EE569 Digital Image Processing: Homework #3

Name: Nazim N Shaikh
USC Id: 8711456229
Email Id: nshaikh@usc.edu
Submission Date: 03/03/2019

Problem 1: Geometric Modification

I. Abstract and Motivation

- Geometric Modification means modifying geometrical properties of an image. It is an operation over image in which image can be translated, rotated, scaled or warped for use cases such as correcting images from lens distortion/camera orientation, Image Zooming, Image Morphing or for creating any special effect distortion. In simple terms, it is rearrangement of pixels in a given image, where, we try to change pixel positions without affecting pixel values. Also known as Spatial Modification, they are extensively used in many applications such as Medical Image Processing – identifying structural differences in an x-ray, Computer Graphics –to give artistic effects to a painting or a sketch, Geographical Mapping, Satellite Imaging and much more. In this assignment, we try to implement all possible geometric transformations utilizing them in various applications.
-

II. Approach and Procedure

- There are two approaches to any geometric transformation: forward and reverse address mapping.
- In forward approach, for each point (u, v) in the original image, we try to find corresponding position (x, y) in the output image by forward transformation. If the points are non-integers, they are rounded to nearest integer values, as result of which data voids can happen.
- An alternative is, reverse computation, in which for each point (x, y) in output image, we find its corresponding point (u, v) in original image using inverse mapping function. If mapped point is not integer, we interpolate from nearby samples from, thereby avoiding data voids. In this assignment, we will use reverse mapping to perform all transformations and perform bilinear interpolation after each transformation.
- One important point to be noted is, we perform all the operations in cartesian coordinate system because geometrical transformations are based on cartesian

coordinate system and it is easy to understand image when viewed from cartesian coordinates.

- The equation for converting from Image to Cartesian Coordinate and back is as below:

$$p = j - width/2 \quad \text{-- (1)}$$
$$q = height/2 - i \quad \text{-- (2)}$$

$$j = q + width/2 \quad \text{-- (3)}$$
$$i = height/2 - q \quad \text{-- (4)}$$

where (j, i) denote pixel positions in image coordinate system and (p, q) denote pixel positions in cartesian coordinates

- This conversion will bring image to center of x-y coordinate system with image center at $(0,0)$
- various tasks involving geometric modification are explained below

(a) Hole Filling:

- Hole filling or Puzzle matching is a geometric modification where we to try fit missing pieces of an image into the desired image. General idea is to find the location, orientation of holes, its corresponding patches and fit the patches by mapping the coordinates.

- Detailed process is explained below:
  1. Find the coordinates of corners in each piece images
     This implies scanning the images in vertical and horizontal direction to find out max-min row and column values where pixel values are not 255.
  2. The coordinates are then converted to cartesian system using equations (1) and (2).
  3. We then perform a composite transformation where the image is first brought to center by translation and then rotated to get a straight image.
     As mentioned before, we will be using reverse mapping in order to get better and complete image.

     The translation function for reverse mapping is given as below:

$$u = x - tx$$

$$v = y - ty$$

where, (u, v) and (x, y) are pixel positions of input and output image,
tx and ty are translation offsets along width direction and height direction

The Rotation function for reverse mapping is stated below:

$$u = x * cos\ (theta) + y * sin(theta)$$
$$v = -x * sin(theta) + y * cos(theta)$$

where, (u, v) and (x, y) are pixel positions of input and output image,
theta = angle of rotation to be calculated as $tan^{-1}(\frac{\Delta y}{\Delta x})$

The rotation is done with respect to origin, which in our case post cartesian conversion is center $(0,0)$.

4. The holes to be filled are of size $(160, 160)$. So, the transformed image is then scaled to match this, using below reverse scaling function

$$u = \frac{x}{fx}$$
$$v = \frac{y}{fy}$$

where, *(u, v)* and *(x, y)* are pixel positions of input and output image,
fx and fy are scaling factors to be calculated as

5. At each stage of geometrical transformation, we convert back the pixels to image coordinate and perform nearest neighbor or bilinear interpolation to obtain pixel values at not integer positions
we use the following formula for bilinear interpolation:

$$F(p', q') = (1 - a)[bF(p + 1, q) + (1 - b)F(p, q)] + a[(1 - b)F(p, q + 1) + bF(p + 1, q + 1)]$$

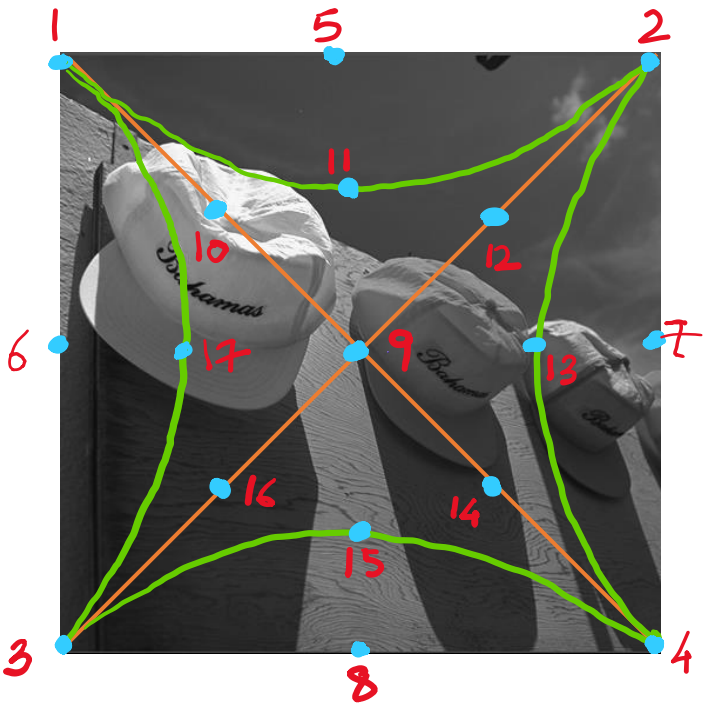
Where $p = floor(p')$, $q = floor(q')$, $a = q' - q$ and $b = p' - p$

6. The main image is now scanned to get coordinates of the all the three white patches to be filled. The scanning is done in a similar way as explain before except here we look pixel values that are 255.

7. Finally, we fill the scaled sub images into appropriate positions obtained from above to get the complete image.

(b) Spatial Warping

- It is a transformation which modifies the spatial configuration of image. It is a kind of special effect distortion mainly used for creative purposes such as for giving facial effects to Instagram or snapchat images, etc.
- Here we make use of triangle based warping technique to apply such effects.
- We divide the image into 4 triangles (highlighted with orange), taking each of the triangles 6 coordinates as control points and then map them into the corresponding regions with similar 6 coordinates (highlighted with green) in the warped image as shown in figure below:



- For example, region enclosed by points 1,5,2,12,9,10 are mapped to region enclosed by points 1,11,2,12,9,10.

- This reverse mapping is given by following second-order polynomial equation:

$$u = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2 \qquad ---(A)$$
$$v = b_0 + b_1x + b_2y + b_3x^2 + b_4xy + b_5y^2 \qquad ---(B)$$

In vector form,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

Where, (u, v) and (x, y) are input and output control points

- This boils to an input-output relationship problem where we have to find the transformation matrix specified by polynomial coefficients $\vec{a}$ and $\vec{b}$ also known as weighting coefficients

  The detail procedure is explained below:

  1. Find 6 input and output control points for each of the 4 triangles from the image
     > For upper triangle, input control points will be $(0,0), (128,128), (256,256), (384, 128), (511,0), (256,0)$ and corresponding output control points are $(0,0), (128,128), (256,256), (384, 128), (511,0), (256,127)$
     > We find these for all triangles
  2. For each triangle with 6 input and output control points do the following:
     i. Convert points into cartesian coordinates using equations (1) and (2)
     ii. Arrange the 6 input points into vector space as below:

$$u^T = [u_0 \quad u_1 \quad u_2 \quad u_3 \quad u_4 \quad u_5]$$

$$v^T = [v_0 \quad v_1 \quad v_2 \quad v_3 \quad v_4 \quad v_5]$$

     iii. Determine polynomial coefficients $\vec{a}$ and $\vec{b}$ by solving minimum mean squared error equation as:

$$\vec{a} = A^{-1}\vec{u}$$
$$\vec{b} = A^{-1}\vec{v}$$

     Where

$$a^T = [a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6]$$
$$b^T = [b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \quad b_6]$$

$$A = \begin{bmatrix} 1 & x_1 & y_1 & {x_1}^2 & x_1y_1 & {y_1}^2 \\ 1 & x_2 & y_2 & {x_2}^2 & x_2y_2 & {y_2}^2 \\ 1 & x_3 & y_3 & {x_3}^2 & x_3y_3 & {y_3}^2 \\ 1 & x_4 & y_4 & {x_4}^2 & x_4y_4 & {y_4}^2 \\ 1 & x_5 & y_5 & {x_5}^2 & x_5y_5 & {y_5}^2 \\ 1 & x_6 & y_6 & {x_6}^2 & x_6y_6 & {y_6}^2 \end{bmatrix}$$

3. Once we have weighting matrix, then for each triangle region apply reverse mapping equations (A) and (B) to get final warped image

(C) Lens Distortion

- Lens distortion occurs due to error in lens design. In this distortion, the straight lines become curved and bend either outward or inward from the image center.

- These distortions also known as Radial distortions can be classified into two – Barrel distortions or pincushion distortions. In Barrel distortion, the straight lines bend outward away from the image center. It is most common to wide angle and fish eye lenses. In Pincushion distortion, the lines bend inward, towards the image center. It is in most common in zoom lenses.

- These radial distortions can be corrected using below mapping equation:

$$x_c = x(1 + k_1r^2 + k_2r^4 + k_3r^6) \qquad ---(I)$$
$$y_c = y(1 + k_1r^2 + k_2r^4 + k_3r^6) \qquad ---(II)$$

Where,

(x, y) and (x_c, y_c) are the original distorted and corrected pixel locations.

k_1, k_2, k_3 are radial distortion coefficients and $r^2 = x^2 + y^2$

- The correction involves converting the image coordinates to camera coordinates. The equation is as follows:

$$x = \frac{u - u_c}{f_x} \qquad ---(a)$$

$$y = \frac{v - v_c}{f_y} \qquad ---(b)$$

The procedure for distortion correction is as follows:

1. Read the distorted image
2. For each pixel position in distorted image
   i. Convert the image coordinates to camera coordinates using equations (a) and (a)
   ii. Apply mapping equations (I) and (II)
   iii. Convert back camera coordinates to image coordinates
   iv. In case of non-integer pixels, perform bilinear or nearest neighbor interpolation
3. Save the corrected Image

III. Experimental Results

(a) Hole Filling

Figure 1 lighthouse1.jpg

Figure 2 transformed lighthouse1.jpg

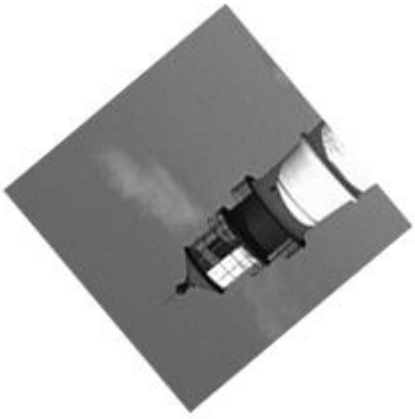


Figure 3 lighthouse2.jpg

Figure 4 transformed lighthouse2.jpg

Figure 5 lighthouse3.jpg

Figure 6 transformed lighthouse4.jpg

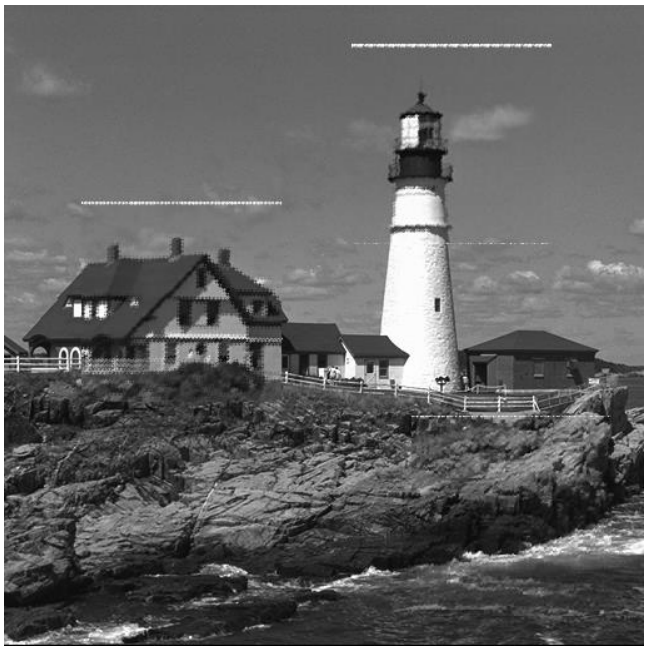


Figure 7 Final Image after Geometric Modification

(b) Spatial Warping

Figure 8 Original Image

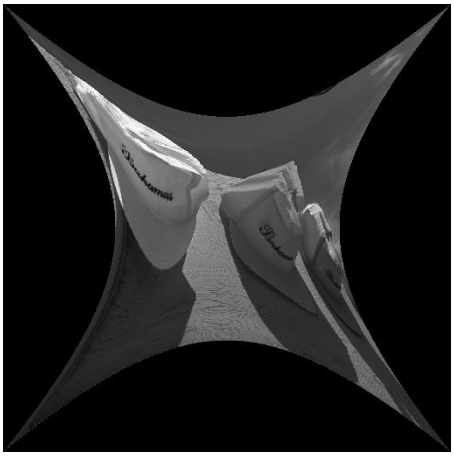


Figure 9 Warped Image

(c) Lens Distortion

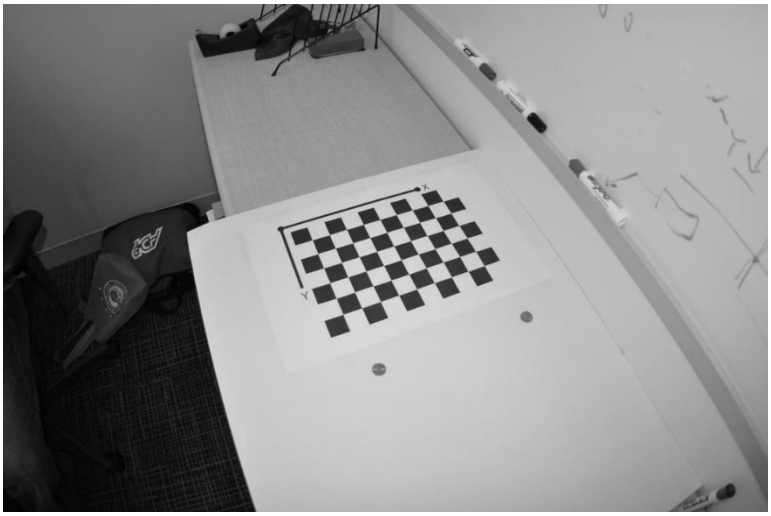


Figure 10 Original Image

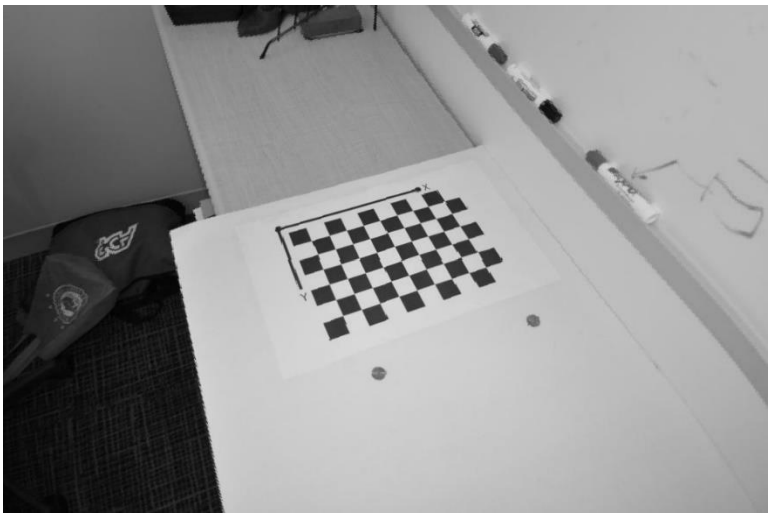


Figure 11 Undistorted Image

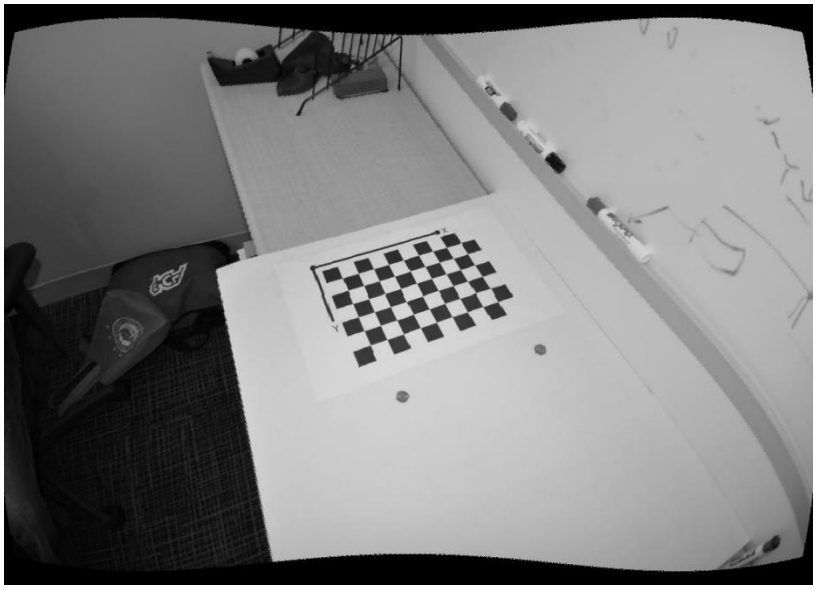


Figure 12 Corrected Image using padding

IV. Discussion

a. Hole Filling/Puzzle Matching

> Looking at the lighthouse output (figure 7), we see that, the sub images in the final image contains white edges with square boundaries. This may be due to white hole size being not entirely square. This can be resolved by scaling the pieces few pixels are larger than original. The overlapping surrounding part would be the average of puzzle piece boundary pixels and hole boundary pixels in the output image.

> On closer inspection, we also find that the sub images are little blurry. This may be because of rotation and scaling, as the operations resulted in non-integer pixel coordinates. Bilinear Interpolation was used to account for non-integer coordinates, which improved the quality to certain extent.

> To further improve quality, one can perform rotation by shearing and also apply cubic interpolation at each transformation stage.

> Rotation also caused other problems such as rotation angle not coming out right or missing image parts or parts going out of boundary. To solve this, first, depending upon required output, the rotation angle was further modified and also adjusted with respect to different sub images. Second, the rotation must be performed with respect to the sub image. This prevents sub image going out of the image boundary.

> Entire problem was implemented in python as here it is much easier to perform matrix multiplications while applying geometric transformations

b. Spatial Warping

> From the spatial warping output (figure 9), we see that the hat image is successfully warped as desired.
> The warping is curved, giving the indication of non-linear mapping between input and output.
> To further improve, one can use more control points to map input and output, resulting in higher order warping. However, calculating pseudoinverse becomes more complicated, hence in most applications second-order polynomial mapping works well along with proper interpolation.

c. Lens Distortion

> From lens distortion output we see that, the curved lines in the distorted image now appear straight.
> Distortion correction was employed using forward mapping. There is a one drawback to this. The image gets cropped a little, cutting out the borders. However, I believe this expected, since its like flattening a bent paper in a frame, which after straightening will surely go out of the frame.
> To solve, this one can add padding to the image, but that might again distort the boundaries of final image (making it curvy – shown in figure 12)
> The safest alternative would be to calculate an inverse mapping function. We obtain image intensities at ideal grid position, by creating a mapping from ideal to distorted position. This is nicely explained in reference [2]

Problem 2: Morphological Processing

I. Abstract and Motivation

- Image Morphology describes a range of techniques used for identifying, analyzing and modifying geometrical structures in an image by relying on non-linear operations related to shapes in an image.
- They rely on relative ordering of pixel values and are indifferent to numerical values, so are widely using in processing of binary images. They can also be applied to gray scale images. Since the transfer functions of gray scale images are unknown, pixel values are of no interest, thereby relying only on their ordering
- They are used in extracting components of image that can be used to represent and describe region shapes such as boundaries holes, skeletons, etc.
- Other uses include Noise removal, feature detection, objection selection, etc.

II. Approach and Procedure

- In Morphological Processing, an image is examined with a small mask or a kernel called Structuring Element which is a binary array. The Structuring element is scanned over the image by placing at all locations in image and comparing neighbourhood pixels.
- We perform a Hit or Miss Transformation where; structuring element is scanned over a binary image. If the pattern of the mask matches the state of the pixels under the mask, we get a 'hit' and an output pixel in correspondence to the center pixel of the mask is set to some desired binary value. For a pattern mismatch or 'miss', the output pixel is set to the opposite binary value.
- In this problem, we implement various basic morphological algorithms such as shrinking, thinning, skeletonizing and their applications in defect detection and object analysis.

a. Basic Morphological Processes

1) Shrinking: It is a method of reducing white pixels to a single dot or a line. In this process, we erase black pixels such that object without holes are shrinked to a single pixel and object with holes are eroded to a connected ring lying midway between hole and its nearest boundary. Shrinking is performed using 2 3x3 hit or miss transformations. Conditional and unconditional masks are applied in 2 stages iteratively until we get a final shrinked image.
   Shrinking can be logically represented as:

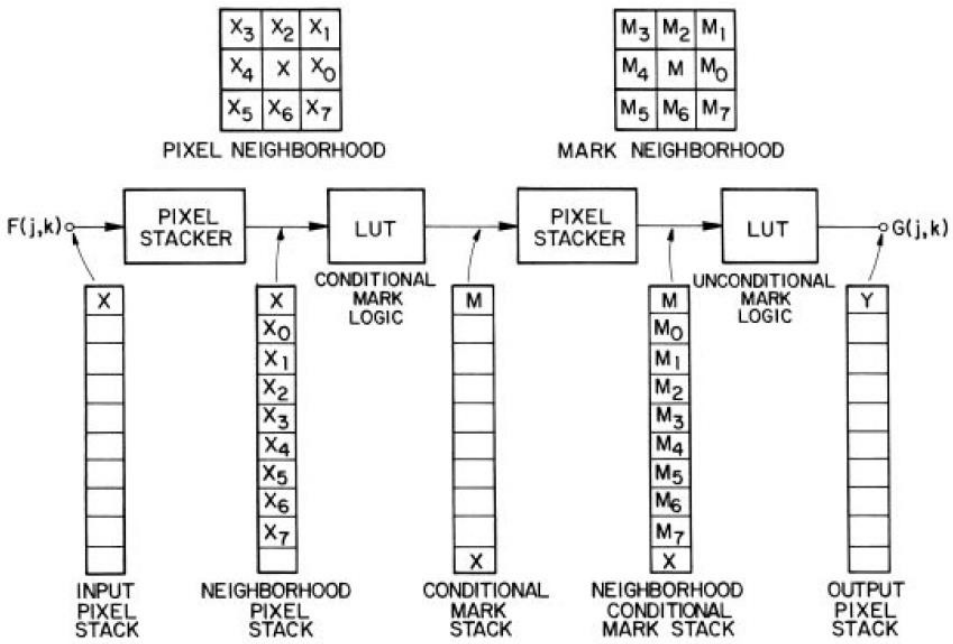
$$G(j,k) = X \cap [\overline{M} \cup P\,(M, M_0, \ldots, M_7)]$$

Where $P\,(M, M_0, \ldots, M_7)$ is erasure inhibiting logical variable defined as per unconditional masks table

The Detailed algorithm is as follows:

1. Create look up table for conditional and unconditional masks
2. Apply conditional masks across whole image to perform hit or miss operation. If there is a hit as per the stage-I look up table, reduce one layer of white pixel to black else keep as it is
3. Store the results in a new intermediate image
4. Apply unconditional masks across new image. If there is a hit, update the pixel to white else keep it to black
5. Compare the final output image with the input image
6. Repeat above process until there are no further erasures

Figure below shows a flowchart for the look up table implementation of different algorithms:



2) Thinning: It is also used to reduce white pixels in binary images. In this process, we erase black pixels such that object without holes erodes to lines and object with holes shrinks to connected rings.

Similar to shrinking, thinning is performed using 2 3x3 hit or miss transformations in which, conditional and unconditional masks are applied in 2 stages iteratively until we get a final thinned image. Here the conditional masks are less compared to shrinking, however unconditional masks remain the same.

The detailed algorithm is as follows:

1. Create look up table for conditional and unconditional masks
2. Apply conditional masks across whole image to perform hit or miss operation. If there is a hit as per the stage-I look up table, reduce one layer of white pixel to black else keep as it is
3. Store the results in a new intermediate image
4. Apply unconditional masks across new image. If there is a hit, update the pixel to white else keep it to black
5. Compare the final output image with the input image
6. Repeat above process until there are no further erasures

3) Skeletonizing: It is a process of reducing white pixels to a skeleton like remnant without harming the connectivity of original region. It is an implementation of thinning that successively erodes away pixels from the boundary until no more thinning is possible, leaving an approximate skeleton like structure. Here as well, 2 3x3 hit or miss transformations are performed in which, conditional and unconditional masks are applied in 2 stages iteratively until there is no further erosion. Here the conditional masks are less compared to shrinking and has a different unconditional masks pattern.

   The detailed algorithm is as follows:

   1. Create look up table for conditional and unconditional masks
   2. Apply conditional masks across whole image to perform hit or miss operation. If there is a hit as per the stage-I look up table, reduce one layer of white pixel to black else keep as it is
   3. Store the results in a new intermediate image
   4. Apply unconditional masks across new image. If there is a hit, update the pixel to white else keep it to black
   5. Compare the final output image with the input image
   6. Repeat above process until there are no further erasures

7. Here, we have to perform bridging operation to restore connectivity which may have been lost while encountering few patterns.

b. Defect Detection and Correction

Defect Detection is guided by analysis of image for distinguishing properties. Morphological processing can be used in identifying defects in an image and also for correcting them.

We can either use some user-defined mask based on visible defect or various standard hit or miss morphological operators can be scanned across the image to find the defect. Here, since we are looking for a black dot in a white background, we can scan the image using subtractive hit or miss operator to identify the defect.

The detailed algorithm is as follows:

1. Create a subtractive 3x3 hit or miss mask with center pixel black and neighboring pixels as white.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

2. Initialize a count variable to store number of defects
3. Apply subtractive operator across whole image to perform hit or miss operation. The mask checks for any black pixel surrounded by all white pixels.
4. If we encounter black pixel (hit) with white background, increase the count of hit. The subtractive operator used is also called as 'isolated pixel removal' where we fix the defect by erasing or replacing the black pixel value with white pixel value.
5. Save the final corrected Image
6. The final count gives the number of defects in the image

c. Object Analysis

Image analysis is one of the many applications of morphological processing. In this section, we discuss how various morphological operations can be used in inspection and counting of rice grains.

i. We first, convert the color image to gray scale
ii. For effective image binarization, we perform canny edge detection which extracts rice with connected edges.

Now we perform a series of morphological operations.

iii. We first introduce dilation, in which areas of white pixels are increased along the edges. A diamond shaped structure element is used to control thickening
iv. We then apply something called flood fill algorithm for filling border connected foreground pixels and also remove region minima not connected to border
v. Image is then eroded to reduce the areas of foreground pixels, smoothen object outlines and breaks thin connections
vi. We then perform shrinking operation
vii. Initialize count variable
viii. Iterate over shrinked image and increment count value whenever we encounter a white pixel. The final count value is the number of grains in the image
ix. For purpose of extracting size of rice grains, we use region props function on eroded image to get areas

III. Experimental Results

a. Basic Morphological Processes:

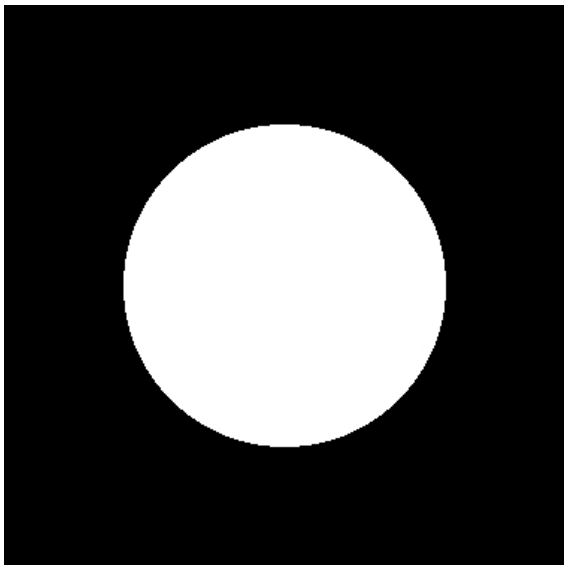


Figure 13 Pattern 1

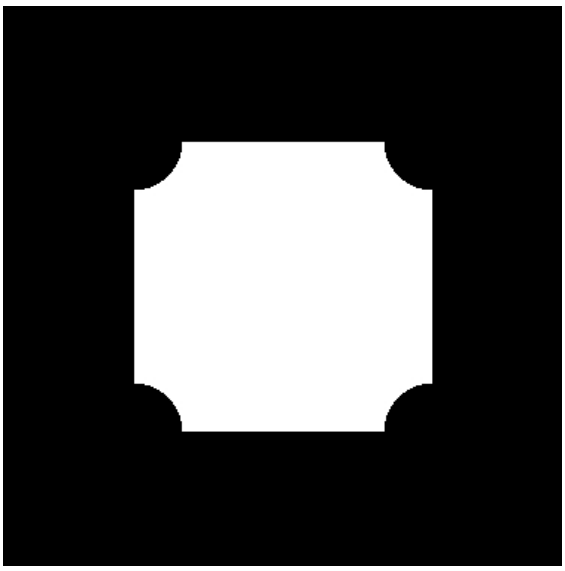


Figure 14 Pattern 2

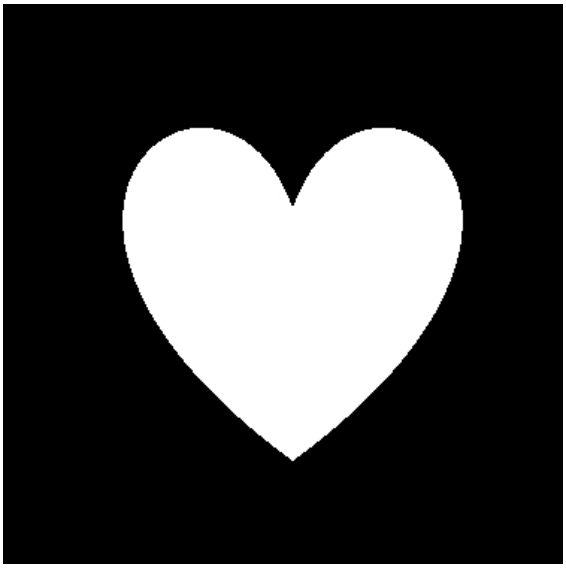


Figure 15 Pattern 3

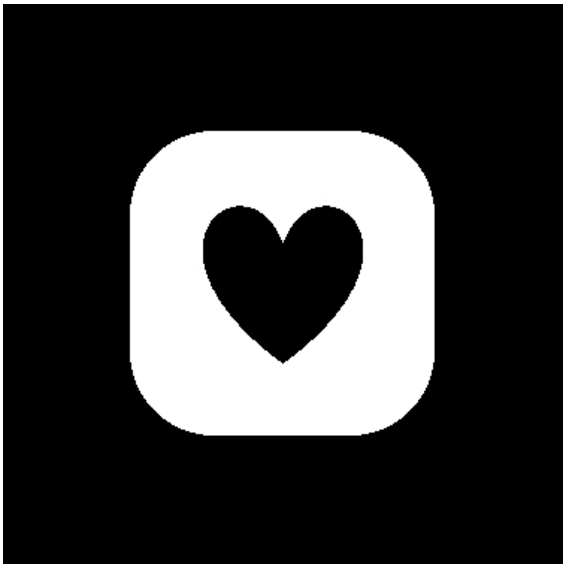


Figure 16 Pattern 4

(1) Shrinking Output:

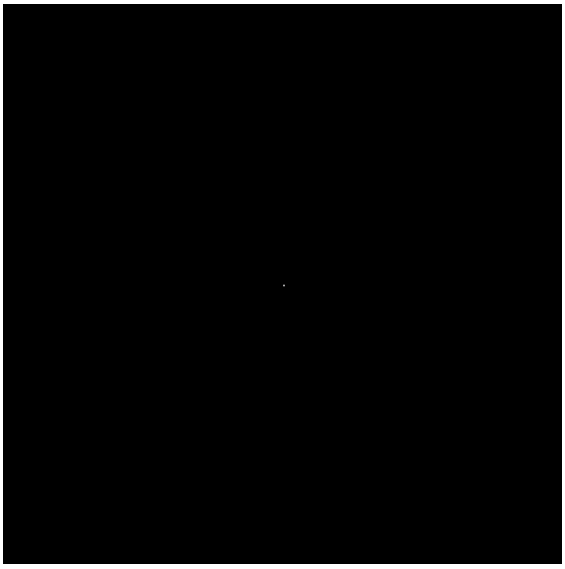


Figure 17 Pattern 1

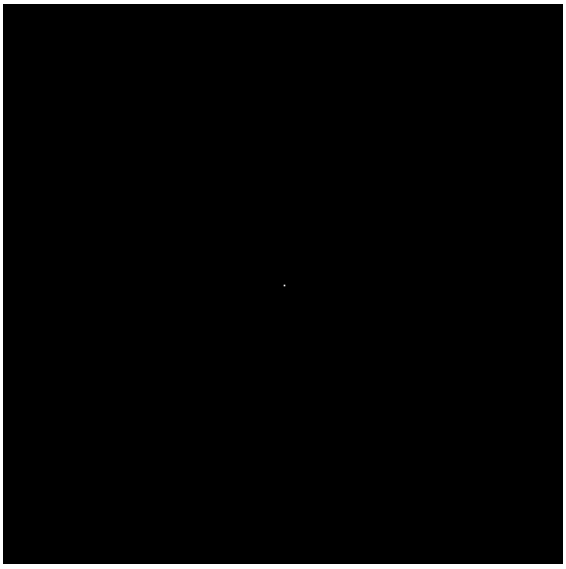


Figure 18 Pattern 2 output

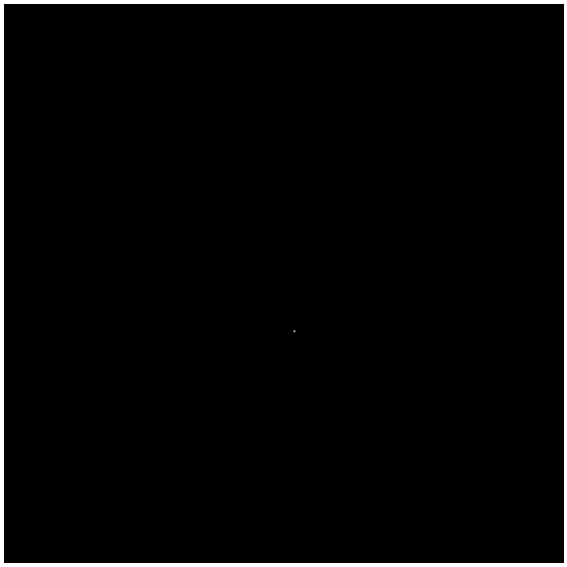


Figure 19 Pattern 3 Output

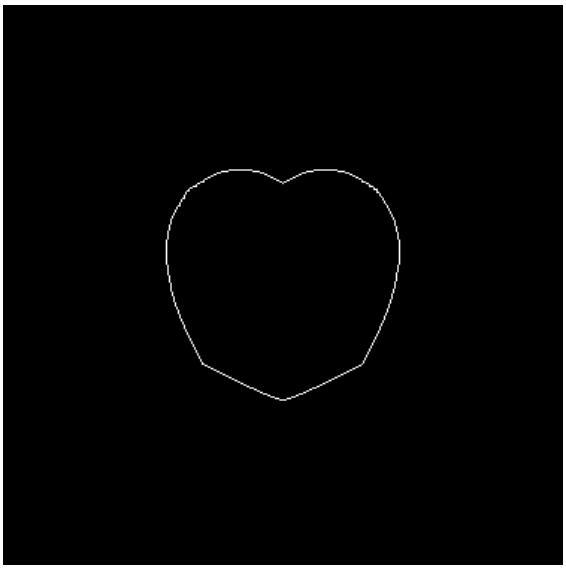


Figure 20 Pattern 4 Output

(2) Thinning Output:

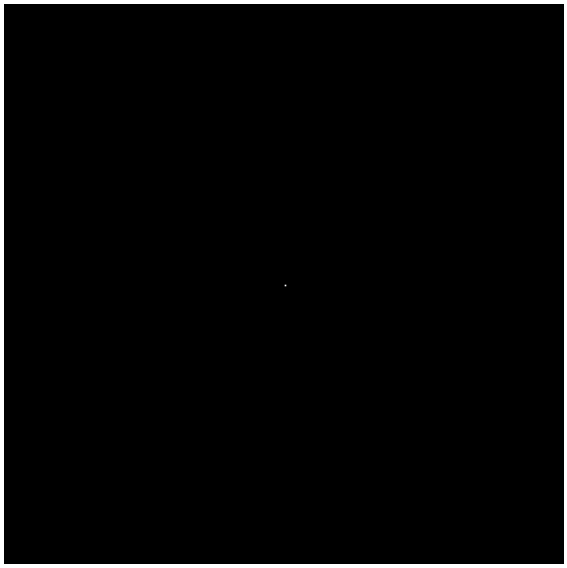


Figure 21 Pattern 1 Output

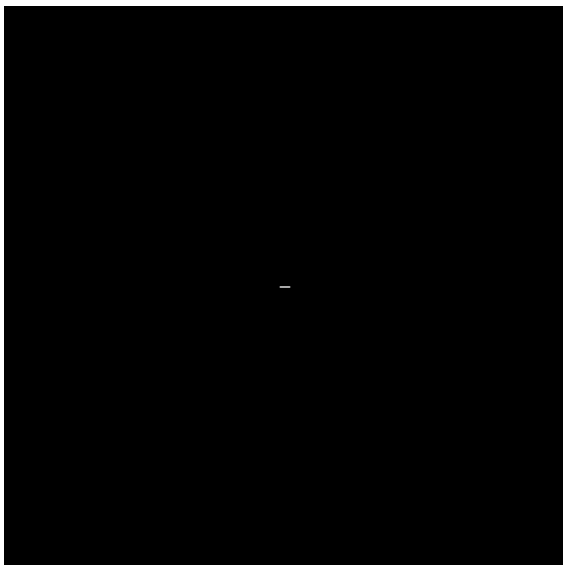


Figure 22 Pattern 2 Output

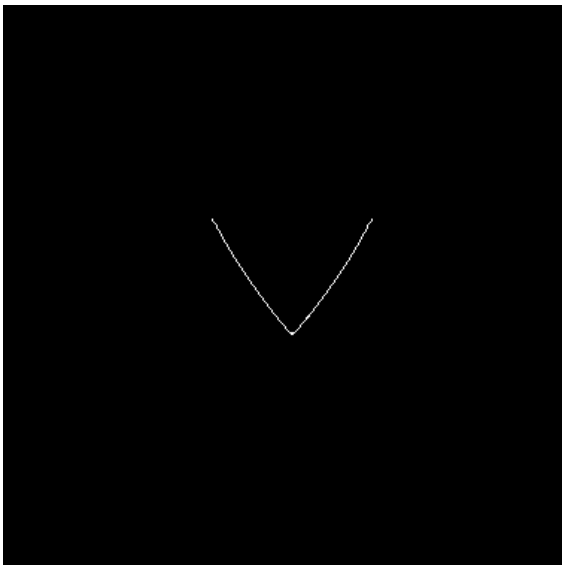


Figure 23 Pattern 3 Output

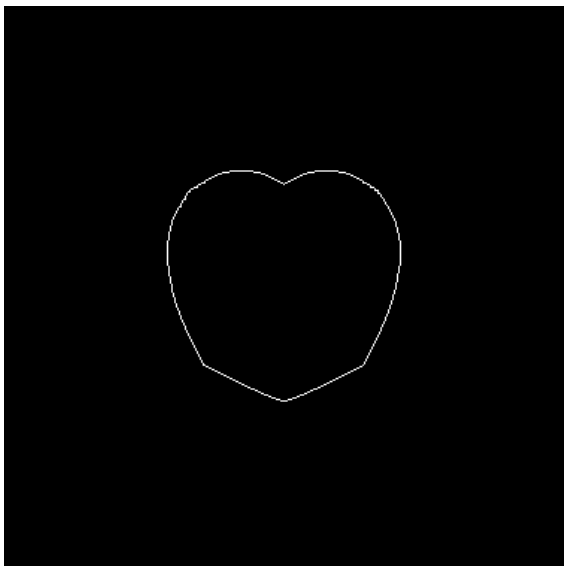


Figure 24 Pattern 4 Output

(3) Skeletonizing Output:

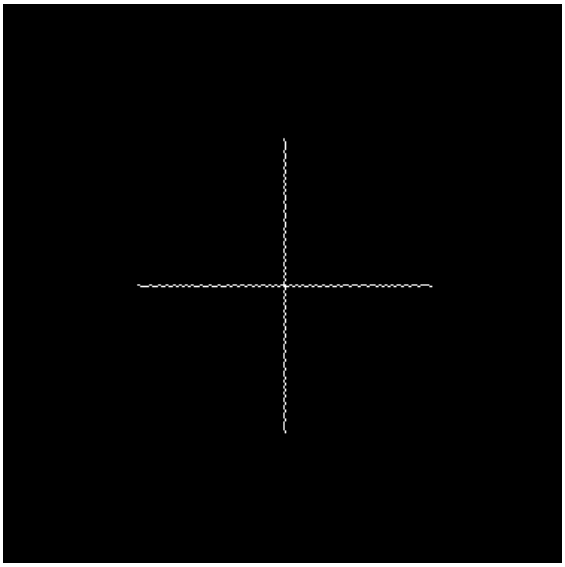


Figure 25 Pattern 1 Output

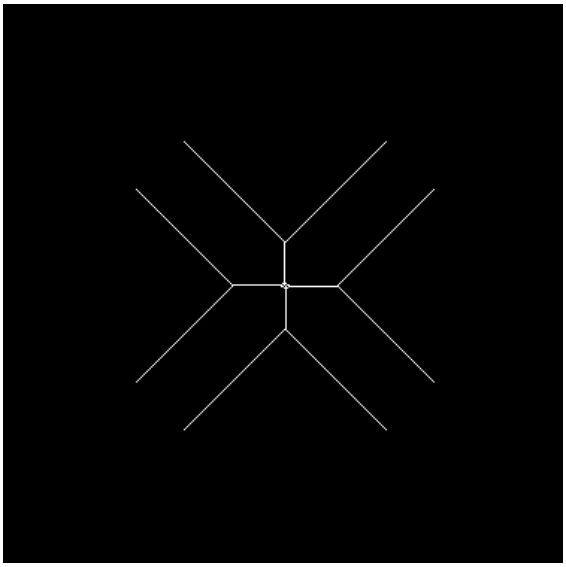


Figure 26 Pattern 2 Output

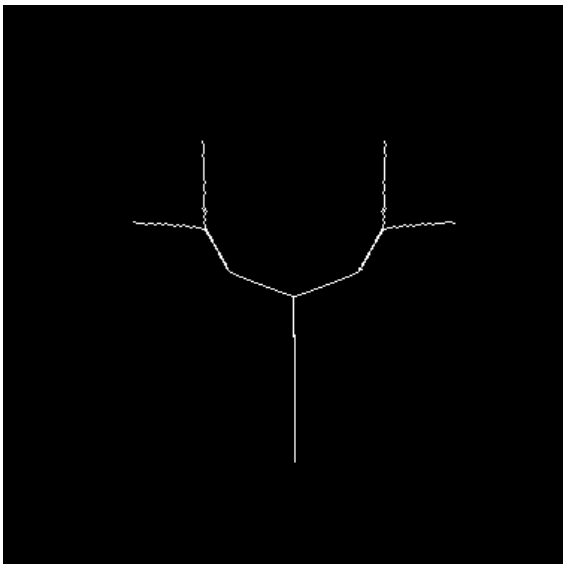


Figure 27 Pattern 3 Output

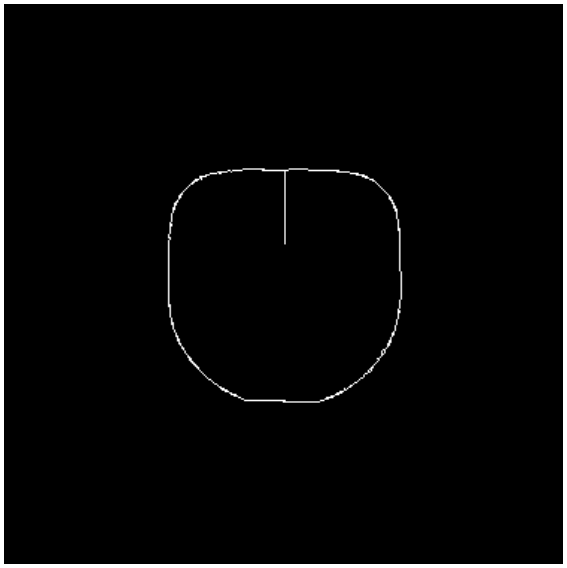


Figure 28 Pattern 4 Output

b. Defect Detection

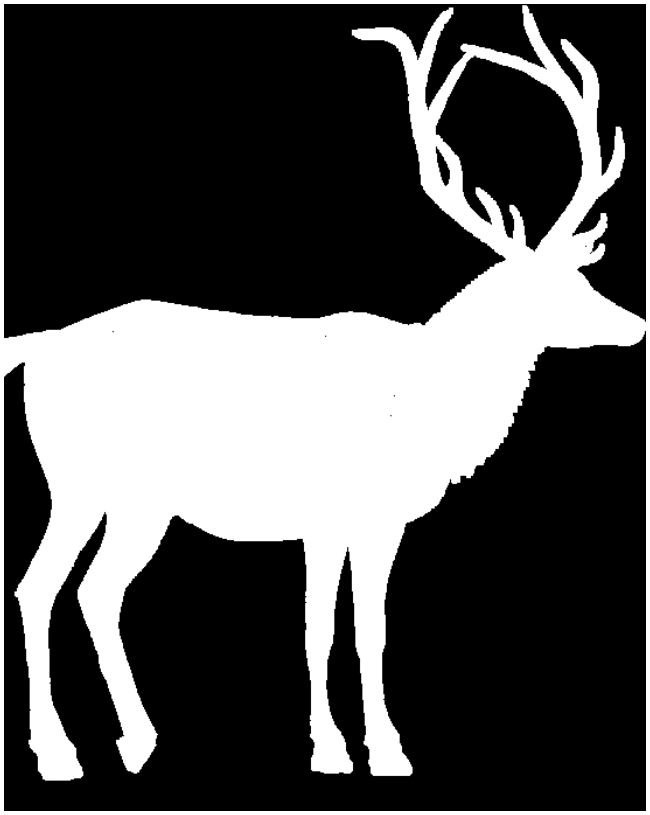


Figure 29 Original Deer Image

c. Object Analysis

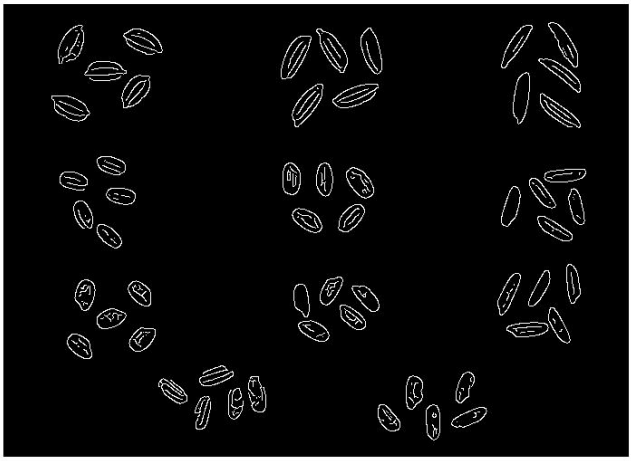


Figure 31 Binarize Rice grain using Canny Edge Detection

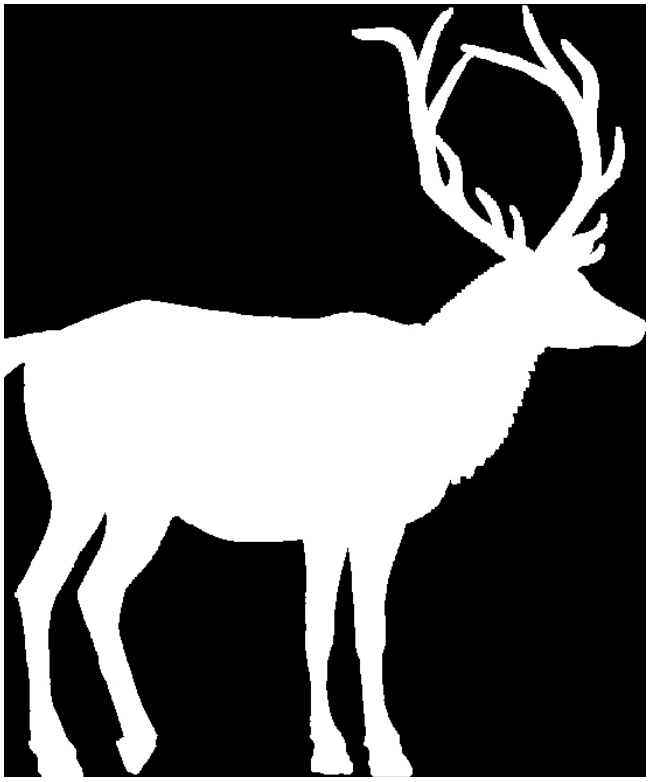


Figure 30 Corrected Deer Image

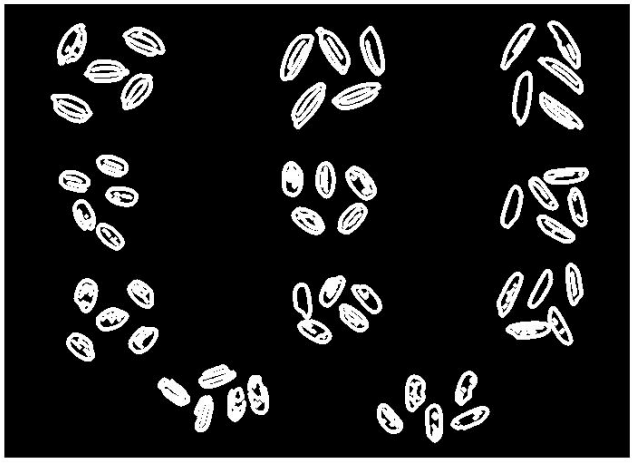


Figure 32 Dilated Rice Image

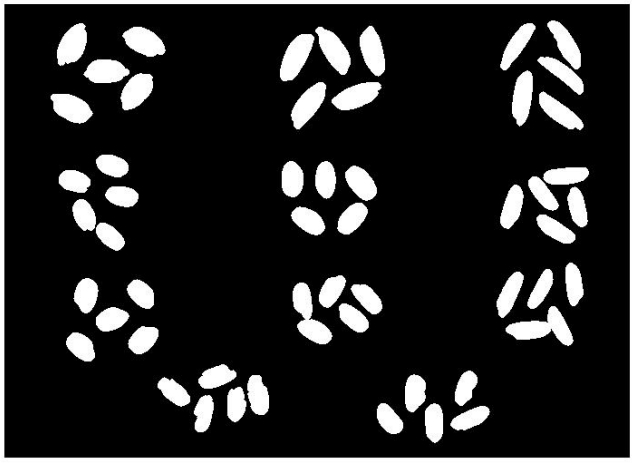


Figure 33 Hole Filled Image

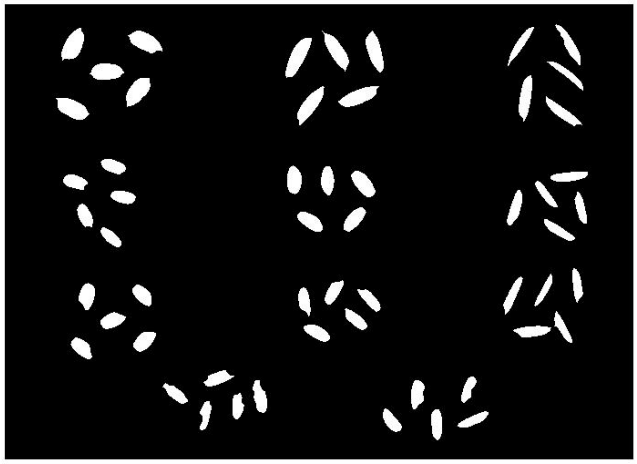


Figure 34 Eroded Rice grain

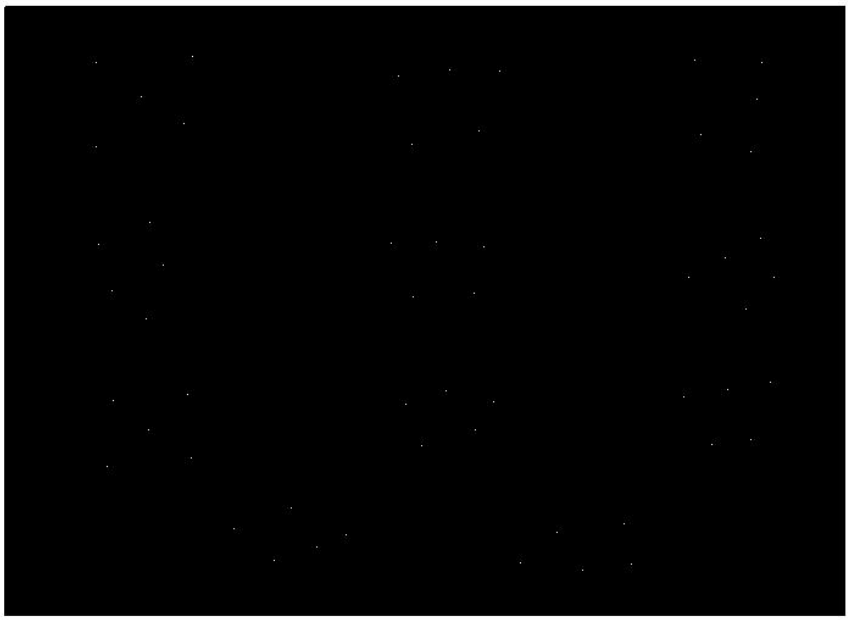


Figure 35 Shrinked Rice grain

IV. Discussion

a. Basic Morphological Processes

> As seen from the shrinking output in fig 11-14, patterns 1-3 contain no holes or rather are solid figures, hence they are eroded to single point, while as pattern 4 contains a heart shape hole enclosed in square structure, it is eroded to a heart like connected ring.

> From table below, we see shrinking is a slow, but as the pattern becomes connected with holes shrinking speed increases as shrinking happens from all side at the same time.(similar is the case for thinning and skeletonizing)

> For thinning output, since pattern 1 is a circle, it gets eroded to single point and rest of patterns are eroded to thin lines as desired. Compared to shrinking, as pattern changes, thinning operation is faster as the erosion happens only till, we get thin lines rather than single dots.

> For skeletonizing, we see the output gives out compact representation of various image patterns, without harming original size of the image patterns.

Skeletonizing is much faster compared to all, as erosion not happens completely but only till, we get topological/rough skeleton of the input image

Below is table indicating number of iterations for different operations on different patterns:

Operation/Pattern	Pattern 1	Pattern 2	Pattern 3	Pattern 4
Shrinking	108	100	142	48
Thinning	108	97	88	49
Skeletonizing	76	96	84	47

b. Defect Detection

> This was a pretty simple application of morphological processing. The subtractive hit or miss operator made it easier to identify as well as fix the defects in the image.

> The defect regions are highlighted in the figure 27. The corrected image is shown in figure 28

> The coordinates of defects are as below:

$$(47, 303), (94, 309), (476, 313), (85, 370), (432, 388)$$

c. Object Analysis

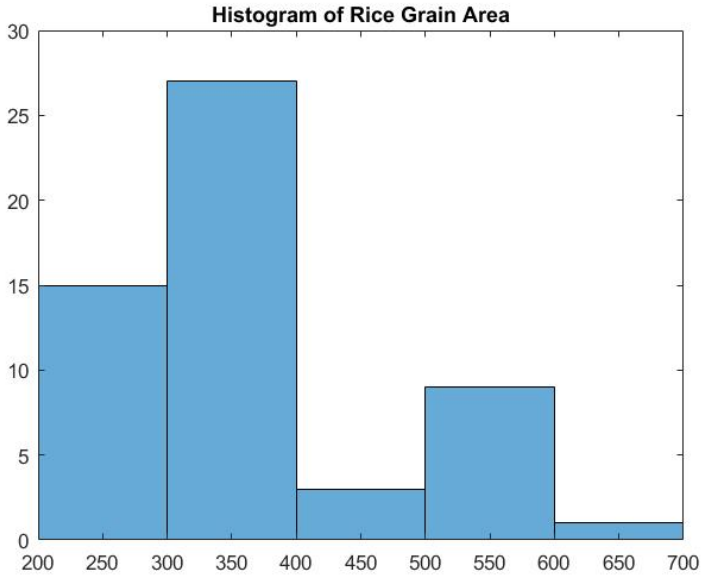
> To start with, for image binarizing various thresholding algorithms were tried. None of them could give appropriate binary image. The bottom left portion of rice grains kept missing or wasn't completely binarized as those objects were darker than foreground. So, to solve this, canny edge detection was employed, which give binarized rice grains with proper connectivity.

> Post that, a series of morphological operations were employed starting with dilation, flood fill, erosion and then shrinking. However, the erosion operator did not completely break thin connections. One grain was still connected as shown in figure. In order to remove that connectivity, I employed a custom mask as below and performed hit or miss operation.

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

> The output after each morphological operation is shown. Total number of rice grains were 55, obtained by counting white pixels in the shrinked image as shown in fig.

> For Comparison of rice grains, we do connected component labelling through inbuild MATLAB functions bwlabel () and use 'Areas' properties of regionprop () function which gives us number of pixels for distinct grains. More number of pixels means larger grain size and vice versa (The sorted area is done using sorted command on the region prop output). Below is histogram representing the same, where x-axis is grain size and y-size is grain count:



> Rice can also be further categorized on the basis of other features such as length, eccentricity or color

> This section was implemented in MATLAB as here it was easier to use inbuilt canny edge detector and also play around various inbuilt morphological filters to get the desired output

V. References

1) Digital Image Processing, Fourth Edition, William K. Pratt, Wiley-Interscience Publication
2) https://www.researchgate.net/publication/224599781_Lens_Distortion_Correction_Using_Ideal_Image_Coordinates/
3) International Journal of Modern Trends in Engineering and Research, Grading of Rice Grains Quality using Image Processing, - Dhara Desai, Prof. Nikunj Gamit, Prof. Kinjal Mistree
4) https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm
5) http://www.astro.caltech.edu/~aam/science/thesis/total/node52.html
6) http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm
7) https://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm