EE569- Introduction to Digital Image Processing

Homework #3

Sourabh J. Tirodkar
3589406164
Submission Date- 3rd March 2020

Problem 1: Geometric Image Modification

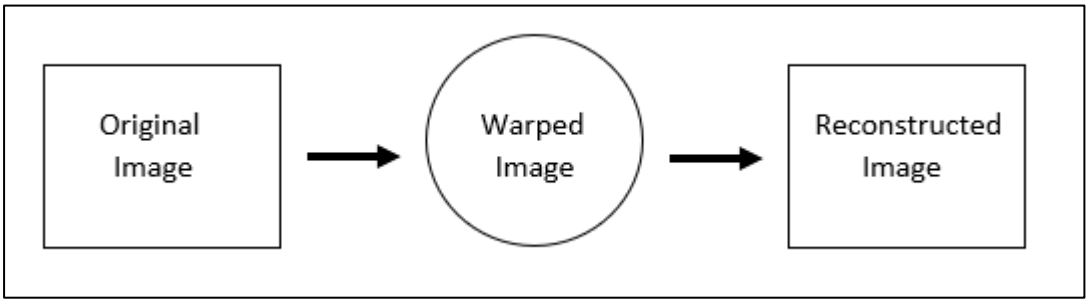
a) Geometric Warping

1. ABSTRACT AND MOTIVATION

   Geometric Image Modification has wide applications which include computer graphics, modification, etc. The operations that are performed are based on one's personal needs. One might want to tilt the image, other might want to zoom in- out or else cropping image. These operations mostly are used as pre-processing steps for much higher order complexity operations. There need to be followed a fix procedure for doing the same.

2. APPROACH

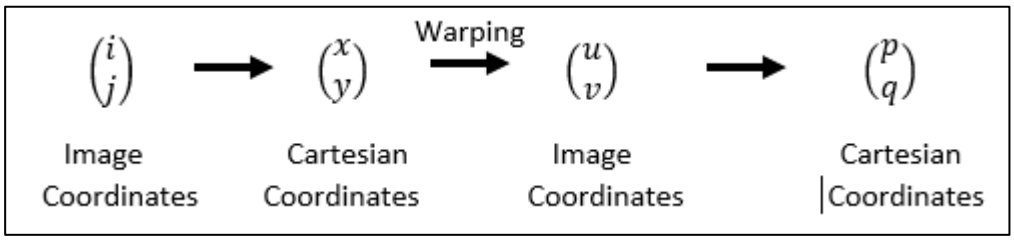
   Warping is mostly done for better visualization of an image. The process followed here is going from Square Image to Circle and then back to Square.



   We need to satisfy these requirements while warping:

   1. Pixels that lie on boundaries of square must lie on boundary of circle.
   2. Center of original image to be warped to center of new image.
   3. Mapping should be reversible.

The basic steps in Geometric Image Modification is to convert image coordinates to cartesian coordinates.



Relation between Image coordinates to Cartesian coordinates are as follows:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & J+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \qquad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & Q+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} p \\ q \\ 1 \end{bmatrix}$$

We do warp operation on cartesian coordinates which maps square image to circle image. We use inverse address mapping. As forward address mapping creates lot of artifacts with black pixels in the image. Inverse address mapping does not have any artifacts and hence used for mapping.

At the recovering step, we need to do bilinear interpolation to remove any unnecessary artifacts if present.

Mapping Procedure:

Points in the square image should be in the range -1 to 1.
The formula derived are:

SQUARE TO CIRCLE: u= $x * \sqrt{1 - \frac{y\text{^}2}{2}}$ v= $y * \sqrt{1 - \frac{x\text{^}2}{2}}$

CIRCLE TO SQUARE:

$$x = \frac{1}{2}\sqrt{2 + 2\sqrt{2}\,u + u^2 - v^2} - \frac{1}{2}\sqrt{2 - 2\sqrt{2}\,u + u^2 - v^2}$$

$$y = \frac{1}{2}\sqrt{2 + 2\sqrt{2}\,v - u^2 + v^2} - \frac{1}{2}\sqrt{2 - 2\sqrt{2}\,v - u^2 + v^2}$$

(u,v) are the coordinates of the circle.
(x,y) are the coordinates of the square.

As mapping from circle to square gives the complex values and MATLAB cannot process that term for an image, I need to add the correction term, which is high enough to not make it complex. I added 99999 in each of the square roots.

3. RESULTS

Fig 1.1 Warped Image of 'hedwig.raw'

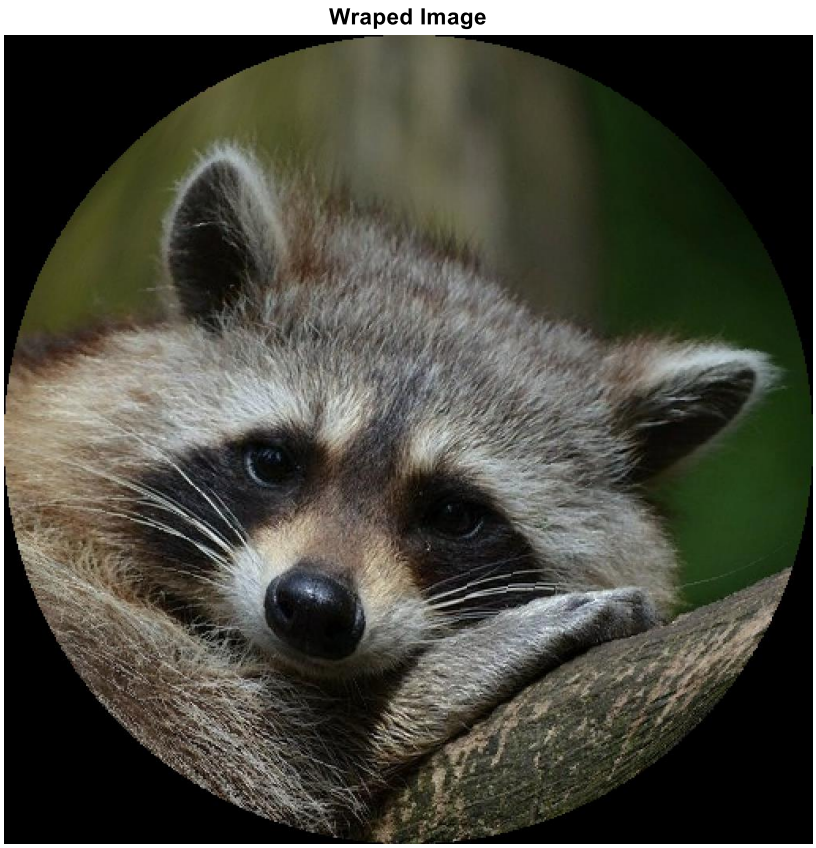


Fig 1.2 Warped Image of 'raccoon.raw'

Wraped Image

Fig 1.3 Warped Image of ‘bb8.raw’

Fig 1.4 Recovered Image of ‘hedwig.raw’

Fig 1.5 Recovered Image of ‘raccoon.raw’

Fig 1.6 Recovered Image of 'bb8.raw'

In 'hedwig.raw', we can see that the there are some pixels missing near the peak of the hedwig.

In 'raccoon.raw', the hair of the animal has got blurred and not clearly prominent as of input image.

In 'bb8.raw', the recovered image has some discontinuity and so has artifacts.

4. DISCUSSION
    a. The mapping approach has been explained with the formulas for square to circle mapping. This mapping techniques does satisfy the 3 requirements which are stated above.
    b. This following algorithm is applied to warped image to recover the square image. As the mapping is reversible, it is possible for us to recover back the original image. If we follow the forward mapping, we can see the black dots all over the image. So, this image has some artifacts, we need to take care of those. To remove these Bilinear Transformation needs to be applied. If we do inverse reverse mapping, there are lesser artifacts which are not clearly seen by naked eye. After doing BT, we still get black borders. We can also use round function which rounds to nearest value of the integer.
    c. When comparing the recovered image with the original image, we do observe the artifacts in the recovered image. This artifacts can be removed by doing bilinear interpolation or using round function in MATLAB.

The source of artifacts are from the fact that we are using forward mapping. This creates black dots as there less points in circle to map in square.
If we follow inverse mapping, we do not get these black dots.
These black dots can be removed by doing Bilinear Interpolation.

b) Homographic Transformation and Image Stitching

1. ABSTRACT AND MOTIVATION

Homographic Transformation is used for the purpose of image stitching which helps to generate the panorama image. This method is very useful as one can get a broader view of an image just by implementing homographic transformation. Recording video is an alternative but it does cost more money and storage. So, merging a set of images to see a virtual world in broader sense is better. Panorama is basically a 360^0 virtual image.

2. APPROACH

The images used for generating the panorama are taking by uncalibrated camera sensors from sweeping left to right along the same plane. This way all the information is captured in a set of images.
These images have features in common. Our aim is to match these features and overlap thus making a panorama image.
Here, we have 3 set of images named- *left, middle* and *right.*

The steps followed in doing image stitching are as follows:
1. Feature Matching using SURF/ SIFT (OpenCV)
2. Match Features- Making pairs
3. Transformation matrix T
4. Warping onto one another
5. Interpolation

I have used open source code for feature detection which is using SURF which is acronym for Speeded Up Robust Features. We make pairs of the matched features.

Transformation Matrix:

$$\begin{bmatrix} h11 & h12 & h13 \\ h21 & h22 & h23 \\ h31 & h32 & 1 \end{bmatrix} * \begin{bmatrix} x1 \\ y1 \\ 1 \end{bmatrix} = \begin{bmatrix} x1^- \\ y1^- \\ w1^- \end{bmatrix} \dots\dots\dots\dots\dots\dots\dots\dots eq(1)$$

$$x1' = \frac{x1^-}{w1^-} \;\&\; y1' = \frac{y1^-}{w1^-} \quad \dots\dots\dots\dots\dots\dots\dots\dots eq(2)$$

$x1 \; and \; y1 \; are \; the \; pixels \; in \; left \; image$

$x1' and \; y1' are \; the \; pixels \; in \; middle \; image$

We have 8 variables to be found, writing the equations:

$h11 * x1 + h12 * y1 + h13 = x1^-$

$h21 * x1 + h22 * y1 + h23 = y1^-$

$$h31 * x1 + h32 * y1 + 1 = w1^{-}$$

$$Substituting\ eq2\ into\ above\ equations\ we\ get,$$

$$h11 * x1 + h12 * y1 + h13 = x1' * (h31 * x1 + h32 * y1 + 1)$$

$$h21 * x1 + h22 * y1 + h23 = y1' * (h31 * x1 + h32 * y1 + 1)$$

$$Using\ solve\ function\ in\ MATLAB, we\ get\ these\ variables.$$

We do the same for middle image and right image to get T matrix. The next I followed is making a canvas large enough to accommodate all 3 images side by side. Using feature matching, these images are then warped onto each other where the points are matched.

3. RESULTS

Fig 1.10 Feature matching between left and middle image

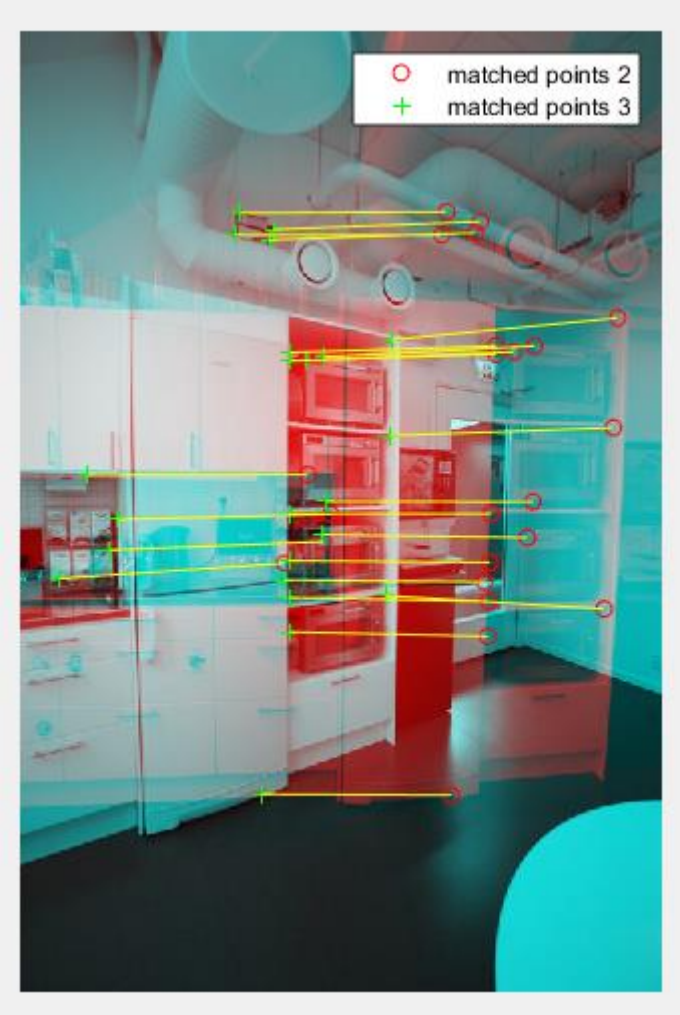


Fig 1.11 Feature matching between middle and right image

Fig 1.12 Left , middle and right image placed for warping onto each other

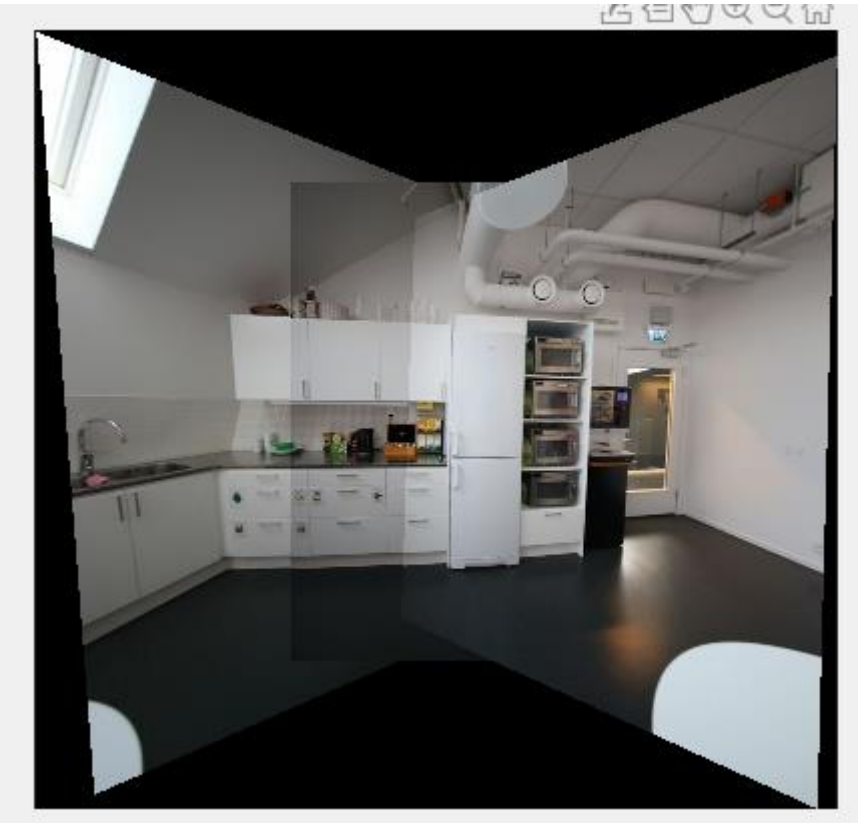


Fig 1.13 Final Panorama

4. DISCUSSION
   a. Fig 1.10 shows the feature matching between left and middle image while fig 1.11 shows the feature matching between middle and right image.
      As told, it is better to take 4 control points that are placed far away from each other. This gives better result and more area is covered.
      I have selected 4 control points for the implementation.

   b. 'matchFeatures' command in MATLAB gives all the feature matched between the two images. The plot does give the data point of the feature matched. I manually selected the points which are father apart for better result.
      SURF or Speeded Up Robust Features is a process which is done in 3 steps:
      Feature Extraction, Feature description and Feature matching.
      I created a large size image, placed all the 3 images in it and after feature matching between images, combined them on those points.

Problem 2: Morphological Processing

a) Basic Morphological process implementation

1. ABSTRACT AND MOTIVATION

   Morphological process includes operations like Shrinking, Thinning and Skeletonizing. Morphological image processing pursues the goals of removing the imperfections of binary image by accounting for the form and structure of the image. It is a nonlinear operation where we use structuring element to be applied on the original binary image.

2. APPROACH

   Connectivity:

   We use concept of connectivity in morphological processing. There are basically 2 types of processing: 4 connectivity and 8 connectivity.

   Consider,

   X3 X2 X1
   X4 X X0
   X5 X6 X7

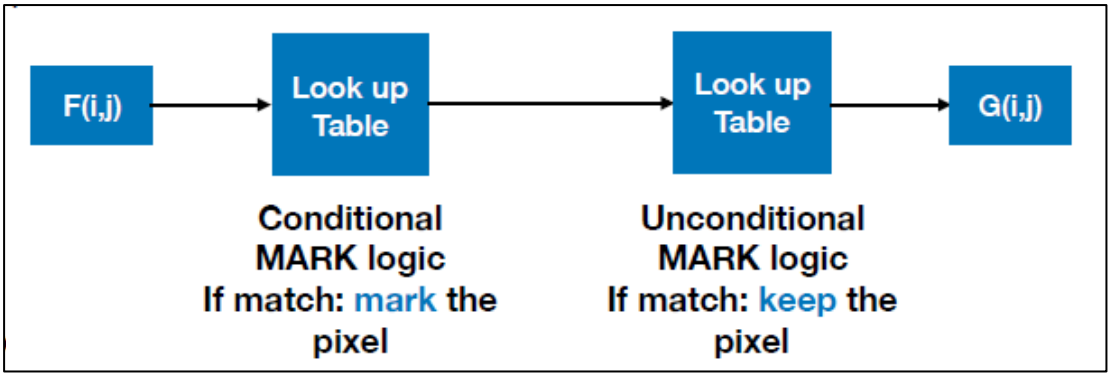
   Here, X is the center pixel which can have value 0 or 1. There are 8 neighbours to X which are X0 to X7. Now, if we take X0, X2, X4 and X6 all equal to 1, it is known as 4 connectivity.
   If all 7 neighbours are 1 then it is said to be 8 connectivity.

   HIT and MISS Transform:

   As described, we use structuring element (mask pattern) in implementing morphological operations, the concept of HIT and MISS transform becomes very important.
   HIT: If all the neighbours of image match with that of structuring element then it is considered as hit. In that case we take action, i.e. if pixel value is 0 then change to 1 and vice versa.
   MISS: If any one of the 8 neighbours do not match with the image then it is MISS. We do not take any action in this case. We retain the center pixel value to the output image.

   Definitions:

   1. Shrinking
      It is the process where the image is erode, the foreground (Object) reduces to a single pixel of value 1. Rest all the image is made 0.
   2. Thinning
      Thinning is the process where the foreground pixels from binary image are reduced to minimal connected ring between its hole and nearest outer boundary.
   3. Skeletonization
      As name suggests, it is process where the skeleton of the image is preserved sending all other data to 0. The edges and connectivity of the image is preserved looking like a skeleton of an image.



Here, F(i,j) is the input binary image. I have designed 2 look up tables, one for Conditional and other for Unconditional. The conditional lookup table consists of 3 subparts: Shrinking Conditional, Thinning Conditional and Skeletonizing Conditional. Likewise, Unconditional has Shrinking Conditional, Thinning Conditional and Skeletonizing Conditional tables.

The process is to generate M map from the conditional look up table. The process of M map is such that if there is match with the structuring element with image, here conditional table and image, we mark the pixel.

The next step is to process these data into Unconditional look up table. Here, if we compare with Unconditional look up table, if the pixel matches with the look up table, then we preserve the pixel of original image. If not match, then we erase those pixel (make 0). This process is carried out until the flag is false. The process followed in both look up table is HIT or MISS transform.

Steps:

1. Boundary extension (padding with 0s)
2. Starting 2^{nd} row and 2^{nd} col, we check the pixel with value=1, if found we check its 8 neighbour and compare with respective lookup table. If HIT, then store M=1 else 0
3. The M matrix is feed into respective unconditional look up table. The same HIT and MISS algorithm is applied.
4. Compare input image matrix with 1^{st} iteration output. If it is not same, then process is carried out till the flag value is true.
5. Output corresponding to S, T or K.

3. RESULTS

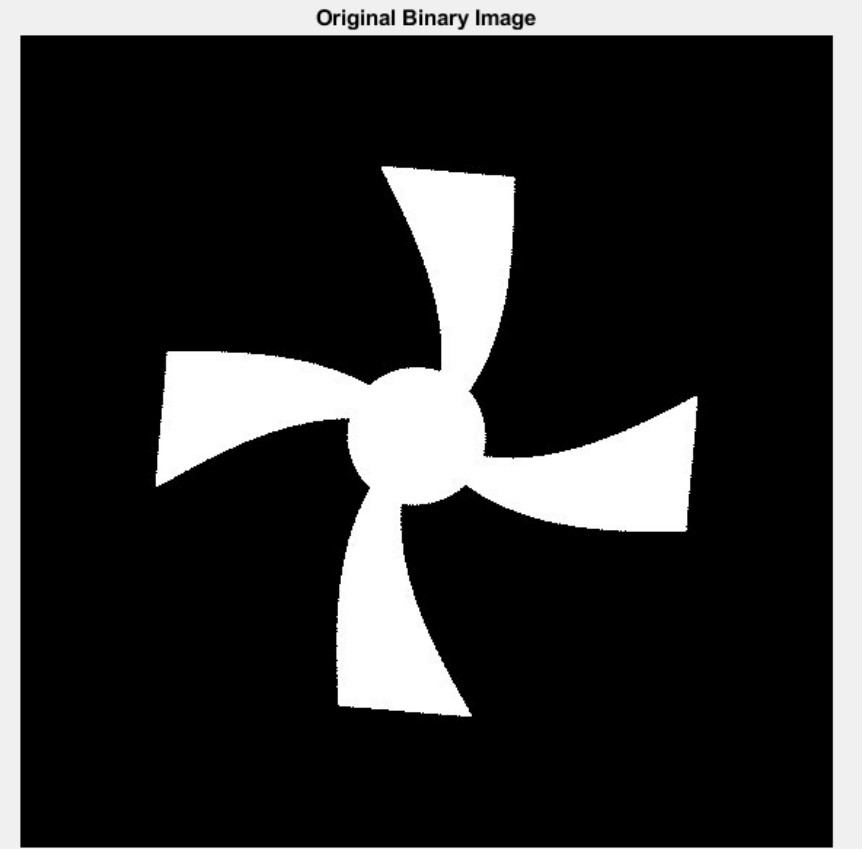


Fig 2.1 ‘Fan.raw’ image

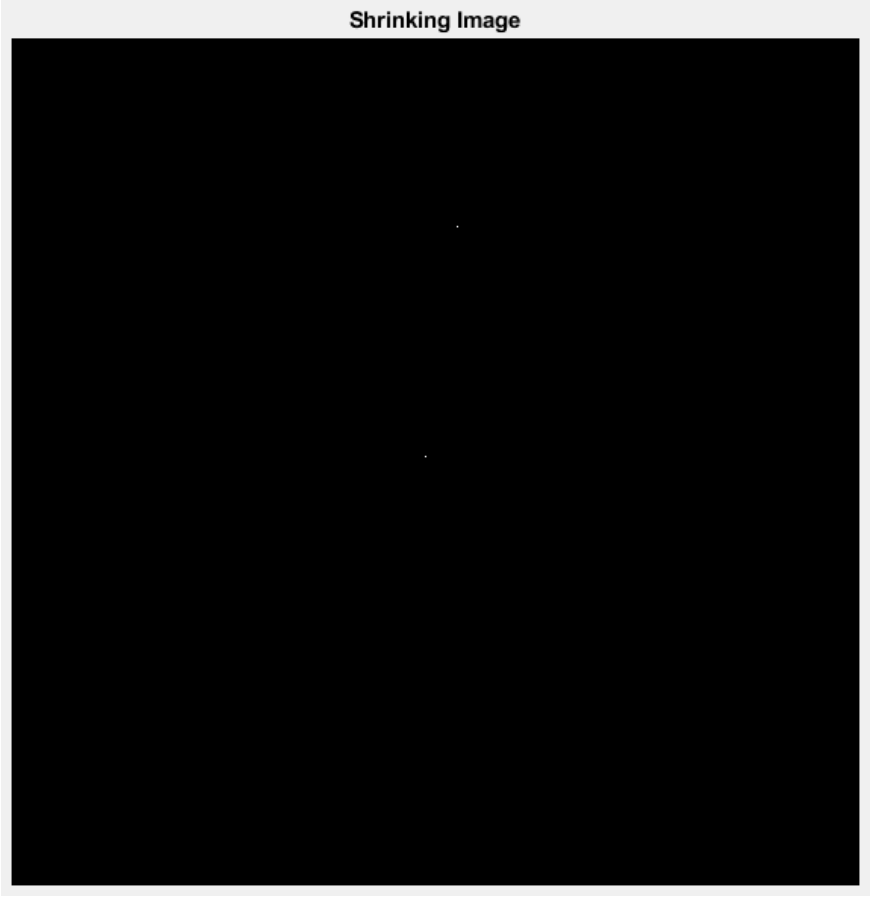


Fig 2.2 ‘Fan.raw’ Shrink image

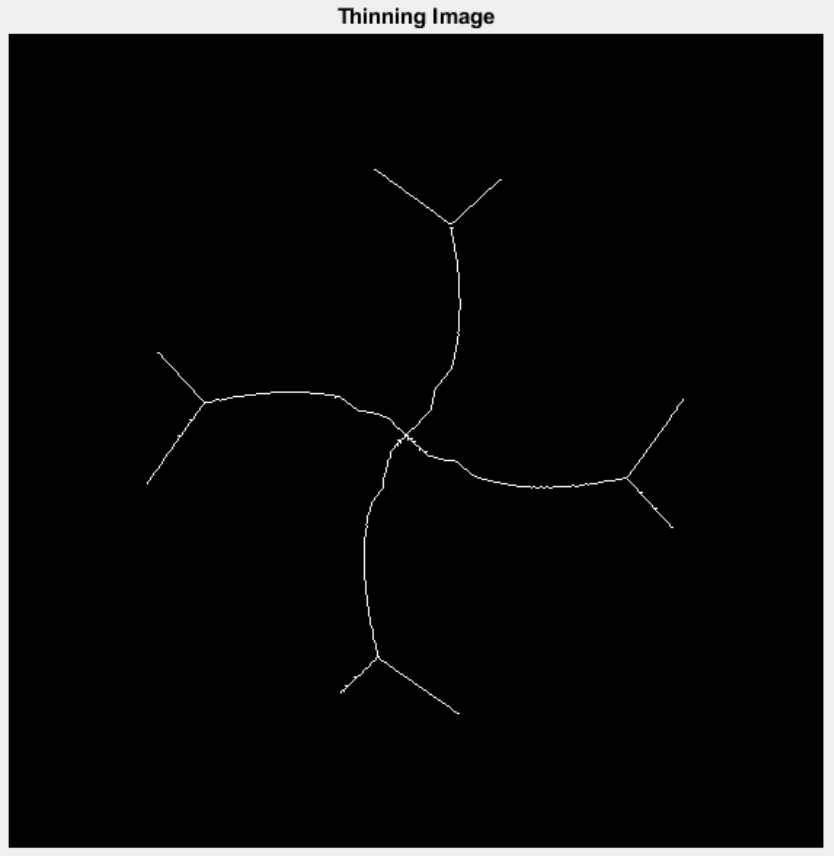


Fig 2.3 ‘Fan.raw’ Thinned image

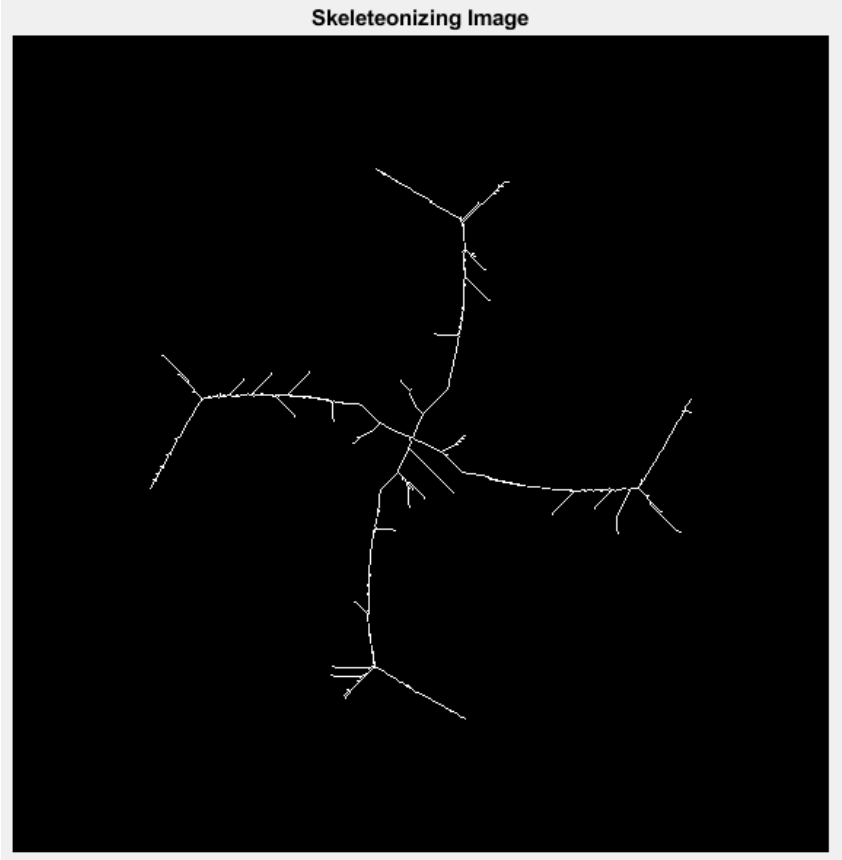


Fig 2.4 ‘Fan.raw’ Skeletonized image

Fig 2.5 ‘cup.raw’ original image

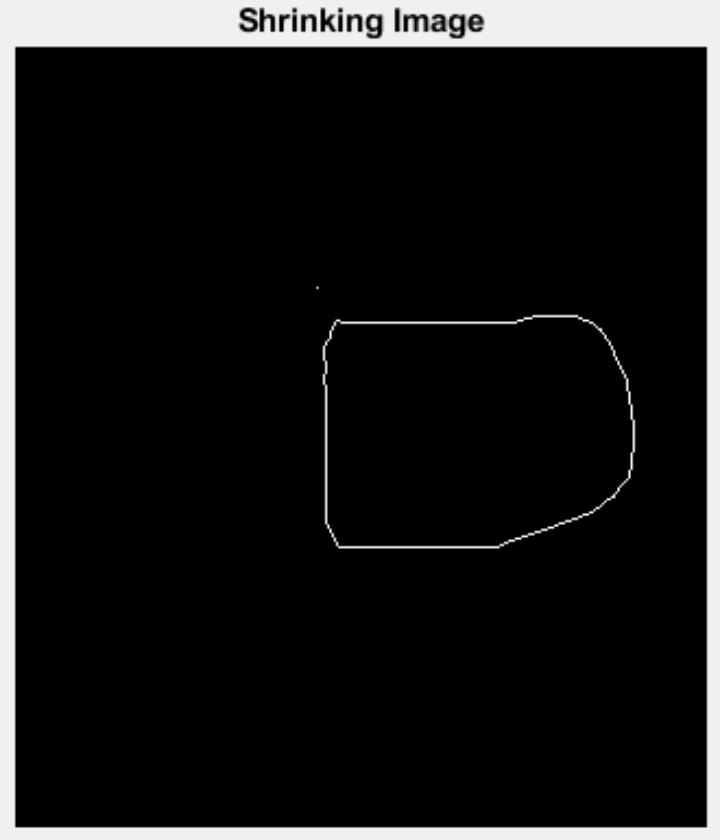


Fig 2.6 ‘cup.raw’ shrink image

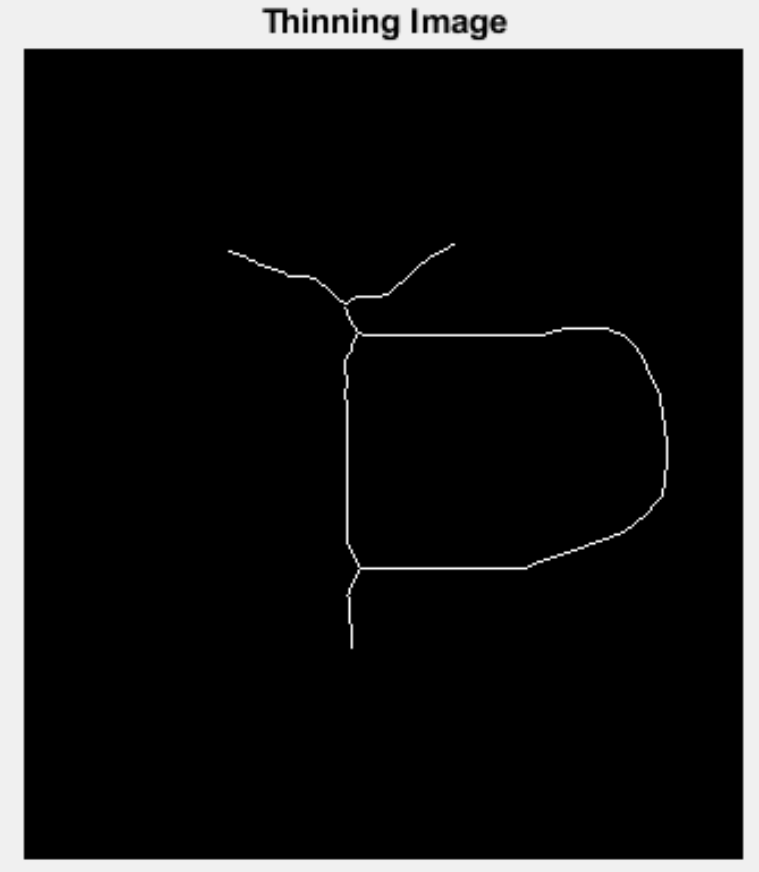


Fig 2.7 ‘cup.raw’ thin image

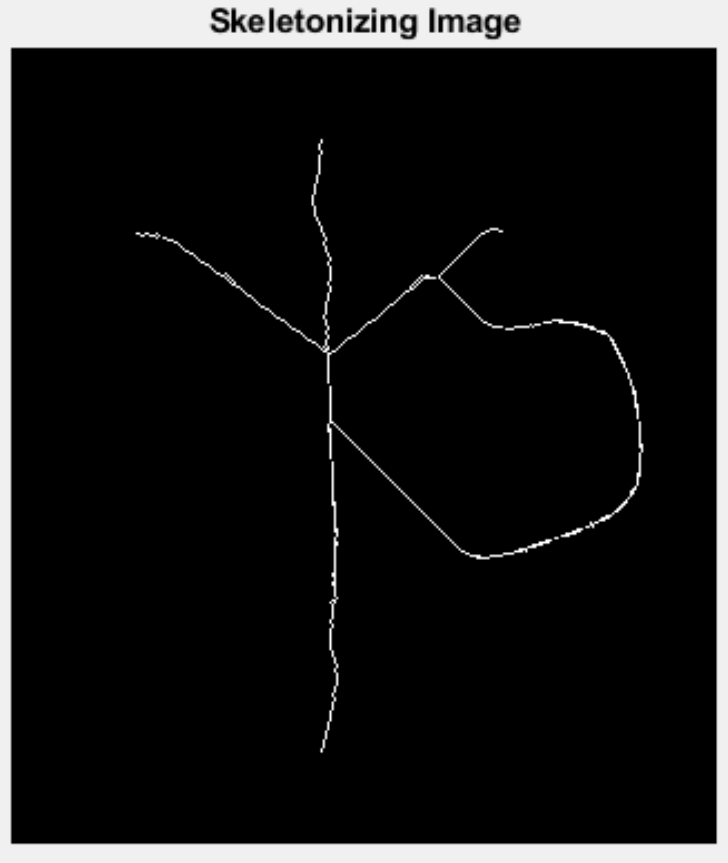


Fig 2.8 ‘cup.raw’ skeletonized image

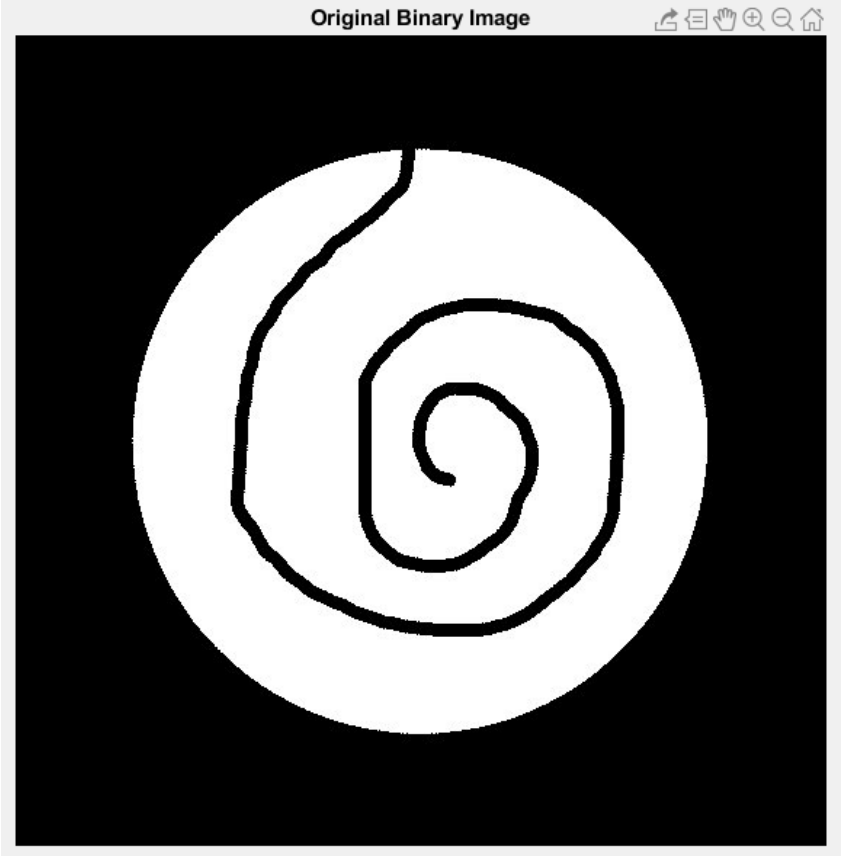


Fig 2.9 ‘maze.raw’ original image

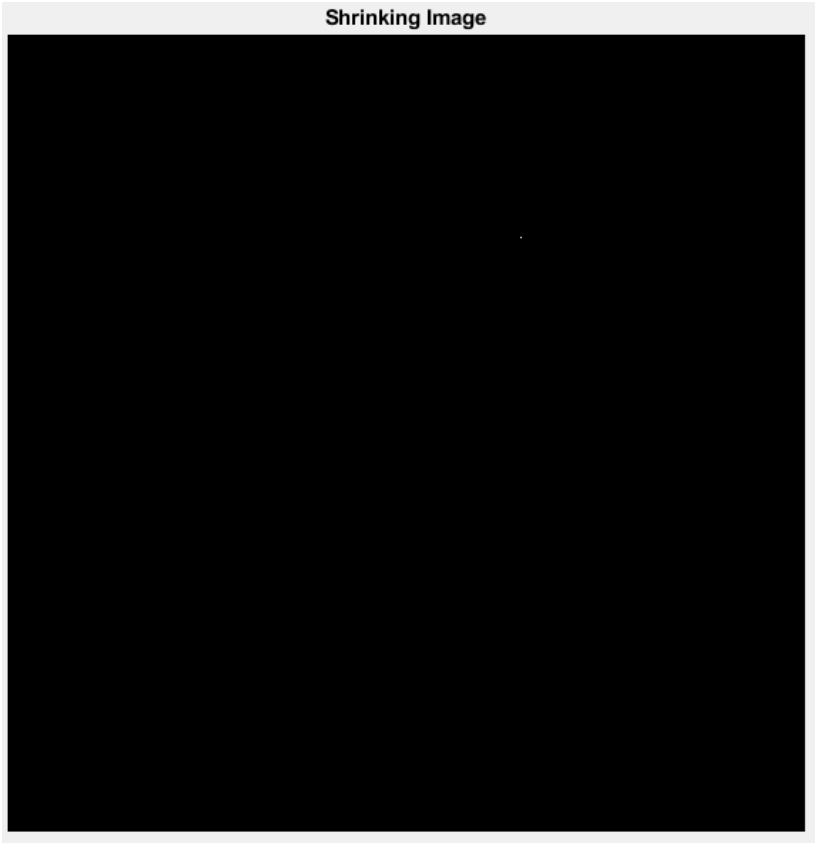


Fig 2.10 ‘maze.raw’ shrink image

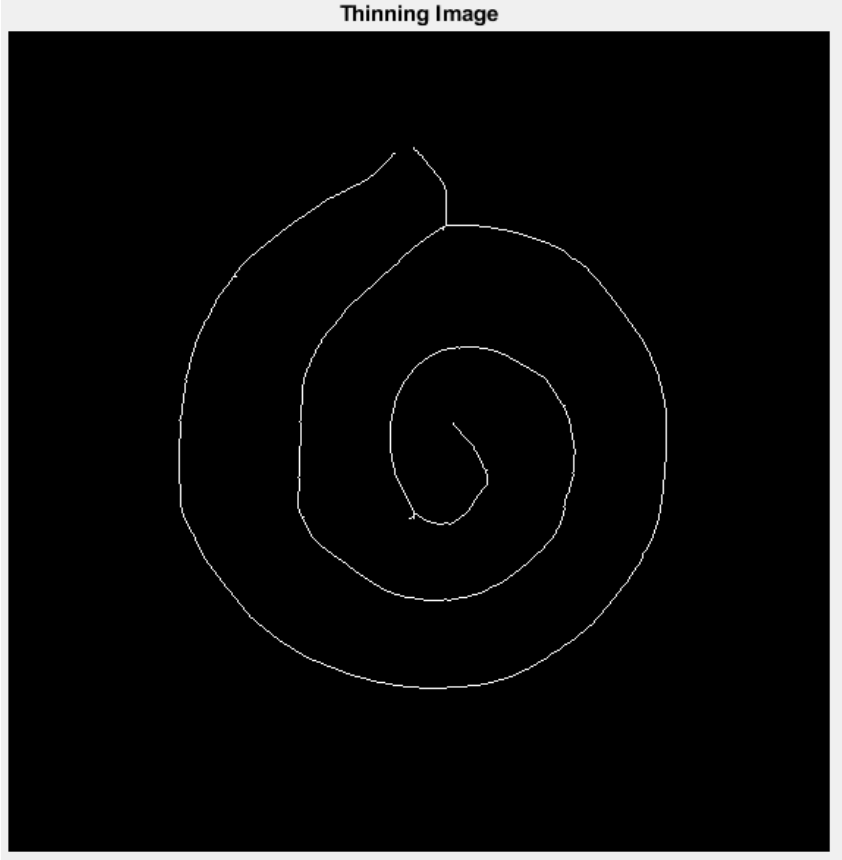


Fig 2.11 ‘maze.raw’ thin image

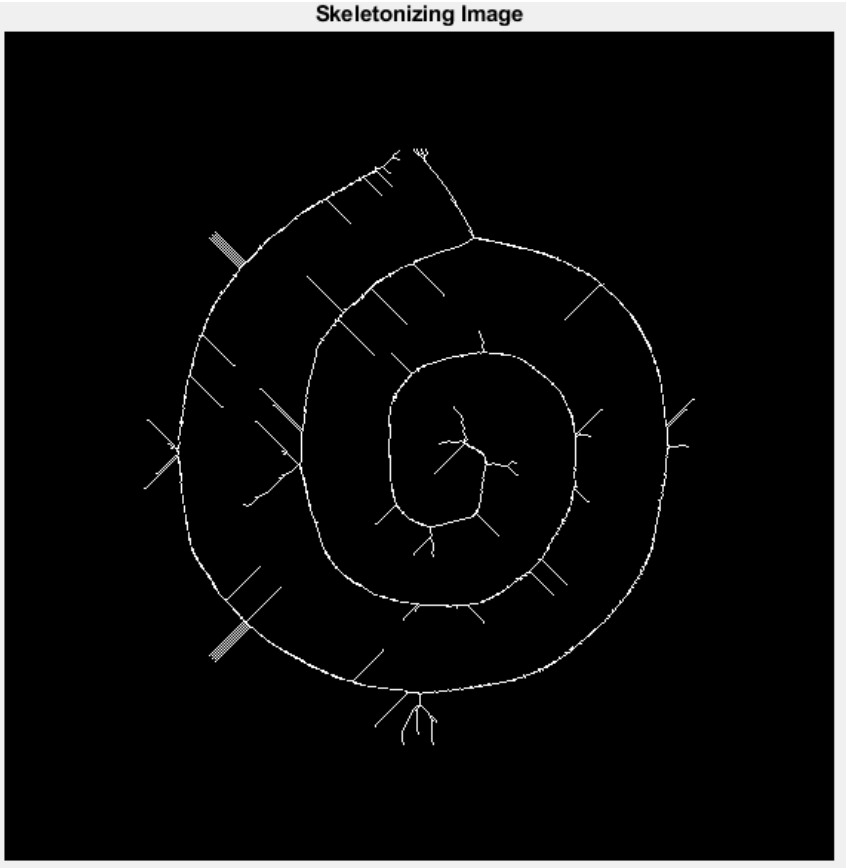


Fig 2.12 ‘maze.raw’ skeletonized image

Intermediate Results

1. Shrinking- '*Fan.raw*'

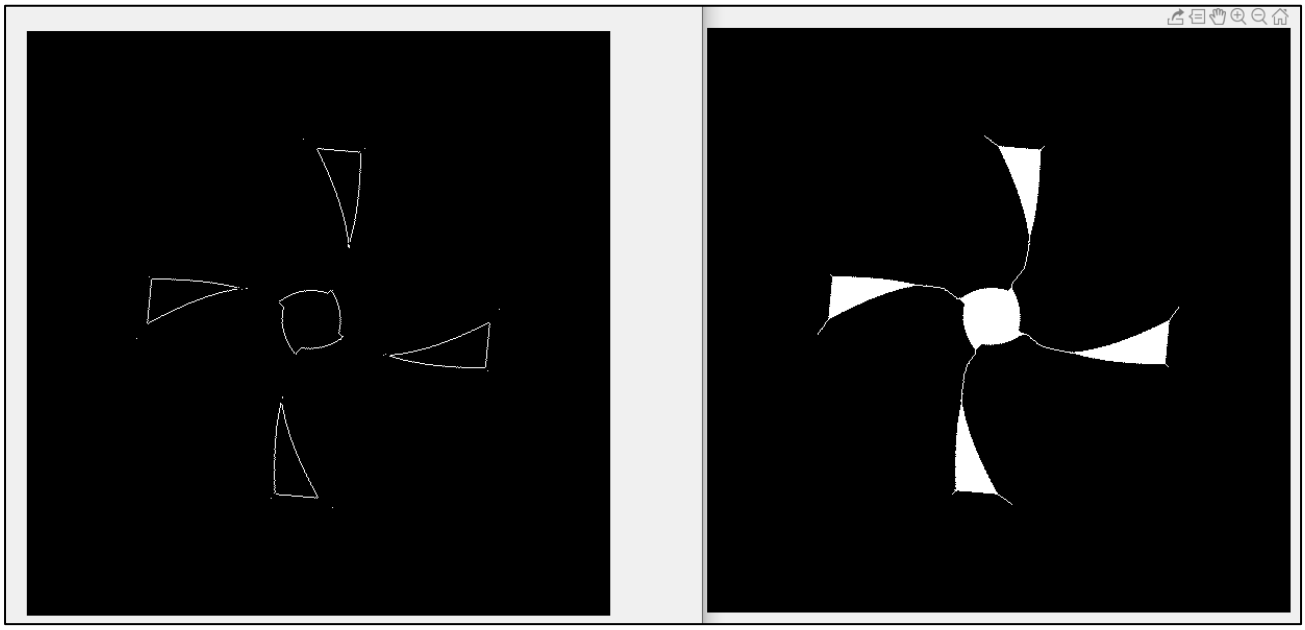


Fig 2.13 'fan.raw' intermediate result

The intermediate result of shrinking shows that it is shrinking to become a single pixel.

2. Thinning- *'Cup.raw'*

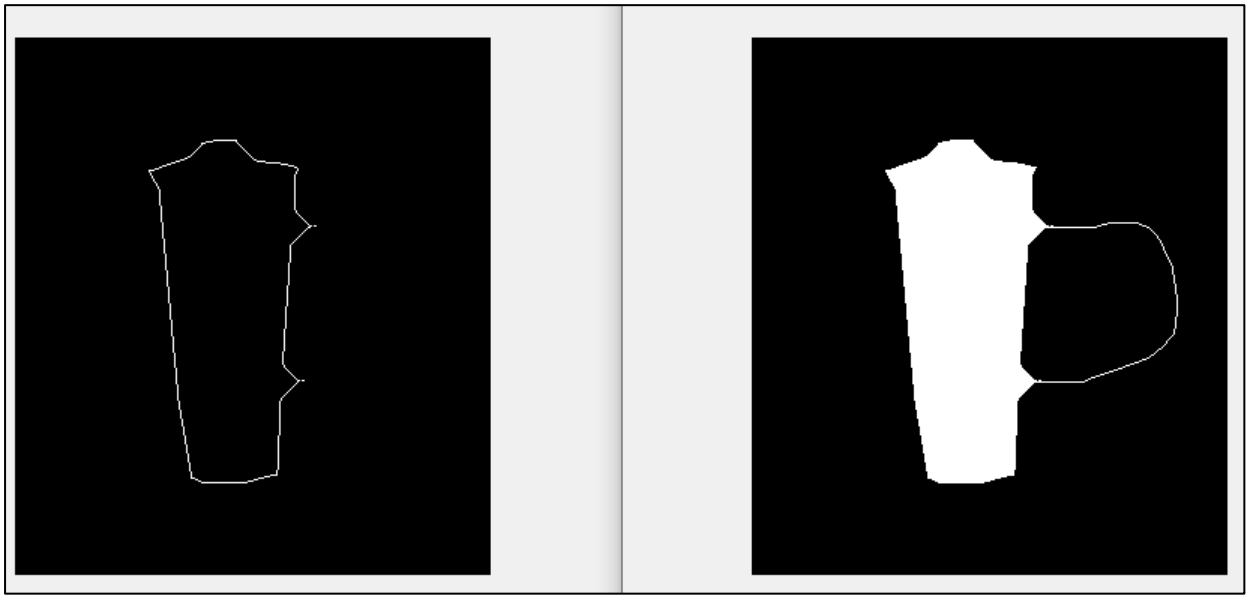


Fig 2.14 'cup.raw' intermediate result

The intermediate result shows the formation of connected component ring in the thinning process.

3. Skeletonizing- ‘mae.raw’

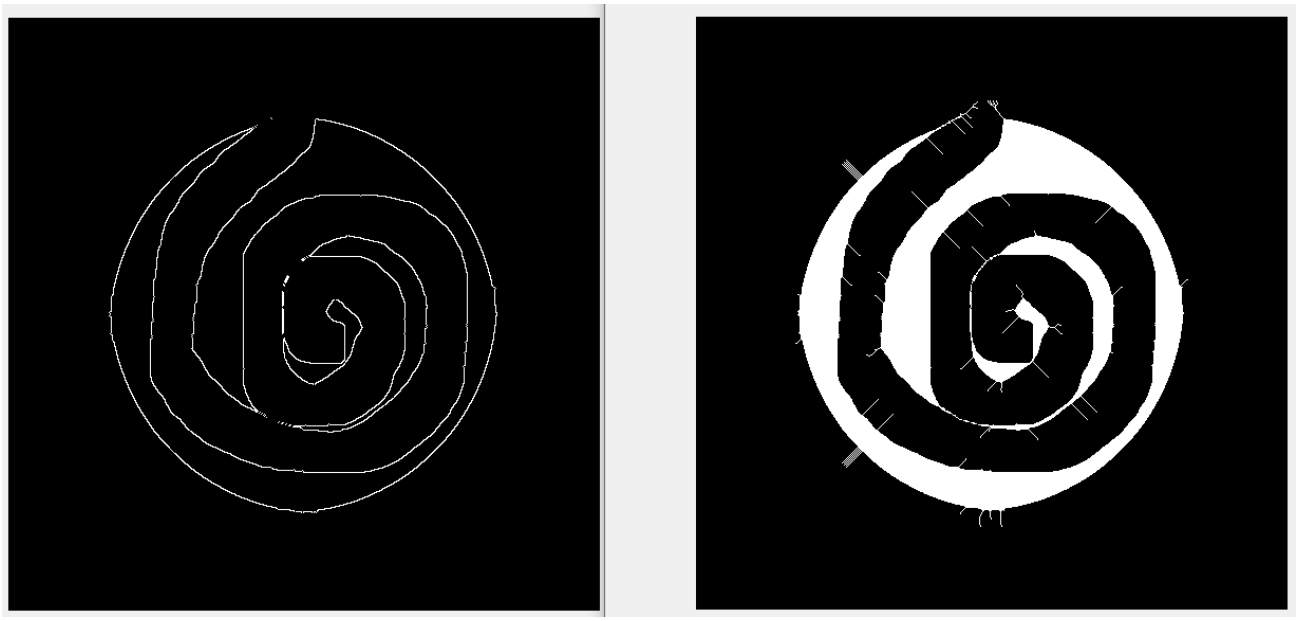


Fig 2.15 ‘fan.raw’ intermediate result

In skeletonizing, we can formation of skeleton like structure in every iteration.

4. DISCUSSION
   a. Shrinking
      As we can see and compare the result of shrinking of 3 given images, we can conclude that if the image is filled, then it shrinks down to one pixel of white near its center. If the object has hole in the middle then the shrinking happens and the dot comes in half way between object and it hole.
      We can see 2 dots in ‘Fan.raw’.
      In ‘cup.raw’, we can see 1 dot and also the pattern which is because we had a hole in the object. (handle of cup).
      In ‘maze.raw’ we can see a single white pixel.

   b. Thinning
      Comparing the 3 results, it can be said that if object does not have a hole in its pattern, then thinning output is minimally generated stroke to nearest outer boundary. And object with hole converts to minimally connected ring between hole and its outer boundary.
      ‘Fan.raw’ doesn’t have any holes. While other 2 images does have the hole which give rise to the connected ring.

   c. Skeletonizing
      In skeletonizing, we basically get all the edges and connected components resulting in the likewise skeleton of an image. We can see all the 3 images of which skeleton is the final output.

General:
As we have used 3*3 structuring element for the HIT and MISS transform, the result would be better if we consider 5*5 structuring element as we would be considering more neighbours but this will increase computational time of the code.

b) Counting Games

1. ABSTRACT AND MOTIVATION

   Morphological Operations has many applications, one of them is 'counting'. Here we can take an image, do the operation and get the count of specific thing. Here, we are given, *'stars.raw'* which is a binary image with stars located all around the image. We need to count the stars in the image. We used Shrinking operation for the same.

2. APPROACH

   'Counting Games' is one of the most fascinating application of Shrinking. The approach is very same which is discussed in above section. The process is similar till the Shrinking operation which converts each star here into a single pixel dot. So, the shrink output is single pixel white dots in the black background.

   Star Counting

   Now, we need to count the number of stars, i.e. number of white dots. To do so, I have added a nested for loop to scan the shrink image and check if the pixel is equal to 1. Thus, we get the count of no of stars.

   Different Stars Sizes Counting

   Adding, we need to mention the different sizes of stars present in the image. For this, I am doing Connected Component Labelling (CCL). There is another method which is finding the L2 distance of the pixel, but CCL seems to be more efficient and faster.

3. RESULTS

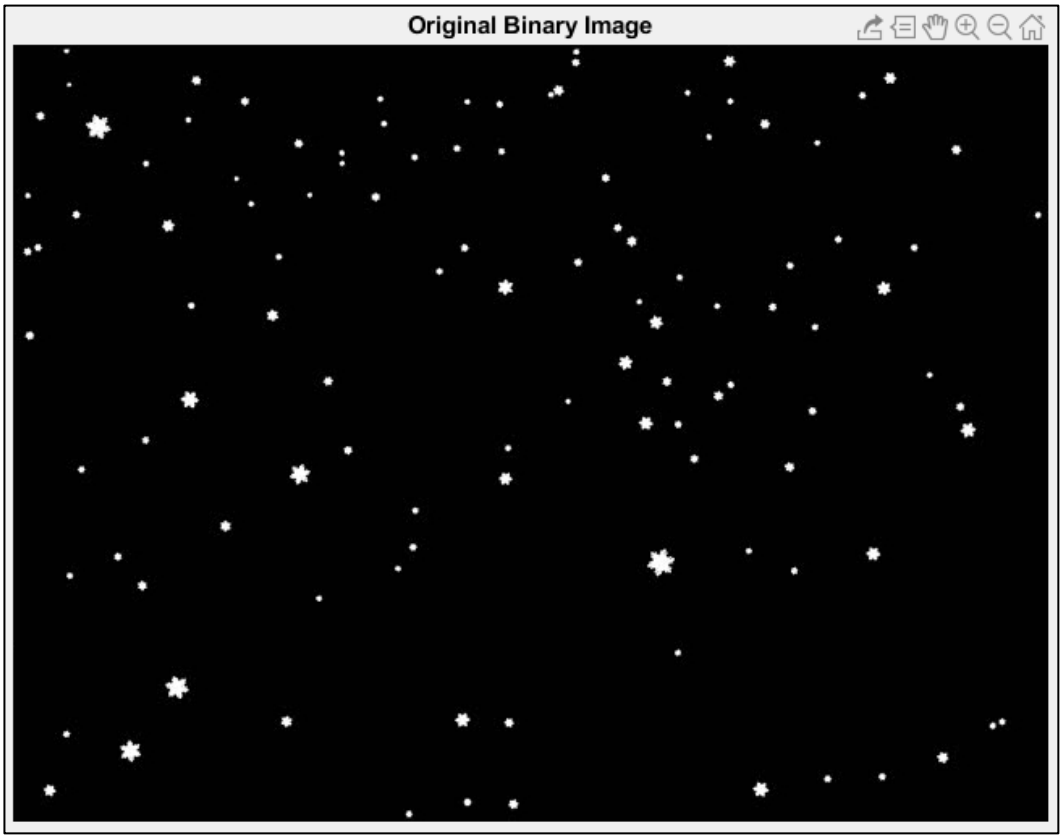


Fig 2.16 ‘stars.raw’ original image

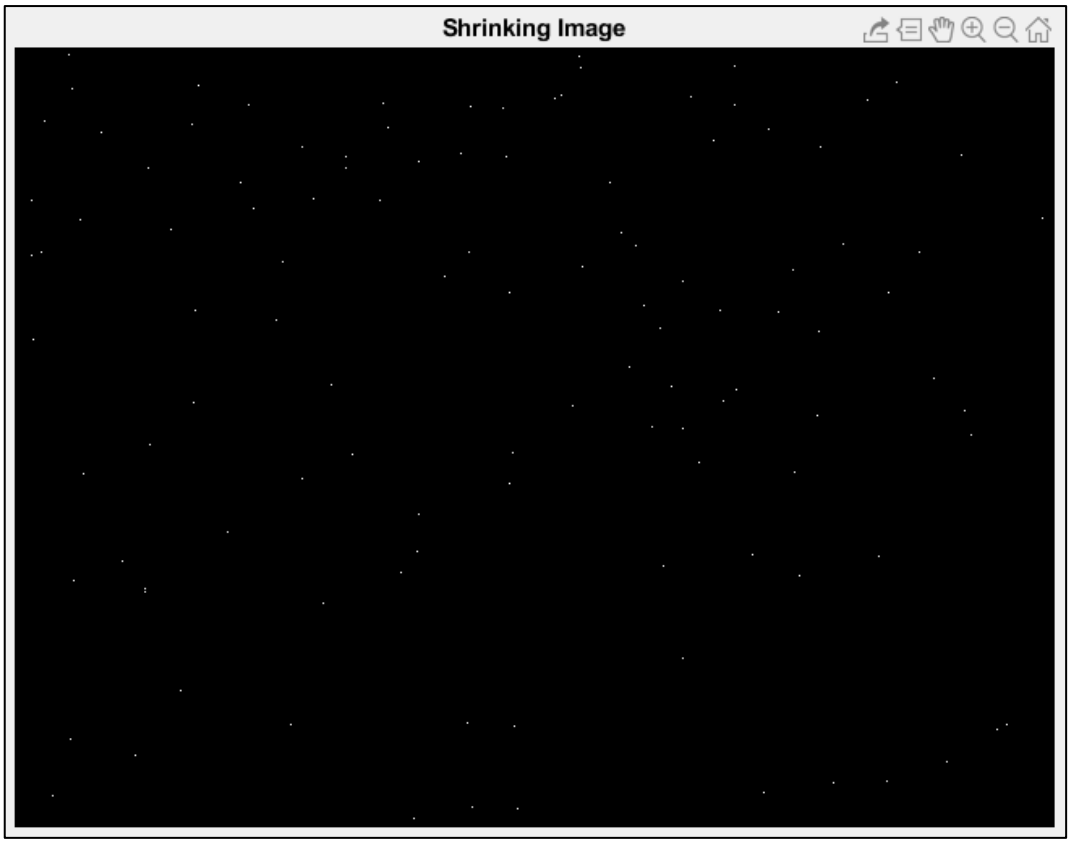


Fig 2.17 ‘stars.raw’ shrink result

count_stars =
113

count_stars_CCL =
113

star_size_count =
41

Frequency Count Table:

Labels	Freq
1	0
2	5
3	10
4	30
5	0
6	15
7	0
8	33
9	0
10	20
11	0
12	5
13	25
14	8
15	9
16	12
17	20
18	9
19	0
20	8
21	8
22	13
23	17
24	129
25	0
26	9
27	23
28	0
29	9
30	0
31	9
32	17
33	9
34	13
35	24
36	13
37	9
38	11

39	10
40	8
41	18
42	6
43	7
44	20
45	8
46	0
47	9
48	15
49	11
50	38
51	0
52	17
53	24
54	13
55	0
56	14
57	15
58	13
59	16
60	0
61	11
62	17
63	0
64	12
65	12
66	9
67	54
68	44
69	0
70	0
71	8
72	13
73	10
74	15
75	32
76	0
77	40
78	0
79	12
80	19
81	0
82	44
83	0
84	8
85	20
86	22
87	0
88	0
89	13

90	66
91	21
92	0
93	0
94	7
95	20
96	0
97	17
98	41
99	0
100	15
101	0
102	48
103	0
104	15
105	12
106	16
107	17
108	0
109	22
110	84
111	0
112	10
113	0
114	38
115	0
116	0
117	10
118	28
119	13
120	11
121	43
122	156
123	0
124	0
125	15
126	0
127	0
128	10
129	13
130	12
131	22
132	11
133	0
134	12
135	115
136	0
137	0
138	0
139	49
140	0

141	27
142	0
143	20
144	0
145	10
146	10
147	13
148	96
149	0
150	0
151	30
152	0
153	14
154	11
155	49
156	0
157	34
158	0
159	16
160	25
161	0
162	12

4. DISCUSSION
   a. The total number of star sizes is 113 by applying shrinking algorithm.
   b. Different Stars size is 41. The frequency of each label is shown above.
   c. Other method to solve above questions is Connected Component Labelling (CCL)

   Explanation for CCL:

   Connected Component Labelling is an algorithm used to detect the connected pixels in computer vision. It is very efficient algorithm in binary images to detect the connected component.

   Steps

   1. Raster Scan the image
   2. If the pixel is 1, then check its 4 neighbours. If center pixel is (i,j) then its neighbours to be checked are (i-1,j-1), (i-1,j), (i-1,j+1) and (i,j-1).
   3. If label is present in any of these neighbours, then assign the same label.
   4. If not then create a new label and make a note of it on the same pixel.
   5. Scan through whole image, increase the label value.
   6. For 2nd pass, again raster scan the image.
   7. Check for pixel which is non zero, then check its 8 neighbours.
   8. If anyone of these 8 pixel has value, then assign this label.
   9. If there are 2 labels in neighbourhood, then select the minimum label.
   10. Final matrix will have the connected components.

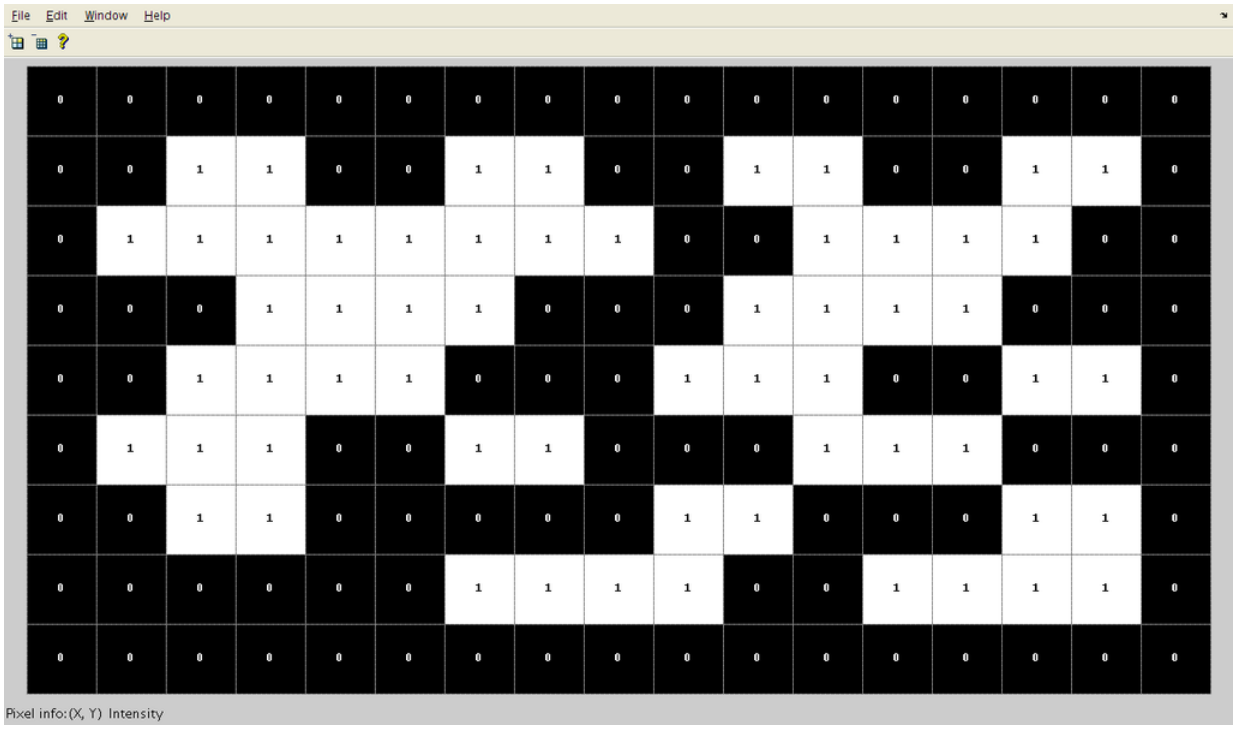


Fig. 2.18 Example Matrix

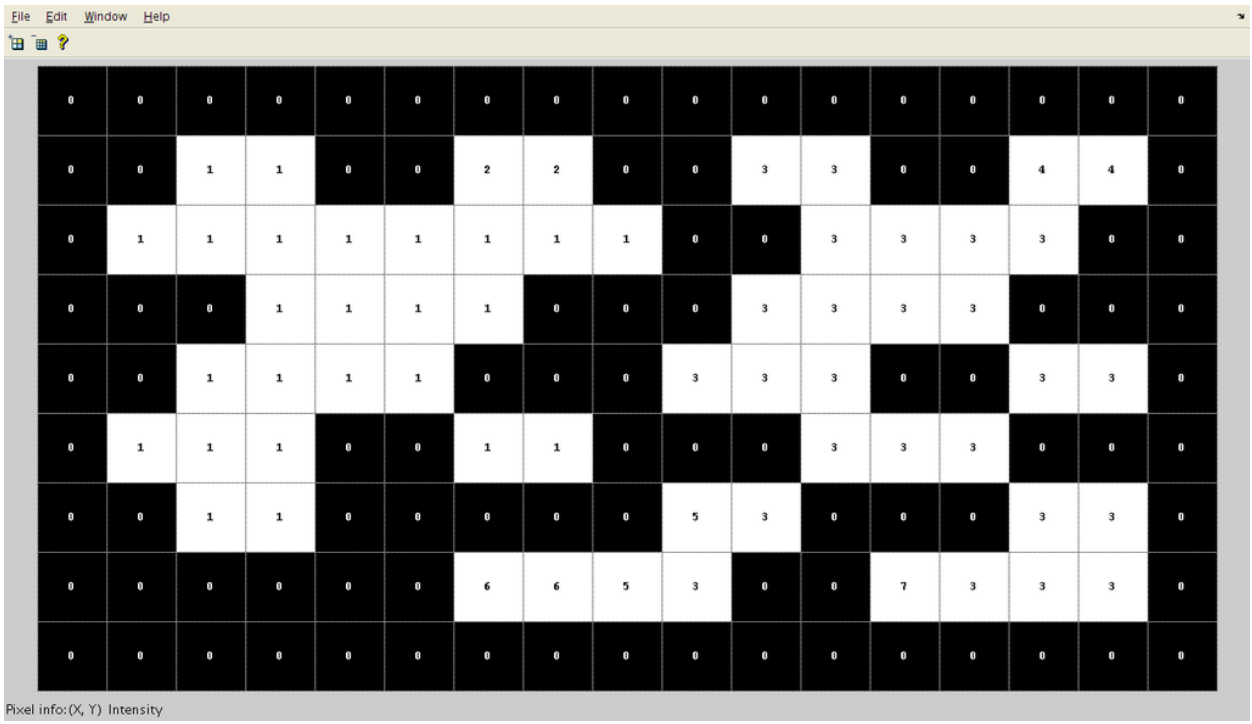


Fig. 2.19 After 1st pass

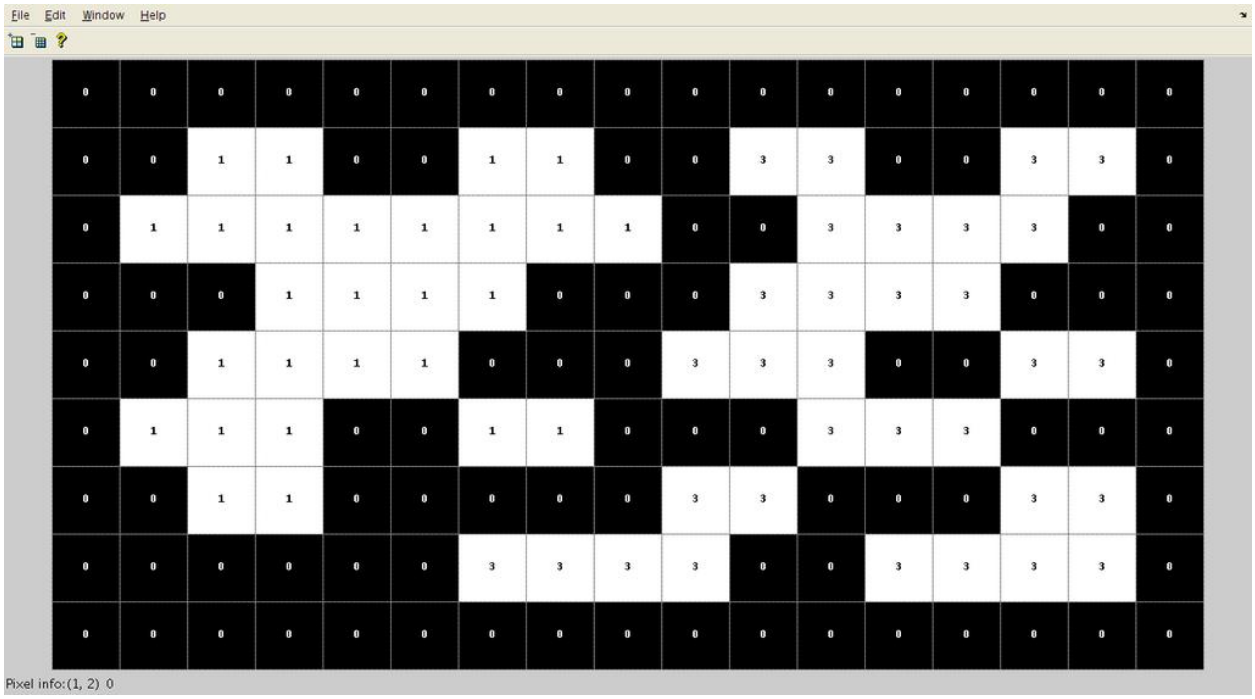


Fig. 2.20 After 2nd pass

c) PCB Analysis

1. ABSTRACT AND MOTIVATION

   This is the extension to the application of 'Counting Games'. We have been provided with PCB and we have to do analysis for the same. Here, we are told to count the number of holes in the PCB, next is we must mention the number of paths in the image.

2. APPROACH

   The approach for the 1st part of the problem is like previous one which is we have to do morphological operation i.e. Shrinking in this case. Here, the given image has object with 0 and background with 1.

   For Shrinking operation, we apply the algorithm to the image provided. The holes in the circuit board reduces to single dot (white pixel) and all the surrounding neighbours are black here. So, I did check if pixel is white and all its 8 neighbours are 0 then it is counted as hole. Thus, we get value of holes count.

   Further, for finding the number of paths, I inverted the image, i.e. foreground to 1 and background to 0. The shrinking operation is then executed. The result of this is all paths with a single pixel value (value=1).
   After this, the algorithm of Connected Component Labelling (CCL) is used. This algorithm will detect all the connected components, i.e. paths. So, we can easily count the number of connected components.
   The detailed explanation of CCL is given in previous section.

3. RESULTS

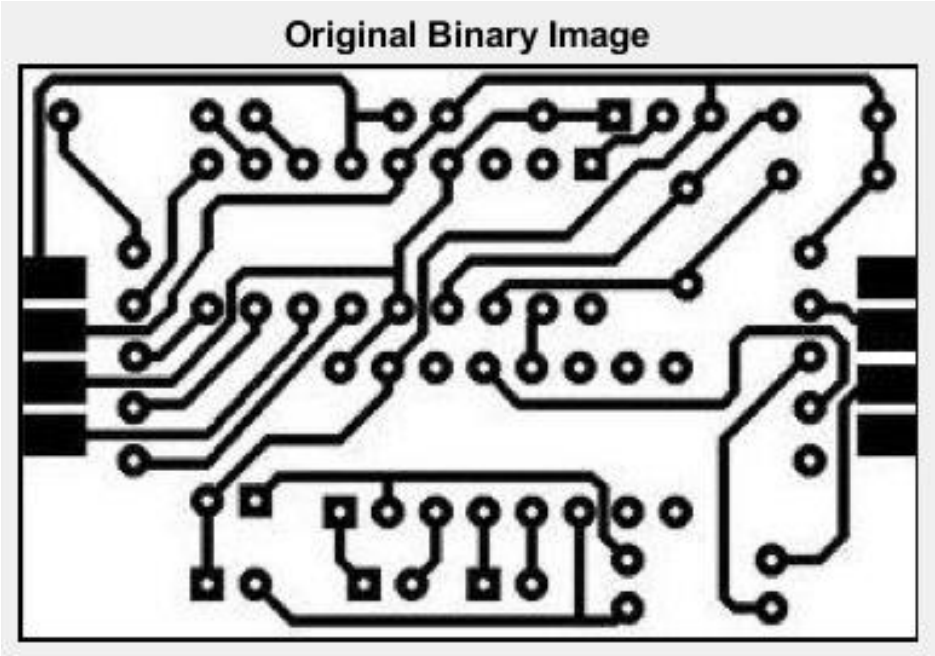


Fig 2.21 'pcb.raw' original image

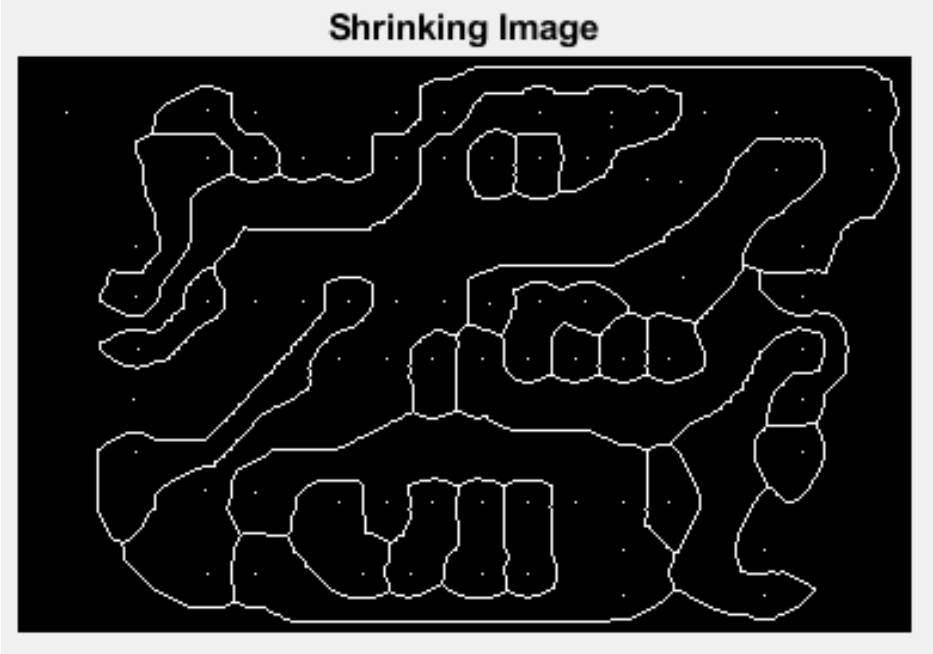


Fig 2.22 'pcb.raw' shrink image

count_holes=
71

The following image is the output of shrinking operation applied to the original image. As we can see, all holes are converted into single pixel while all paths got connected.

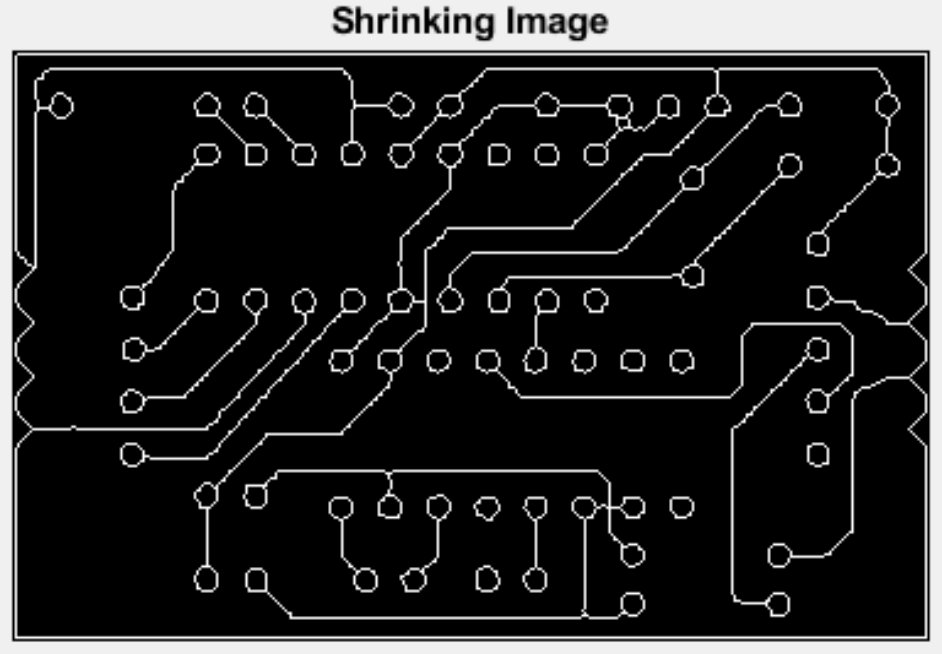


Fig 2.23 'pcb.raw' shrink image after inverting

count_pcb_path=
25

The following image is the output of shrinking operation applied to the inverted image of original. As we can see, the paths are clearly seen and they are all in single pixel.
CCL algorithm is applied to found out the count of PCB paths.

4. DISCUSSION
   a. The number of holes in given PCB image is 71.
      Operation used is Shrinking which reduces every hole into a single pixel and then count of total no of ones with all background in surrounding come out to be 71
   b. The total number of paths in PCB image is 25. The path is defined as the connection link between the 2 PCB holes. There could be branches in the paths and counted as the single path.

d) Defect Detection

1. ABSTRACT AND MOTIVATION
   Morphological Processing as said is used to remove the imperfections in the image. So, if a image has some defect, it can be removed using morphological processing. This application becomes very useful and valuable in industry as one can easily find the pixels that are defected in the image.

2. APPROACH
   The morphological operation used for defect detection is 'Shrinking'.
   The steps followed are as follows;
   a. Finding center pixel of gear. This is done by Shrinking operation applied to the image directly.

b. The center pixels of 4 black holes are found out. This is done by inverting the image and then doing the shrinking operation.
c. Making stencil such that it has 12 tooth with 30 degree separation between each tooth.
d. Compare the stencil image with the shrinked output, so that it will detect the missing tooths.
e. Note down the pixel and mark.

For better understanding, I named each tooth by numbers starting from 1 to 12. 1 is the rightmost and following the anticlockwise direction.

3. RESULTS

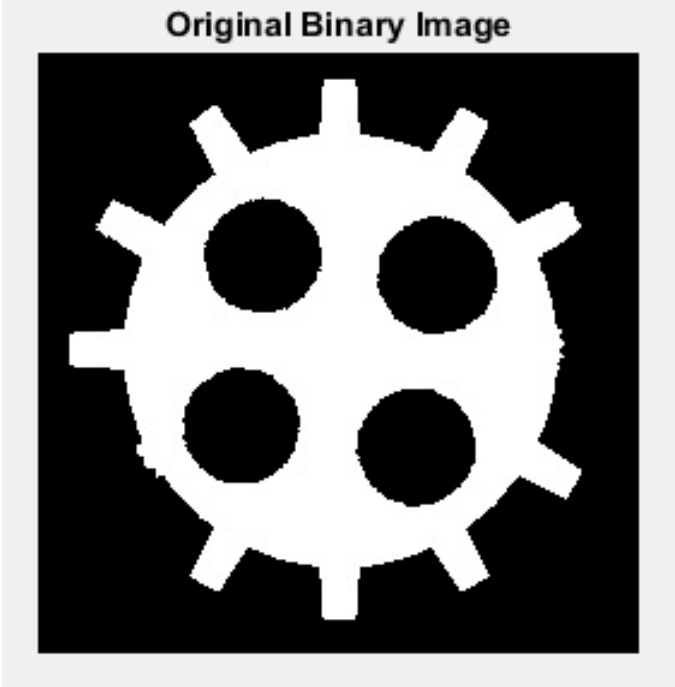


Fig 2.24 'geartooth.raw' original image

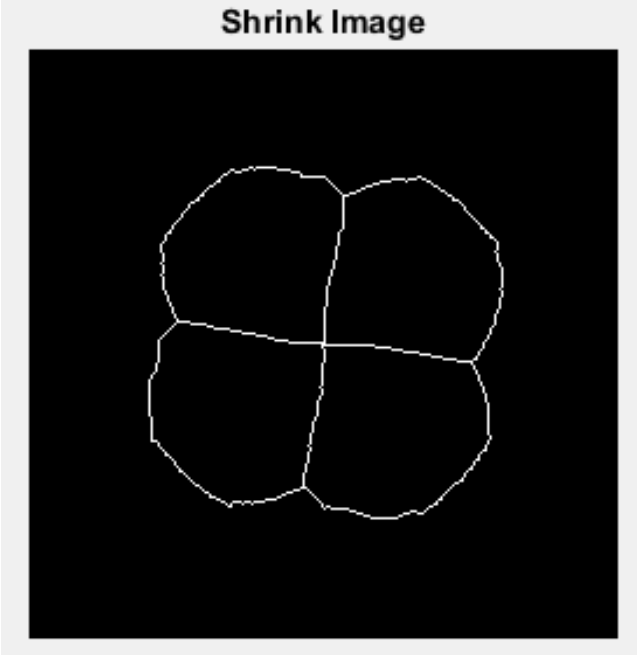


Fig 2.25 'geartooth.raw' shrink image to calculate center

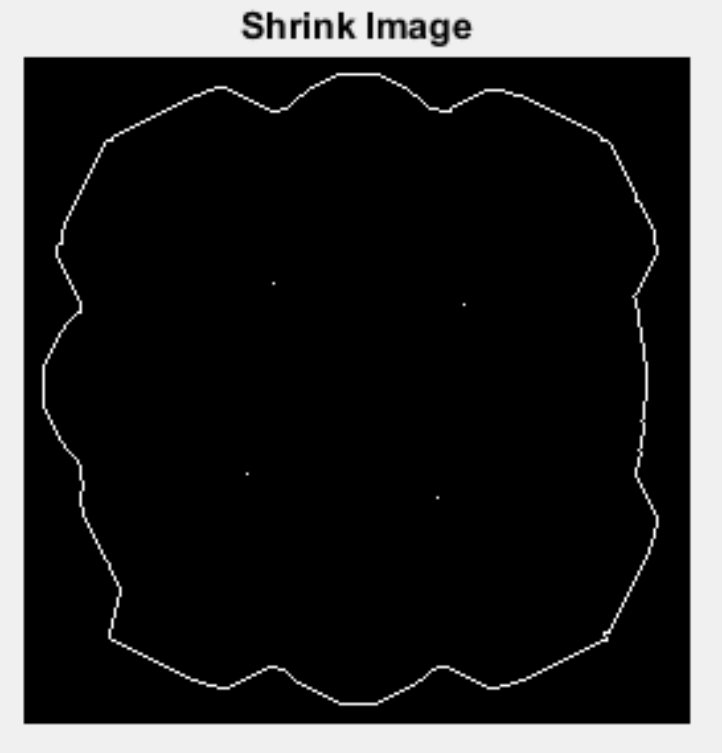


Fig 2.26 ‘geartooth.raw’ shrink image to calculate center of black holes

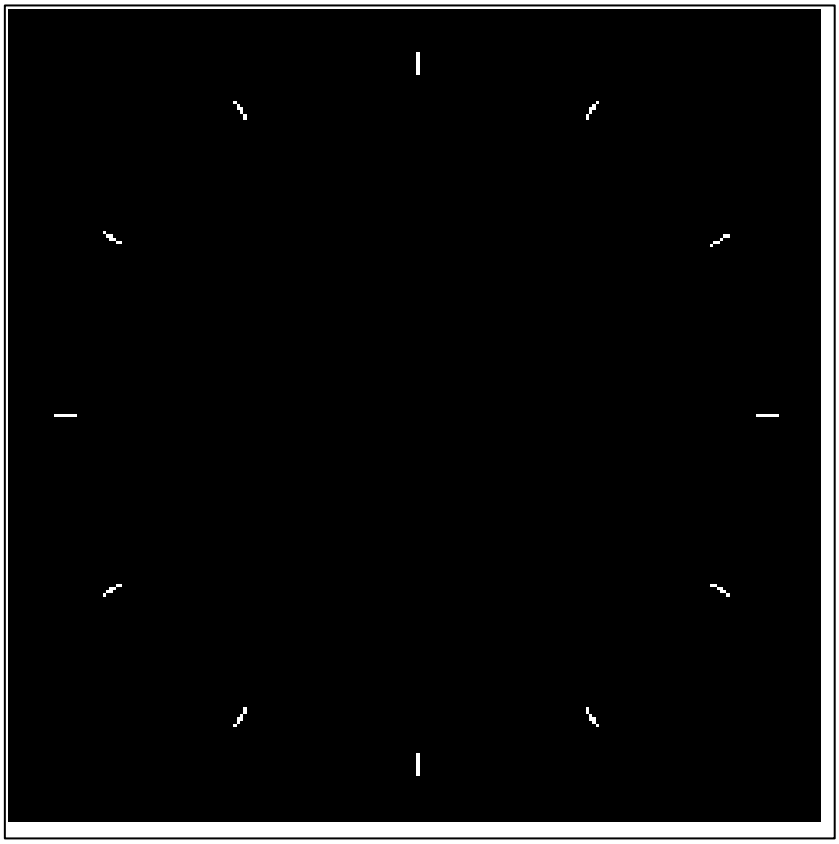


Fig 2.27 Reference image showing gear teeth

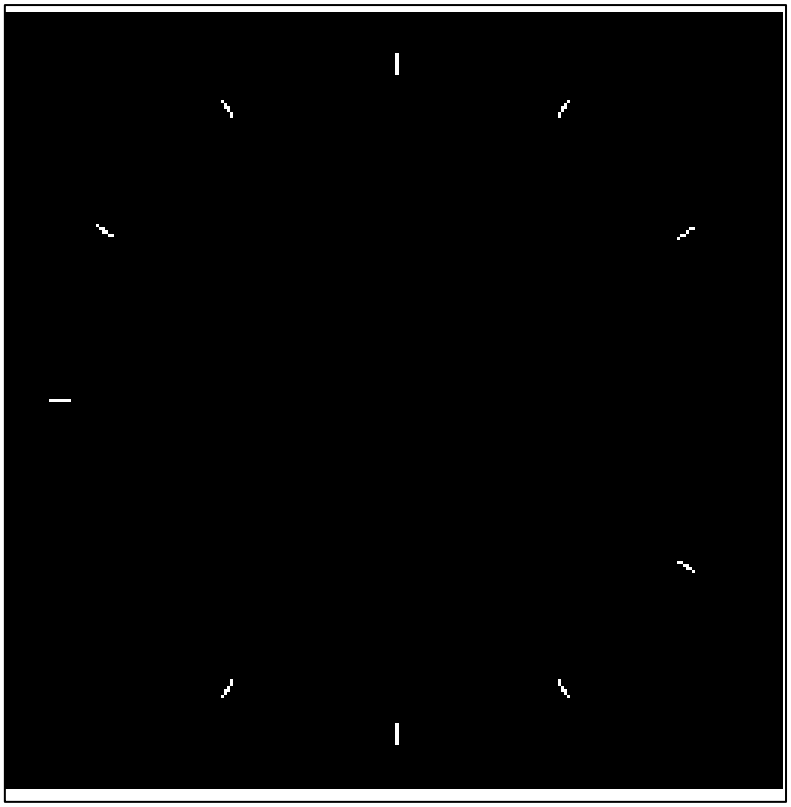


Fig 2.28 Image showing gear teeth missing at 2 positions

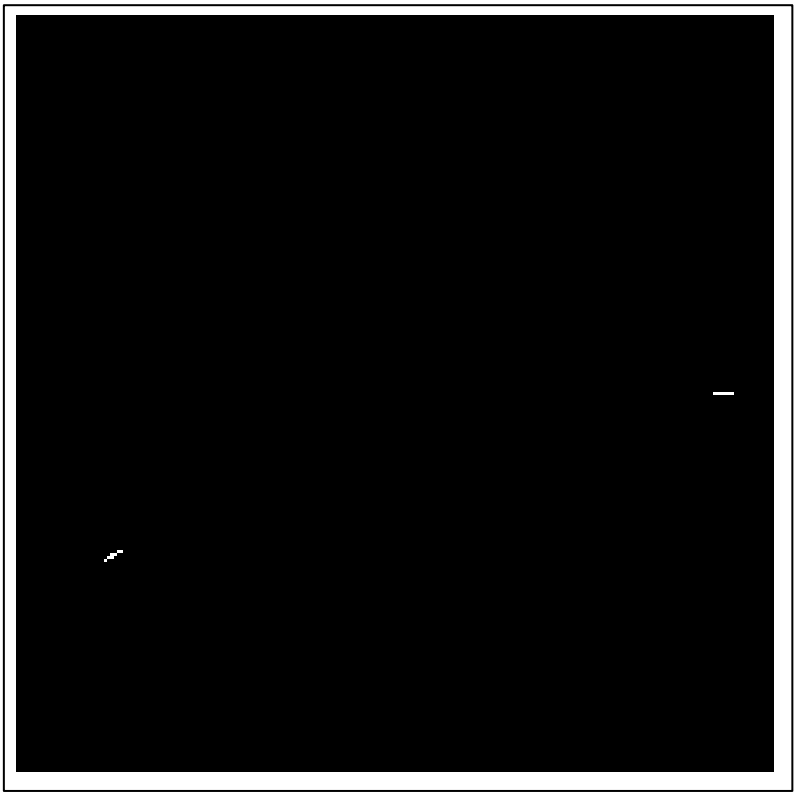


Fig 2.29 Image showing missing gear teeth

Fig 2.25 shows the shrinking image to know the center of the gear tooth.
Fig 2.26 show the shrinking image which is of inverted input so that we get center of the 4 black holes present in the image. These black holes reduce to a single pixel, white dot.

The reference image has 12 tooths (lines for representation) placed separating each other at 30 degrees. Fig 2.28 shows that 2 tooths are missing as seen in the original image has well. Fig 2.29 image shows only the teeth's which were missing from the original image.

4. DISCUSSION

As we can see from the original image above, the missing teeth are teeth 1 and 8. So after applying Shrinking and all other operations mentioned in above section, we get the missing tooth which are indicates by white pixels in the black background. The positions of missing teeth are successfully found using morphological operations.

REFERENCES:

1. http://squircular.blogspot.com/2015/09/mapping-circle-to-square.html
2. http://mathproofs.blogspot.com/2005/07/mapping-square-to-circle.html
3. https://en.wikipedia.org/wiki/Connected-component_labeling
4. https://in.mathworks.com/help/vision/ref/matchfeatures.html
5. https://in.mathworks.com/help/gpucoder/examples/feature-extraction-using-surf.html