

Qidi Han (6814891757)
EE-569
Homework #3
March-1-2020

Problem 1: Geometric Image Modification

I. Abstract and Motivation

In many times, geometric modification of images is popular in our daily life. There are many ways to do the geometric modification. To do those geometric modification, we also will use some linear equations to design and implement our transformation. Those techniques are very useful in our daily life. For example, when we need to enlarge the image, there is a interpolation techniques used to enlarge the image. Homographic transformation and image stitching are also important technique tools that used in our daily life. There are many times that people need to combined images from different perspective into a big image. There are two experiment that we are going to do in this project. The first experiment is called geometric warping, it will make us to design spatial warping technique to achieve our three requirements. The second experiment is called homographic transformational and image stitching. We are going to combined different images from different perspective into one big image. For both image, we need to calculate the transformation function.

II. Approach and Procedures

a. Geometric warping

There are three different ways of doing geometric warping. The first way is translation. It basic moving around the image in different position. The translation matrix provided below. The x and y will represent the original data point, x+x0 and y+y0 will represent our destination point.

$$\begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ 1 \end{bmatrix} \quad (1)$$

The second way is rotation. It changes the orientation of the image. The rotation matrix is provided below. The x and y represent the original data point, x' and x' represent our new destination data point.

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \quad (2)$$

The third way is scaling. It enlarges or narrow the image by using the scaling matrix below. The s1 and s2 represent the x and y scaling parameter. The x and y represent the original data point, and s1x and s2y represent the new destination data point.

$$\begin{bmatrix} S1 & 0 & x \\ 0 & S2 & y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x * S1 \\ y * S2 \\ 1 \end{bmatrix} \quad (3)$$

If we combined those three different transformations together, it will be called affine transformation. The affine transformation matrix combined those three different matrices together provided below.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x0 \\ 0 & 1 & y0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} S1 & 0 & x \\ 0 & S2 & y \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

For the first question in the project, I only used the matrix 3 that proved above. The first step of my project is that I cut the image into four different pieces. Then I will use the boundary pixel position to calculate the S1 and S2 for matrix 3. The S1 and S2 will be different for each row. The Pythagoras theorem will use for calculating our S2. The length of our small image will be 256. And our column number is from 1 to 256.

$$|S2| = \frac{\sqrt{(\text{length of each small image})^2 + (\text{column number})^2}}{\text{length of each small image}} \quad (5)$$

After, I calculate the S1 and S2. I will apply inverse matrix to do our project. I will build the new image first and use inverse matrix to find the pixel value in our original image. Finally, I will fill the pixel value into our new image. The formula that I used below

$$\text{oldposition} = \text{inv}(S) * \text{newposition} \quad (6)$$

When I use the inverse matrix, it will have decimals for our number, I use simply used round() function to round the number. For the next problem, I will introduce a bilinear interpolation method to do the issue of decimals. Figure 1a, 1b, 1c, 1d shows the comparison between our original small image with our four processed small images. Figure 2 shows the combination of processed image.

After, I warp the original image to the disk-shaped image, I will reverse the disk-shaped image to square image. I will use the same method for the last question, I will build the new square image first, then use the matrix to find pixel value from our disk-shaped image. The formula below is used for this process. Figure 3 shows the reversed image of Hedwig image.

$$\text{disk_shaped_image_position} = S * \text{square_image_position} \quad (7)$$

Repeated the same process for image Raccoon and bb8 image. Figure 4a, 4b, 4c, 4d and Figure 7a, 7b, 7c, 7d shows the comparison between our original small image with our four processed small images for Raccoon and bb8 images. Figure 5 shows the combination of processed image for Raccoon. Figure 8 shows the combination of processed image for bb8. Figure 6 shows the reversed image for Raccoon image. Figure 9 shows the reversed bb8 images. There are three requirements for our question. The first requirement is the boundaries of original image still lie on the boundaries of circle. The second requirement is the center of the original image is still the center of disk-shaped image. The third requirement is the mapping should be reversible. Since we use the boundary pixel position to calculate the S. I will achieve the first requirement. The center never changed in our case. And Figure 3, 8, 9 shows the reversed image. Therefore, I also achieve the third requirement.

b. Homographic Transformation and Image Stitching

For this question, I will use detectSURFFeatures() function to detect and extract features between left and middle image. And the feature between right and middle image. After I have the features, I will matchFeatures() function to store all the similar feature from both image. From those similar features, I will pick 4 different pairs. I will use those 4 different pairs to calculate the H function. I combined three images into a big image with size 1440*1920. I put the original in the center of my new image. And the left image in the left hand side, the right images in the

right hand side. I will put those 8 different pairs data point into the new image. I use the new position for those 8 different pair data points to calculate two different H. The equation below is for calculate the H. Figure 10 shows the H values. Figure 10a shows the matching point that I chose for left and middle image. Figure 11 shows the H values. Figure 11a shows the matching point that I chose for right and middle image.

$$inv \begin{pmatrix} x1 & y1 & 1 & 0 & 0 & 0 & -x1X1 & -y1X1 \\ x2 & y2 & 1 & 0 & 0 & 0 & -x2X2 & -y2X2 \\ x3 & y3 & 1 & 0 & 0 & 0 & -x3X3 & -y3X3 \\ x4 & y4 & 1 & 0 & 0 & 0 & -x4X4 & -y4X4 \\ 0 & 0 & 0 & x1 & y1 & 1 & -x1Y1 & -y1Y1 \\ 0 & 0 & 0 & x2 & y2 & 1 & -x2Y2 & -y2Y1 \\ 0 & 0 & 0 & x3 & y3 & 1 & -x3Y3 & -y3Y1 \\ 0 & 0 & 0 & x4 & y4 & 1 & -x4Y4 & -y4Y1 \end{pmatrix} * \begin{bmatrix} X1 \\ X2 \\ X3 \\ X4 \\ Y1 \\ Y2 \\ Y3 \\ Y4 \end{bmatrix} = \begin{bmatrix} h11 \\ h12 \\ h13 \\ h21 \\ h22 \\ h23 \\ h31 \\ h32 \end{bmatrix} \quad (8)$$

After I have H values, I will use equation below to calculate the new position for our images. And warping each of pixel into a big image. The x and y represent the original pixel position, The x'/w' and y'/w' represent the processed pixel position.

$$\begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x'/w' \\ y'/w' \\ 1 \end{bmatrix} \quad (9)$$

After I warp the original image into another. There are many black dots in our image. To fixing the black dots, we have to do the bilinear interpolation. The bilinear interpolation takes the inverse H matrix multiple with processed image to get original pixel value. There is issue of decimal in this problem. The formula for interpolation is provided below. Figure 12 shows the parameter for interpolation process.

$$f(p + \Delta x, q + \Delta y) = [1 - \Delta x \quad \Delta x] \begin{bmatrix} f(p, q) & f(p, q + 1) \\ f(p + 1, q) & f(p + 1, q + 1) \end{bmatrix} \begin{bmatrix} 1 - \Delta y \\ \Delta y \end{bmatrix} \quad (7)$$

Finally, Figure 13a shows the before interpolation image. Figure 13b shows the image after interpolation. For the overlap area, I will average the value for each pixel.

III. Experimental Results

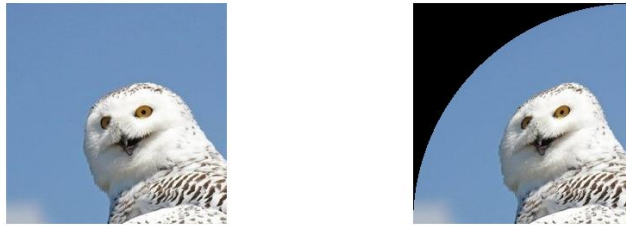


Figure 1a shows the comparison between our original top left image with our processed top left images.

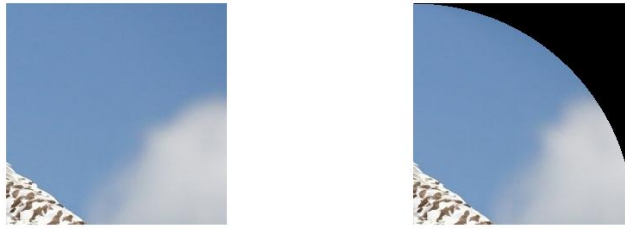


Figure 1b shows the comparison between our original top right image with our processed top right images.



Figure 1c shows the comparison between our original bottom left image with our processed bottom left images.

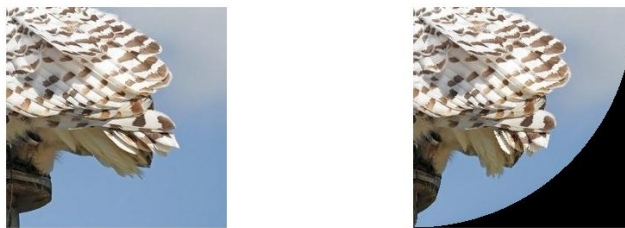


Figure 1d shows the comparison between our original bottom right image with our processed bottom right images.



Figure 2 shows the combination of processed image for Hedwig image.



Figure 3 shows the reversed image for Hedwig image.

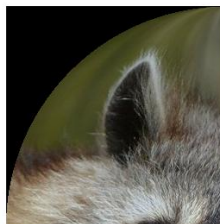
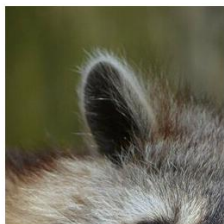


Figure 4a shows the comparison between our original top left image with our processed top left images for Raccoon image.

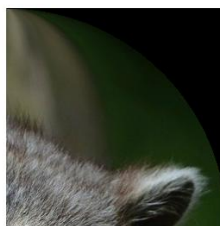
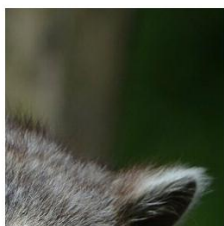


Figure 4b shows the comparison between our original top right image with our processed top right images for Raccoon image.

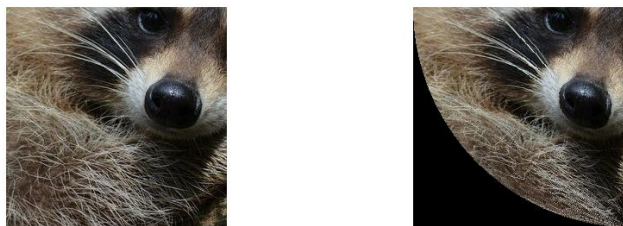


Figure 4c shows the comparison between our original bottom left image with our processed bottom left images for Raccoon image.

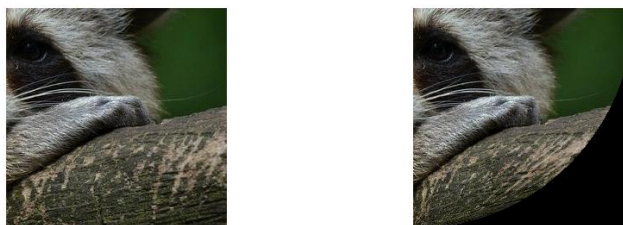


Figure 4d shows the comparison between our original bottom right image with our processed bottom right images for Raccoon image.



Figure 5 shows the combination of processed image of Raccoon image.

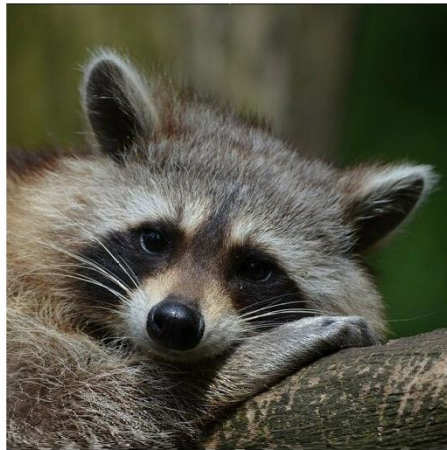


Figure 6 shows the reversed image for Raccoon image.



Figure 7a shows the comparison between our original top left image with our processed top left images.

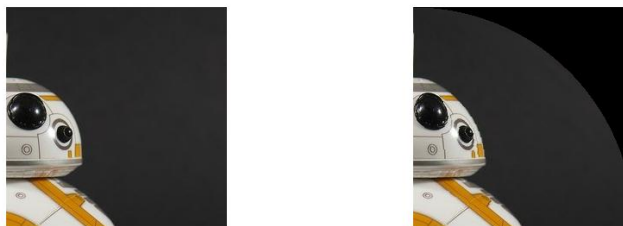


Figure 7b shows the comparison between our original top right image with our processed top right images.

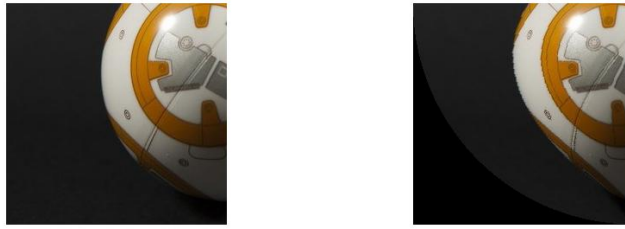


Figure 7c shows the comparison between our original bottom left image with our processed bottom left images.

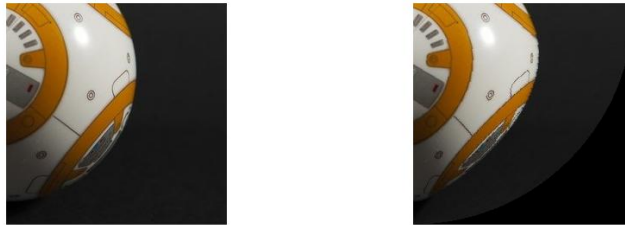


Figure 7d shows the comparison between our original bottom right image with our processed bottom right images.



Figure 8 shows the combination of processed image of bb8 image.



Figure 9 shows the reversed bb8 images.

```
h_result =
| 3x3 single matrix
|
| 1.0e+03 *
|
| 0.0024    0.0018   -1.0177
| 0.0000    0.0048   -0.5329
| -0.0000    0.0000    0.0010
```

Figure 10 shows the H values



Figure 10a shows the matching point that I chose for left and middle image.

```
h_result2 =
3x3 single matrix
    0.4198    -0.2902    397.4494
    0.0064    -0.0366    480.3925
    0.0000    -0.0004     1.0000
>>
```

Figure 11 shows the H values



Figure 11a shows the matching point that I chose for right and middle image.

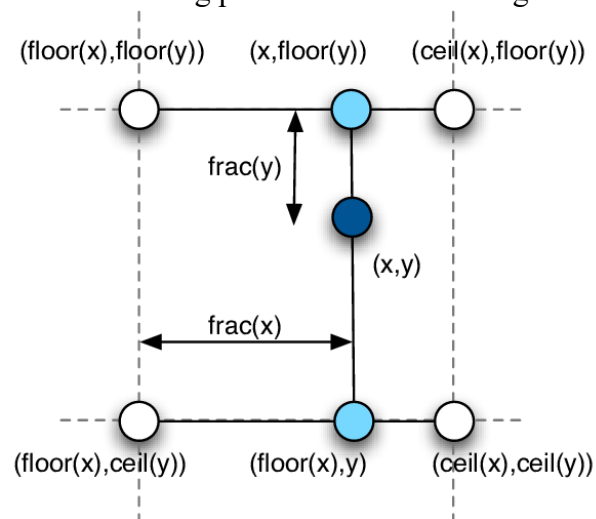


Figure 12 shows the parameter for interpolation process.

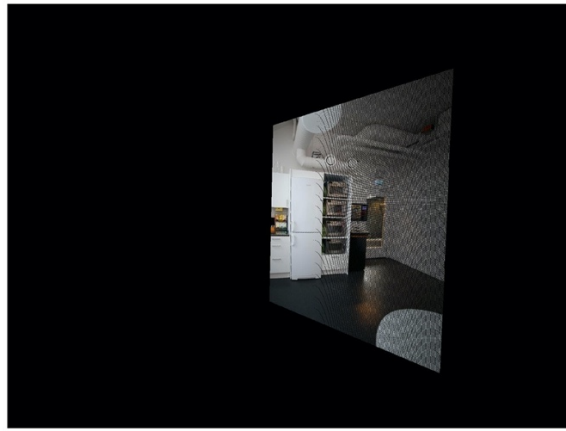


Figure 13a shows the before interpolation image.



Figure 13b shows the image after interpolation.

IV. Discussion

a-1

Describe the approach in the approach section. Figure 1 to Figure 9 shows the result.

a-2

Figure 3,6 and 9 shows the result for the reverse spatial warping image to the original image.

a-3

Compare the recovered square image with the original image, there are some blurred area.

From the Figure 3, the tree stump has many zig-zag area, there are two reason, the first reason is that I didn't do the bilinear interpolation, the second season is that we absolutely lose some pixel value from the disk-shaped image to the square image. All the blurred area in the boundary, the center area of three image still remind the same, because the center still not changing for our question. Center area will not change too much. But for general case, the reversed image still have good quality.

b-1

I used four control points. For the left and middle image, I used

[563.65851,743.60175;696.68274,940.08038;935.40356,954.08997;722.90656,879.45782]

for middle image.

[581.68347,436.85150;695.57135,613.11145;951.55493,619.98193;721.83929,549.06317]

for the left image. Those point are relevant points from my big image. For the right and middle image, I used
[888.33752,1477.5094;948.05652,1305.7500;574.81750,1397.8083;760.68750,1231.8198]
for the right image, and
[912.27100,1153.6377;925.97223,975.19519;568.92627,1071.8191;750.45605,919.31403]
for the middle image. Figure 10a and Figure 11a shows the control points.

b-2

There are four steps of choosing the control points. The first step is using `detectSURFFeatures()` function to detect the features for both image. The second step is using `extractFeatures()` function to extract all the features from the image. The third step is using `matchFeatures()` to matching two images with same features and save the coordinate in a matrix. The final step is selecting those matching features to use in our experiment. For the left and middle image, I choose the left corner of the refrigerator, the second microwave, the charger on the wall and the top of the kitchen cupboard. For the right and middle image, I choose the top of the kitchen cupboard, two concern on the bottom door of the kitchen cupboard and the storage box of cereal. After I choose the control point, I do the step that described above. I do the left and middle image first, then do the right and middle image. Then I combined all three image together, the overlap area, I will average of those areas.

Problem 2: Morphological processing

I. Abstract and Motivation

Morphological processing is one of the useful tools for image processing. Morphological processing also includes shrinking, thinning and skeletonizing. There are two tables for checking the pattern. Each shrinking, thinning and skeletonizing have different results. In this project, I will use those three different technical to solve some problem.

II. Approach and Procedures

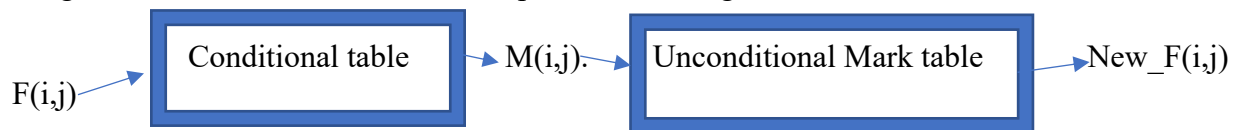
a. Basic morphological process implementation

Before all the process, I will do the boundary extension to each image, but for each boundary, I will use 0 as the boundary. It also called zero padding. Our object marked as 1. The background marked as 0. Before the process, I will make the images with binary number. The threshold for each problem will be different. For the basic morphological process, I will make the threshold as 127. After I make the binary image, I will do the following process.

1. Shrinking

In our project, there are two pass filter for this shrinking process. The first table called conditional table. The second table called unconditional table. Below is the demonstration for the processing. For each $F(i,j)$, I used 3×3 matrix, because our table is 3×3 . For my program, I only check the matrix when the middle point is 1. There are 58 matrices for the conditional table. For the unconditional Mark table, I listed out all the possible matrix. I set M as 1, and D as -1. For the case of A, B and C, I make A B and C as '001' to '111' for each term. There are 42 matrices for the unconditional table. Each process will update one single value. $M(i,j)$ is the mark matrix,

it's the matrix that processed by conditional table. When the 3×3 matrix have similar pattern in the conditional table, I will make 1 in the mark matrix in the same position. Then I will take each 3×3 matrix in the mark matrix to compare with the unconditional mark table. If there are no matrix match with our 3×3 matrix, I will update the original image with 0 in the same position, otherwise, I will stay the same. For each iteration, we will have a new mark matrix. When the original image never changes, I will break the while loop. There are three different image that need to do the shrinking. Figure 14a, 14b and 14c shows the intermediate processing and final image after the first iteration for fan, cup and maze image.



2. Thinning

For the thinning process, I will have the similar process with shrinking. The only different is the conditional and unconditional table. For the conditional, there are 46 matrices in the thinning table. For the unconditional mark table, I will use the same table as the shrinking unconditional mark table. Figure 15a, 15b and 15c shows the intermediate processing image and the final image for the fan, cup and maze image.

3. Skeletonizing

For the skeletonizing, I will have the same process as the shrinking and thinning. The only difference is the conditional and unconditional table. For the conditional table, there are 40 different matrices. For the unconditional table, there are 46 different matrices. Figure 16a, 16b and 16c shows the intermediate processing image and final image for the fan, cup and maze image.

b. Counting games

For the first question of the counting games, I shrink the image first, then count the matrix with $[0\ 0\ 0; 0\ 1\ 0; 0\ 0\ 0]$, because the process of shrinking is making the connected white space as one dot. There are 112 stars in our image. I make the threshold as 127. Figure 17 shows the final result for the star image after shrinking process.

For the second question, I will save all the matrix with $[0\ 0\ 0; 0\ 1\ 0; 0\ 0\ 0]$ after each iteration. There are only 5 iteration for this image. The different star number between each iteration will give us each star sizes in 5 different frequency. Figure 18 shows the table of the iteration number with respect to star number. The second method is using the maximum L2 distance between two-pixel value for each label. The first step is I will set each star to different label. The label is mentioned in the discussion class. The completely steps will describe in next paragraph. Figure 18a shows the histogram for each star size with respect to number of stars. I use the round() function to round the star size that didn't have decimals. For the third question, I will use the method that mentioned in the discussion class. Firstly, the program will quickly scan through the whole image. if the pixel value is 0, we will skip it. If the pixel values are not 0, we will see the minimum neighboring pixel value that not equal to 0. If the minimum neighbor pixel value is 0, I will assign a new label to this pixel value. Finally, I will repeat whole process until the image never changes. Figure 19 show the result for this method. This method is given by the TA in the discussion.

c. PCB analysis

For this question, I will make the image into binary image first. The threshold for this question is 55. Then, I will do the shrinking process to the image. Do the same process with as counting games. We will count the matrix with $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. There are 71 dots in our final image. Therefore, there are 71 holes in our question. Figure 20 shows the shrinking image for the PCB.

For the second question, I reverse the binary image first, because we are going to detect the pathway as white pixel value. Figure 21 shows the image with reverse binary image. I use the same method that mentioned in the discussion. But before label each pathway, I shrink the image first by using the MATLAB `bwmorph()` function to find the hole position, because from part a, we already realize our table is not correct. Therefore, I use the MATLAB function as the TA discuss in the discussion section. Then I use those holes position to remove the hole from the original image. Each hole is contained with 14 pixels. I make those 14 pixels to 1. Figure 22 shows the image after remove the holes. After I remove the hole, there are only pathways reminds in our images. The path with connected with square on left- and right-hand sides of the image will not consider as a path. Finally, I label the pathways. There 32 pathways in our experiment. Figure 23 shows the result from my MATLAB.

The second way of this question is using the labeling technical first, then subtract the number of individual holes. There are 30 lines, and 10 individual holes. Therefore, there are 20 lines in our second methods. Figure 23b shows the MATLAB output for the results.

d. Defect detection

There are six steps of my design.

1. The first step is that shrinking the reverse image. Find the four dots inside of gear teeth. Use those four dots to find the center locations of the image. Figure 24 shows the fours dots after shrinking.
2. The second step is that shrinking the image once and use this shrink image subtract the original reverse image, we will have the boundary for this image. Figure 25 shows the boundary of this gear.
3. Use the boundary of this gear and MATLAB `pdist()` function to caculate the inside and outside radius for this gear.
4. Then I will use this inside radius to move the four circles inside the gear.
5. Then I will use matrix equation 4 above to rotation this non-circles image from step 4 with 90 degrees along the center that I find in the step 1. Figure 26 shows the gear rotate 90 degrees.
6. Finally, I will combine the image from step 5 with our original image. Figure 27 shows the combination image

Experimental Results

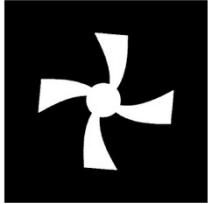
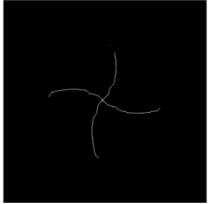
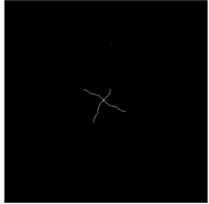
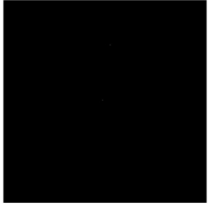
			
Original image	Iteration=50	Iteration=80	Final image with iteration=106

Figure 14a shows the intermediate processing image for shrinking and final image for fan image.



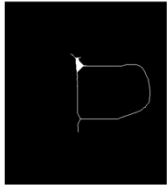
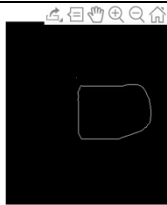
			
Original image	Iteration=50	Iteration=80	Final image with iteration=106

Figure 14b shows the intermediate processing image for shrinking and final image for cup image.


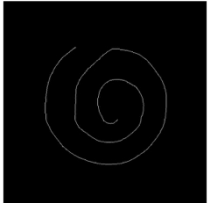
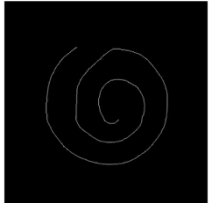

			
Original image	Iteration=50	Iteration=80	Final image with iteration=902

Figure 14c shows the intermediate processing image for shrinking and final image for maze image.

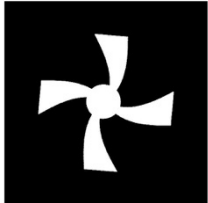
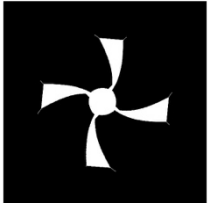
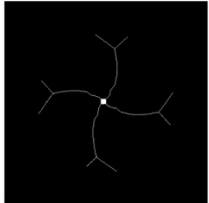
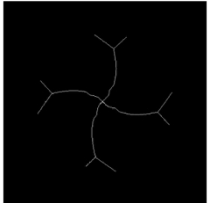
			
Original image	Iteration=10	Iteration=40	Final image with iteration=49

Figure 15a shows the intermediate processing image for thinning and the final image for fan image.




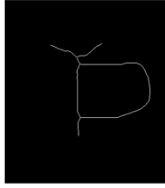
			
Original image	Iteration=10	Iteration=50	Final image with iteration=90

Figure 15b shows the intermediate processing image for thinning and the final image for cup image.



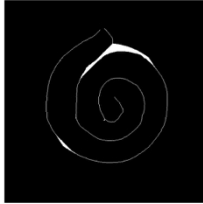
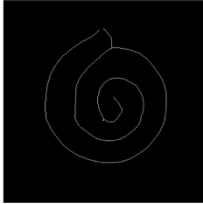
			
Original image	Iteration=10	Iteration=40	Final image with iteration=54

Figure 15c shows the intermediate processing image for thinning and the final image for maze image.

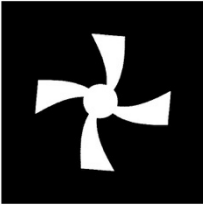
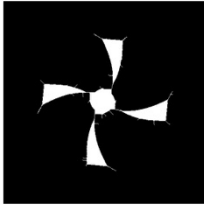
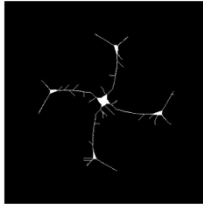
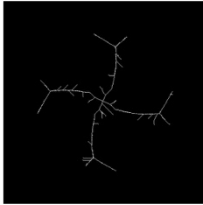
			
Original image	Iteration=10	Iteration=25	Final image with iteration=34

Figure 16a shows the intermediate processing image for skeletonizing and the final image for fan image.




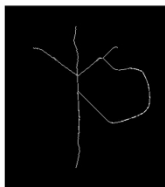
			
Original image	Iteration=10	Iteration=25	Final image with iteration=80

Figure 16b shows the intermediate processing image for skeletonizing and the final image for cup image.

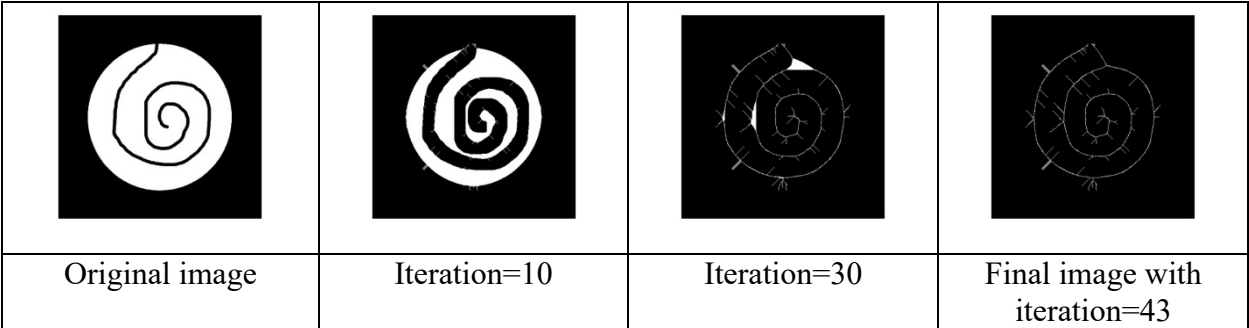


Figure 16c shows the intermediate processing image for skeletonizing and the final image for maze image.

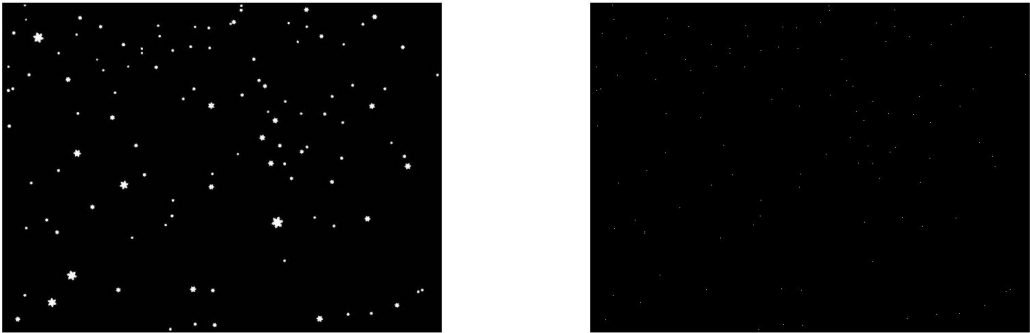


Figure 17 shows the original image and final result for the star image after shrinking process.

Number of Iteration	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Number of star number	31	95	107	110	112

Figure 18 shows the table of the iteration number with respect to star number.

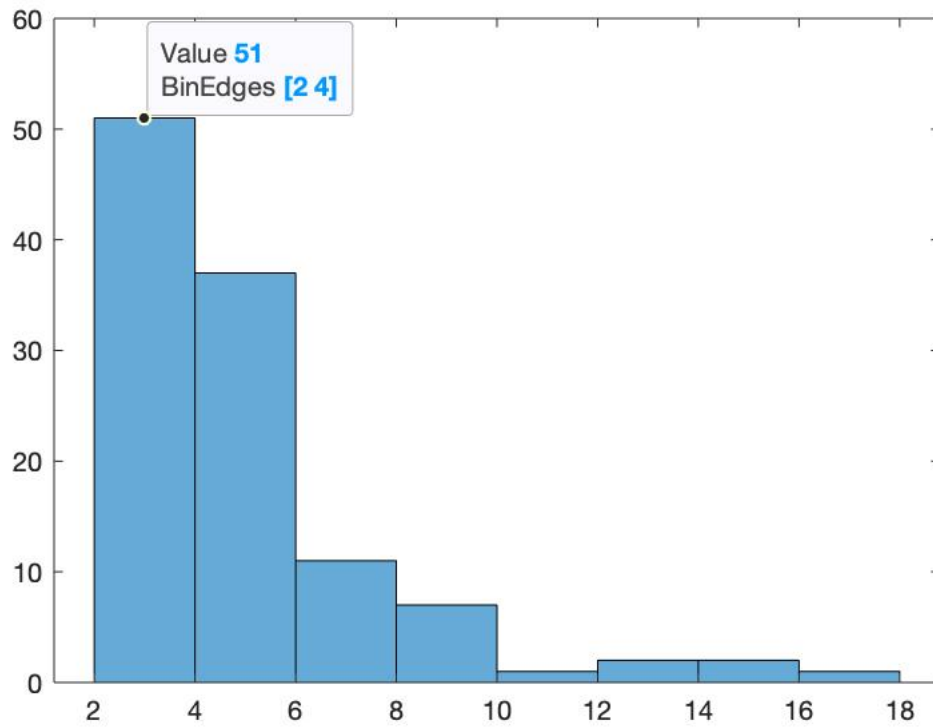


Figure 18a shows the histogram for each star size with respect to number of stars.

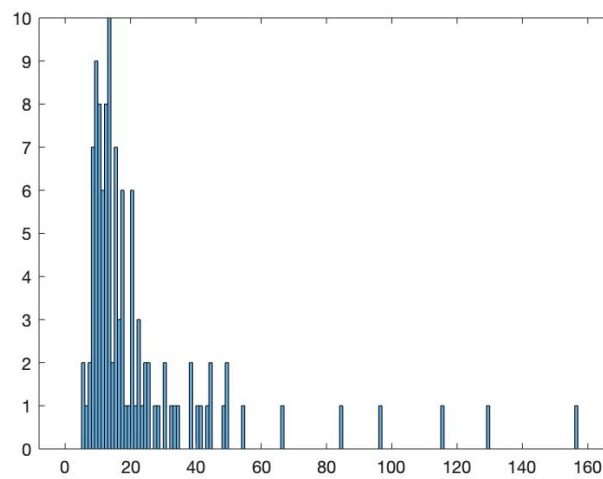


Figure 19 show the histogram of star size (pixel number) vs number of sizes for discussion method.

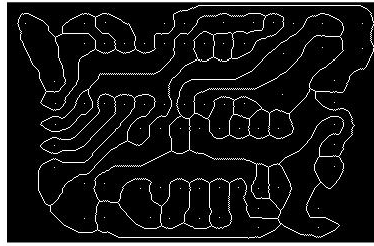


Figure 20 shows the shrinking image for the PCB.

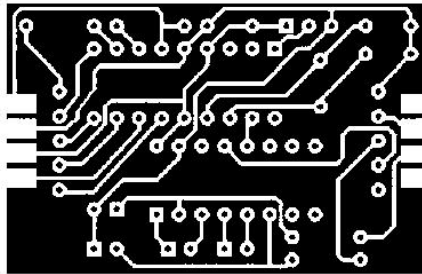


Figure 21 shows the image with reverse binary image.

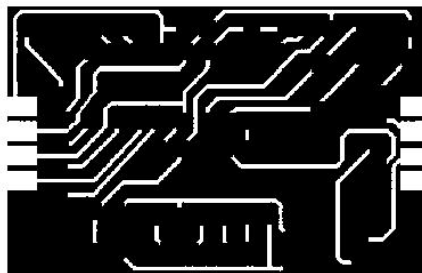


Figure 22 shows the image after I remove the holes.

Command Window

```
Retrieving Image PCB.raw ...  
number lines  
32
```

```
f_x >> |
```

Figure 23b shows the MATLAB output for the results for first methods.

```
Retrieving Image PCB.raw ...  
number lines  
    20  
|  
fx >> |
```

Figure 23b shows the MATLAB output for the results for second methods.

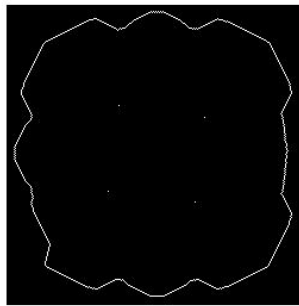


Figure 24 shows the four dots after shrinking.

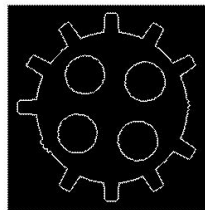


Figure 25 shows the boundary of this gear.

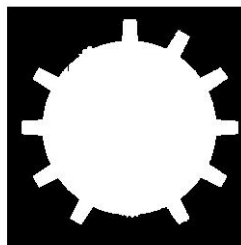


Figure 26 shows the gear rotate 90 degrees.

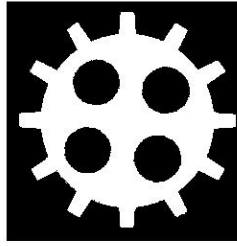


Figure 27 shows the combination image

III. Discussion

2a

For the shrinking process, I think the final answer should be one dots. But for our result from figure 14a, there are two dots. As the TA mentioned on the email. The table for our question is not correct. I believe Figure 14 will give us a good performance if the table is correct. For the Figure 14b, because there is a black area for the handle of the cup, therefore, the shrinking of cup is a circuit.

For the thinning process, it gives us the skeleton of the image, but it didn't give us details, as the Figure 15, we can see the give us the major skeleton of the image. For the maze image, it gives us a pattern of maze. For the cup image, it gives us a two line to the top of the cup. For the fan image, it gives us a similar fan image.

For the skeletonize process, it's similar with thinning image, but there are more details for the skeletonize process, it gives us more skeletons of the image. For the Figure 15 and Figure 16, we can tell the different between the process of the skeletonize and thinning. For the fan image, there are more details lines to the edge of fan. For cup image, there is an addition line to the top of the cup. For the maze image, there are more detail to the edge of the original maze image.

2b

(1) Figure 17 shows the result for the total star number. There are 112 stars.

(2) From Figure 18a, we also can see that the total number star has 112. From the Figure 18a, the first iteration have 31 stars become $[0\ 0\ 0; 0\ 1\ 0; 0\ 0\ 0]$. The second iteration have 95 dots. The third iteration have 107 dots. The fourth iteration have 110 dots. The fifth iteration have 112 dots. The sixth iteration still 112 dots. Therefore, we only use five iteration. From the Figure 18b, I used round number for my pixel size, because the maximum L2 distance have decimal, and for plotting purpose, I use round() function to round the star size.

(3) I already mentioned all the step in the approach section. From Figure 19, we can see that the total number star is 112, and there are 41 different star size. The histogram shows all the information. The x-axis shows the star size, and the y-axis shows the number of stars that belongs to this star size. The totally of the y-axis equal 112.

2c

- (1) I already mentioned all the step in the approach section, there are 71 holes. The holes also include the square holes.
- (2) There are 32 pathways for the first method, there are 20 pathways for the second methods. As I mentioned in the approach section, the path for connecting the eight square box in left-hand and right-hand of the image will not count as a path for the first method. And the individual hole will not consider as a path for the second method.

2d

I already provided all the step and figure for this question in the approach and result section. The Figure 27 is the final answer.