1 Geometric Image Modification

1.1 Motivation

Geometric image modification is one of the techniques for traditional image processing. There are three basic modifications, translation, rotation and scaling. Geometric modification is a combination of translation, scale, rotation and shear.

1.1.1 Changing coordinates

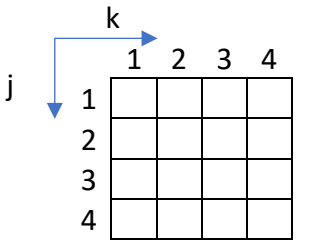


Image coordinate

Image to Cartesian coordinate

$$x = k - 0.5$$

$$y = J + 0.5 - j$$

Cartesian to Image coordinate

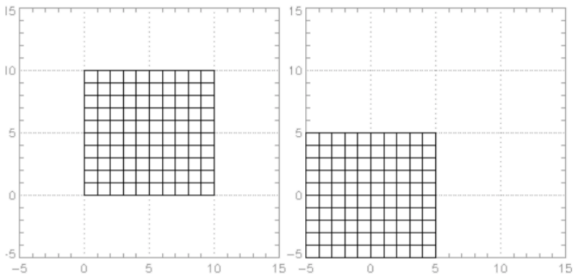
$$k = x + 0.5$$

$$j = J + 0.5 - y$$

1.1.2 Translation

Translation is moving the value of each pixels with the same value follow the x-axis or y-axis. When using the linear algebra, it can be shown as:

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

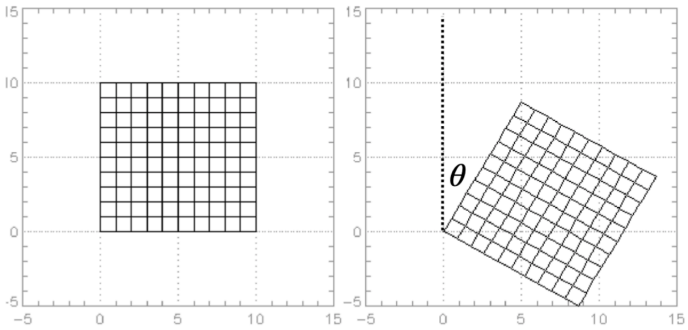


1.1.3 Rotation

Rotation is to rotate every pixel with a constant angel.

It can be shown as:

$$\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

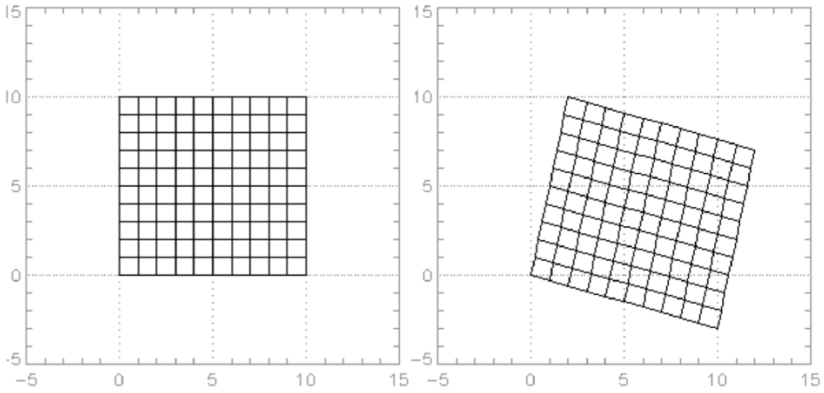


1.1.4 Scaling

$$\begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 x \\ s_2 y \\ 1 \end{bmatrix}$$

Shear

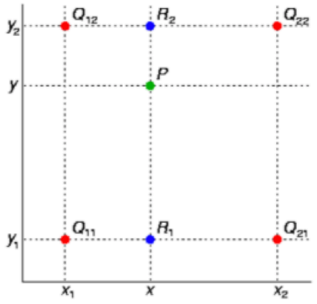
$$\begin{bmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$



1.1.5 Bilinear interpolation

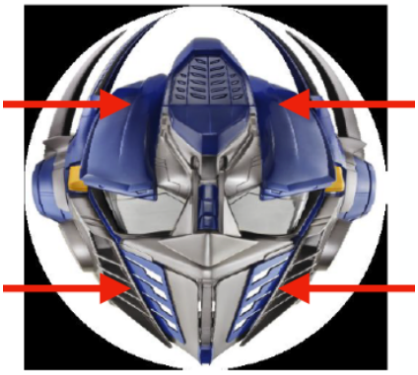
For every x y coordinates obtained from geometric modification, they may not be integers. Thus, we need to use bilinear interpolation to estimate the target value.

$$
\begin{aligned}
f(x,y) \approx & \frac{f(Q_{11})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y_2-y) \\
& + \frac{f(Q_{21})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y_2-y) \\
& + \frac{f(Q_{12})}{(x_2-x_1)(y_2-y_1)}(x_2-x)(y-y_1) \\
& + \frac{f(Q_{22})}{(x_2-x_1)(y_2-y_1)}(x-x_1)(y-y_1).
\end{aligned}
$$



1.2 Geometric warping

In this problem we will warp three images to disk images which is the same way given by the sample image. I will compress the image horizontally. Which means compress the width of image to the relevant chord.



1.2.1 Method

1.2.1.1 Warping

Step1: Find central point's cartesian coordinate $(x_{center}image, y_{center}image)$ of image;

Step2: Calculate the radius of the output disk.

Step3: Calculate every pixel mapping value in the output image based on the following method

(1) Justify if this pixel is in the disk, if not the value of the pixel is 0, else do the following

(2) Transfer the image coordinates$(x_{in}image, y_{in}image)$ of the pixel to cartesian coordinate$(x_{in}cartesian, y_{in}cartesian)$. Then calculate the length of horizontal chord where the pixel is lying on.

$chord = 2 * \sqrt{radius^2 - (y_{in}cartesian - y_{center}cartesian)\text{^}2}$

(3) Calculate the distance from the edge of disk to the pixel on chord

$$distance = x_{in}cartesian - x_{center}cartesian + \frac{chord}{2}$$

(4) Calculate mapping pixel image coordinate, L is equal to the length of input image.

$$x_{map}image = x_{in}image$$
$$y_{map}image = distance * L/chord$$

(5) Bilinear interpolation $(x_{map}image, y_{map}image)$
The value of each pixel is equal to $(x_{map}image, y_{map}image)$ correspond bilinear interpolation

1.2.1.2 Reverse

Step1: Find central point's cartesian coordinate $(x_{center}image, y_{center}image)$ of image;

Step2: Calculate the radius of the input disk.

Step3: Calculate every pixel mapping value in the output image based on the following method

(1) Transfer the image coordinates$(x_{in}image, y_{in}image)$ of the input image pixel to cartesian coordinate$(x_{in}cartesian, y_{in}cartesian)$. Then calculate the length of horizontal chord where the mapping pixel is lying on.
$chord = 2 * \sqrt{radius^2 - (y_{in}cartesian - y_{center}cartesian)\text{^}2}$

(2) Calculate the distance from the edge of disk to y-axis

$$distance = x_{center}cartesian - \frac{chord}{2}$$

(3) Calculate mapping pixel image coordinate, L is equal to the length of input image.

$$y_{map}cartesian = y_{in}cartesian$$
$$x_{map}cartesian = x_{in}cartesian * \frac{chord}{L} + distance$$

Change coordinates from cartesian $(x_{map}cartesian, y_{map}cartesian)$ to image$(x_{map}image, y_{map}image)$

(4) Bilinear interpolation $(x_{map}image, y_{map}image)$
The value of each pixel is equal to $(x_{map}image, y_{map}image)$ correspond bilinear interpolation

1.2.2 Results

Bb8

Bb8 disk

Bb8 reverse

Raccoon

Raccoon Disk

Raccoon reverse

Hedwig

Hedwig Disk

Hedwig reverse

1.2.3 Discussion and question answers

The reverse image preserves information from the original image well. This transformation is a one-to-one mapping from original image to disk image and from disk image to recovery image, which keep boundary pixels at boundary and keep the center of image still at center.

When comparing to the original image, the upper and lower parts of the reverse image become blurred. Since when making disk image, the horizontal chord of upper part and lower part of disk image is fairly short, which will lead to the loosen of information. So at the reverse image, the upper and lower parts of the reverse image become blurred.

1.3 Homographic Transformation and Image Stitching

Image stitching is useful to combine multiple images to creat a new panorama image.

1.3.1 Method

The homographic transformation uses two planes, real plane and projection plane. In this method we will use MATLAB build in function SURF to find out eight pairs of points to calculate transformation matrix H for mapping left to middle and mapping right to the middle. Then warping left and right image to the middle image plane to construct a panorama.

$$\begin{bmatrix} x_2' \\ y_2' \\ w_2' \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{x_2'}{w_2'} \\ \frac{y_2'}{w_2'} \end{bmatrix}$$

$$x_2 = \frac{H_{11} * x_1 + H_{12} * y1 + H_{13}}{H_{31} * x_1 + H_{32} * y_1 + H_{33}}$$
$$y_2 = \frac{H_{21} * x_1 + H_{22} * y1 + H_{23}}{H_{31} * x_1 + H_{32} * y_1 + H_{33}}$$

To simplify the calculation, the H_{33} is set to be 1.

In this algorithm we set plane of middle image to projection plane. And left or right images are in their own real plane. After the calculation of H matrix, each point (x_1, y_1) in projection plane has a corresponding (x_2, y_2) from real plane, then the value of (x_1, y_1) is equal to the interpolation result of (x_2, y_2).

Control points are generated by SURF and selected manually.

1.3.2 Result

1.3.2.1 Control points

Left and middle

Middle and right

1.3.2.2 Result

1.3.3 Discussion and question answers

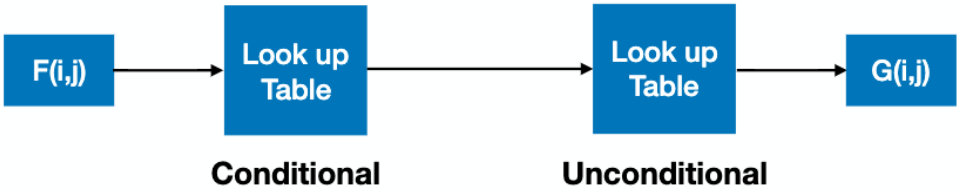
I have used four pairs of control points for each mapping. So, there are eight pairs of control points used in my program. The point chosen by me based on the spatial distribution of the points. The more distance between the four points, the better. The reason for the points chosen is that if the selected control points are far away from each other, the effect of errors in the selection of points may be improved

2 Morphological processing

There are three basic morphological processing method. Named shrinking, thining and skeletonizing.

2.1 Basic morphological process implementation

2.1.1 Method



There are two look up tables.

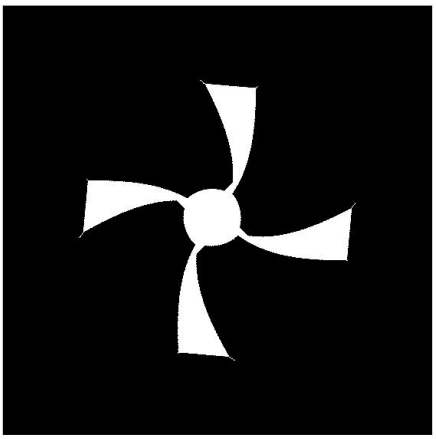
For conditional table, it is used to mark suspect boundary. If the center pixel and its l2 neighborhood is match with one of the tables from the conditional tables then the center pixel will be marked. It is called hit.

For unconditional table, it is used to determine if the marked pixel will be switched from 1 to 0. If hit then the pixel will be protected, if miss then the pixel will be switched to 0.

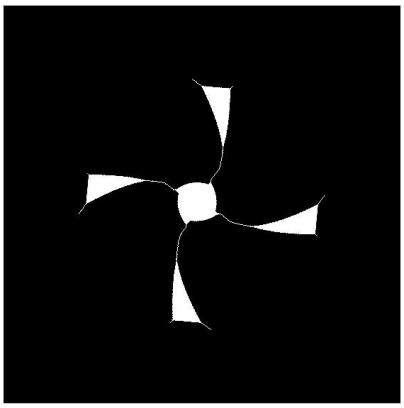
2.1.2 Result

2.1.2.1 Shrinking

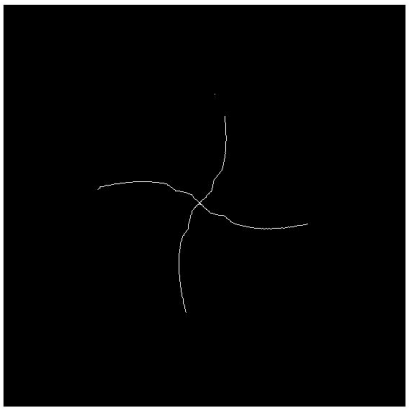
2.1.2.1.1 Fan



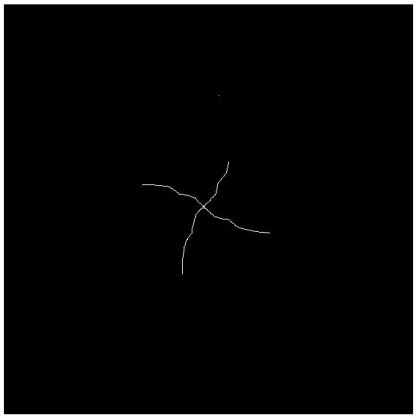
T = 10



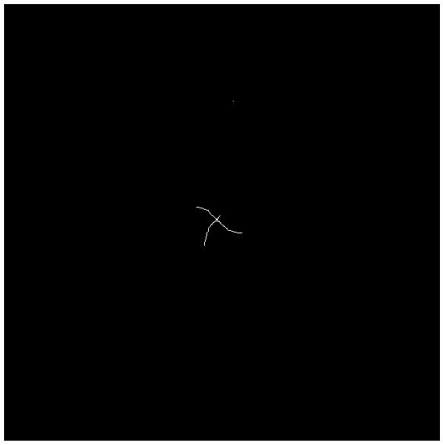
T = 20



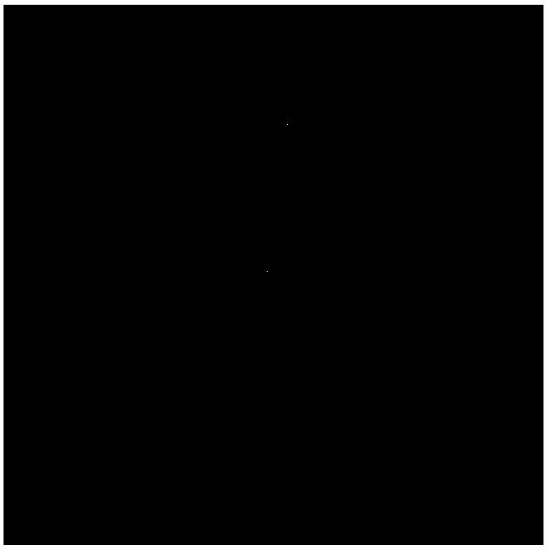
T = 60



T = 120

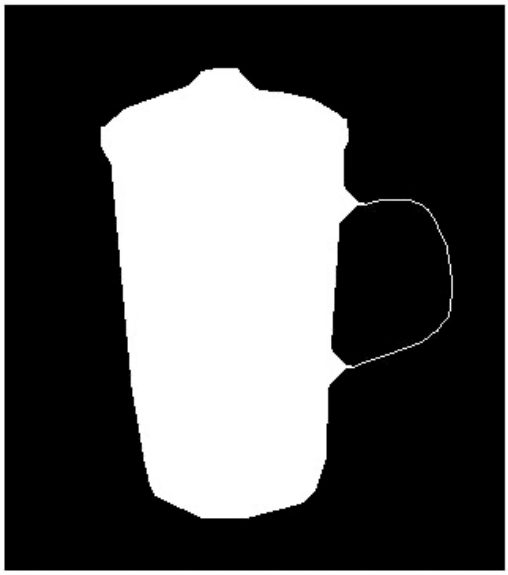


T = 180

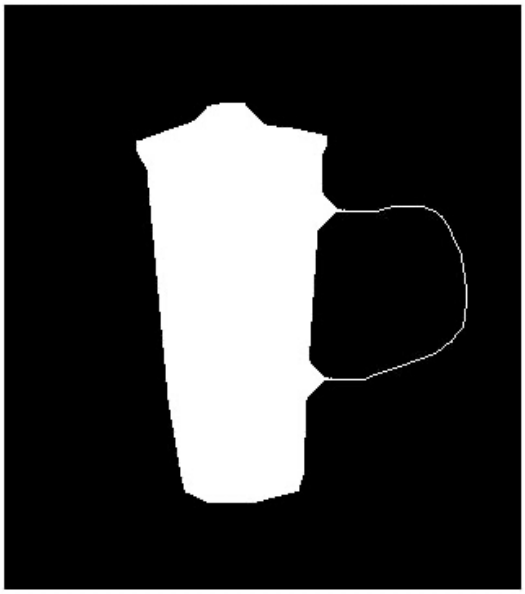


Final result

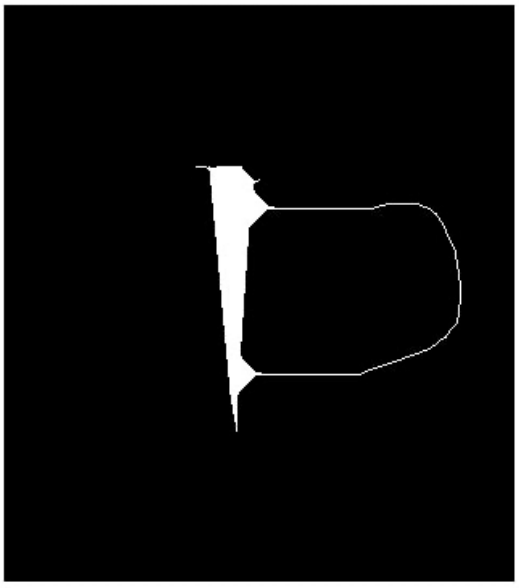
2.1.2.1.2 Cup



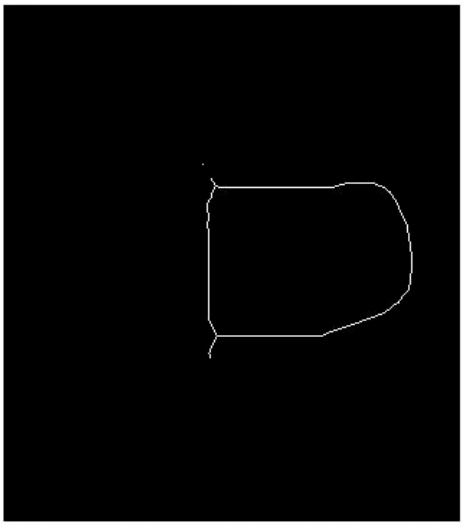
T = 10



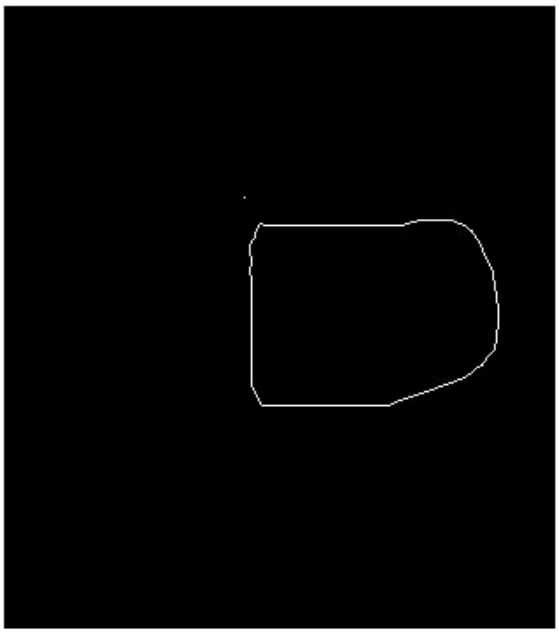
T = 30



T = 70



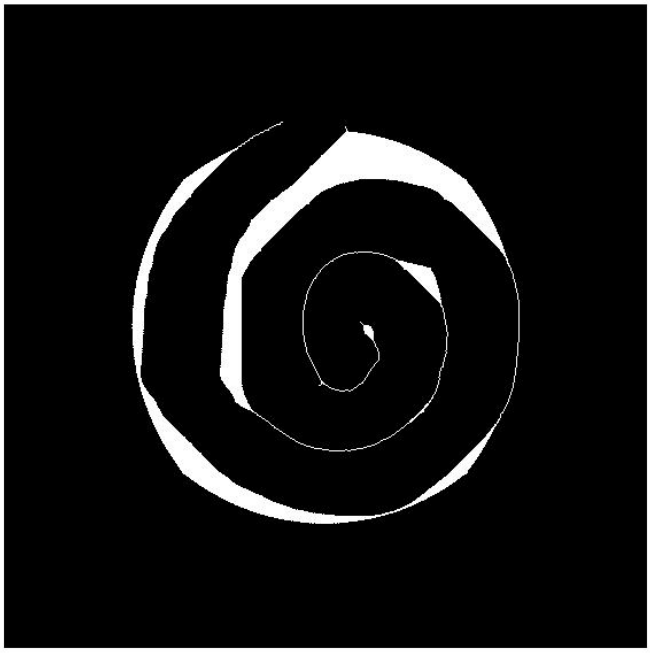
T = 90



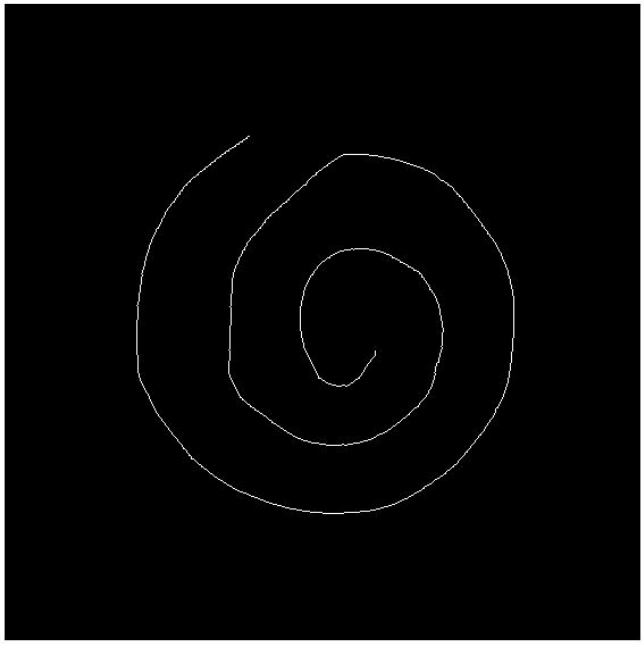
Cup Shrinking result

2.1.2.1.3 Maze

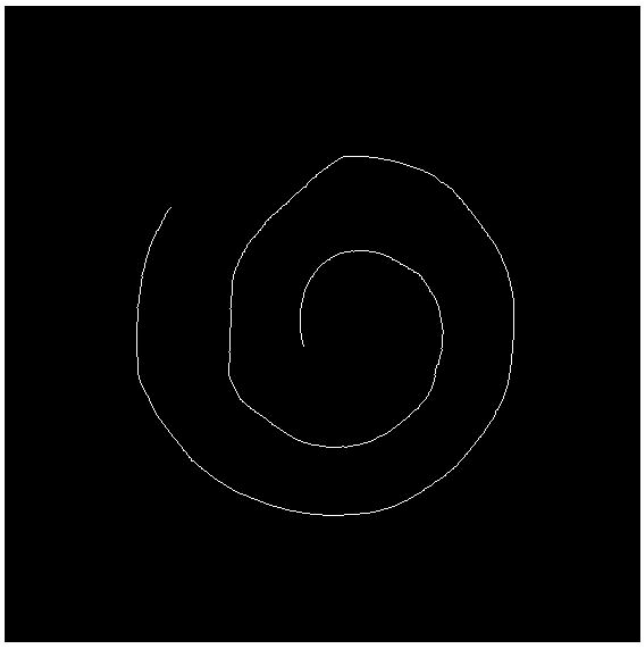
T = 10



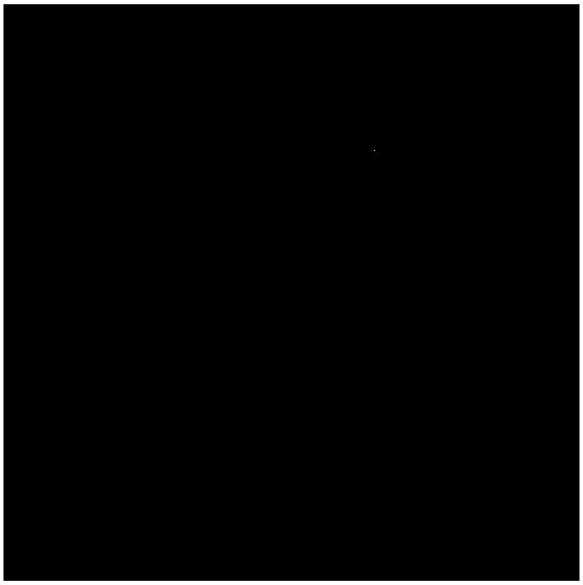
T = 30



T = 60



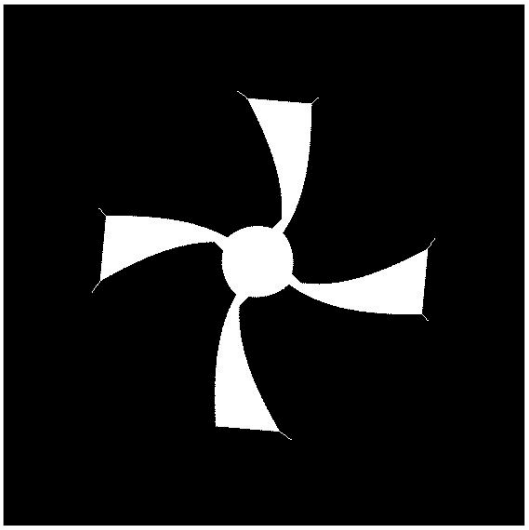
T = 200



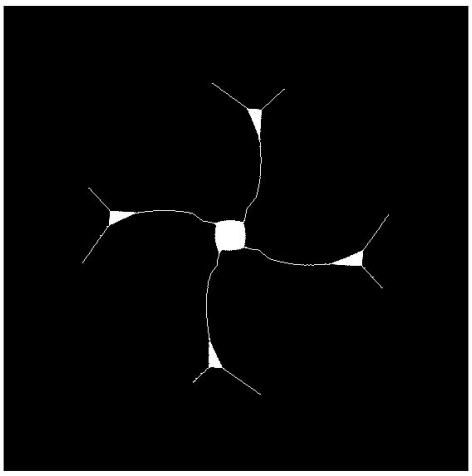
Maze shrinking result

2.1.2.2 Thinning

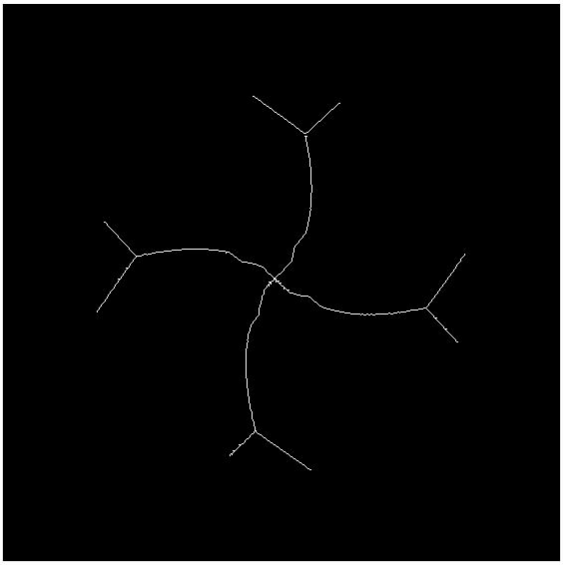
2.1.2.2.1 Fan



T = 10

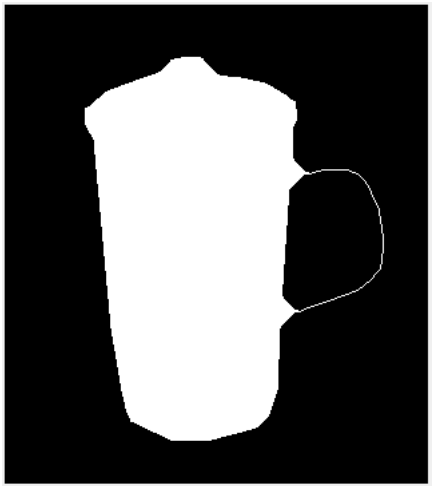


T = 30

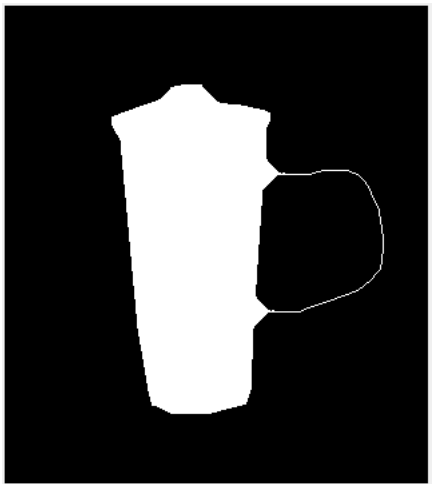


Fan thinning result

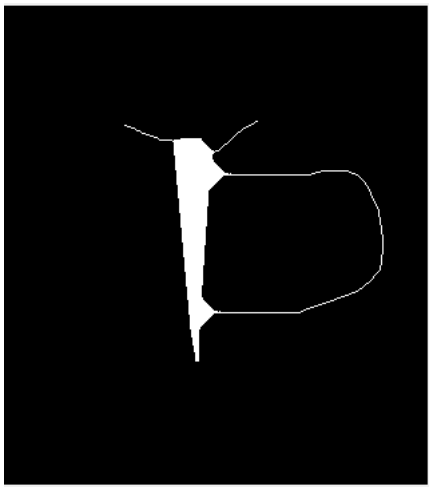
2.1.2.2.2 Cup



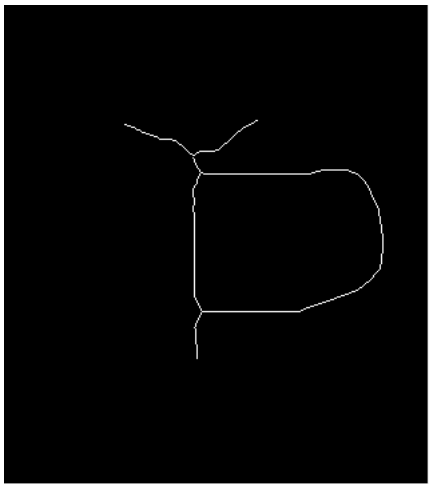
T = 10



T = 30



T=70

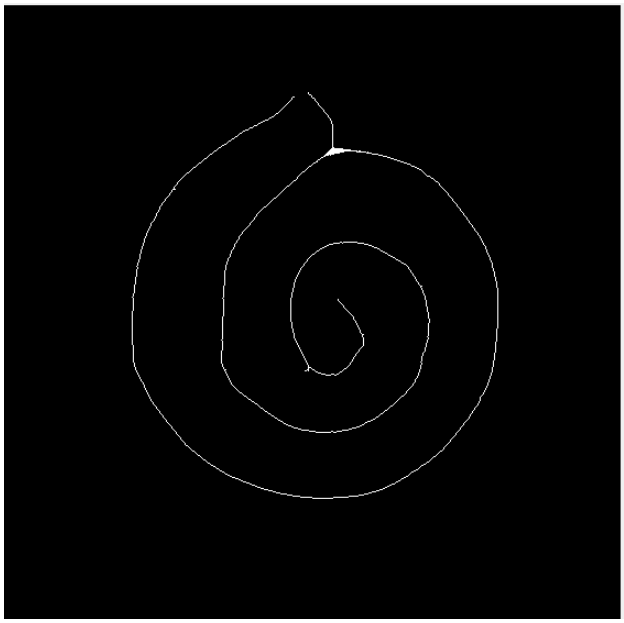


Cup thinning result

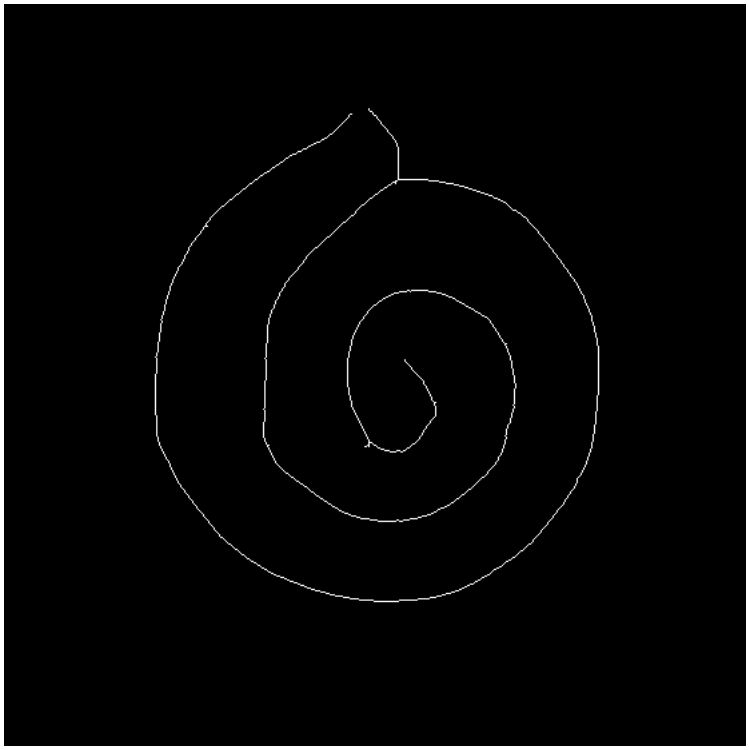
2.1.2.2.3 Maze

T = 10

T = 20



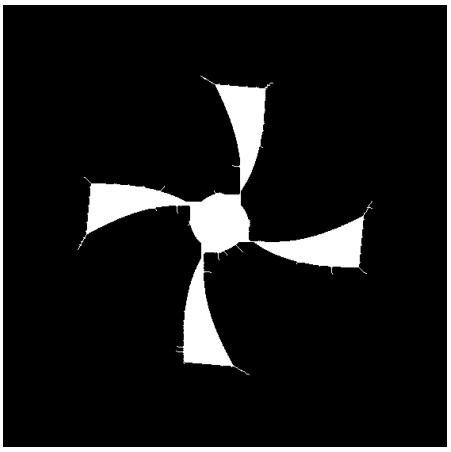
T = 50



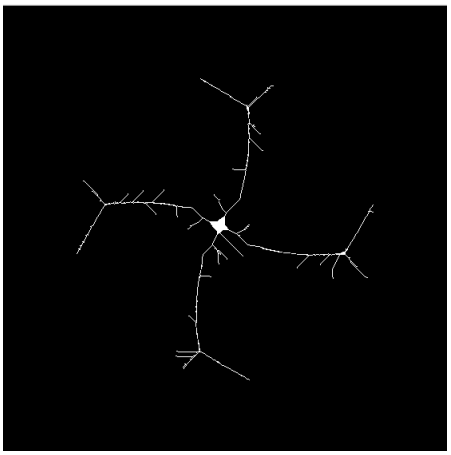
Result thinning maze

2.1.2.3 Skeletonizing

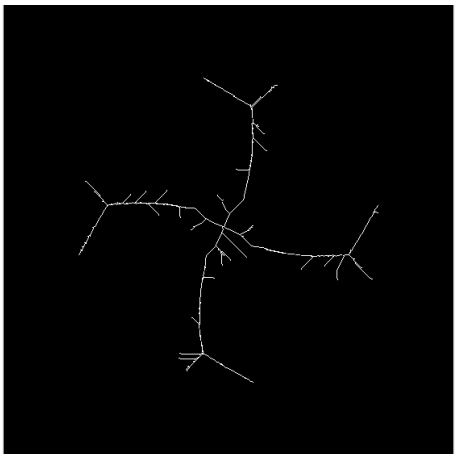
2.1.2.3.1 Fan



T = 10

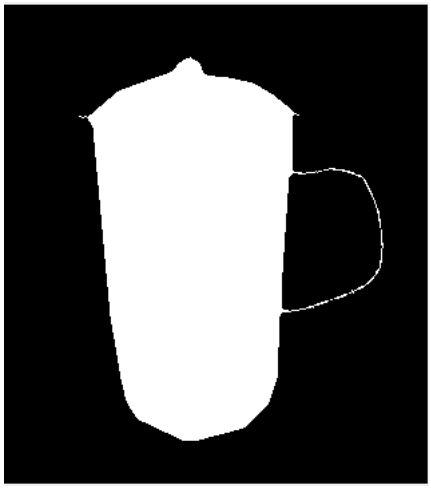


T = 30

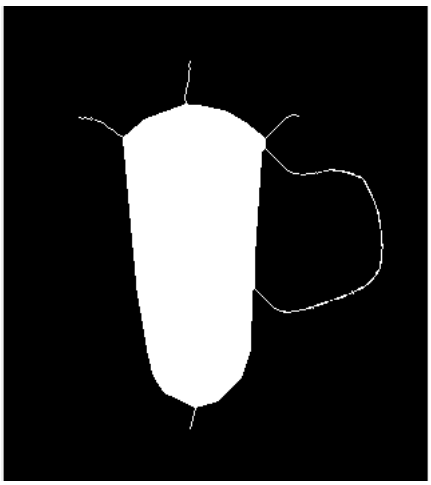


Fan skeletonizing result

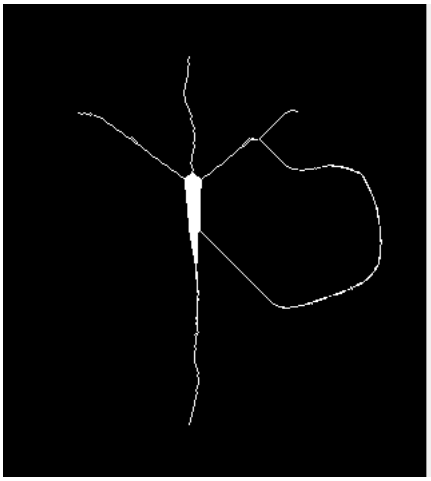
2.1.2.3.2 Cup



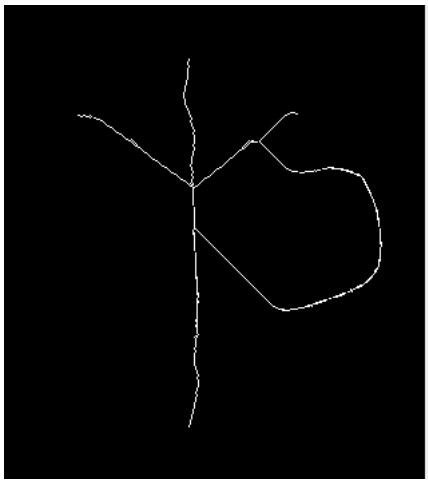
T = 10



T = 30



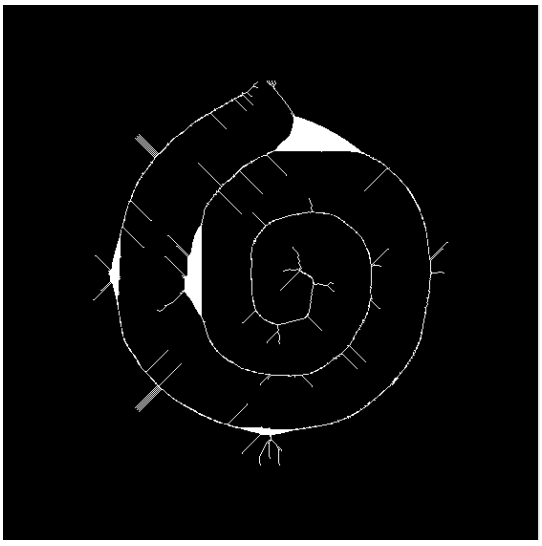
T = 70



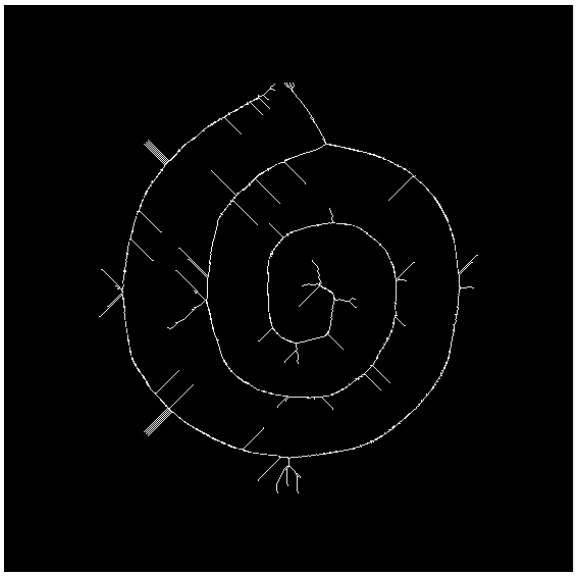
Cup skeletonizing result

2.1.2.3.3 Maze

T = 10



T =30



Maze skeletonizing result

2.1.3 Discussion and question answers

2.1.3.1 Shrinking

The table provided has some errors, which leads to image separated in to two points. For example, cup and fan.

Shrinking will eliminate edge pixels until a point. For example, the image of fan is shrunk to be two points and maze is shrunk to become a point. But if there is a hole in the original target, then shrinking will make a circle around the hole. Like the cup image, the handle of cup is preserved. It is useful for locate the target.

2.1.3.2 Thinning

Thinning is useful for preserving skeletonization, in the image of Fan, Cup and maze the skeleton is shown by smooth curves. This method is useful for understanding target's connectivity of the original target region.

2.1.3.3 Skeletonizing

On the one hand it shows the skeleton of original target, on the other hand it could also reflect contour change. It is useful for keeping the extent of target and the connection of region.

2.2 Counting games

In this problem I will implement three different method. First using the shrinking program in problem 2(a), second using the build in shrinking function and the last using deep first searching to find out stars.

2.2.1 Method

2.2.1.1 Method 1 using shrinking program from 2(a)

In this method, the original image is shrunk once in each step with shrinking program from 2(a). The threshold for "stars.raw" is 40. After each shrinking calculate the number of isolated points in image, save the number then set all isolate points to 0. (the isolated point is defined as the point has all 0 values of second order pixels). Then the size of stars in defined as the steps of shrinking to eliminate each star.

2.2.1.2 Method 2 using build in function.

Same method with 2.2.1.1, but change shrinking function to build in function.

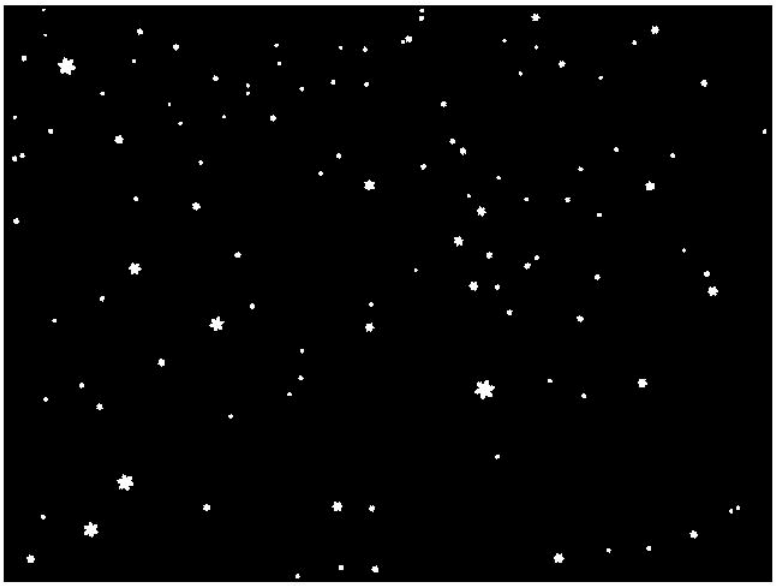
2.2.1.3 Method 3 Deep First Searching

Using two loops to find stars in original image. If original_image(x,y)=1 then using dfs to mark neighborhood pixels. The DFS algorithm is shown below

```
value = dfs(x,y)
    sum = 0;
    image(x,y) = 0;
    sum = sum+1;
   for i = 1 to 8
      if image(xi,yi) = 1 //(xi,yi) is eight second order neighbor points of (x,y)
          sum = sum+dfs(x1,y1);
       end
   end
   value = sum
```

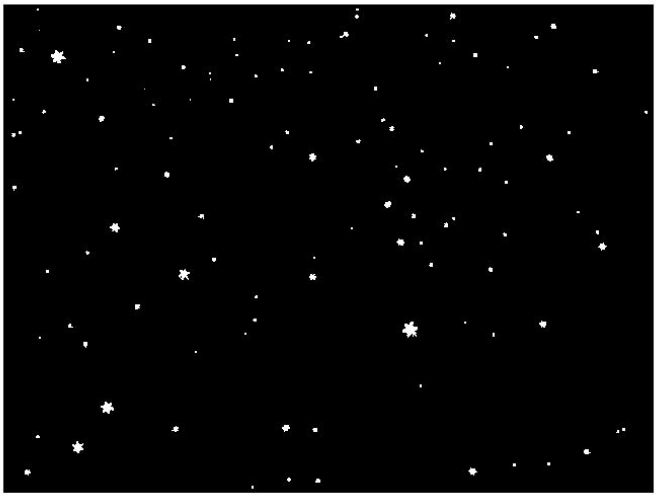
In each star storage pixels' location then calculate maximum l2 distance. The classification of star is equal to the round l2 distance.

2.2.2 Result



Stars.raw

2.2.2.1 Method 1 Shrinking program from 2(a)



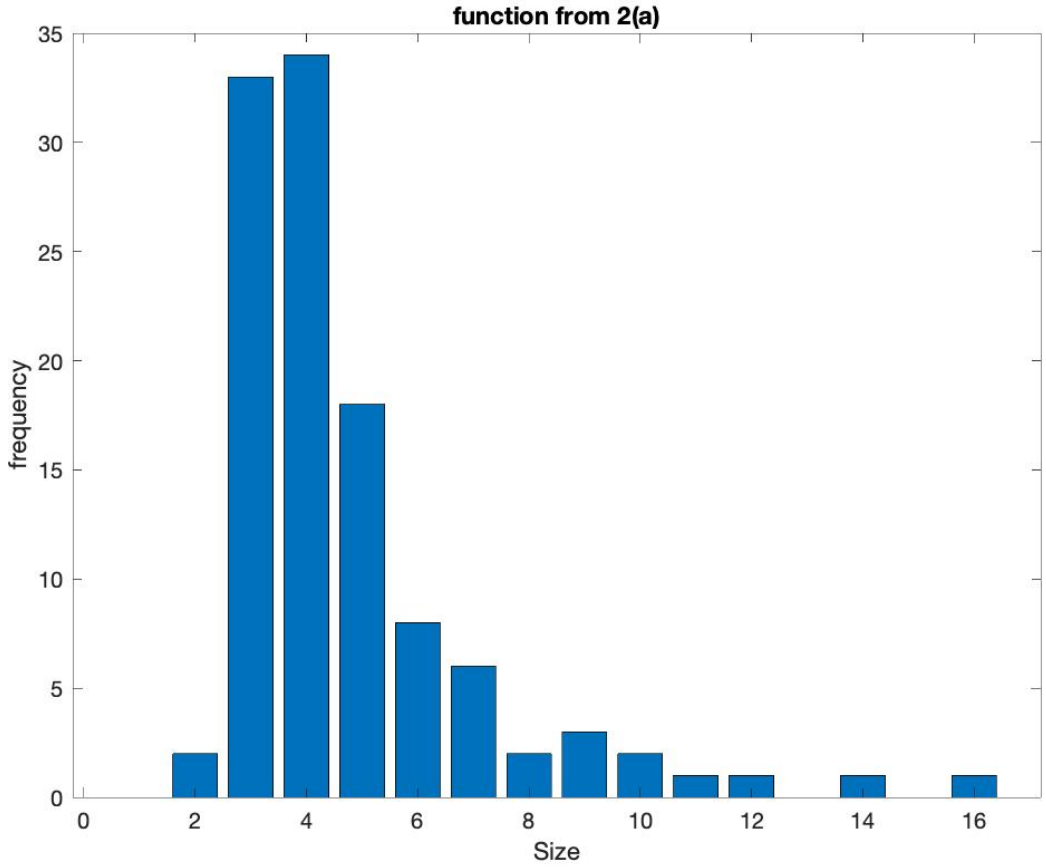
First step

Third step

sixth step

Ninth step

fifteenth step



Size	1	2	3	4	5	6	7	8
Frequency	0	2	33	34	18	8	6	2
Size	9	10	11	12	13	14	15	16
Frequency	3	2	1	1	0	1	0	1

The total number of stars is 112, the total levels is 16

2.2.2.2 Method 2 Build in function (bwmorph)
There are 5 levels

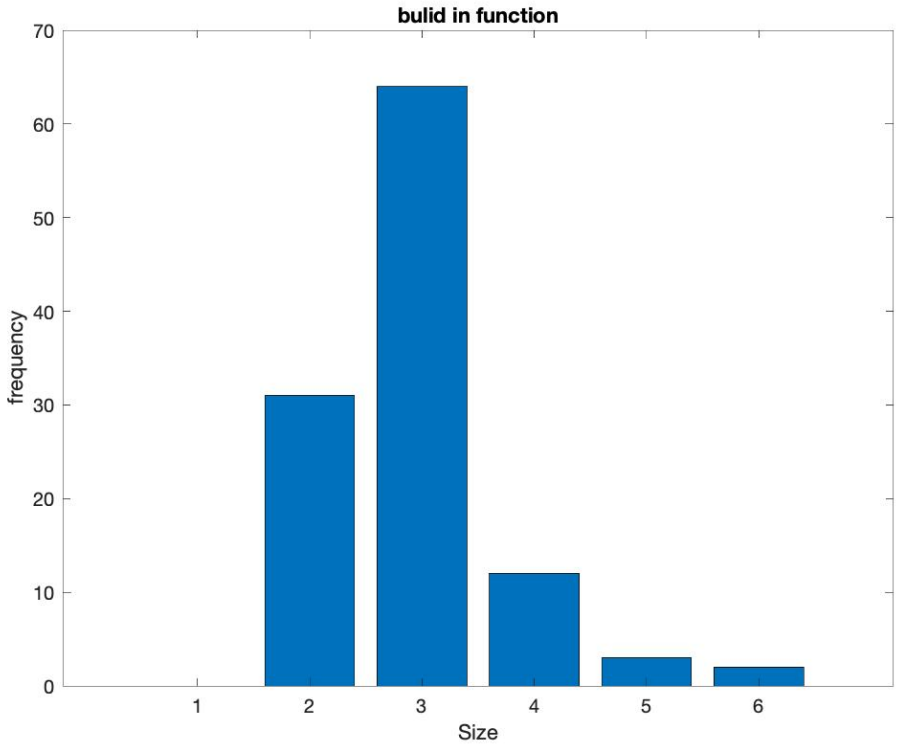
First step

Second step

Third step

Fourth step

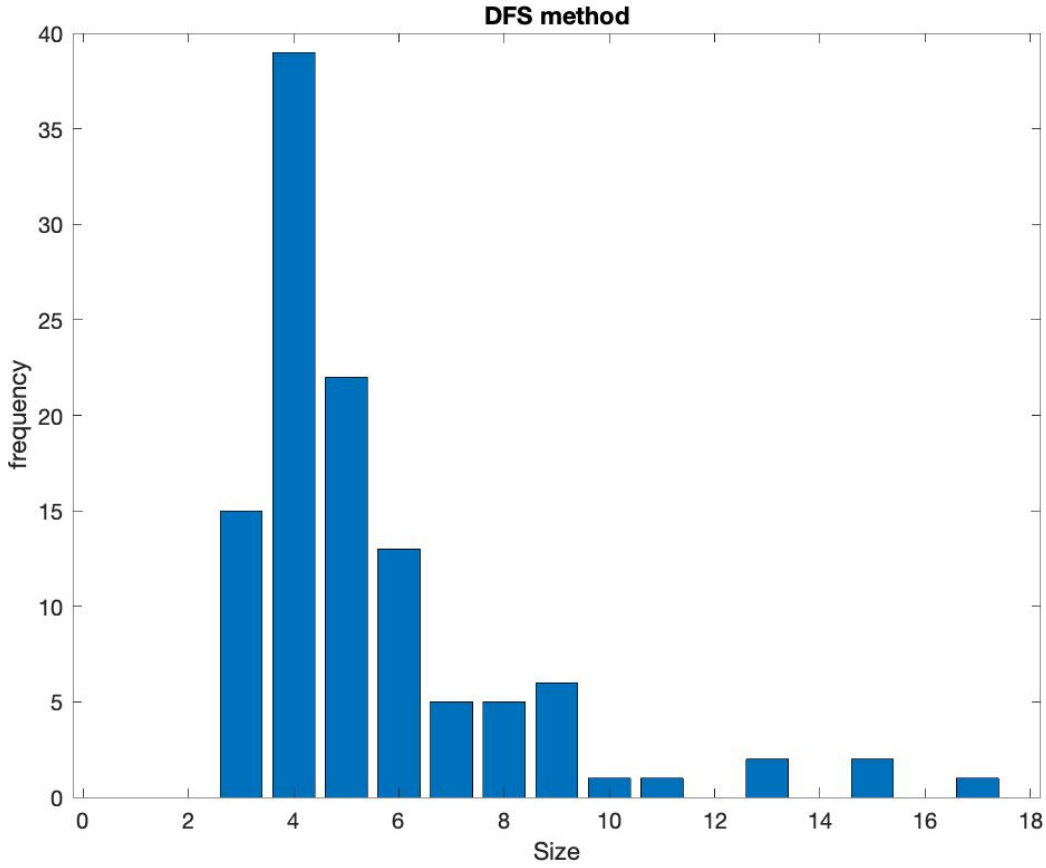
Final step



Size	1	2	3	4	5	6
frequency	0	31	64	12	3	2

The total number of stars is 112, the total levels is 6

2.2.2.3 Method 3 DFS method



The size of stars is equal to round up(max l2 distance)

Size	1	2	3	4	5	6	7	8	9
Frequency	0	0	15	39	22	13	5	5	6
Size	10	11	12	13	14	15	16	17	
Frequency	1	1	0	2	0	2	0	1	

The number of stars is 112, The total levels is 17

2.2.3 Discussion and question answers

Combine the results above, there are 112 stars in image and different algorithm has a different classification of stars' level. According to the program of 2(a) there are 13 different levels and 5 for the build in function 32 for DFS. Because the build in function has a better shrinking behavior, so the it has fewer levels comparing to the other two method. And for function from 2(a), since shrinking is not based on the l2 distance so it has smaller amount of levels than DFS method. DFS method can calculate the exact maximum l2 distance. These three methods have the same functional trend.

2.3 PCB analysis

2.3.1 Assumption

Basic assumption: 8 black blocks at left and right edge of image will not be calculated in to pathway.

A pathway is a connection of holes where current can flow. Which means in the figure above there is only one pathway.

2.3.2 Method

Step 1: remove image black border (image 1)

Step 2: Using DFS algorithm and change background to black

This algorithm is the same as 2(b). Using DFS start from (1,1) to switch all connected background to black. (image 2)

Step 3: Shrink and calculate the number of holes

Step 4: The value of pixels in image three will equal to pixel values from image1 subtract that from image 2

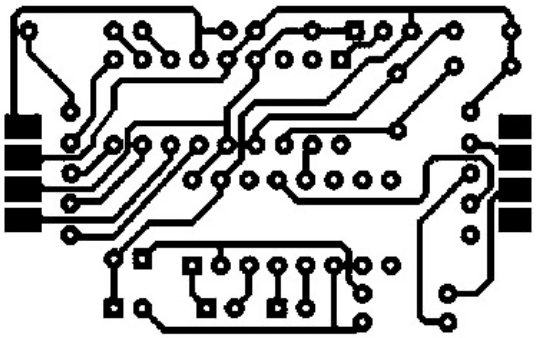
Step 5: Using DFS to eliminate isolated holes.

The DFS is modified to storage every removed pixel location. If the number of pixels in cluster is larger than a threshold, use the recover location to recovery pixels. If the number of pixels in cluster is lower than threshold then this is the hole need to be removed.

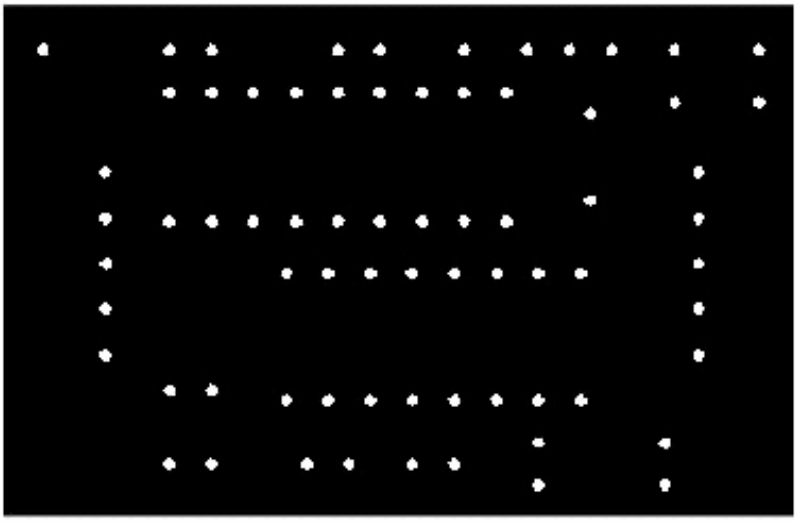
Step 6: Shrink and calculate the number of pathways

2.3.3 Result

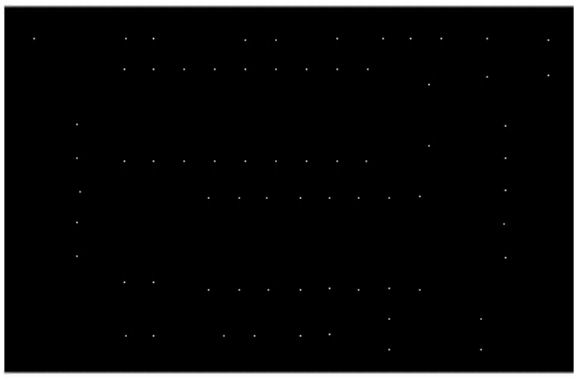
Step 1: remove image black border (image 1)



Step 2: Using DFS algorithm to change background to black

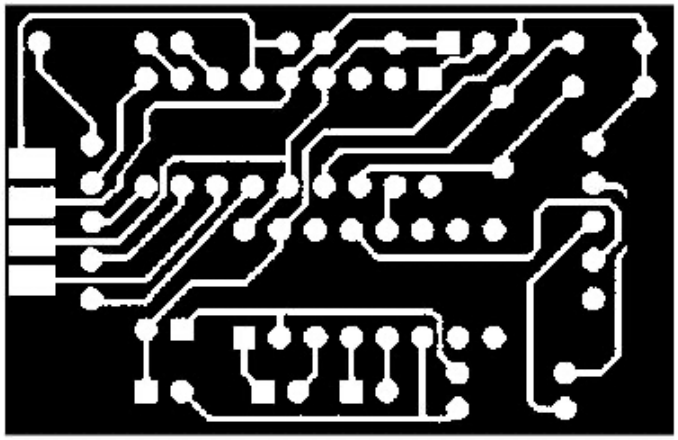


Step 3: Shrink and calculate the number of holes

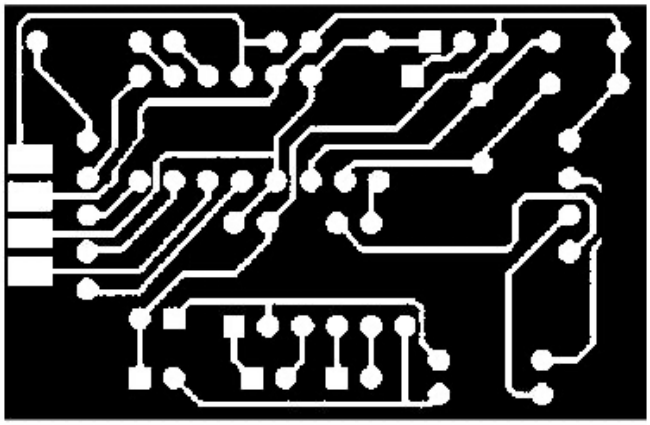


The total number of holes is 71

Step 4: The value of pixels in image three will equal to pixel values from image1 subtract that from image 2. Remove right four black blocks and switch black and white.



Step 5: using DFS algorithm to eliminate isolate holes



Step 6: Switch black and white of image then shrink and calculate the number of pathways

There are 25 pathways.

2.3.4 Discussion and question answers

In this problem threshold will influence the result. So when reading raw image, I chosen 50 as the threshold. As the result there are 71 holes and 25 pathways.

2.4 Defect detection

2.4.1 Method

Step 1: find the center of the four black circle holes. Then calculate the center of this gear.

Step 2: Rotate the gear 90 clockwise.

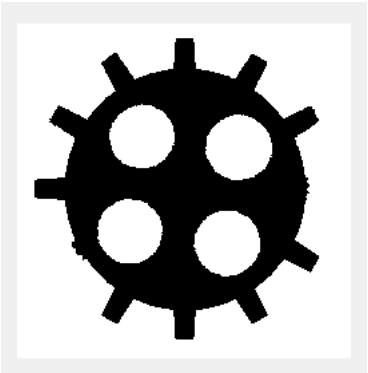
Step 3: Find pixels equal to 1 in step2 but equal to 0 in original image

Step 4: Eliminate pixels other than gear teeth using the modified DFS which is same in 2(c).

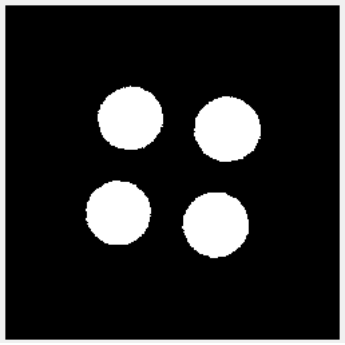
2.4.2 Result

Step 1: find the center of the four black circle holes. Then calculate the center of this gear.

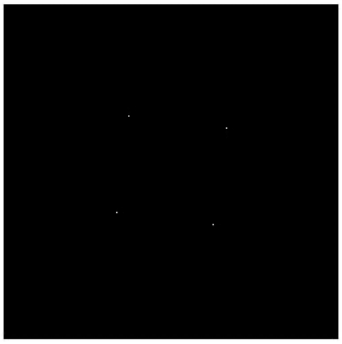
At first reverse image color



Second using DFS to mark background as black

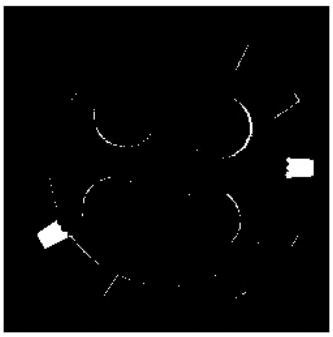


Then do shrinking and find out the image center



Step 2: Rotate the gear 90 clockwise.

Step 3: Find pixels equal to 1 in step2 but equal to 0 in original image



Step 4: Eliminate pixels other than gear teeth using the modified DFS which is same in 2(c).

Show the teeth in the original image