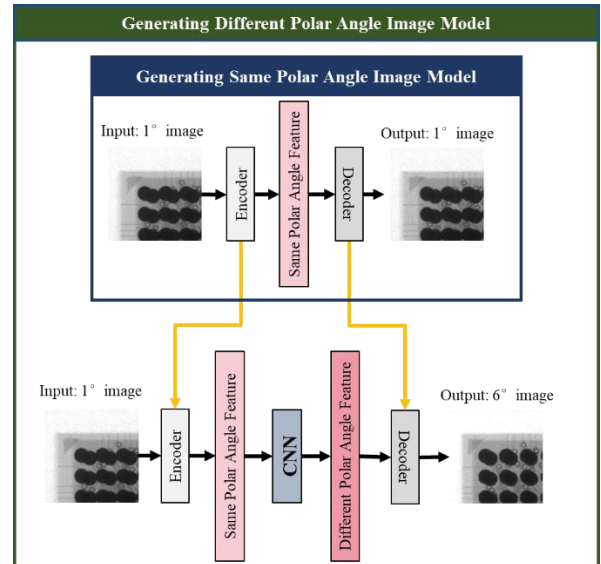


Fig. 2: Our method for generating bigger-angle images based on AE with two models: Generating Same-Polar-Angle Image Model and Generating Different-Polar-Angle Image Model. Besides the basic Encoder and Decoder of the AE, we add CNN to convert small-polar-angle extracted features into bigger-polar-angle extracted features.

Our HsuKit includes two models: The first model generates the same-polar-angle images, mainly training the ability of the Encoder to extract the features and the

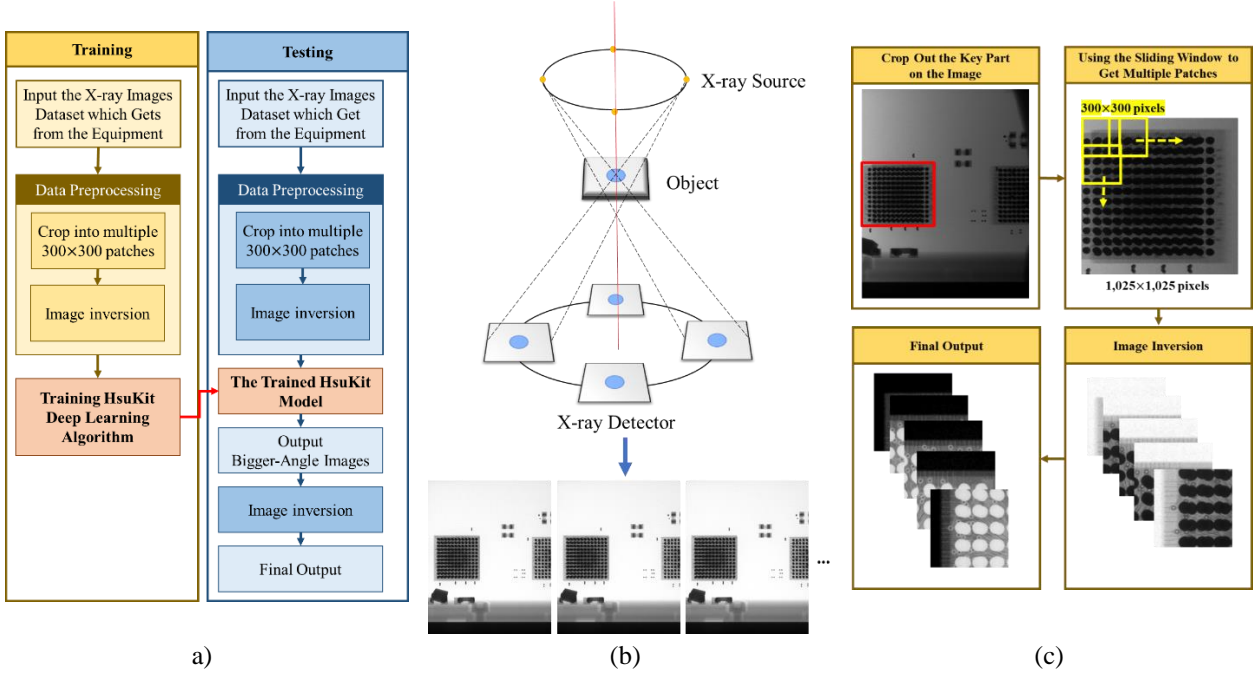


Fig. 3: (a) The flowchart of our proposed method. (b) X-ray image system setup. (c) Image preprocessing.

ability of the Decoder to reconstruct back to the image. The second model generates different-polar-angle images and mainly trains the ability of the Convolutional Neural Network (CNN) to transform the same-polar-angle features into different-polar-angle features, and the encoder and decoder are directly adopted from the previous model. Using our method and reconstructing a 3D image, we can obtain a relatively normal solder ball shape and a better-quality 3D PCB image helpful for defect inspection.

In Section 2, we will introduce our dataset and deep learning algorithm, HsuKit, in detail. Our experimental results will be described in Section 3. Finally, Section 4 draws the conclusion.

## 2. METHOD

Therefore, this section describes our dataset and preprocessing. Then, we will illustrate the architecture of the HsuKit deep learning algorithm in detail to generate a bigger-polar-angle image of PCB X-ray images in Figure 3 (a).

### 2.1. Dataset

The dataset used in this paper is to photograph PCB at different angles through the X-ray machine, to obtain multiple 2D X-ray images of different angles in Figure 3 (b). This study will take a set of images every  $5^\circ$  according to the center Z-axis, starting from  $1^\circ$  and reaching  $41^\circ$ . And the X-ray tube of each angle will surround  $360^\circ$  and take an image every  $11.25^\circ$ , so each group can get 32 images. Finally, we can obtain a total of 288 ( $=32 \times 9$ ) images. Each image will contain many

overlapping solder balls, electronic elements, and complex circuit backgrounds.

### 2.2. Image Preprocessing

The process of image preprocessing is shown in Figure 3 (c). We will select and capture part of the solder balls in the 288 2D X-ray images. After many experiments, the width and height of the solder ball images captured by this research are  $1,025 \times 1,025$  pixels to capture the solder ball part in this dataset and cut off the uninteresting parts, such as the background. Besides, to increase the dataset of this research, we will use the sliding window method to cut out multiple  $300 \times 300$  pixel patches from the extracted images, so each  $1,025 \times 1,025$  pixel 2D extracted image will cut out 512 patches of the size of  $300 \times 300$  pixels. Finally, we can get a total of 4,096 images.

Since the value of the solder balls in the image is 0: black part of the image. However, when using the deep learning architecture model for training, the main learning area with a value other than 0 will be relatively easy to learn. Moreover, although we have removed the uninteresting parts as much as possible by selecting key parts on the images, there are still a few parts that will be retained, but the pixel values of these parts are bright on the image, which may cause the model to learn the unimportant place, which indirectly affects the final output of the model. Therefore, the image will be processed by image inversion in this research, that is, the value of each pixel in the image will be inverted in black and white. In the end, we will use this inverting image to train the deep learning model.

### 2.3. HsuKit

Our method is divided into two models in Figure 2. In this subsection, we will introduce the architecture of the two models and the loss used by the optimizer.

#### 2.3.1 Generating Same-Polar-Angle Image Model

In the first model, the encoder is composed of 8 convolutional layers in Figure 4. The kernel size of each convolutional layer is  $3 \times 3$ , and the activation function adopts Leaky Rectified Linear Units (LReLU) [20], which can avoid the negative neurons cannot get updated problems while training. Except for the first convolutional block whose filter number is 32, the filter number will gradually enlarge by twice in each convolutional block, and our initial filter number is set to 128, so the filter numbers of the third and fourth blocks are 256 and 512.

The main design concept of the decoder is to reconstruct it to the original image shape. Therefore, we use 8 deconvolutional layers in the deconvolutional layer, and the activation function also uses LReLU. The filter number of those deconvolutional layers is gradually reduced, and the only thing to note is that the filter number of the last deconvolutional layer should be 1 because our image is a grayscale image with only 1 channel.

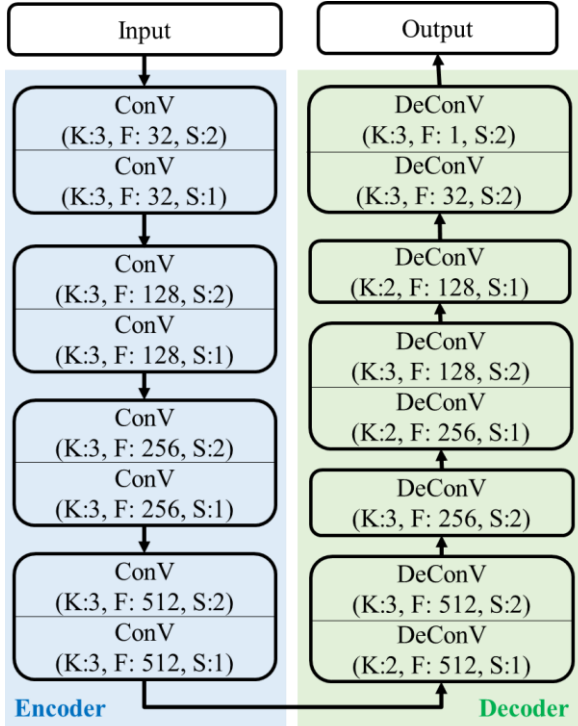


Fig. 4: The architecture to generate the same-polar-angle image model.  $K$  is the kernel size;  $F$  is the filter number;  $S$  is the strides.

To obtain a model for generating the same-polar-angle images, minimizing the differences between the input and the output images generated by the model is the main aim of this task. We use the mean square error [21] as the loss function:

$$\mathcal{L}_{GSPAIM}(I) = \sum_i^H \sum_j^W \|f_D(f_E(I)) - I\|_2, \quad (1)$$

where  $I$  is the input image;  $H$  and  $W$  are the width and height of the input image, respectively;  $f_D$  and  $f_E$  are the transformation functions of the trained model, respectively.

Our optimizer to optimize the loss function in Equation (1) is Adam (Adaptive Moment Estimation) [22]: one of the most used optimizers for training neural networks.

#### 2.3.2 Generating Different-Polar-Angle Image Model

In the second model, the main task is to generate the different-angle image model. The current goal is to let the model learn the difference between the input image and its image with an increased photography angle of  $5^\circ$ . For example, the input image is photography according to the center Z-axis angle of  $1^\circ$  in Figure 4, and the output image is the  $6^\circ$  image.

In addition to using the Encoder and Decoder of generating the same-angle image model, the most special part of this model is to add a CNN model to the middle part to convert the features in Figure 5. The CNN model is composed of 2 convolutional layers and 2 deconvolutional layers. The kernel size of each layer is  $3 \times 3$ , and the activation function is still LReLU.

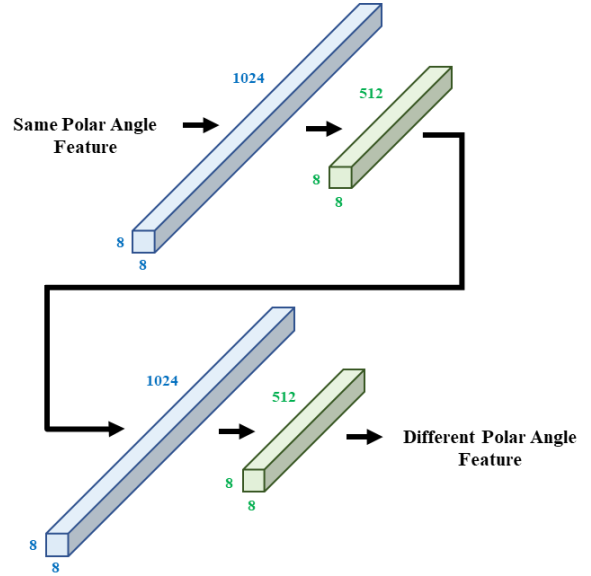


Fig. 5: The architecture to generate the different-angle image model.

In this model, we still use Adam as the optimizer to optimize the loss function. We apply two loss functions to make sure the converting features are as similar to the object features:

$$\mathcal{L}_{GDPAIM}(I, G) = \mathcal{L}_{Middle}(I, G) + \mathcal{L}_{Final}(I, G) \quad (2)$$

where  $\mathcal{L}_{Middle}$  is the loss function which can minimize the difference between the object features and the converting features:

$$\mathcal{L}_{Middle}(I, G) = \sum_i^H \sum_j^W \|f_c(f_E(I)) - f_E(G)\|_2, \quad (3)$$

where  $I$  is the input image (e.g. an image with a polar angle of 1);  $G$  is the ground truth image (e.g. an image with a polar angle of 6);  $f_c$  is the converting function of the trained CNN model, respectively.

Besides, we also want to ensure the final output images are as similar to the ground truth images as possible, to double-check if the converting features are able to reconstruct back to the bigger-polar-angle image:

$$\mathcal{L}_{Final}(I, G) = \sum_i^H \sum_j^W \|f_D(f_c(f_E(I))) - G\|_2, \quad (4)$$

### 3. RESULT

In this section, we will describe the environment for deep learning network. Then, we will show the output results of the HsuKit and comment on the results. Besides, we will also compare our model with other networks.

#### 3.1. Experimental Environment

Our Hsukit deep learning network environment are: 32GB of RAM, AMD Ryzen 7 5800X 8-Core of CPU, and NVIDIA GeForce GTX 3090 of GPU, and the operating system is Windows 10. Moreover, we use Tensorflow and Keras deep learning framework using Python to implement the architecture of Hsukit.

#### 3.2 Results and Comparison

In this experiment, we want to describe the structural similarity between our generated images and the ground truth images, and the difference between each pixel, so, we use three common evaluation metrics to measure the image quality, namely SSIM (Structural Similarity Index), PSNR (Peak Signal-to-Noise Ratio), and MAE (Mean Absolute Error) [23]. The higher the SSIM and PSNR values, the better the output image quality, and the lower the MAE value, the better. Therefore, we also compare our method with the original AE [6] to show that our modified AE model, HsuKit, promotes the output result efficacy. Moreover, we also compare the results with and without data preprocessing.

The evaluation metrics results of generating same-polar-angle image model results are presented in Table 1 and Figure 6. It is obvious that our Hsukit model is better than the original AE in every evaluation metric. Then, we also observe the results with and without data preprocessing. We find out that although PSNR and SSIM values are very close to each other, it can be seen that there is a clear gap in MAE values. This shows that the images generated by the model with preprocessing are more similar to the ground truth images, so we can say that the Hsukit model with preprocessing performs better. The same result is shown in figure 6. The output

results of the HsuKit model with preprocessing have a clearer solder ball shape, and the background of images is also clearer than other methods.

Table 1. Each method results in generating same-polar-angle images. (↑ the higher, the better; ↓ the lower, the better) (Bold: the best)

| Method                 | PSNR ↑          | SSIM ↑        | MAE ↓          |
|------------------------|-----------------|---------------|----------------|
| AE [6]                 | 34.1803         | 0.8623        | 102.5361       |
| HsuKit                 | 36.53186        | 0.9098        | 84.2259        |
| (No Preprocessing)     |                 |               |                |
| <b>HsuKit</b>          | <b>35.69703</b> | <b>0.9025</b> | <b>62.7230</b> |
| <b>(Preprocessing)</b> |                 |               |                |

Table 2 presents the results of generating different-polar-angle images, and the output images of each model are shown in Figure 7. Our method with preprocessing has the best results for those evaluation metrics values. The output images in Figure 7 also shows that the HsuKit method has the better result. The solder ball shape is clearer, the image contrast is higher, and the image has less noise. However, the evaluation metric value and the output images are worse than the result of the generating same-polar-angle image model, probably because the task of transforming the features is indeed more difficult and requires future improvement.

Table 2. Each method results in generating different-polar-angle images.

| Method                 | PSNR ↑         | SSIM ↑        | MAE ↓          |
|------------------------|----------------|---------------|----------------|
| AE [6]                 | 32.10941       | 0.8164        | 73.0072        |
| HsuKit                 | 34.1713        | 0.8550        | 103.2555       |
| (No Preprocessing)     |                |               |                |
| <b>HsuKit</b>          | <b>34.7486</b> | <b>0.8574</b> | <b>75.8217</b> |
| <b>(Preprocessing)</b> |                |               |                |

### 4. CONCLUSIONS

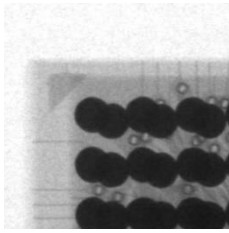
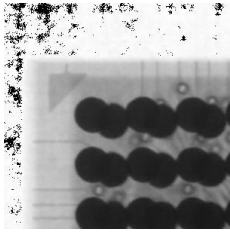
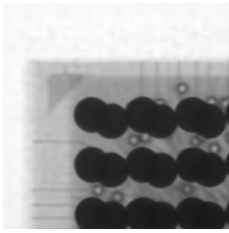
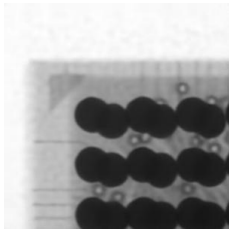
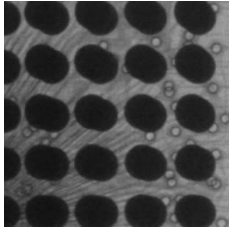
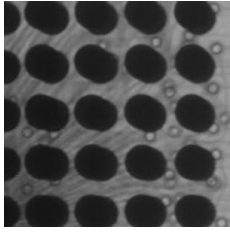
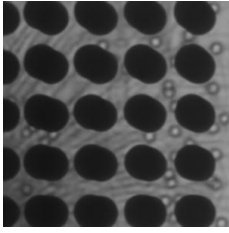
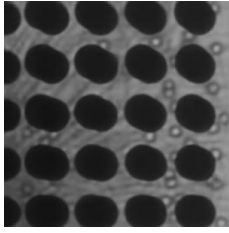
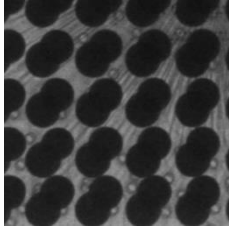
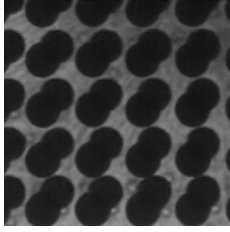
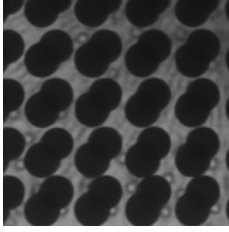
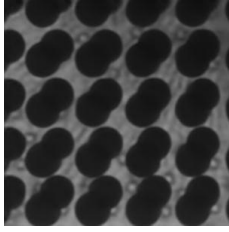
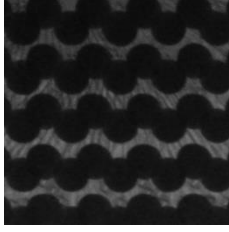
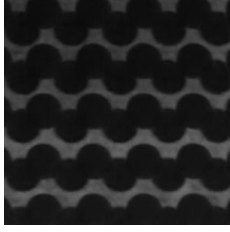
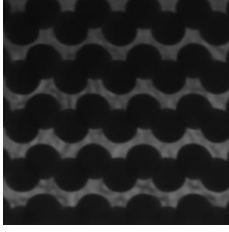
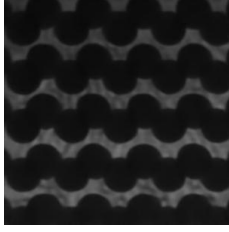
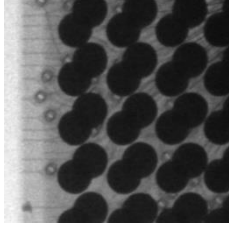
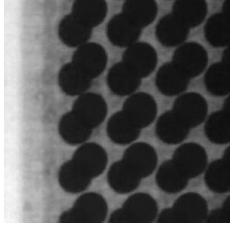
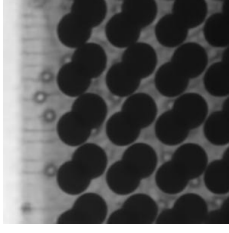
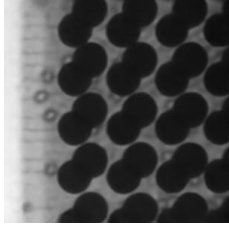
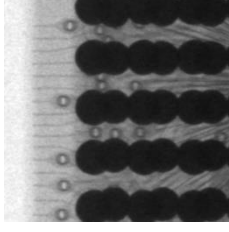
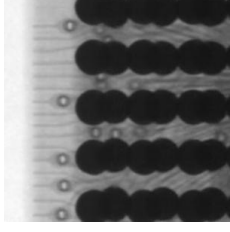
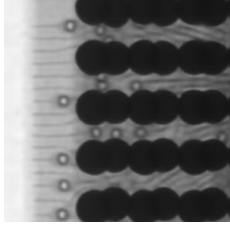
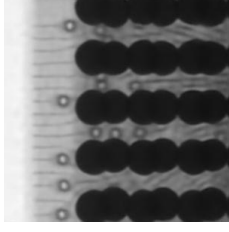
We develop HsuKit, generating bigger-polar-angle images model, to solve the problem of overlapping electronic components on the multi-layer PCB losing some information while reconstructing and the photography angle of the equipment is limited. The final evaluation metrics values are: MAE: 75.8217, PSNR: 34.7486, and SSIM: 0.8574. These results show that our model can already generate bigger-polar-angle images with better image quality. However, there is still some mistake in transferring the feature from same-polar-angle to different-polar-angle, and it still has some noise on the image requiring improvement. We will continue working in this direction, trying more models and finding out the best solution for this task.

### ACKNOWLEDGMENT

This research was supported by the Ministry of Science and Technology of Taiwan, R.O.C., under Grants MOST 109-2221-E-002-158-MY2 and MOST

108-2221-E-002-140, and by Test Research, Jorgin Technologies, III, Chernger, Jeilin Technology, Otus Imaging, D8AI, PSL, and LVI. Thanks to TRI (Test

Research, Inc.) [24] for providing the PCB 2D X-Ray images for us to implement this experiment.

| Input Image/<br>Ground Truth Image  | AE [6]  | HsuKit<br>(No Preprocessing)   | HsuKit<br>(Preprocessing)   |
|---|---|--|---|
|    |    |    |    |
|    |    |    |    |
|   |   |   |   |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |



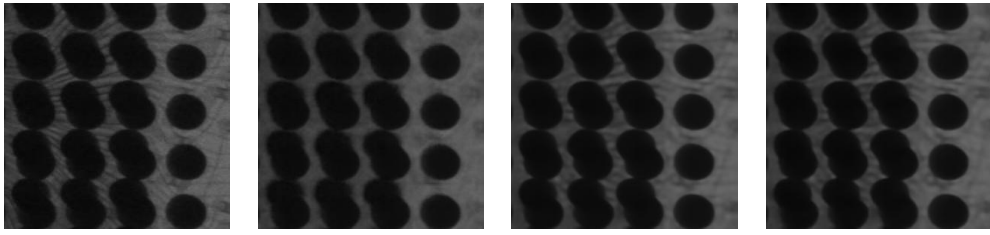


Fig. 6: The result of generating same-polar-angle images of each model.

| Input Image | AE [6] | HsuKit<br>(No Preprocessing) | HsuKit<br>(Preprocessing) | Ground Truth Image |
|-------------|--------|------------------------------|---------------------------|--------------------|
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |
|             |        |                              |                           |                    |

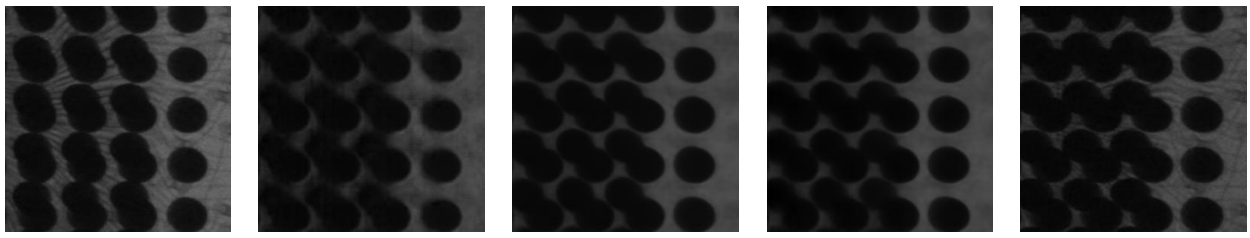


Fig. 7: The result of generating different-polar-angle images of each model.

## REFERENCES

- [1] S. Ren, L. Lu, L. Zhao, and H. Duan, "Circuit Board Defect Detection Based on Image Processing," *Proceedings of International Congress on Image and Signal Processing*, Shenyang, China, pp. 899-903, 2015.
- [2] E. M. Taha, E. Emary, and K. Moustafa, "Automatic Optical Inspection for PCB Manufacturing: A Survey," *International Journal of Scientific and Engineering Research*, vol. 5, no. 7, pp. 1095-1102, 2014.
- [3] T. H. Kim, T. H. Cho, Y. S. Moon, and S. H. Park, "An Automated Visual Inspection of Solder Joints Using 2D and 3D Features," *Proceedings IEEE Workshop on Applications of Computer Vision*, Sarasota, FL, pp. 110-115, 1996.
- [4] H. Boerner and H. Strecker, "Automated X-ray Inspection of Aluminum Casting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 1, pp. 79-91, 1988.
- [5] A. K. Schnurr, K. Chung, T. Russ, L. R. Schad, and F. G. Zöllner, "Simulation-Based Deep Artifact Correction with Convolutional Neural Networks for Limited Angle Artifacts," *Zeitschrift für Medizinische Physik*, vol. 29, no. 2, pp. 150-161, 2019.
- [6] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv: 2003.05991*, 2020.
- [7] H. Steck, "Autoencoders that Don't Overfit Towards the Identity," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1-11, 2020.
- [8] P. Vincent, H. Larochelle, Y. Bengio, and P. A. Manzagol, "Extracting and Composing Robust Features with Denoising Autoencoders," *Proceedings of International Conference on Machine Learning*, Helsinki, Finland, pp. 1096-1103, 2008.
- [9] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive Autoencoders: Explicit Invariance During Feature Extraction," *Proceedings of International Conference on Machine Learning*, Bellevue, Washington, pp. 833-840, 2011.
- [10] J. Masci, U. Meier, and J. Schmidhuber, "Stacked Convolutional Autoencoders for Hierarchical Feature Extraction," *Proceedings of International Conference on Artificial Neural Networks*, Espoo, Finland, vol. 1, pp. 52-59, 2011.
- [11] D. P. Kingma, and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv: 1312.6114*, 2013.
- [12] I. Goodfellow, J. Pouget-Abadie, and M. Mirza et al., "Generative Adversarial Nets," *Proceedings of Advances in Neural Information Processing Systems*, Montreal, Quebec, Canada, vol. 33, pp. 2672-2680, 2014.
- [13] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial Autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.
- [14] B. Ghogh, A. Ghodsi, F. Karray, and M. Crowley, "Generative Adversarial Networks and Adversarial Autoencoders: Tutorial and Survey," *arXiv:2111.13282*, 2021.
- [15] C. Y. Liou, W. C. Cheng, J. W. Liou, and D. R. Liou, "Autoencoder for Words," *Neurocomputing*, vol. 139, no. 2, pp. 84-96, 2014.
- [16] F. Farahnakian and J. Heikkonen, "A Deep Auto-Encoder Based Approach for Intrusion Detection System," *Proceedings of International Conference on Advanced Communication Technology*, Chuncheon, Korea, pp. 178-183, 2018.
- [17] X. Dai, J. Bai, T. Liu, and L. Xie, "Limited-View Cone-Beam CT Reconstruction Based on an Adversarial Autoencoder Network with Joint Loss," *IEEE Access*, vol. 7, pp. 7104-7116, 2018.
- [18] L. Beggel, M. Pfeiffer, and B. Bischl, "Robust Anomaly Detection in Images Using Adversarial Autoencoders," *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Würzburg, Germany, pp. 206-222, 2019.
- [19] Q. Song, F. Xu, and Y. Q. Jin, "SAR Image Representation Learning with Adversarial Autoencoder Networks," *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, Yokohama, Japan, pp. 9498-9501, 2019.
- [20] H. H. Aghdam, E. J. Heravi, and D. Puig, "Toward an Optimal Convolutional Neural Network for Traffic Sign Recognition," *Proceedings of International Conference on Machine Vision*, Barcelona, Spain, vol. 9875, pp. 108-112, 2015.
- [21] P. Golik, P. Doetsch, and H. Ney, "Cross-Entropy vs. Squared Error Training: A Theoretical and Experimental Comparison," *Proceedings of Interspeech*, Lyon, France, vol. 13, pp. 1756-1760, 2013.
- [22] D. P. Kingma, and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, pp. 1-15, 2014.
- [23] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600-612, 2004.
- [24] Test Research, Inc., "TRI Innovation," <https://www.tri.com.tw/tw/index.html>, 2022.