3D Reconstruction of Solder Balls

[1]Yi-Bing Zhang (張易冰), [2]Chiou-Shann Fuh (傅楸善), Hung-Yi Chen (陳弘毅), Cheng-Shi Wong (翁丞世), Tsung-An Hsieh (謝尊安)
[1] Dept. of Electronic Science and Engineering, Nanjing University, Nanjing, China
[2] Dept. of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan
E-mail: 568420827@qq.com, fuh@csie.ntu.edu.tw

ABSTRACT

In our project, we attempt to reconstruct the 3D model of nine solder balls on the PCB (Printed Circuits Board) according to 128 projections of X-ray images of those solder balls. We mainly use two methods: 3D software Unity [1] and our original method for 3D reconstruction. Meanwhile, in order to have a better understanding of ART, SART, we also try to conduct an experiment on ART, SART for 2D reconstruction. Consequentially, we finally obtain the 3D model of those solder balls, which can be used to inspect their inner structure.

Keywords: *Solder Balls; 3D Reconstruction; ART; SART; Unity; X-ray; Original Method*

1. INTRODUCTION

Manufacturers of PCB (Printed Circuit Board) need to ensure the quality of their products. Therefore, it is necessary for them to have an effective method to inspect their product in order to find out if there are any defects. For example, there may be some cracks or voids in the solder ball, which cannot be seen easily in visible light. However, if we can get the X-ray image of solder ball, we can find the cracks in it. Furthermore, it would be helpful if we can obtain the 3D model of solder ball because 3D model contains more information and can be easy to understand. For instance, we can look at the cross-section of the 3D model in order to find if there is a hole or crack in the solder ball.

After realizing the significance of this research, we have tried different methods to reconstruct the 3D model of solder balls. Moreover, the results are useful to the Automatic Optical Inspection (AOI). Unity is a famous software to generate and visualize 3D objects. Thus, we attempt to build the whole X-ray system by which we obtain images of solder balls. Our 3D model designed in Unity aims to compute the accurate positions of those nine solder balls and their sizes. We also use laminography to reconstruct the cross-section image on the focal plane of our X-ray system. Further, in order to get a more accurate 3D model, we create an original method which can provide us with a more accurate result. And this original method can benefit the industry in inspecting the quality of objects on PCB (Printed Circuits Board).

2. 3D RECONSTRUCTION BY UNITY

Unity is a famous 3D design software which helps us to reconstruct the solder balls based on the data calculated from 128 X-ray projection images of solder balls.

2.1 Locate the center of solder ball on the X-ray Image

In order to build the 3D model of the X-ray system and the solder balls in Figure 1, we first attempt to find the coordinates of centers of images of solder balls.

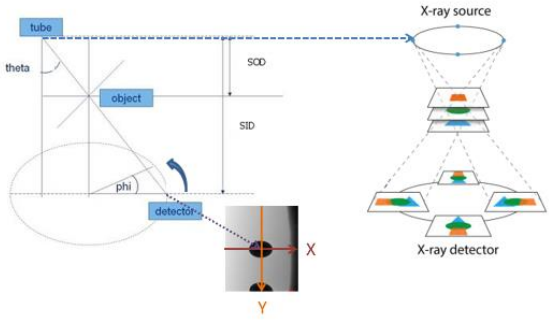


Figure 1 System setup (*theta* = 35°, SOD=Source to Object Distance = 18.432mm, SID=Source to Image Distance= 201.541mm, 1496X1496 pixels, 4um/pixel).

There are many methods to determine centers of those ellipses which are X-ray images of solder balls, as in Figure 2. Moreover, we choose to use a bounding box which is a rectangle to circumscribe the ellipse. Then, we calculate the center of the rectangle and regard it as the center of the ellipse. The result of this method is shown in Figures 3 and 4. Figure 3 shows nine bounding boxes generated to measure the centers of those nine solder balls. Moreover, Figure 4 shows the results of coordinates in pixels of nine centers of those nine solder balls.

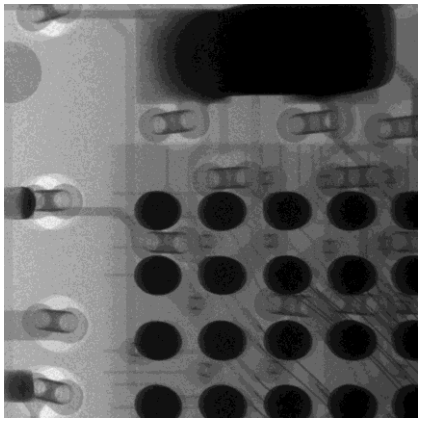


Figure 2 X-ray projection image.

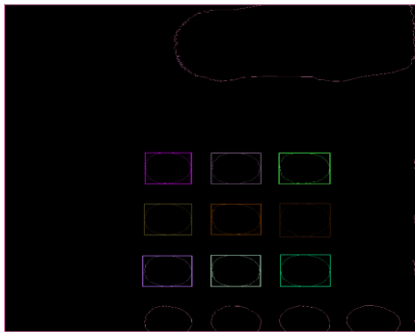


Figure 3 Bounding boxes circumscribing ellipses.

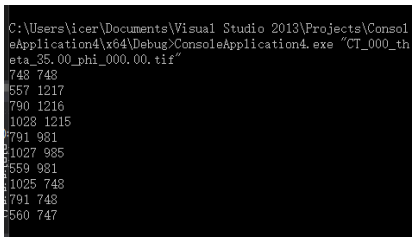
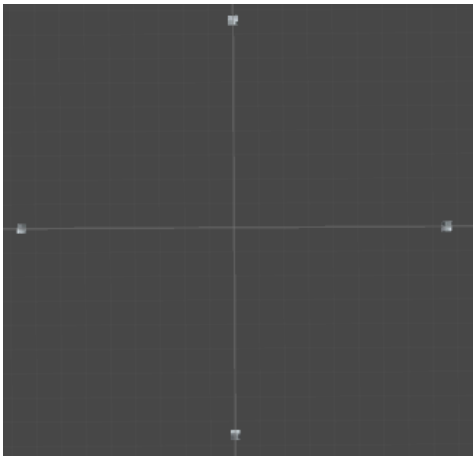


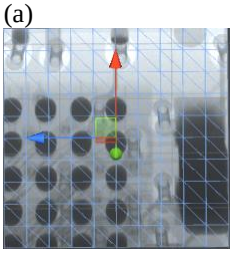
Figure 4 Coordinates of centers of ellipses.

2.2 Locate solder balls in the world coordinate system

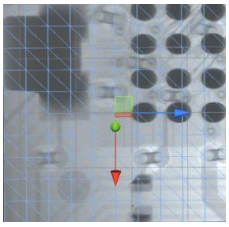
We choose X-ray images of 0°, 90°, 180°, and 270° to calculate the coordinate of those nine solder balls in the world coordinates system. According to the actual size of the X-ray system, after scaling down, we build a model of the system in Unity.



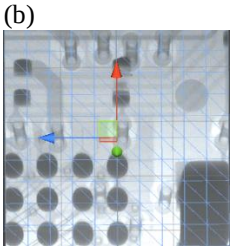
(a)



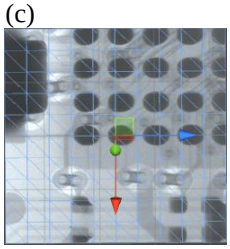
(b)



(c)



(d)



(e)

Figure 5 (a) The four X-ray projection images in the X-ray system. (b) X-ray projection image of 0^{o}. (c) 90^{o}. (d) 180^{o}. (e) 270^{o}.

We use geometry to calculate the coordinates of the solder balls in Figure 6. For the same ball in two images, theoretically, we can compute its coordinates.

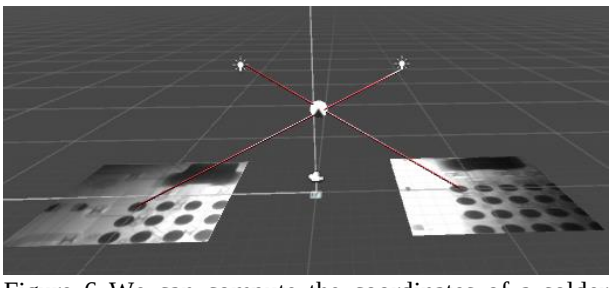
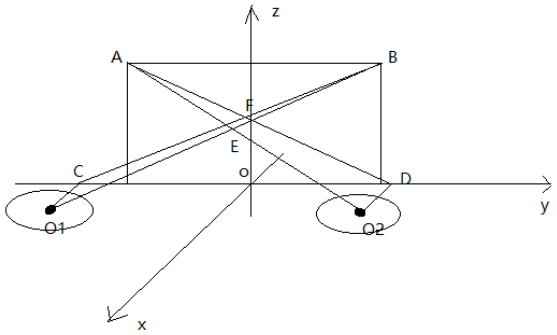


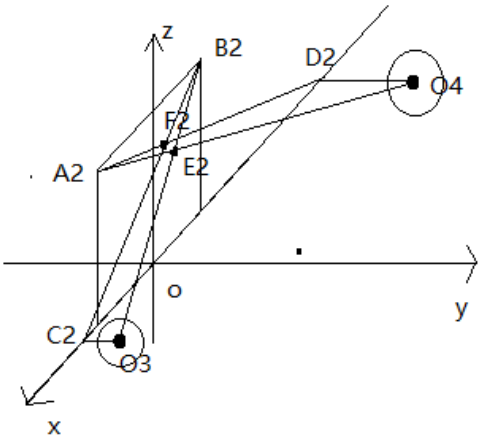
Figure 6 We can compute the coordinates of a solder ball in the world coordinate system based on geometry.

For the same ellipse in 0°, 90°, 180°, and 270° Projection images, the coordinates (in pixels in the image) of its center are (560,747), (750,936), (941,746), and (752,555) respectively. Based on those coordinates and the parameters of our X-ray system, we figure out a

fast way to compute the coordinates of that solder ball in the world coordinates system.



(a)



(b)

Figure 7 The method we use to compute the position of solder ball. (a) O2 is the center of the ellipse on 0° projection and O1 is the center of the ellipse on 180° Projection. A is the position of X-ray tube when 0° projection is generated and B is the position of X-ray tube when 180° projection is generated. BO1 and AO2 are the projection rays which intersect at E which is exactly the center of our solder ball. BC and AD are the projections on y-o-z plane of BO1 and AO2. F is the projection of E. (b) O4 is the center of the ellipse on 90° projection and O3 is the center of the ellipse on 270° projection. A2 is the position of X-ray tube when 90° projection is generated and B2 is the position of X-ray tube when 270° projection is generated. B2O3 and A2O4 are the projection rays which intersect at E2 which is exactly the center of our solder ball. B2C2 and A2D2 are the projections on x-o-z plane of B2O3 and A2O4. F2 is the projection of E2.

On the 1496X1496 image, the center is (748,748). Thus, if we want to know the coordinates of D, we first calculate the distance between (748,748) and O2(560,747) on y axis. Then, multiply the distance by resolution (4um/pixel).

$$(748-560)*4/1000=0.752 \text{ mm}$$

According to parameters of our X-ray system, the center of the 0° image (748,748), locates at (0,128.215, 0).

$$128.215-0.752=127.463 \text{ mm}$$

Thus, D(0,127.463,0). Similarly, C(0,-127.443,0), C2(127.443,0,0), and D2(-127.463,0,0).

Because A(0,-12.905,201.541),B(0,12.905,201.541), A2(12.905,0,201.541), and B2(-12.905,0,201.541).

We can use linear functions to describe AD, BC, A2D2, and B2C2.

AD: $z=-1.436y+183.012$……(1)
BC: $z=1.436y+183.009$……..(2)
A2D2: $z=1.436x+183.012$……...(3)
B2C2: $z=-1.436x+183.009$……..(4)

F is the solution of (1) and (2), when F2 is the solution of (3) and (4).

F(0,0.001,183.010) F2(0.001,0,183.014)

In this way, we can obtain the coordinate of the center E (E2) of the solder ball because F is the projection of E on y-o-z plane when F2 is the projection of E2 on x-o-z plane.

E(0.001,0.001,183.012).

Similarly, we use this method to compute the coordinates of centers of nine solder balls.

(0.001,0.001,183.012) (0,0.086,183.011)
(-0.001,0.172,183.011) (0.085,0,183.011)
(0.085,0.085,183.011) (0.085,0.172,183.012)
(0.171,0.001,183.012) (0.172,0.085,183.012)
(0.171,0.173,183.011)

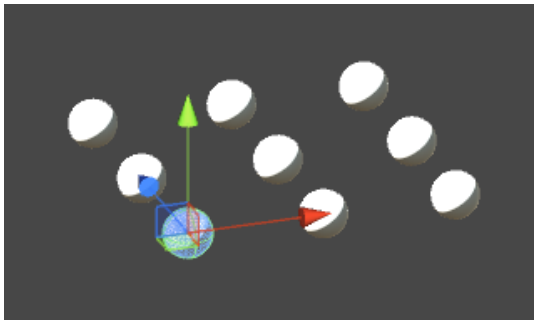
And in order to make the process automatic and fast, we write a program in Matlab [2] to solve those coordinates.

```
y1=939;
y2=557;
y11=-128.215-(748-y1)*4*0.001;
y22=128.215-(748-y2)*4*0.001;
syms k1 b1
f1=y11*k1+b1;
f2=12.905*k1+b1-201.541;
[k11,b11]=solve(f1,f2,k1,b1);
k11=double(k11);b11=double(b11);

syms k2 b2
f1=y22*k2+b2;
f2=-12.905*k2+b2-201.541;
[k22,b22]=solve(f1,f2,'k2','b2');
```

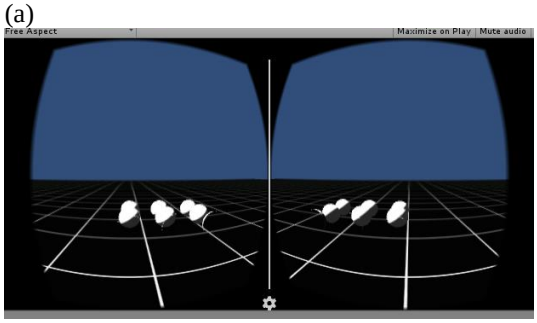
```
k22=double(k22);b22=double(b22);

syms x z
f1=k11*x+b11-z;
f2=k22*x+b22-z;
[x,z]=solve(f1,f2,'x','z');
x=double(x);z=double(z);
```

Therefore, we obtain their positions in our X-ray system.

Further, we introduce Virtual Reality (VR) technology into our research and then we use Google Cardboard [3] to see their 3D model.



(a)



(b)

Figure 8 (a) 3D model of solder balls built with Unity. (b) 3D model seen by Google Cardboard.

2.3 Fitness

In order to determine the size and shape of the solder ball, we compare the shadow of our solder ball model with the ellipse in the X-ray projection image. If they fit perfectly, then we can obtain the size and shape of our solder ball.

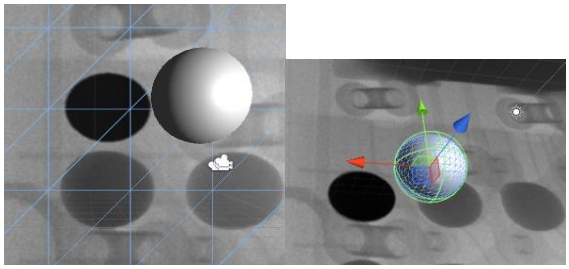
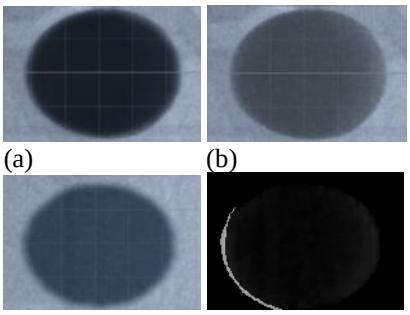


Figure 9 The shadow fits well with the X-ray image when the size and shape of solder ball model are correct.



(a) (b) (c) (d)

Figure 10 (a) The image of the combination of shadow of solder ball model and the X-ray image. (b) The X-ray image. (c) The shadow of the solder ball model. (d) The difference between the shadow and the X-ray image.

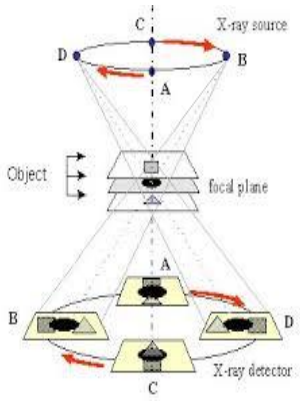
$$\text{Fitness}=\frac{shadow \cap image}{shadow \cup image}$$

In our research, we use Matlab to generate Figure and calculate the fitness.

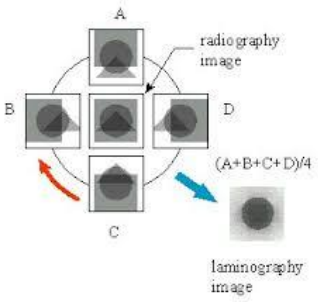
Fitness=0.9573

3. LAMINOGRAPHY

We can obtain the cross-section image on the focal plane of our X-ray system by averaging those 128 projections.



(a)



(b)

Figure 11 [3] (a) An example of our X-ray system. (b) The result of laminography method.

First, we compute the average image of 128 projections. Second, we conduct threshold on that image to generate a binary image.

(a) (b)

Figure 12 (a) The result of laminography where upper right corner is dark capacitor, but solder balls are dispersed and unrecognizable as spheres because they are below focal plane. (b) The binary image generated by thresholding.

As we can see, on the focal plane of our system, there is a rectangle as shown in Figure 12.

4. ART (ALGEBRAIC RECONSTRUCTION TECHNIQUE)

4.1. ART (Algebraic Reconstruction Technique)

In the field of Computed Tomography, ART (Algebraic Reconstruction Technique) is widely used to reconstruct the cross-section of an object. It is an iterative method and can gradually reconstruct the image based on projections of that image in different angles.

The iterative algorithm [4] is defined as,

$$f_j^{(i)} = f_j^{(i-1)} + \lambda \frac{p_i - \sum_{k=1}^{N} f_k^{(i-1)} w_{ik}}{\sum_{k=1}^{N} w_{ik}^2} w_{ij}$$

$i = 1, 2, 3 \ldots M \qquad j = 1, 2, 3 \ldots N$

M : total number of projection rays.
N : total number of pixels.
λ : relaxation factor.
p_i : the projection value of the i-th ray.
w_{ik} : the weight of the k-th pixel on the i-th ray.
$f_j^{(i)}$: the j-th pixel value after being updated by the i-th ray.
$\sum_{k=1}^{N} f_k^{(i-1)} w_{ik}$: the current projection value of the i-th ray.

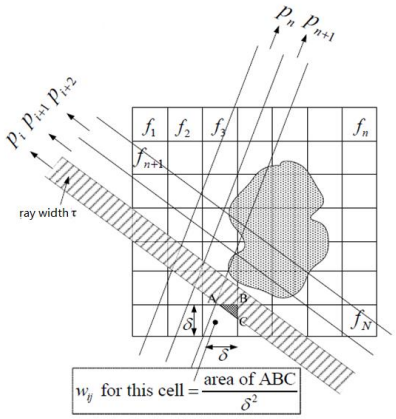


Figure 13 How projections are generated [5].

4.2. Reconstruct 2D image by ART

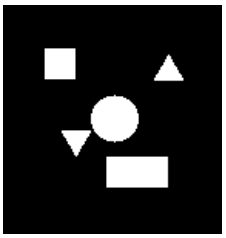


Figure 14 Shapes to be reconstructed.

First, we need to obtain projections in different angles used to reconstruct the image. We generate projections of this image from 0° to 179°, 1° for each projection, just according to the method shown in Figure 13. And we put them together in one image.

Figure 15 Projections of the image. Every row of this figure is one projection of a certain angle of that image (1 degree/projection).

According to the principles of ART, we revised a Matlab program [6] of ART and successfully reconstruct the original image.

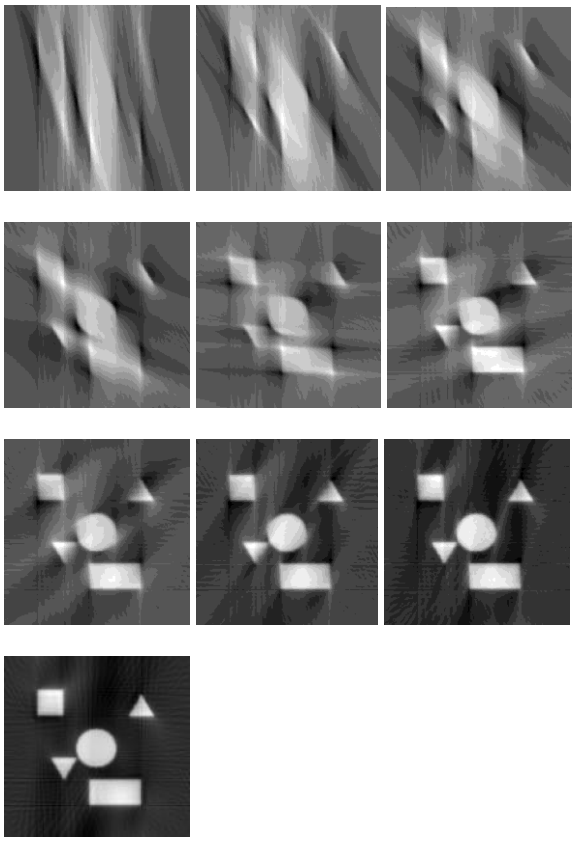


Figure 16 Above images show results after increasing number of iterations (1100 iterations/image). The last image is our final result.

5. SART (SIMULTANEOUS ALGEBRAIC RECONSTRUCTION TECHNIQUE)

SART is another popular method in the field of computed tomography. Moreover, SART considers all projection rays in one iteration to reconstruct the image, unlike that ART only uses one ray to reconstruct the image in one iteration. Generally, SART is believed to resist severe noise and can guarantee a better quality of the reconstructed image than ART.

5.1. SART (Simultaneous Algebraic Reconstruction Technique)

$$f_j^{(i)} = f_j^{(i-1)} + \frac{\sum_{p_i \in P_\varphi} \left[\lambda \frac{p_i - \sum_{k=1}^{N} f_k^{(i-1)} w_{ik}}{\sum_{k=1}^{N} w_{ik}} \right] w_{ij}}{\sum_{p_i \in P_\varphi} w_{ij}} \qquad j = 1, 2, \ldots, N$$

P_φ : All projection rays.

Therefore, based on this formula, we can know that in one iteration, SART takes all rays into consideration, which means that after one iteration, the image will mostly appear.

Again, we take Figure 14 as an example and attempt to reconstruct it. Successfully, we get the result after just one iteration.

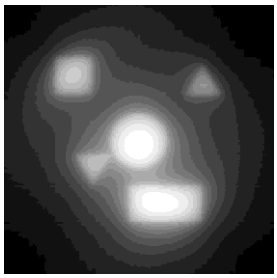


Figure 17 The result after one iteration by SART, good but still significant reconstructed halo noise.

5.2. 3D Reconstruction by SART

3D reconstruction is based on the knowledge of 2D reconstruction by SART. We can reconstruct the 3D object layer by layer. Moreover, the reconstruction of just one layer is the same as what we do in 2D reconstruction. After all layers are reconstructed, we stack them together to obtain the 3D object.

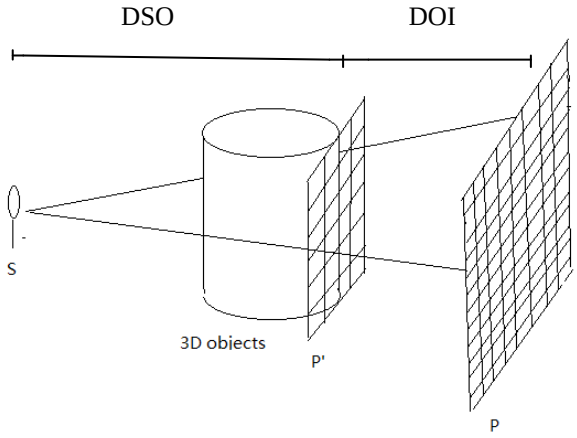
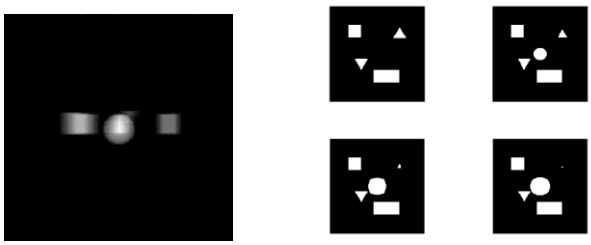


Figure 18 The way we get projections. DSO: distance between source and object. DOI: distance between object and image.

In Figure 18, it shows how we get the projection of the 3D object efficiently. First, we reduce the physical size of image *P* but maintain its size in pixels to fit image *P'* in physical size. Second, because images *P* and *P'* have different pixels and we can easily get image *P'*, which is the parallel projection of the 3D object, so we use image *P'* to interpolate image *P*. Therefore, we obtain image *P*, which is the actual projection in that system.

As for the process of 3D reconstruction, we just conduct a reverse operation. We use image *P* to interpolate image *P'*. After acquiring *P'* in different angles, we can use SART to reconstruct the 3D object layer by layer.

Figure 19 shows objects which we want to reconstruct. Figure 19 (a) corresponds to image *P'* in Figure 18.



(a) (b)

Figure 19 (a) 3D objects we want to reconstruct, including a cube, a sphere, a cuboid, a triangular pyramid, and a triangular prism. (b) Cross-sections of those objects obtained at different heights.

We find a program written by Kyungsung Kim [7] and revise it to fit our research. After running it, we successfully reconstruct the 3D objects.

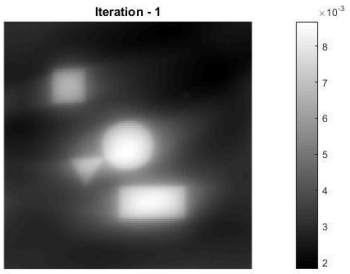


Figure 20 A cross-section of the reconstructed 3D object (after one SART iteration) at the height of 15 pixels.

6. 3D RECONSTRUCTION FOR SOLDER BALLS USING OUR ORIGINAL METHOD

6.1 Store projections into a 3D matrix

In our task to reconstruct solder balls, we are given 128 X-ray projections (1496X1496 pixels) in a file, just as the image in Figure 2. Thus, we first need to automatically read those images into a 3D matrix in Maltab so that we can visit them easily later.

Below is a part of our MATLAB code to finish this job.

```
for  j = 1:length(listOfImages)
  fileName = listOfImages(j).name;
  rfid=sprintf(fid,fileName);
  Irgb=imread(rfid);
  Irgb=imresize(Irgb,[500,500]);
  proj(:,:,j)=Irgb;
end
```

After running this program, we successfully store those projection images into a 3D matrix where 1496/3=500.

6.2 3D Reconstruction of solder balls

According to characteristics of our X-ray system shown in Figure 1, projections are generated in a rotational track and the projection angle is 35°. Therefore, after knowing how those projections are obtained, we create our own method to reconstruct the 3D information of solder balls.

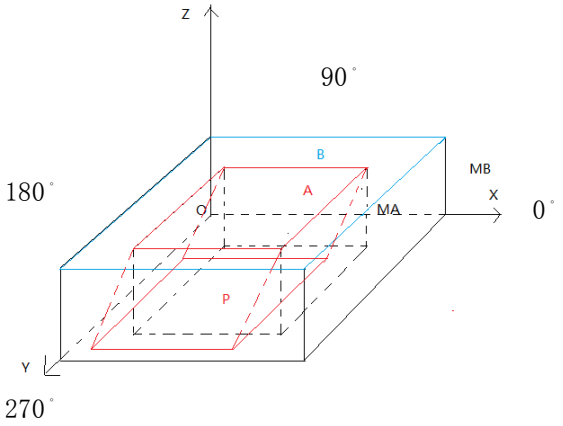


Figure 21 Our original method. MB is a 700X700X200 pixels matrix. MA is a 500X500X200 pixels matrix in MB. P is the X-ray projection image resized to 500X500 pixels. Plane A is in the center of plane B. MB (700X700X200 pixels) is larger than MA (500X500X200 pixels) due to 35° tilt of X-Ray source, inspected PCB board, and X-Ray sensor in all 360° directions (700=500+2*200*sin35°).

First, we build a 3D matrix as MB in Figure 21 to contain the 3D model of solder balls and the whole PCB (Printed Circuit Board). Plane *P* is the projection and plane *A* is the focal plane of our X-ray system because the objects on focal plane keep consistent in all projections (laminography can reconstruct the image on focal plane, in Figure 12). X-ray passes some voxels in our 3D matrix and the values of those voxels passed by the X-ray are accumulated into the corresponding pixel in the projection image *P*. We create the following formula to restore the values of those voxels based on projections.

$$f_j = \sum_{i}^{M} \frac{p_i}{n_i} \qquad j=1,2\ldots N$$

N : total number of voxels in MB
M : total number of projection rays that pass by voxel *j*.

p_i: the value of the i-th projection ray.
n_i: number of voxels passed by the i-th projection ray.

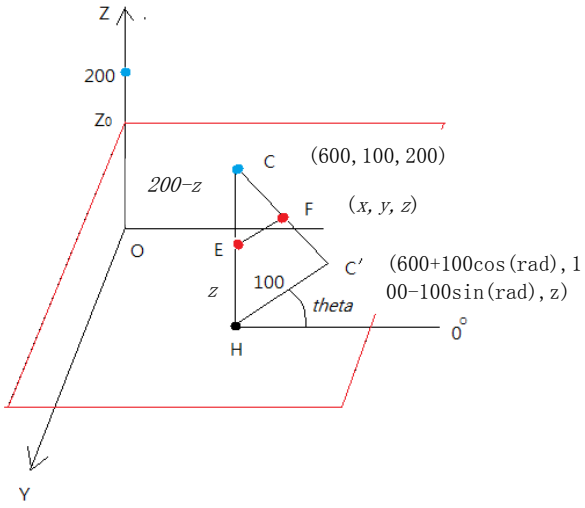


Figure 22 The method to find the voxel on $z=z_0$ plane passed by a certain X-ray.

Based on Figure 21, we draw Figure 22 to illustrate our method concretely. Upper right top corner C is a point on the focal plane z=200 and C' is its corresponding point in the X-ray image. And they belong to a same projection ray. Projection ray CC' intersect with plane $z=z_0$ at F. CH is a perpendicular line with plane X-O-Y, which intersect with plane $z=z_0$ at E. Now, we will illustrate how to calculate the coordinates of F, which is the voxel we want to reconstruct.

Assume that

$C(x_c,y_c,200)$,

Thus,

$C'(x_c+100*\cos(theta),y_c-100*\sin(theta),0)$,

$E(x_c,y_c, z_0)$, $H(x_c,y_c,0)$.

Triangle CHC' is similar with triangle CEF.

Thus,

$$\frac{EF}{HC'} = \frac{CE}{CH}$$

$$\frac{x_f - x_c}{100*\cos(theta)} = \frac{200 - z_0}{200}$$

$$\frac{y_f - y_c}{-100*\sin(theta)} = \frac{200 - z_0}{200}$$

By solving equations above, we obtain the coordinates of F where $x_c=j+100$, $y_c=i+100$, $\sin 35^0=0.57=100/200$.

$$x_f = x_c + (1 - \frac{z_0}{200})*100*\cos(theta)$$

$$y_f = y_c - (1 - \frac{z_0}{200})*100*\sin(theta)$$

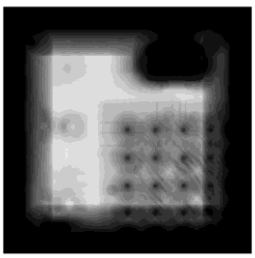
Therefore,

$F(x_f, y_f, z_0)$

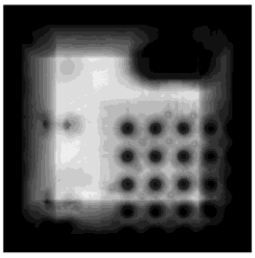
Based on the analysis, we write a program in Matlab to fulfill our idea. Below is the main part of our code.

```
for theta=0:2.81:357.19
  p=p+1;
  rad=theta*pi/180;
   prothe=proj(:,:,p);
   for z=1:1:200
   for i=1:1:500
   for j=1:1:500
   x=round((j+100)+((200-z)/200)*100*cos(rad));
   y=round((i+100)-((200-z)/200)*100*sin(rad));
      image(y,x,z)=image(y,x,z)+(1/200)*prothe(i,j);
   end
   end
   end
end
```

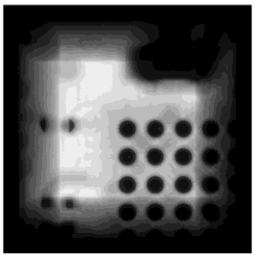
Our code is based on our formula and easy to understand. After running this program, as consequence, we successfully reconstruct the 3D information of those solder balls. Meanwhile, we discover the inner structure of solder balls.



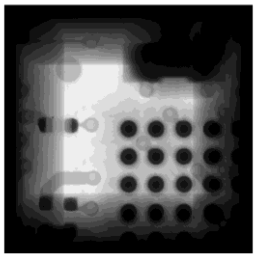
(a)



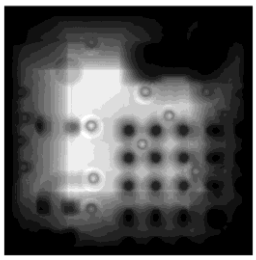
(b)



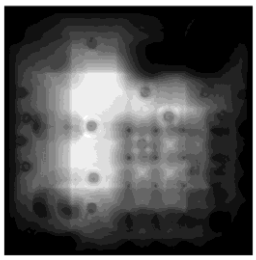
(c)



(d)



(e)



(f)

Figure 23 Cross-sections of our reconstructed 3D matrix MB (Figure 21) at different heights. (a) z=120 (b) z=100 (c) z=80 (d) z=60 (e) z=40 (f) z=20 pixels.

In cross-sections of our 3D reconstructed model, we can see the structures of different objects on the PCB (Printed Circuit Board).

```
>> m3Dmymethod
Elapsed time is 1438.336690 seconds.
>>
```

Figure 24 Time consuming of our program (Intel(R) Core(TM) i5-3230M CPU, 2.60 GHz, 11.6 GB available RAM, Win 10 Professional, NVIDIA GeForce 710M).

Further, we pick the cross-section at z=85 pixels and increase its contrast in order to find more information in those solder balls. And we obtain the following result.

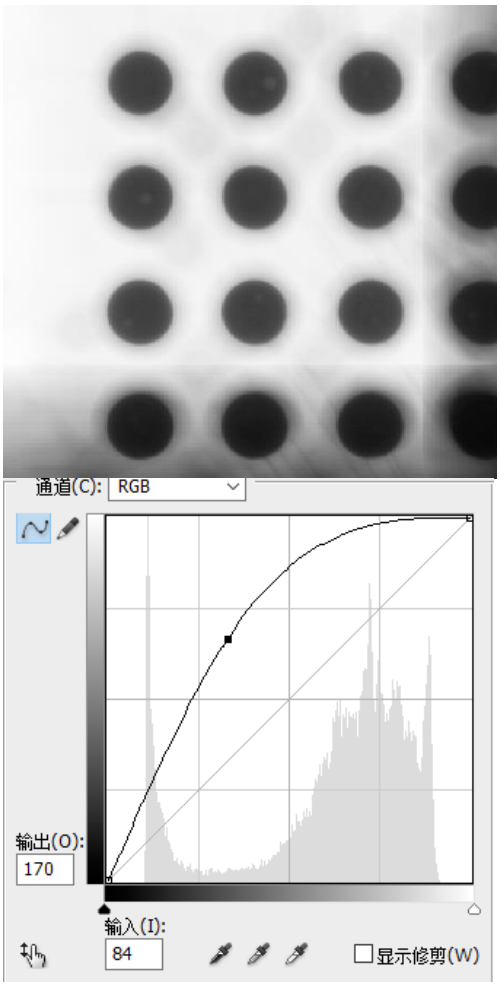


Figure 25 Cross-section of solder balls after adjusting the input-output curve (Photoshop input-output curve) and enhancing contrast. Voids (white disks) inside solder balls (black disks) are visible.

We can find some small voids in those solder balls in Figure 25. Therefore, we successfully create a new original algorithm to inspect the quality and inner structure of solder balls, which is supposed to benefit the industry related.

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, Lumens Digital Optics, and Lite-on.

REFERENCES

[1] Unity Technologies, “Unity,” *http://unity3d.com/*, 2016.

[2] MathWorks, “Matlab,” *http://www.mathworks.com*, 2016.

[3] Google, “Google Cardboard,” *https://vr.google.com/cardboard/index.html*, 2016.

[4] H. K. Lin, “Algebraic Reconstruction Technique,” Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, 2016.

[5] H. K. Lin, “Algebraic Reconstruction Technique,” Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, 2016.

[6] Ligong Han, “Tools for 2-D Tomographic Reconstruction ,” *http://www.mathworks.com/matlabcentral/fileexchange/43008-tools-for-2-d-tomographic-reconstruction?s_tid=srchtitle*, 2015.

[7] Kyungsang Kim, “3D Cone beam CT (CBCT) projection backprojection FDK, iterative reconstruction Matlab examples,” *http://www.mathworks.com/matlabcentral/fileexchange/35548-3d-cone-beam-ct--cbct--projection-backprojection-fdk--iterative-reconstruction-matlab-examples?s_tid=srchtitle*, 2016.