EE 569 HOMEWORK 3

Problem 1: Geometric Modification

(a) Geometric Transformation

1.a.1 Abstract and Motivation

Geometric image modification is widely used in Computer graphics. Shift, rotate, enlarge, scale, shrink, non-linear warp are the most common operations involved. These transformations are frequently used as pre-processing steps in applications like scanning image, to correct miss-aligned images, to create effects on images, to convert 2D to 3D images or vice versa, to view the model from different positions. With geometric transformations, the spatial relationship between pixel and image are modified.

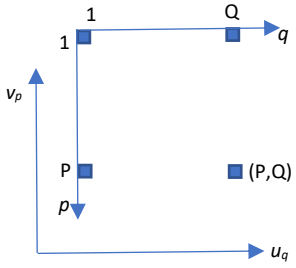
1.a.2 Approach and Procedure

Fig 1. Representation of input and output image co-ordinates

Let F(p,q) be the input image with image co-ordinates p, q and G(j,k) be the output image with image co-ordinates j,k.

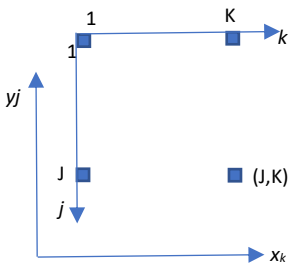
The relationship between the image co-ordinates and the cartesian co-ordinates is well established by:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & J+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} j \\ k \\ 1 \end{bmatrix} \qquad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & P+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \qquad \text{eqn(1)}$$



$(p,q) = (P,1) \longleftrightarrow (u_p, v_q)=(0.5,0.5)$

u_p, v_q are cartesian co-ordinates of input image



$(j,k) = (J,1) \longleftrightarrow (x_k, y_j)=(0.5,0.5)$

x_k, y_j are cartesian co-ordinates of output image

The basic steps involved in geometric transformations are:

1. Conversion from image co-ordinates to cartesian co-ordinates.
2. Perform operations in the cartesian co-ordinates.
3. Output co-ordinates have to be integers, while input co-ordinates can be fractional numbers. Using inverse/reverse address mapping, the fractional pixel values are estimated via bilinear interpolation of the input image.

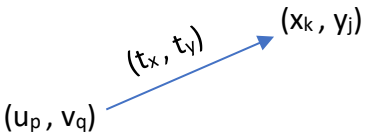
Ref 1: Operations:

1. Translational

$x_k = u_q + t_x$

$y_j = v_p + t_y$

(t_x, t_y) is the translational vector

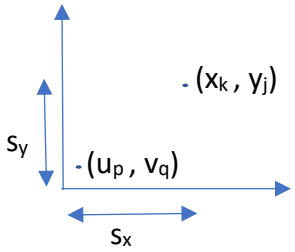


Translational Matrix $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$

2. Scaling with respect to origin

$x_k = s_x * u_q$

$y_j = s_y * v_p$

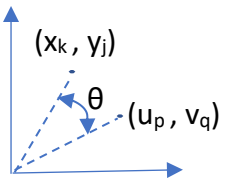


Scaling Matrix $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$

3. Rotation

$x_k = \cos(\theta) * u_q - \sin(\theta) * v_p$

$y_j = \sin(\theta) * u_q + \cos(\theta) * v_p$



Rotation Matrix $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & tx \\ sin(\theta) & cos(\theta) & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$

Ref 2: Algorithm for bilinear interpolation:

Case1: If both input image co-ordinates (p,q) are integers, extract the pixel value directly at (p,q)

G(j,k)=F(p,q)

Case2: If p is integer and q is fractional image co-ordinate, weighted average is calculated as

G(j,k)= (ceil(q)-q)*F(*p,floor(q)*)+ (q-floor(q))*F(*p,ceil(q)*)

Case3: If p is fractional and q is integer image co-ordinate, weighted average is calculated as

G(j,k)= (ceil(p)-p)*F(*floor(p),q*)+ (p-floor(p))*F(*ceil(p),q*)

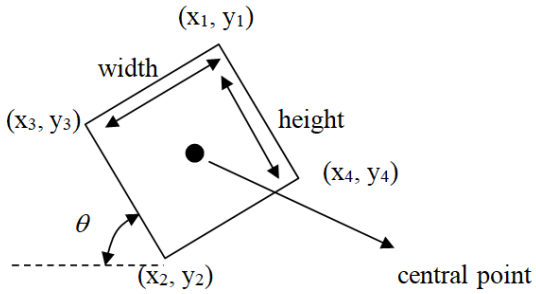
Case4: If p is fractional and q is fractional image co-ordinate, weighted average is calculated as

G(j,k)= (ceil(p)-p)*F(*floor(p),q*)+ (p-floor(p))*F(*ceil(p),q*) + (ceil(q)-q)*F(*p,floor(q)*)+ (q-floor(q))*F(*p,ceil(q)*)

Geometric Transformation:

Steps involved for fitting the sub-images in the main image are as follows:

Step 1: To identify the co-ordinates of corners in each sub-image: (Discussion 1.4.1 includes improved algorithm)



a. Since the background of the image is white (255), for each row, the number of non-white pixels is counted.
b. The occurrence of first and last non-zero location/row indicates the x1 and x2 co-ordinate of the image.
c. Since the background of the image is white (255), for each column, the number of non-white pixels is counted.
d. The occurrence of first and last non-zero location/column indicates the y3 and y4 co-ordinate of the image.
e. To find y1 and y2, across x1 and x2 rows, the non-white pixel locations/columns are identified and their corresponding mean yields y1 and y2.
f. To find x3 and x4, across y3 and y4 columns, the non-white pixel locations/rows are identified and their corresponding mean yields x3 and x4.
g. Width, height, theta and centre point are evaluated from estimated (x1,y1),(x2,y2),(x3,y3) and (x4,y4) as

width= $\sqrt{(x1-x3)^2+(y1-y3)^2}$

height = $\sqrt{(x1-x4)^2+(y1-y4)^2}$

theta = $\arctan\left(\frac{y2-y3}{x2-x3}\right)$

centre-point = $\left(\frac{x3+x4}{2}, \frac{y3+y4}{2}\right)$

Step 2: Image is tilted and scaled by performing the following :

a. For j=1:J and for k=1:K :output image co-ordinates are first converted to output cartesian co-ordinate by inverse of eqn(1)
b. theta= theta' + constant is estimated with theta' from the co-ordinate extraction and constant manually by looking at the image orientation. For each of the subimages, constants are as follows:
   1. lighthouse1: constant=(3*pi)/2
   2. lighthouse2: constant=pi
   3. lighthouse3: constant=0
c. Compound operation is performed on image as follows:
   1.Translational to centre of output image co-ordinate
   2.Translation to origin in output cartesian co-ordinate
   3. Rotate by theta
   4. Scale with respect to origin
   5. Shift back to centre of output image co-ordinate

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \left[\begin{bmatrix} 1 & 0 & \text{-tx2} \\ 0 & 1 & \text{-ty2} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \text{sx} & 0 & 0 \\ 0 & \text{sy} & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} cos(\theta) & \text{-}sin(\theta) & tx \\ sin(\theta) & cos(\theta) & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & \text{tx2} \\ 0 & 1 & \text{ty2} \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & \text{tx1} \\ 0 & 1 & \text{ty1} \\ 0 & 0 & 1 \end{bmatrix} \right]^{-1} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Centre of the image co-ordinates (xy_centre(x,y)) are converted to cartesian co-ordinates by inverse of eqn(1)

tx1= xy_centre(x)-centre-point(x);

ty1= xy_centre(y)-centre-point(y);

tx2= - xy_centre(x)

ty2= - xy_centre(y)

sx=160/width

sy=160/height

d. (P,Q) input image co-ordinates are calculated from (u,v) input cartesian co-ordinate using eqn(1)

e. If (P,Q) input image co-ordinates are within the size of the image, the pixel intensity value is obtained from bilinear interpolation algorithm mentioned in Ref 2 of 1.2.1 else pixel intensity value is assigned as 255.

f. The same procedure is carried out for all the lighthouse1~3 images.

Step3: To find the co-ordinates of the holes:

a. Through raster parsing, at the encounter of each white pixel (intensity=255) at (*i,j*) position.

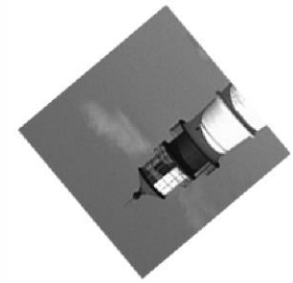
b. *i*+159 and *j*+159 should be within the size of image boundary, only then check if P(i:i+159,j:j+159) is equal to 160*160 matrix with intensity 255.

c. If the above condition is met, store (*i,j*) as the top- left corner.

Step4: Each of the 160*160 modified lighthouse1~3 images are copied to the locations of top-left corner positions of the lighthouse image.

1.a.3 Results

lighthouse~2 original image



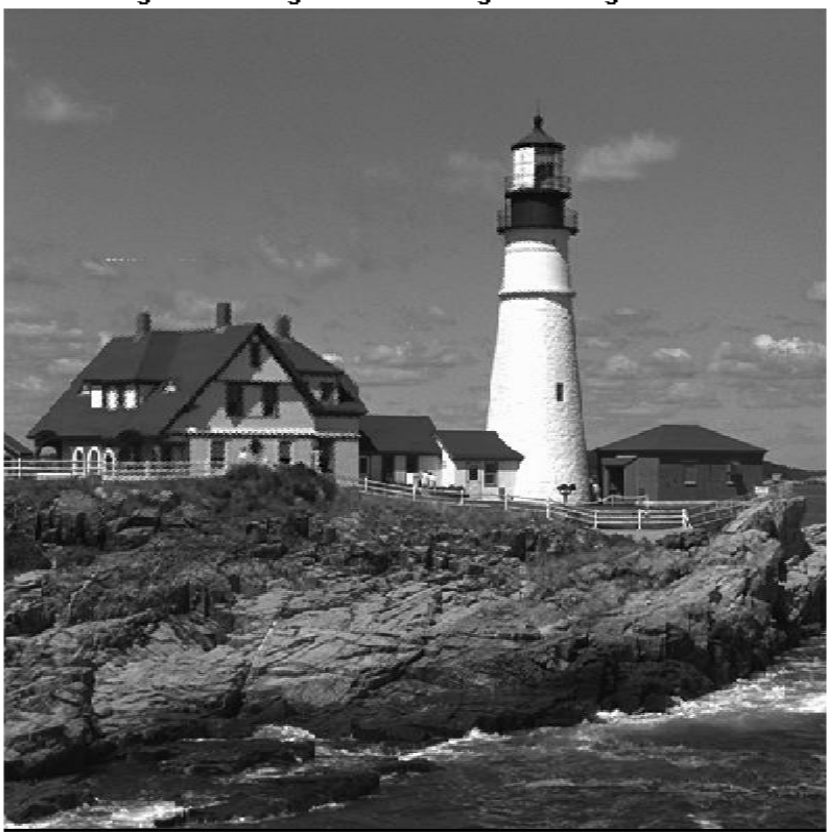
lighthouse~2 modified image of 160*160

lighthouse~3 original image

lighthouse~3 modified image of 160*160

lighthouse image

lighthouse image after correcting the scaling factor



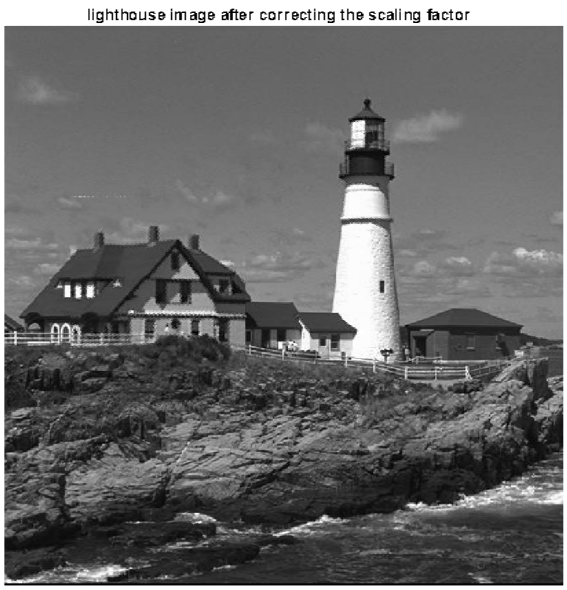
lighthouse image after improving the corner point extraction algorithm

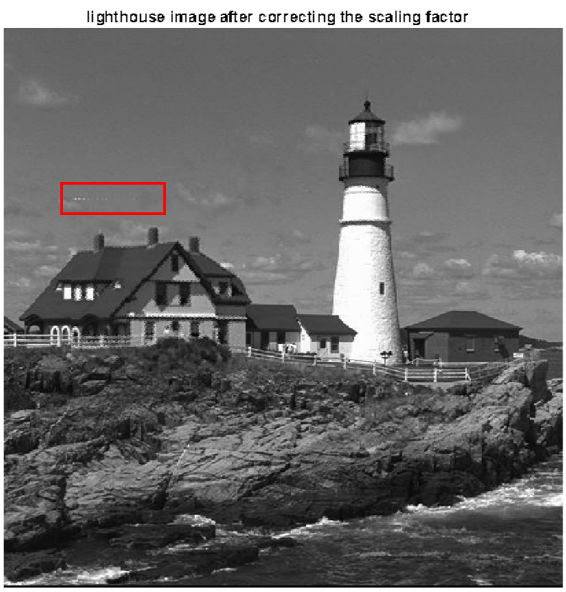
1.a.4 Discussion

1. The lines along the edges of the filling images after applying all the necessary geometric modification, is due to the near approximation/ estimation of the corner points which determines the degree of rotation and the scaling.

2.Following are the methods to remove the lines along the edges

a. We need to extract the 160*160 image from modified lighthouse~1-3, but the size of the image is 256*256, with centre at position(128.5,128.5).Extraction should start from top-left corner (48.5,48.5) which is not possible(fractional), we approximate starting from (48,48).But the image has to be scaled to 160.6,160.5 for the approximation we have done. This avoids the line along the edges.





b. In the above image even after correcting the scalar factor, there are edges along the boundary which can be improved by correcting the corner point extraction algorithm.

lighthouse image after improving the corner point extraction algorithm

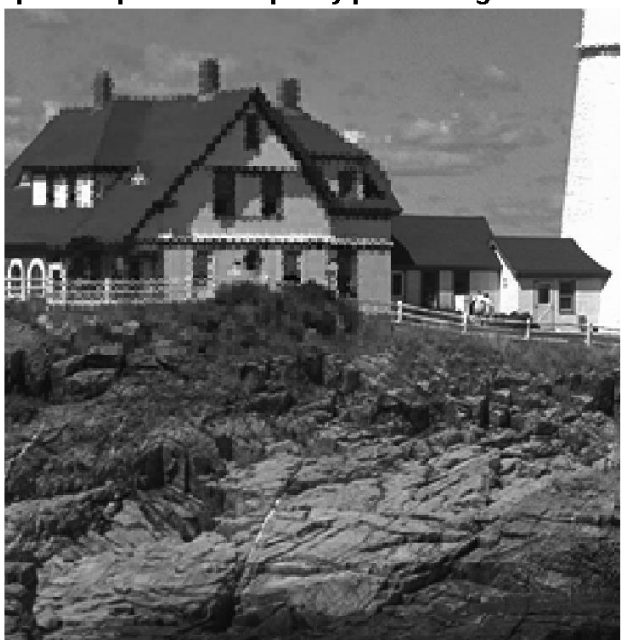
The updated algorithm includes the first step of thresholding, few pixels which are close to 255 cause mis-identification of the corner points. Therefore, a threshold of 249 is chosen, if pixel intensities are greater than the threshold, they are scaled to 255, and the remaining procedure is same as 1.a.2. Step1 a~g.

3. Bilinear interpolation plays the key role in the quality of the image output. Weighted average (Ref 2 of 1.a.2) renders better quality than simple average of fractional indexes. The image results in aliasing effect, artifacts, zipper effect.

4. The compound operations performed in Step 2c of 1.2.1 in a single step yields better results than splitting the operations by carrying one at a time, since bilinear interpolation each time will hinder the quality of the output image as shown below.

Compound operation in a single step

Compound operation is split by performing one at a time



(b) Spatial Warping

1.b.1 Abstract and Motivation

Image warping is in essence a transformation that changes the spatial configuration of an image. Warping may be used for correcting image distortion as well as for creative purposes. Images may be viewed as if they had been projected onto a curved or mirrored surface or are partitioned into polygons and each polygon distorted. Spatial warping have applications in image mosaicing and image blending. Image mosaicing involves the spatial combination of a set of partially overlapped images to create a larger image of a scene. Image blending is a process of creating a set of images between a temporal pair of images such that the created images form a smooth spatial interpolation between the reference image pair

1.b.2 Approach and Procedure

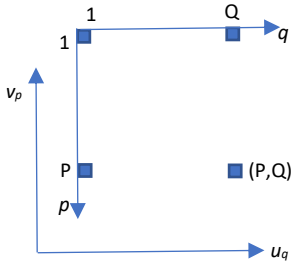
Fig 2. Representation of input and output image co-ordinates

Let F(*p,q*) be the input image with image co-ordinates *p, q* and G(*j,k*) be the output image with image co-ordinates *j,k*.

The relationship between the image co-ordinates and the cartesian co-ordinates is well established by:

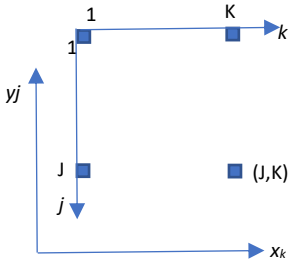
The relationship between the image co-ordinates and the cartesian co-ordinates is well established by:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & J+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} j \\ k \\ 1 \end{bmatrix} \qquad \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -0.5 \\ -1 & 0 & P+0.5 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \qquad \text{eqn(1)}$$



$(p,q) = (P,1) \longleftrightarrow (u_p, v_q)=(0.5,0.5)$

u_p , v_q are cartesian co-ordinates of input image



$(j,k) = (J,1) \longleftrightarrow (x_k, y_j)=(0.5,0.5)$

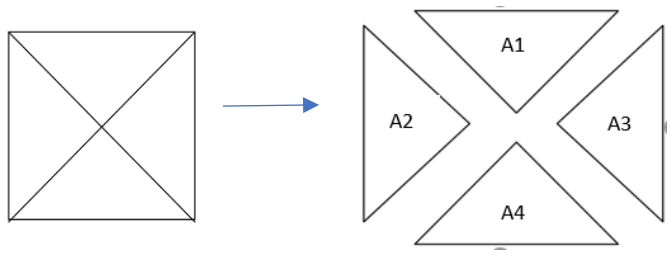
x_k , y_j are cartesian co-ordinates of output image

The input image cartesian co-ordinates and the output image cartesian co-ordinates are related as a non-linear function as follows:

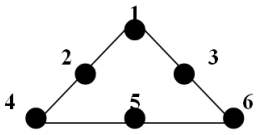
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a0 & a1 & a2 & a3 & a4 & a5 \\ b0 & b1 & b2 & b3 & b4 & b5 \end{bmatrix} * \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix} \qquad \text{eqn(2)}$$

Spatial warping is performed as follows:

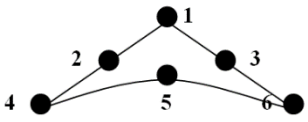
1. Input image is divided into 4 triangles and for each triangle, the co-efficients a0~a5, b0~5 are calculated.



2. The output (*j,k*) and input image (*p,q*) co-ordinates for 6 points on each triangle are converted to corresponding output (*x,y*) and input (*u,v*) cartesian co-ordinates using eqn(1)



Input image (*p,q*)



Output image(*j,k*)

3. The co-efficients are estimated by inverse operation using eqn(2)

$$\begin{bmatrix} a0 & a1 & a2 & a3 & a4 & a5 \\ b0 & b1 & b2 & b3 & b4 & b5 \end{bmatrix} = \begin{bmatrix} u1 & u2 & u3 & u4 & u5 & u6 \\ v1 & v2 & v3 & v4 & v5 & v6 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x1 & x2 & x3 & x4 & x5 & x6 \\ y1 & y2 & y3 & y4 & y5 & y6 \\ x1^2 & x2^2 & x3^2 & x4^2 & x5^2 & x6^2 \\ x1y1 & x2y2 & x3y3 & x4y4 & x5y5 & x6y6 \\ y1^2 & y2^2 & y3^2 & y4^2 & y5^2 & y6^2 \end{bmatrix}^{-1}$$

4. for j=1:height and for k=1:width of the image, the output image co-ordinates(j,k) are converted to output cartesian co-ordinates (x,y) using eqn(1)
5. Depending on the triangle the output image co-ordinate falls into, the a0~5 and b0~5 co-efficients are determined, thus obtaining the corresponding input cartesian co-ordinates(u,v).
   Reference points used are : (N= size of image; 128= height of the arc)

Traingle	Input image reference point (P,Q)
A1	[1;1;1]
	[1;0.5*(N+1);1]
	[1;N;1]
	[0.5*(1+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(1+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A2	[1;1;1]
	[0.5*(N+1);1;1]
	[N;1;1]
	[0.5*(1+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A3	[1;N;1]
	[0.5*(N+1);N;1]

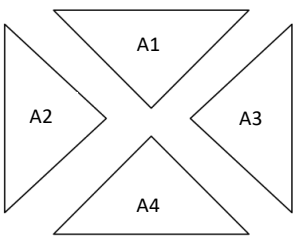
	[N;N;1]
	[0.5*(1+0.5*(N+1));0.5*(N+0.5*(N+1));1] [0.5*(N+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A4	[N;1;1]
	[N;0.5*(N+1);1]
	[N;N;1]
	[0.5*(N+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]

(N= size of image; 128= height of the arc)

Traingle	Output image refernce point (J,K)
A1	[1;1;1]
	[128;0.5*(N+1);1]
	[1;N;1]
	[0.5*(1+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(1+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A2	[1;1;1]
	[0.5*(N+1);128;1]
	[N;1;1]
	[0.5*(1+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A3	[1;N;1]
	[0.5*(N+1);N-128;1]
	[N;N;1]
	[0.5*(1+0.5*(N+1));0.5*(N+0.5*(N+1));1] [0.5*(N+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]
A4	[N;1;1]
	[(N-128);0.5*(N+1);1]
	[N;N;1]
	[0.5*(N+0.5*(N+1));0.5*(1+0.5*(N+1));1]
	[0.5*(N+0.5*(N+1));0.5*(N+0.5*(N+1));1]
	[0.5*(N+1);0.5*(N+1);1]

6. The criteria for identifying the triangle to which the output image co-ordinate (j,k) belongs:

```
if(J>=K && (K<=(N-J+1)))        A=A2;
elseif(J>=K && (K>=(N-J+1)))_A=A4;
elseif(J<=K && (K<=(N-J+1)))  A=A1;
elseif(J<=K && (K>=(N-J+1)))  A=A3;
```



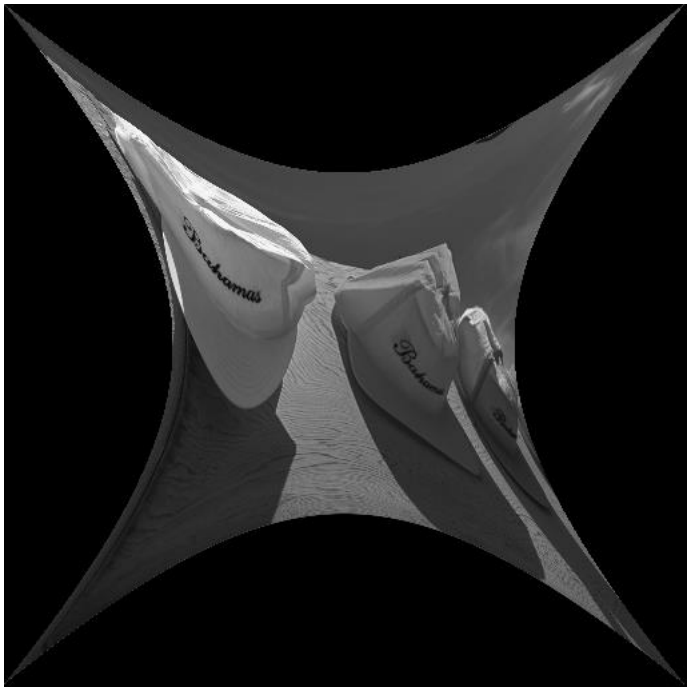
$A = \begin{bmatrix} a0 & a1 & a2 & a3 & a4 & a5 \\ b0 & b1 & b2 & b3 & b4 & b5 \end{bmatrix}$ co-efficents estimated from the corresponding triangle points.

7. Input image co-ordinates are obtained through inverse mapping of eqn(1) and pixel intensity at (p,q) is obtained using bilinear interpolation (Ref 2: of 1.a.2) and if (p,q) is out of index, the pixel intensity is assigned 0.

1.b.3 Results

Input Image

Output Image



1.b.4 Discussion

1. The co-efficient matrix for 4 triangle:

A1

2x6 double

	1	2	3	4	5	6	7
1	-4.5475e-11	1.0000	2.0162e-13	1.7251e-16	2.2556e-16	-2.5847e-16	
2	2.2510e-10	3.9379	-2.9379	-0.0077	1.5127e-15	0.0077	

A2

2x6 double

	1	2	3	4	5	6	7
1	1.1301e-10	4.9379	-3.9379	-0.0077	-1.8258e-16	0.0077	
2	-3.2067e-12	1.1768e-14	1.0000	-1.9602e-16	3.9001e-18	-2.5153e-17	

A3

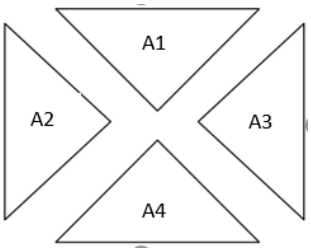
2x6 double

	1	2	3	4	5	6	7
1	-3.8222e-10	-3.0314	4.0314	0.0079	-1.0746e-15	-0.0079	
2	-8.3446e-11	1.9185e-13	1.0000	-2.9490e-17	-1.2341e-16	-8.6814e-17	

A4

2x6 double

	1	2	3	4	5	6	7
1	1.0882e-11	1.0000	-1.5432e-13	-2.9910e-16	-2.4758e-16	5.7419e-16	
2	-2.5426e-11	-4.0314	5.0314	0.0079	5.0644e-16	-0.0079	



2. For higher order polynomials, calculating the inverse or pseudo inverse of matrix A accurately may arise concerns. Computational complexity will increase for higher order polynomials.
3. If the image is further decomposed into smaller triangles, more A co-efficients have to be derived, which is computationally expensive and time-consuming.
4. The limitation of the process is, it cannot be generalized for any type of image like non-square image.
5. Bilinear interpolation will lead to artifacts, zipper effect.

(c) Lens Distortion Correction

1.c.1 Abstract and Motivation

Lens Distortion correction are often used as a pre-processing step for image registration. Lens distortions are phenomena that prohibit applying the simple pinhole camera principle in most of photogrammetric applications. Distortions belong to optic deficiencies called aberrations that cause a degradation of the final image. In contrary to other aberrations, distortions do not affect quality of the image but have a significant impact on the image geometry. Radial distortion has a significant influence on the image geometry. Radial distortion is a deficiency in straight lines transmission. The effect of radial distortion is that straight lines are bended as general curves and points are moved in the radial direction from their correct position. Together with a spatial transformation, the correction of radial distortion is the key step in the image rectification.

1.c.2 Approach and Procedure

Radial distortion often happens when an image is captured on a non-flat camera's focal plane. The rlationship between the actual image and its distortion is as follows:

$x_d = x\,(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$ eqn(1)

$y_d = y\,(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$ eqn(2)

where x, y are undistorted pixel locations, K1, K2, K3 are called radial distortion coefficients of the lens, and $r^2 = x^2 + y^2$. The x, y, xd , yd are defined in camera coordinate system, to get these values from a given digitized image, you need to convert the pixel locations from the image coordinate (u, v) to the camera coordinate(x, y)as follows:

$$x = \frac{u - uc}{fx} \qquad y = \frac{v - vc}{fy} \qquad \text{eqn(3)}$$

where (uc, vc) is the center of the image, fx and fy are the scaling factors. To recover the undistorted image, the (x, y), given the (xd, yd), the key lies on finding the inverse function so that

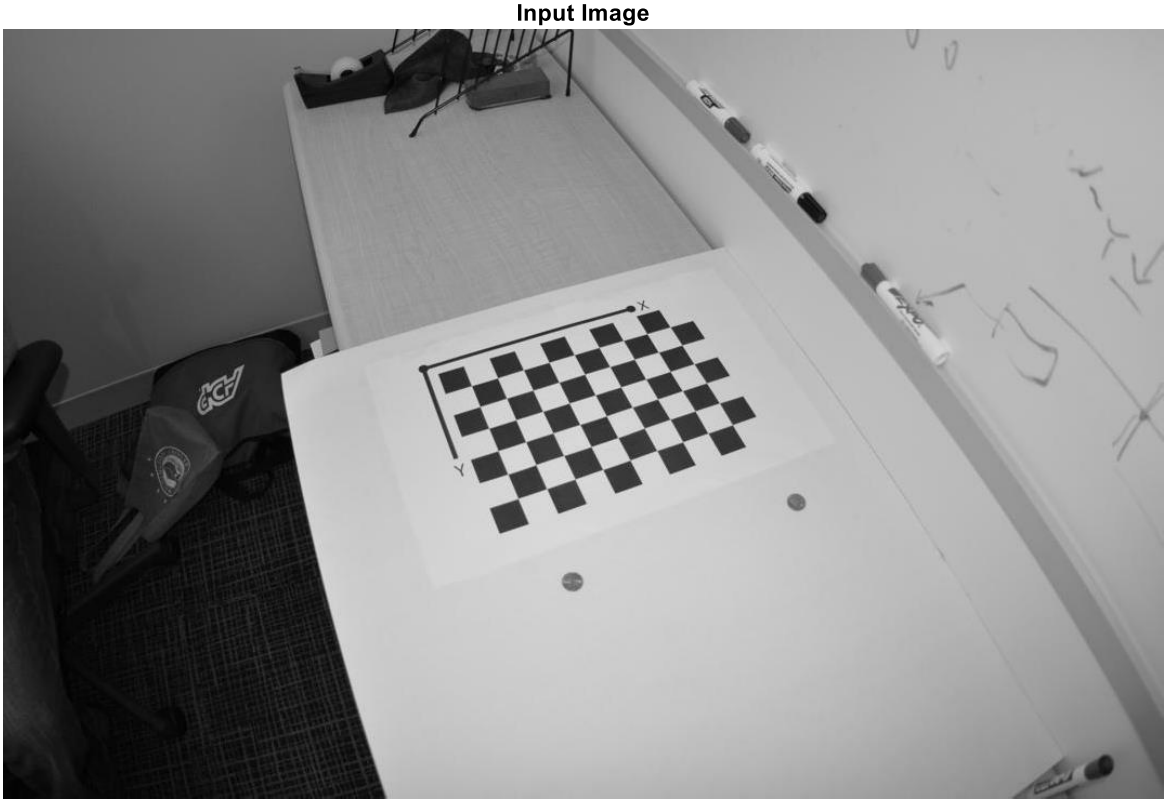
$x = f(x_d, y_d) \quad y = g(x_d, y_d)$ eqn(4)

However, there is no exact inverse function for Eq. [1] since the forward mapping from (x, y) to (xd, yd) is not linear. A common solution is first to project (x, y) to (xd, yd) which ends up in triplets of (x, y, xd) and (x, y, yd). Given that K1 = -0.3536, K2= 0.1730, K3= 0, fx= fy= 600.

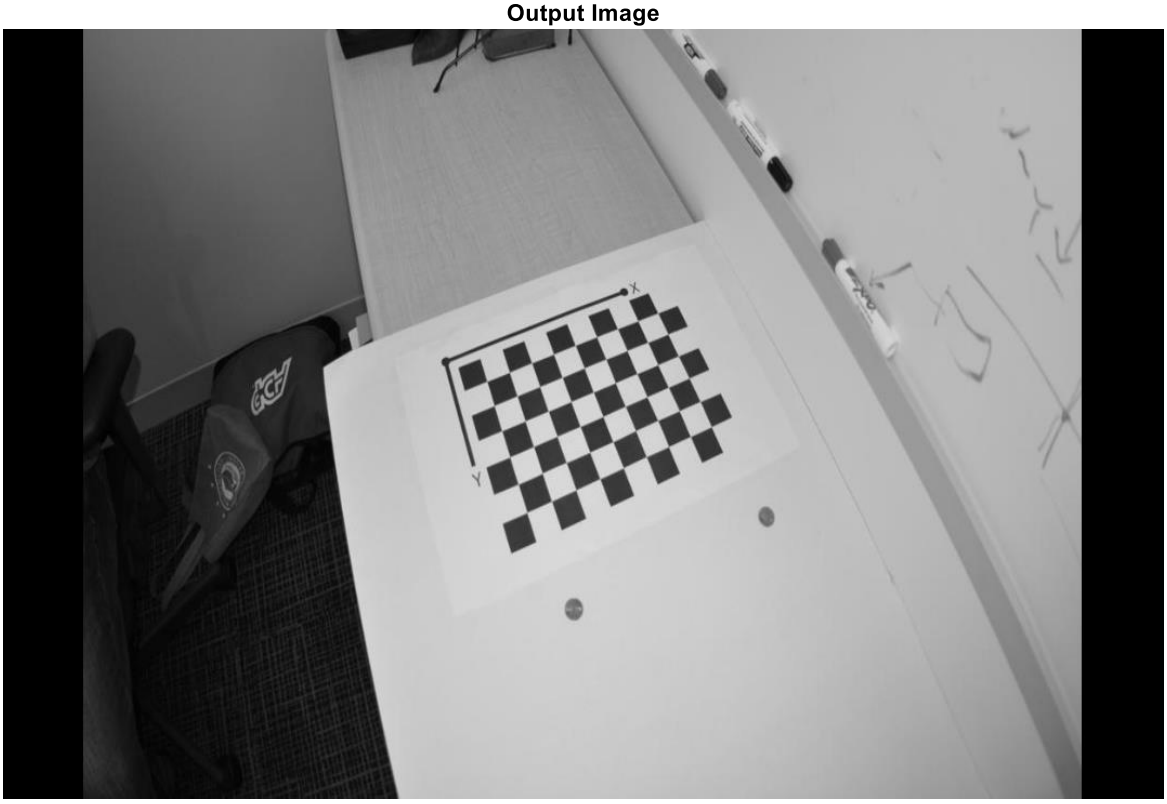
Step 1: (uc,vc) are calculated as 0.5*(height+1) and 0.5*(width+1), since height and width are even.
Step 2: For each output image co-ordinate(u,v), corresponding camera co-ordinate is calculated using eqn(3).
Step 3: xd,yd are also calculated using forward transformation provided in eqn(1) and eqn(2)
Step 4: Perform linear regression on(x,y,xd) and (x,y,yd) as Y=XB where Y is xd/yd and X=[ones x y]
Step 5: From the obtained co-efficient matrix, derive an expression for x and y in-terms of xd,yd and the co-efficients.
Step 6: Each image co-ordinate (u',v') is converted to camera co-ordinate using eqn(3). x and y are calculated from step 5. (x,y) camera co-ordinate are converted back to image co-ordinate(u,v) using the inverse function in eqn(3)
Step 7: Bilinear interpolation is performed as in section 1.2.1, if the pixel index is beyond the size of image, 0 intensity is assigned.

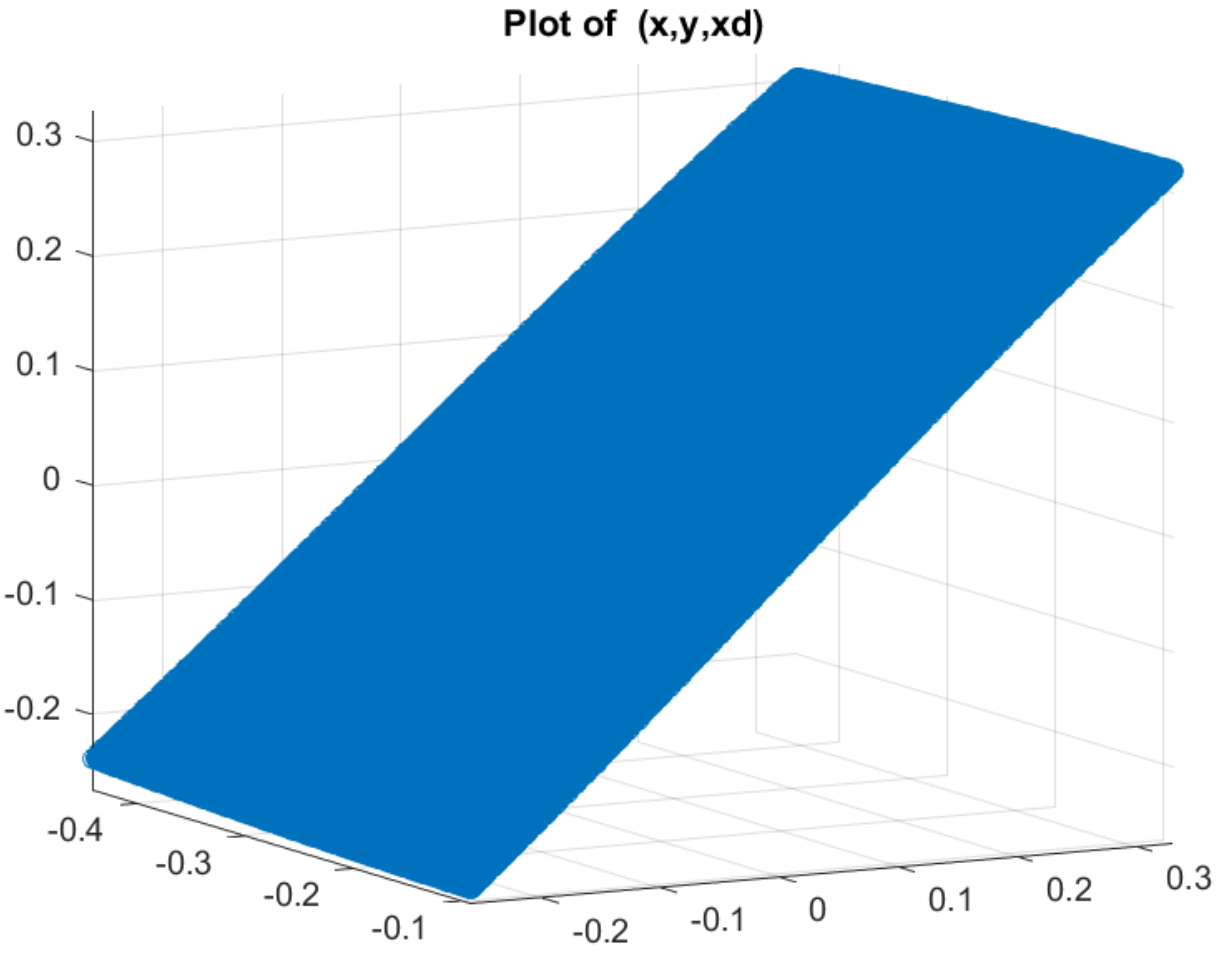
1.c.3 Results

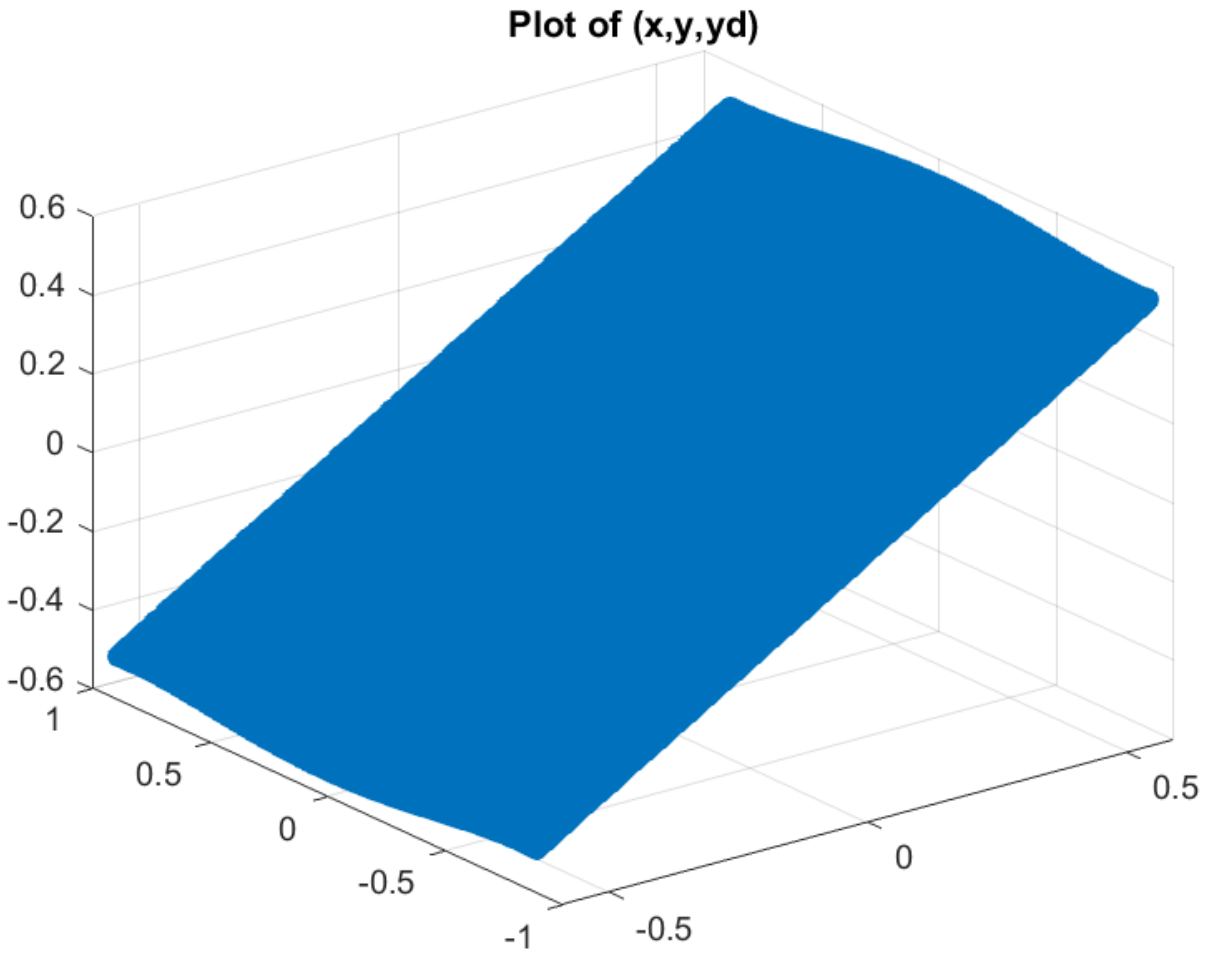
Input Image



Output Image




Plot of (x,y,xd)
0.3
0.2
0.1
0
-0.1
-0.2
-0.4
-0.3
-0.2
-0.1
-0.2
-0.1
0
0.1
0.2
0.3



1.c.4 Discussion

1. The co-efficients of the linear regression are as follows:

xd = 4.24374930874109e-16 + 0.881913888254870*x + 3.26621323246559e-18 * y;
yd= -1.01753439871659e-16 +1.07580397920992e-17*x + 0.860142087187569 *y ;

2.Classical linear regression, weighted least squares, singular value decomposition are few other methods that can be used to find the inverse function.

3. The division model and it's revision proposed by Fittzgibbon can be used to formulate the inverse mapping for fewer terms than the polynomial model, but the method fails in accuracy to deliver for high distortion lenses, since it may be necessary to include higher order in the distortion model.

4. Linear regression models the relation between a dependent, or response, variable y and one or more independent, or predictor, variables $x_1, \ldots, x_n$. Simple linear regression considers only one independent variable using the relation

$$y = \beta_0 + \beta_1 x + \epsilon,$$

where β_0 is the y-intercept, β_1 is the slope (or regression coefficient), and ϵ is the error term.

Start with a set of n observed values of x and y given by (x_1, y_1), (x_2, y_2), ..., (x_n, y_n). Using the simple linear regression relation, these values form a system of linear equations. Represent these equations in matrix form as Y=XB

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}.$$

The relation is now Y=XB which minimizes ||Y-Xb||
Where B=inv(X)*Y if inverse exists or it is B=pinv(X)*Y (pinv is pseudo-inverse)

5.Weighted linear regression can be performed by providing high weightage to data points that are derived from reliable sources (like the centre pixels which are less distorted)

6. The quality of a fit is evaluated by by considering R2 value, higher R2 better is the fit. R2 measures how much of the observed variance is explained by the observed xi.
R2 is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}.$$

Problem 2. Morphological Processing

(a) Basic Morphological Process Implementation

2.a.1 Abstract and Motivation

Morphological image processing is a type of processing in which the spatial form or structure of objects within an image are modified. When images are processed for enhancement and while performing some operations like thresholding, more is the chance for distortion of the image due to noise. As a result, imperfections exist in the structure of the image. The primary goal of morphological operation is to remove this imperfection that mainly affects the shape and texture of images. It is also very helpful in image segmentation.

2.a.2 Approach and Procedure

The main operations performed are Shrinking, Thinning and Skeletonization.

Ref 1: Binary Image Connectivity

Binary image morphological operations are based on the geometrical relationship or *connectivity* of pixels that are deemed to be of the same class. Consider the following neighbourhood pixel pattern:

*X*3 *X*2 *X*1
*X*4 *X* *X*0
*X*5 *X*6 *X*7

Binary valued image with X=0 or 1, with its eight neighbours *X*0, *X*1, ……, *X*7.
Pixel X is four connected if *X*0 =*X*2= *X*4= *X*6 =1

Pixel X is eight connected if $X0 = X1 = X2 = X3 = X4 = X5 = X6 = X7 = 1$

The connectivity relationship between a centre pixel and its eight neighbours can be quantified by the concept of a *pixel bond*, the sum of the bond weights between the centre pixel and each of its neighbours. Each four-connected neighbour has a bond of two, and each eight-connected neighbour has a bond of one.

Ref 2: Hit - or - Miss Transformation

Morphological techniques probe an image with a small shape or template called a **structuring element**. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels. Some operations test whether the element "fits" within the neighbourhood, while others test whether it is "hit" / “miss”. If the binary-valued pattern of the mask matches the state of the pixels under the mask (hit), an output pixel in spatial correspondence to the centre pixel of the mask is set to some desired binary state. For a pattern mismatch (miss), the output pixel is set to the opposite binary state.

It is often possible to establish simple neighbourhood logical relationships that define the conditions for a hit. Hit-or-miss morphological algorithms are often implemented in digital image processing hardware by a pixel stacker followed by a look-up table (LUT).

Basic Definitions:

Shrink: Erase black pixels such that an object without holes erodes to a single pixel at or near its centre of mass, and an object with holes erodes to a connected ring lying midway between each hole and its nearest outer boundary.

Thin: Erase black pixels such that an object without holes erodes to a minimally connected stroke located equidistant from its nearest outer boundaries, and an object with holes erodes to a minimally connected ring midway between each hole and its nearest outer boundary.

Skeleton: A skeleton or stick figure representation of an object can be used to describe its structure.

Approach

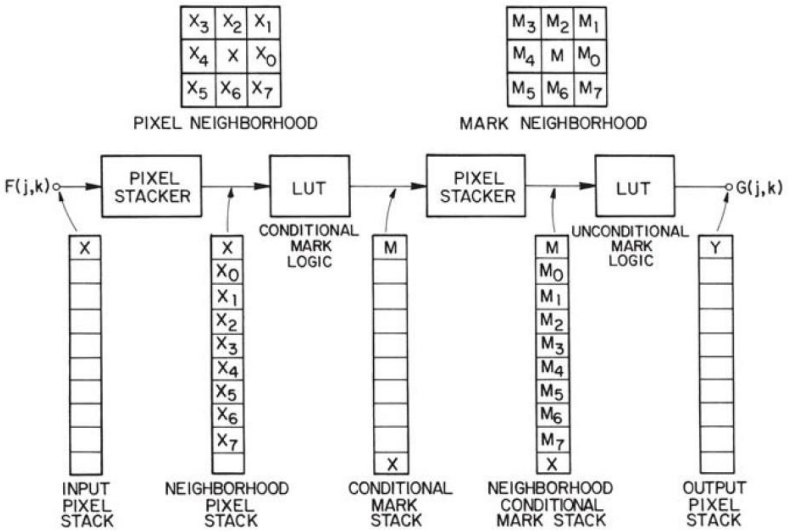


Fig 3: Flowchart for conditional mark operations.

In the algorithm, two concatenated hit-or-miss transformations are performed to obtain indirect information about pixel patterns within a window. In the first stage, the states of nine neighboring pixels are gathered together by a pixel stacker, and a following look-up table generates a conditional mark M for possible erasures by referring to the pattern table, which will be conditionally marked for erasure. In the second stage

of the algorithm, the center pixel X and the conditional marks in a neighborhood centered about X are examined to create an output pixel.

Algorithm: Shrinking/Thinning/Skeletonization

Step 1: The input image has to be boundary extended with zeros (background is 0)

Step 2: For i=2:N+1 and for j=2:N+1 (where N is the size of the image), if the centre pixel (i,j) is equal to 1, then eight neighbour pixel *X*0, *X*1, …, *X*7 are compared against the standard conditional mask lookup tables of shrinking/thinning/skeletonization (depending on the operation of interest). If there is a match, then M(i,j) value is set to 1 else set to 0.

Step 3: M(i,j) is the input for the second stage, G(i,j) is set only if the below condition is satisfied:
G(i,j) = X ∩ [(~M) U P(M0,M1,….M7)]

where P(M0,M1,…,M7) are the unconditional masks for shrinking/thinning/skeletonization (depending on the operation of interest). It is set to 1, only if there is a 'hit' by comparing the neighbour M0,…,M7 in the lookup table.

Step 4: Compare whether G is same as previous iteration output, if the condition is false, then the flag = false and the input=G , temp=G (which stores previous iteration G, initially set to all zeros) and steps 2~3 are repeated again. If G is same as previous iteration then flag=true and the while loop is exited.

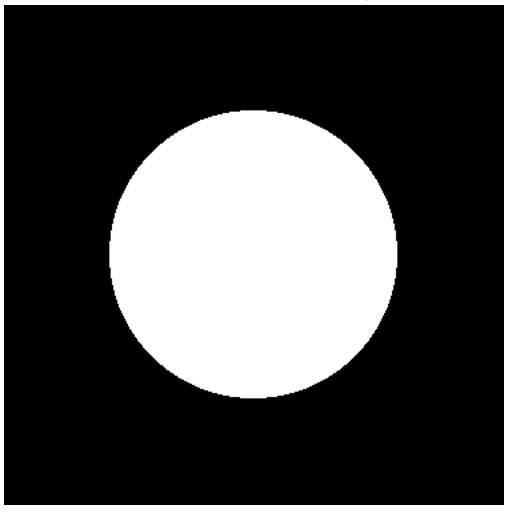
Step 5: (*Only to be carried out, if the operation chosen is skeletonization) The output G(i,j) along with surrounding eight neighbour pixel intensities are evaluated for the following condition:
G1(i,j) = X U [P1 U P2 U P3 ……U P6] and is set to '1' if the condition yields logical '1'

Note: All the conditional and unconditional masks of Shrinking, Thinning, Skeletonization, Bridging are stored as an array with M0,...,M7 values.

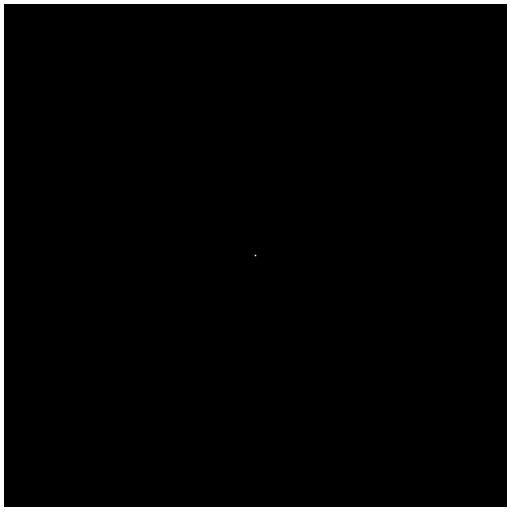
2.a.3 Results

a) Shrinking

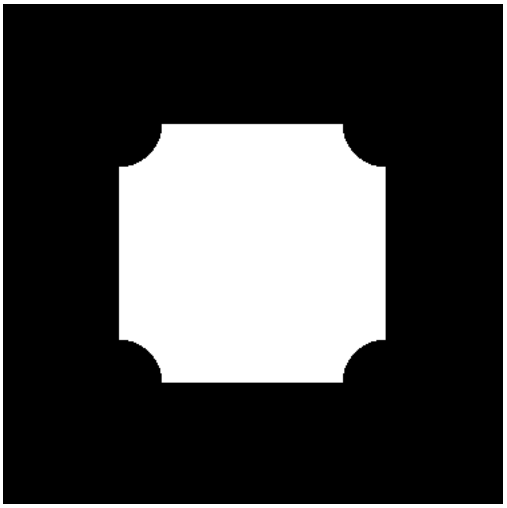
Pattern1 input image



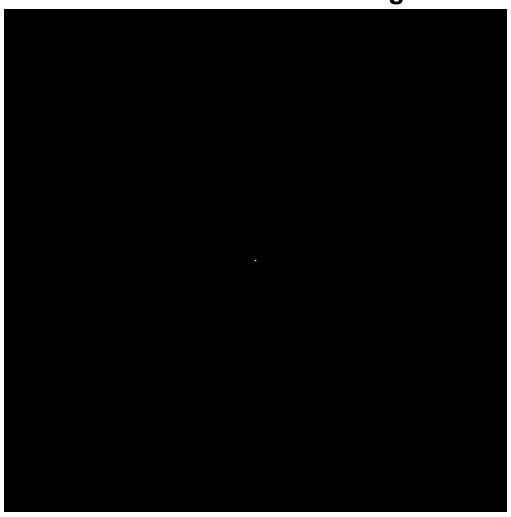
Pattern1 shrinked image



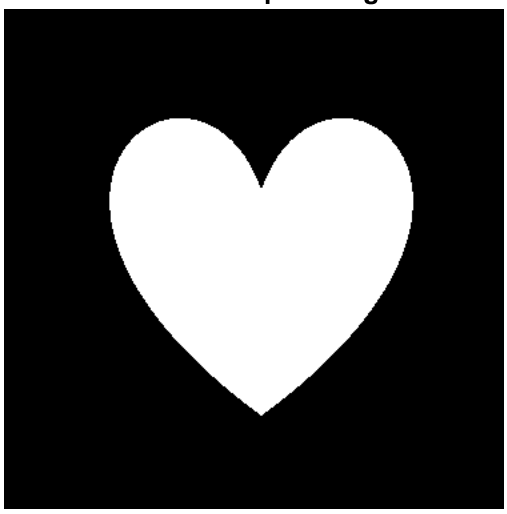
Pattern 2 input image

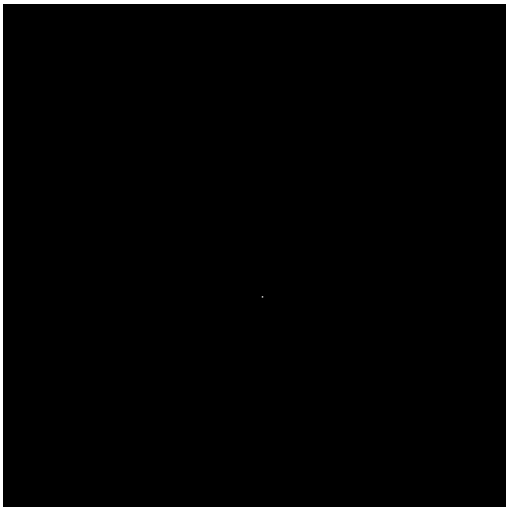


Pattern 2 shrinked image

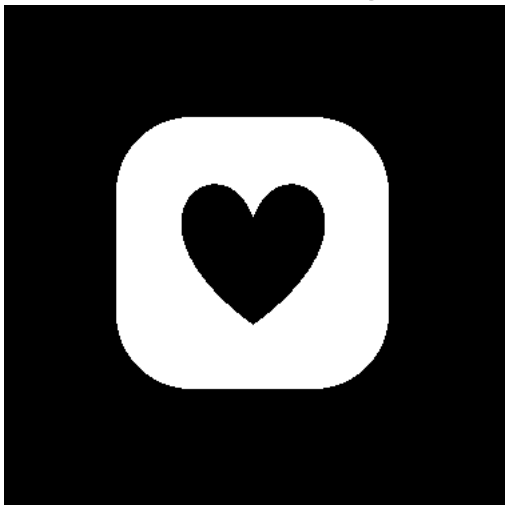


Pattern 3 input image

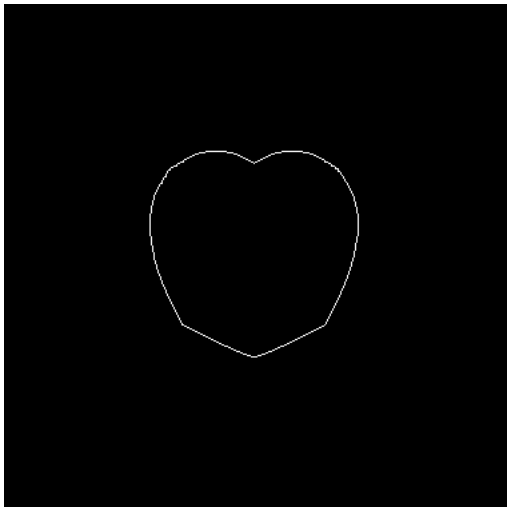




Pattern 4 input image

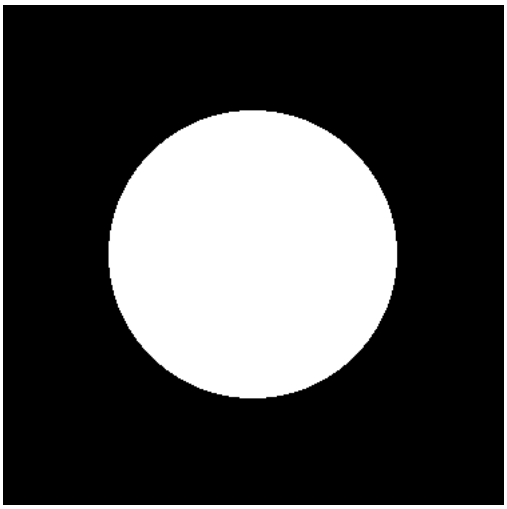


Pattern 4 shrinked image

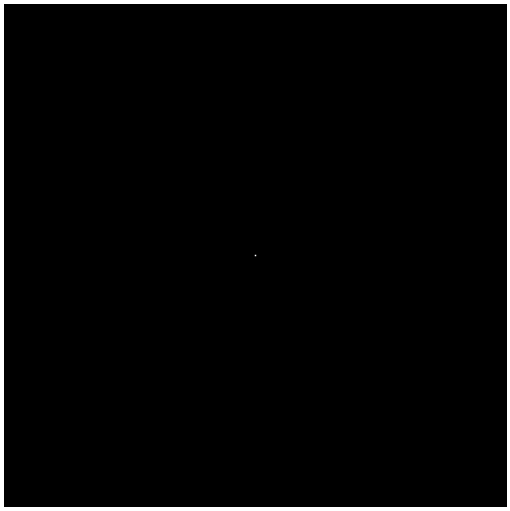


b) Thinning

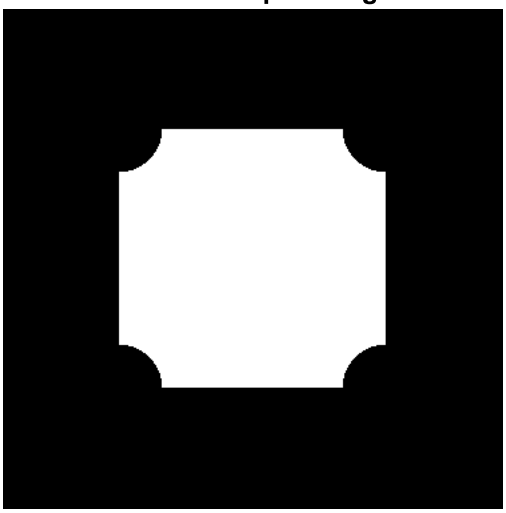
Pattern1 input image



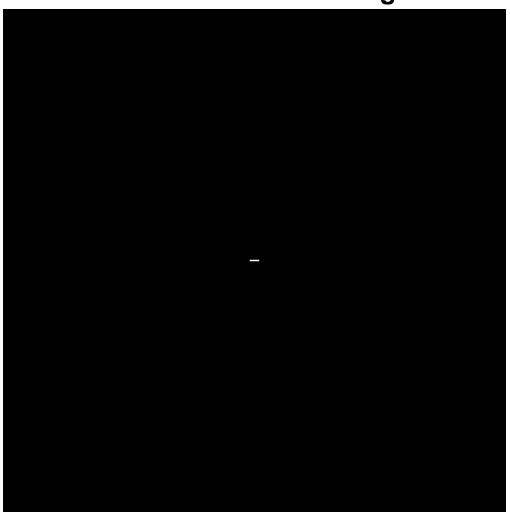
Pattern 1 thinned image



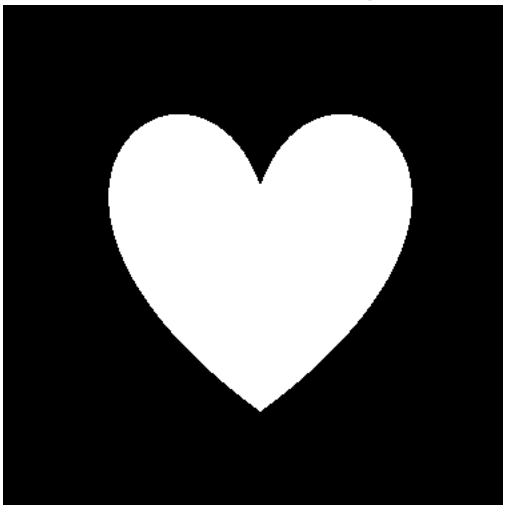
Pattern 2 input image



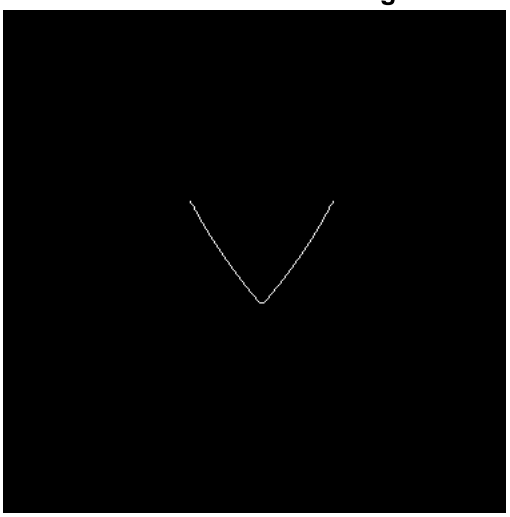
Pattern 2 thinned image



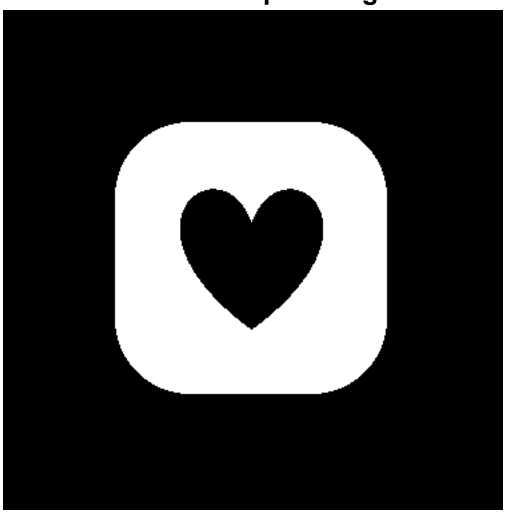
Pattern 3 input image



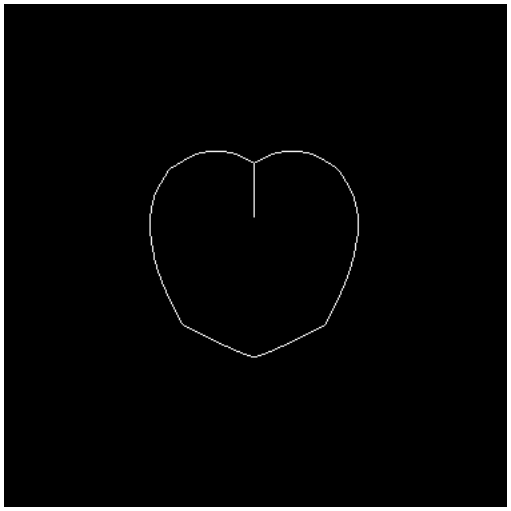
Pattern 3 thinned image



Pattern 4 input image

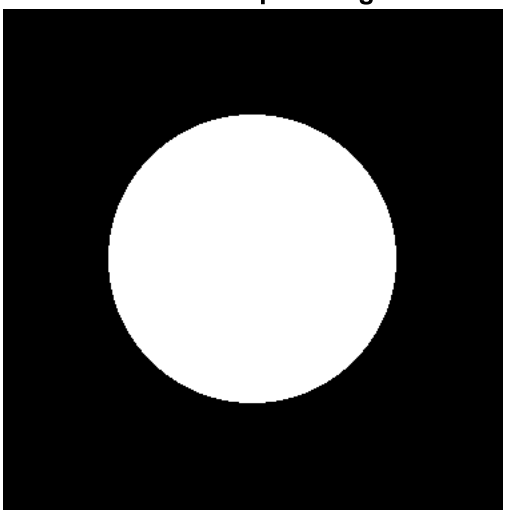


Pattern 4 thinned image

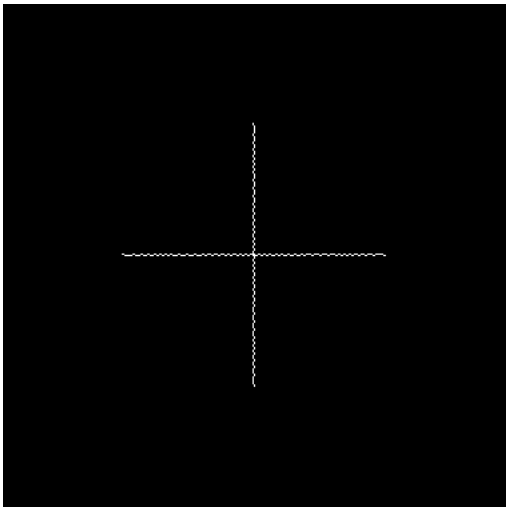


c) Skeletonization

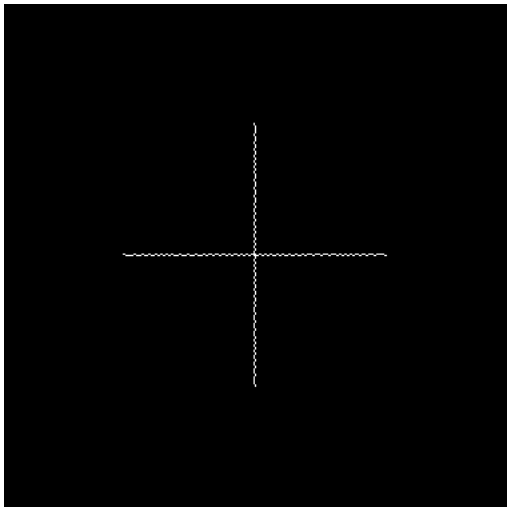
Pattern1 input image



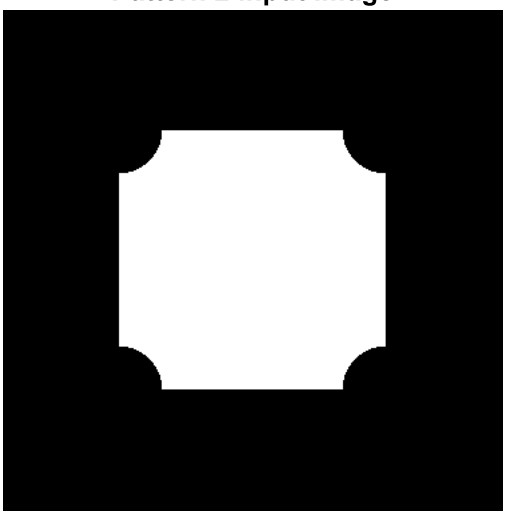
Pattern1 Skeletonized image



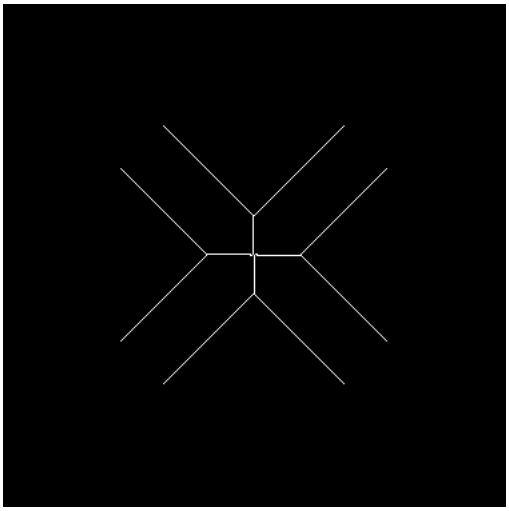
Pattern 1 skeletonized image with bridging



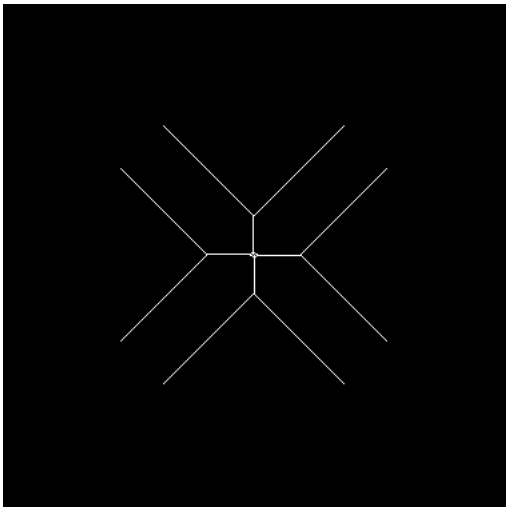
Pattern 2 input image



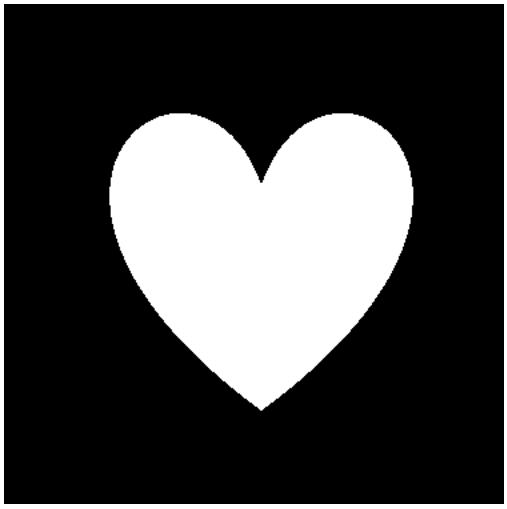
Pattern 2 skeletonized image



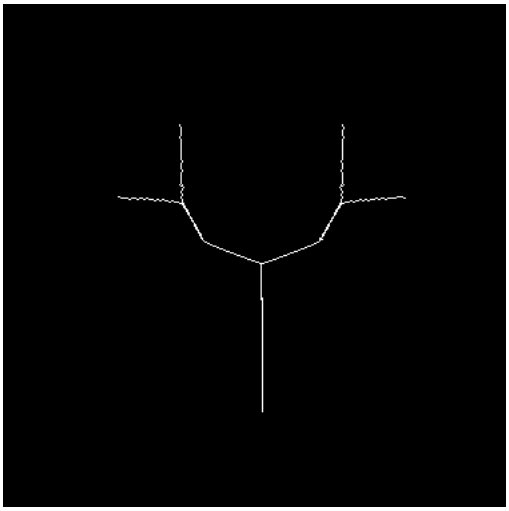
Pattern 2 skeletonized image with bridging



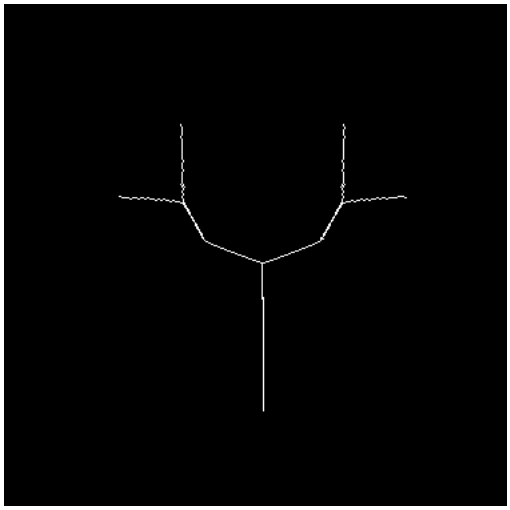
Pattern 3 input image



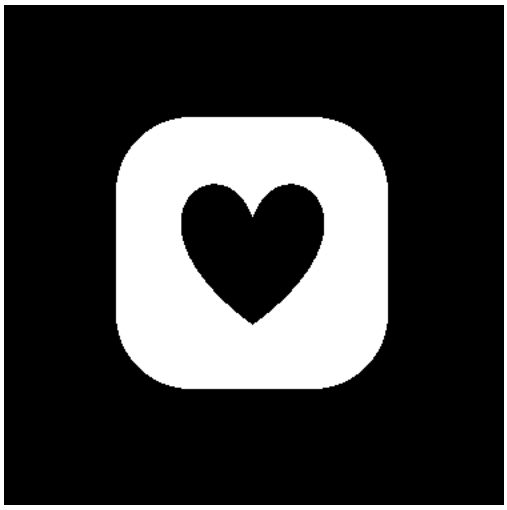
Pattern 4 skeletonized image



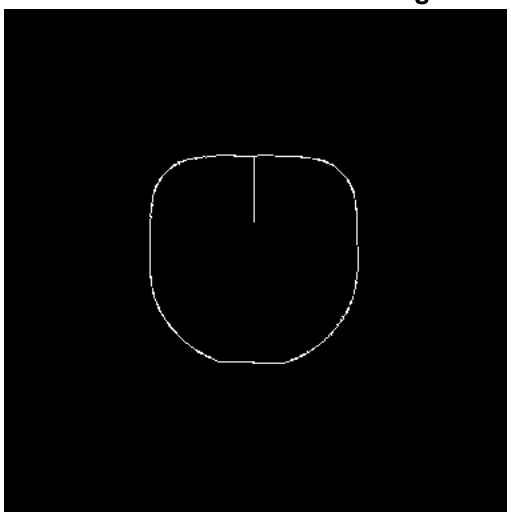
Pattern 4 skeletonized image with bridging



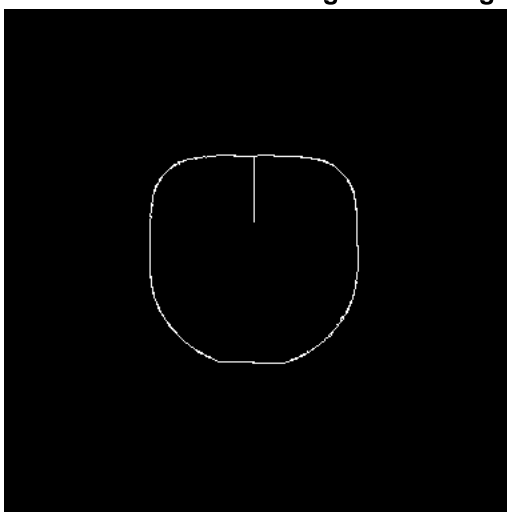
Pattern 4 input image



Pattern 4 skeletonized image



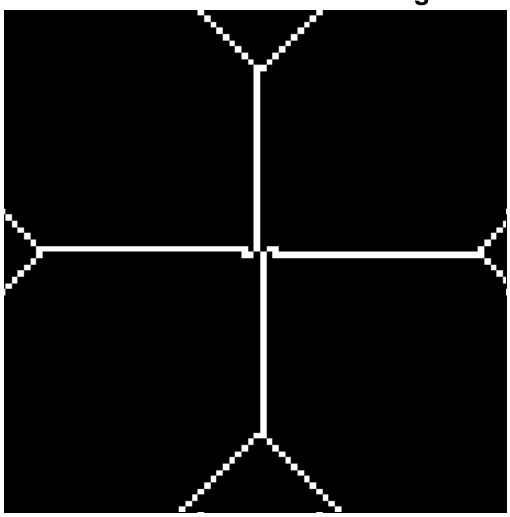
Pattern 4 skeletonized image with bridging



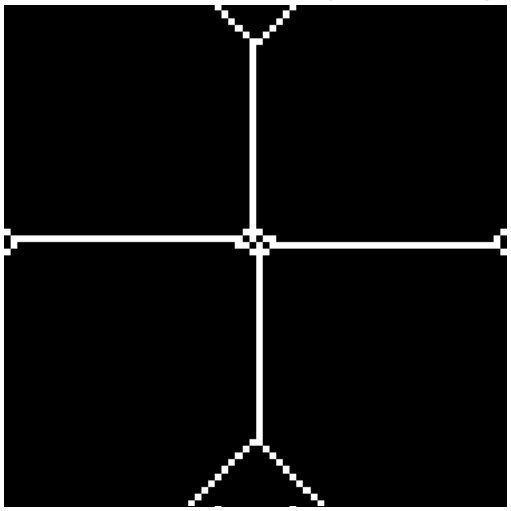
2.a.4 Discussion

1. The process of shrinking, thinning, skeletonization are computationally not efficient. All the possible combination of masks is to be included in-order to achieve desired output.
2. Window size constraint: The 3x3 window does not provide enough information to prevent total erasure and to ensure connectivity. A 5x5 hit-or-miss transform could provide sufficient information to perform proper shrinking. But such an approach would result in excessive computational complexity in determining (225 possible patterns).
3. From the shrinking results in section 2.a.3 a), it can be derived that a filled image (inside white or without holes) will shrink down to a single pixel near its centre of mass (Pattern3~ dot at bottom), if there is no break in the boundary (Pattern1~3). In case of Pattern4, since the image contains background within the filled image(hole), the black pixels inside the white region tries to expand upon shrinking while the white/ foreground region tries to shrink, resulting in a mid-way between hole and its outer boundary.
4. From the thinning results in section 2.a.3 b), it can be derived that for objects without any holes results in minimally connected stroke located equidistant from its nearest outer boundaries (for pattern1: only centre of the circle is equidistant), and an object with holes erodes to a minimally connected ring midway between each hole and its nearest outer boundary (Pattern 4)
5. From the skeletonization results in section 2.a.3 c), it can be derived that for objects, a stick figure representation can be observed according to the structure of the foreground (white region/object).The obtained images are not unique and it depends on the masks used for comparison and setting condition.
6. In case of skeletonized images and skeletonized images with bridging, the following images are evident for the differences between the duo. It is vital to perform bridging after skeletonization to retain the connectivity. The below images reflect the broken edges connected after bridging.

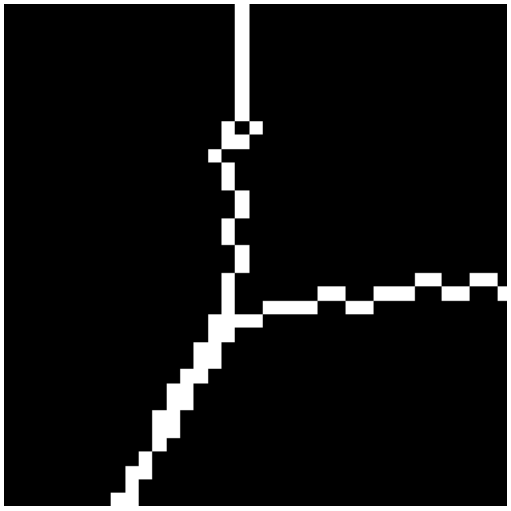
Pattern 2 skeletonized image



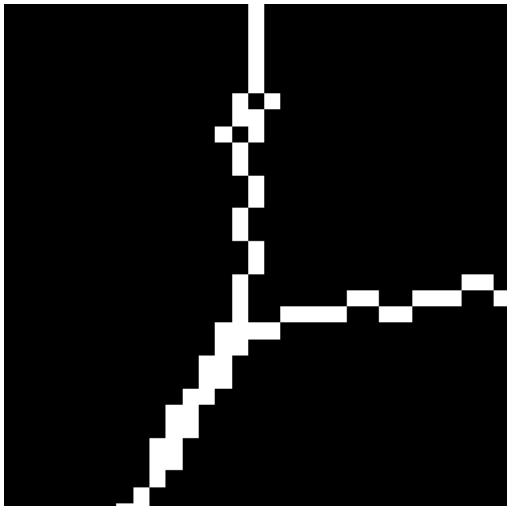
Pattern 2 skeletonized image with bridging



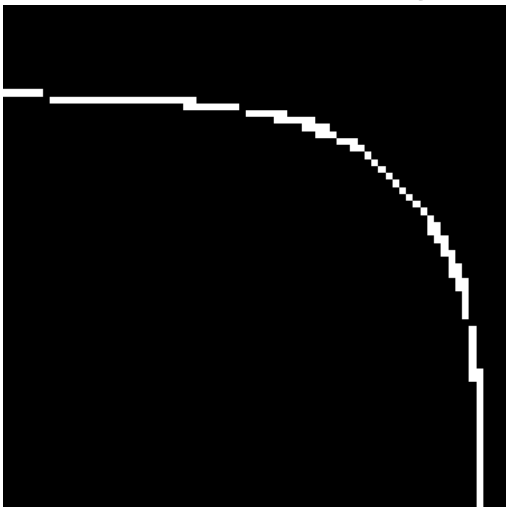
Pattern 4 skeletonized image



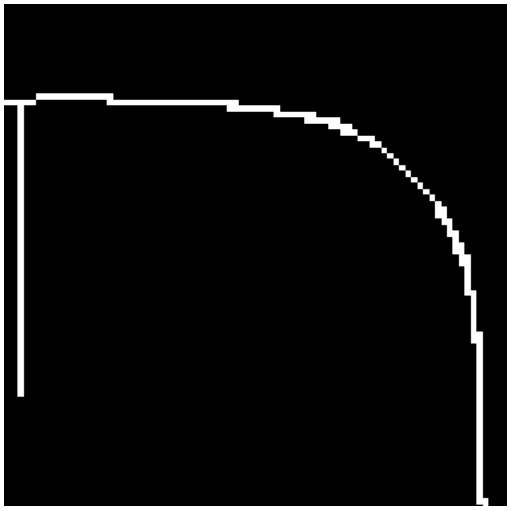
Pattern 4 skeletonized image with bridging



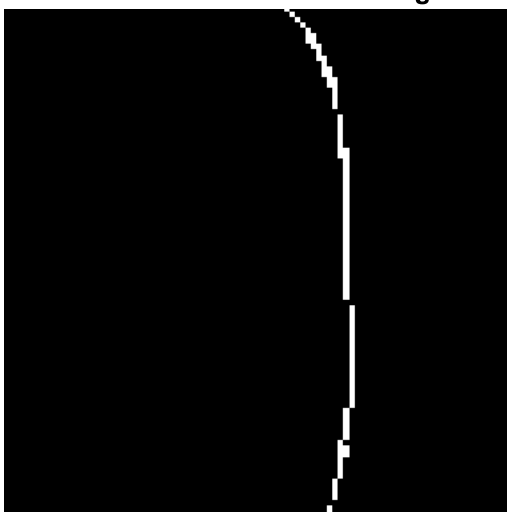
Pattern 4 skeletonized image



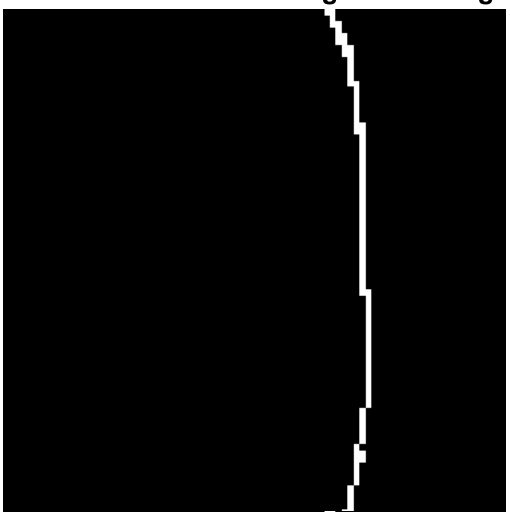
Pattern 4 skeletonized image with bridging



Pattern 4 skeletonized image



Pattern 4 skeletonized image with bridging



(b) Defect Detection and correction

2.b.1 Abstract and Motivation

The primary goal of morphological operation is to remove the imperfection that mainly affects the shape and texture of images. It is also very helpful in image segmentation. One of the applications of morphological processing technique is defection detection.

2.b.2 Approach and Procedure

Since the deer image is designed for product decoration, and it will be enlarged to fit multiple product sizes later. Thinning process can be made use of to identify the defect, since it is enlarged and shrinked. For consistent product appeal, image defect detection to insure no black dots in the main deer body is necessary.

Step1: Image boundary has to be extended with zero padding, since the background is black (0 intensity pixel)

Step 2: For i=2:N+1 and for j=2:N+1 (where N is the size of the image), if the centre pixel (i,j) is equal to 1, then eight neighbour pixel X0, X1, …, X7 are compared against the standard conditional mask lookup tables of thinning. If there is a match, then M(i,j) value is set to 1 else set to 0.

Step 3: M(i,j) is the input for the second stage, G(i,j) is set only if the below condition is satisfied:
$G(i,j) = X \cap [(\sim M) \cup P(M0,M1,\ldots.M7)]$
where P(M0,M1,…,M7) are the unconditional masks for thinning. It is set to 1, only if there is a 'hit' by comparing the neighbour M0,…,M7 in the lookup table.

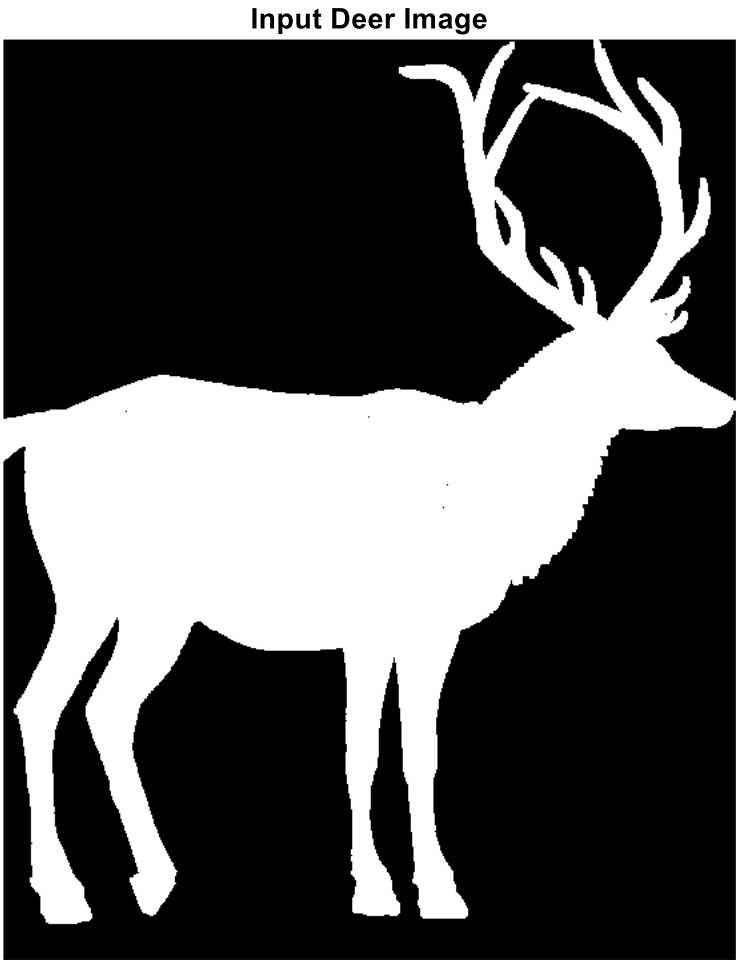
Step 4: Compare whether G is same as previous iteration output, if the condition is false, then the flag = false and the input=G , temp=G (which stores previous iteration G, initially set to all zeros) and steps 2~3 are repeated again. If G is same as previous iteration then flag=true and the while loop is exited.

Step 5: If the image has no defects, then there would be no closed loops or connected components, else If there is any black pixels in the main body, then those might get enlarged to form connected components or form closed loops

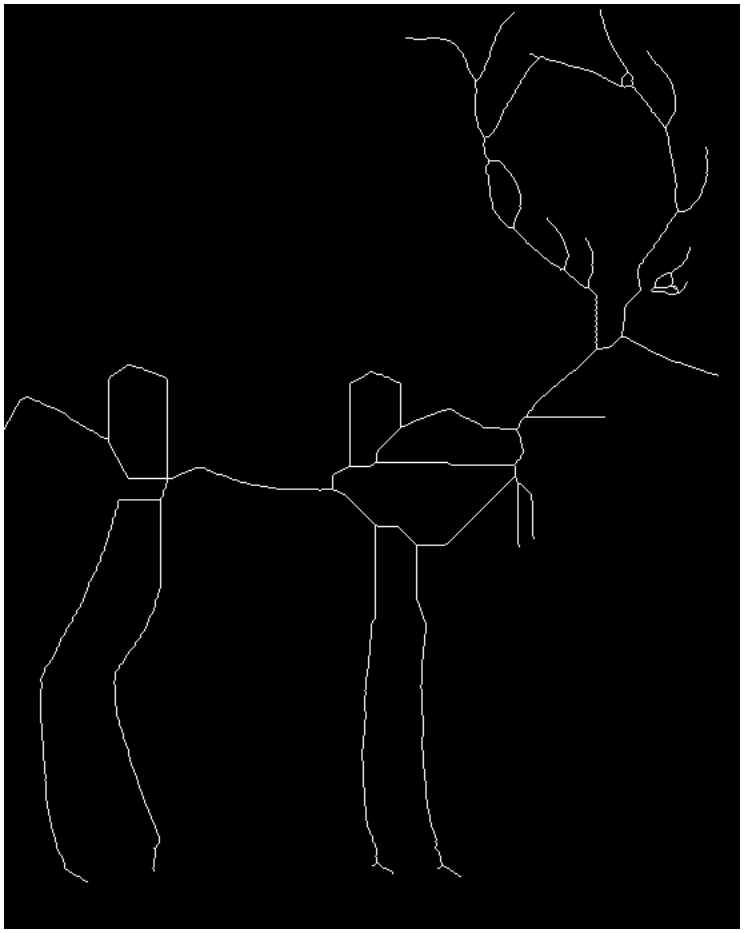
Step 6: If there are any defect regions in the main body, the black pixels would be surrounded by white pixels which needs to be corrected to white pixel. [Ref 1 of section 2.a.2] 4-Connectivity of the pixels is looked for pixel intensity '0' and is changed to pixel intensity '1' if pixel has 4 connectivity.

Step 7: Thinning process is applied for the corrected image to cross-verify if there are any defects, in case any.

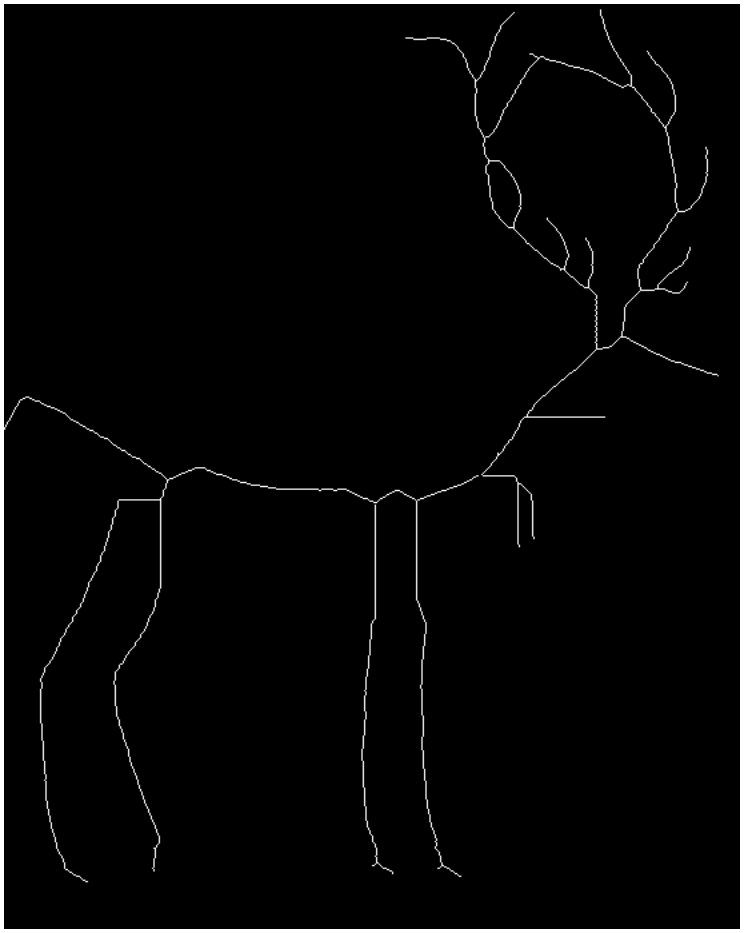
2.b.3 Results



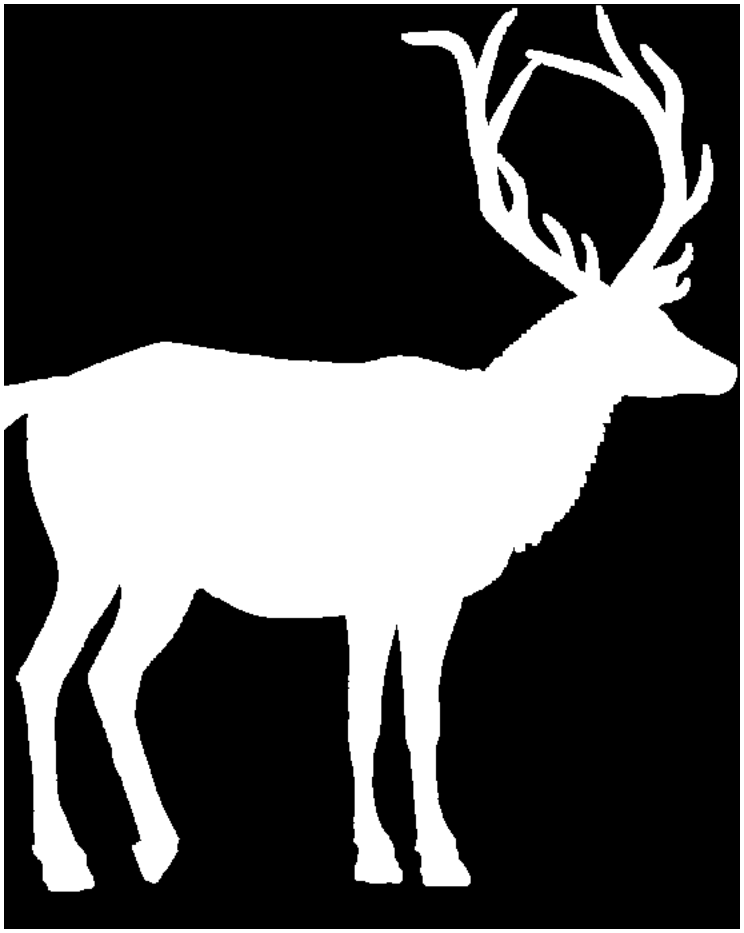
Thinned image of input image



Thinned image after correcting defects



Output image with defects removed



2.b.4 Discussion

1. Even if the black pixels in the body of the deer has been removed, the deer image may still suffer from other defect due to the formation of the closed loop near it's stag.

2. This defect has to be avoided by breaking the bridges (connectivity) near it's stag.
   For instance

Zoomed

The highlighted box have a T connection, which needs to be broken down, by changing the pixel at the intersection/ T joint as '0' instead of '1'

3. Location of defects (black pixels in white deer body)

Location of defect in the boundary extended image	Location of defect in the original image
[57 466] [214 505] [215 504] [282 95] [286 277] [337 336] [354 333]	[56 465] [208 499] [213 504] [214 503] [281 94] [285 276] [336 335] [353 332]

(c) Object Analysis

2.c.1 Abstract and motivation

The primary goal of morphological operation is to remove the imperfection that mainly affects the shape and texture of images. It is also very helpful in image segmentation. One of the applications of morphological processing technique is for pre-processing and post-processing the images.

2.c.2 Approach and Procedure

Since the image contains lot of noise and holes, the image has to be preprocessed.
Step 1: Convert RGB image to gray scale image
Step 2: imadjust() function is used to adjust the brightness of the image, since one of the grain is dark

and a threshold is set to convert to binary image.

Step 3: bwareopen() function is used to remove small objects from binary image

Step 4: Structuring element of disk shape is used of radius 1 is used to perform morphological closing operation

Step 5: imfill() function is used to fill the image with 'holes' (black regions inside the object.

Step 6: The pre-processed image is then fed as an input to perform shrinking operation.

Step 7: Steps for shrinking is mentioned in 2.a.2

Step 8: The shrinked image contains 1 pixel/grain. Therefore, sum all the white pixels retained after shrinking to obtain the count of rice grain.

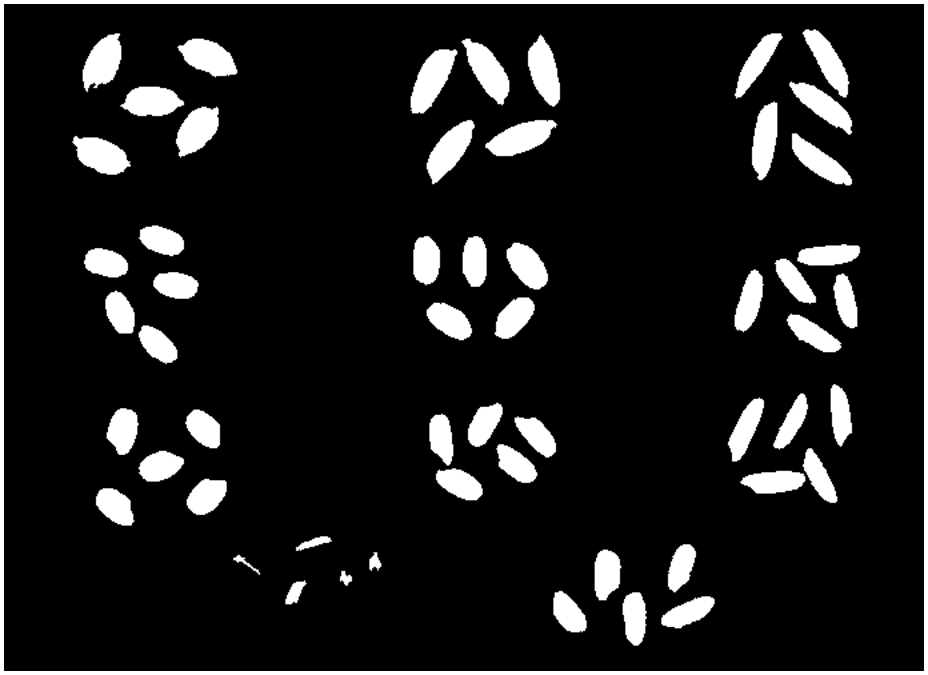
Step 7: The pre-processed image is then fed as an input to perform thinning operation

Step 8: The image obtained after thinning yields a thin line, the trace along the line would provide the length of the rice grain.

2.c.3 Results

Input Image

Binarized pre-processed input image

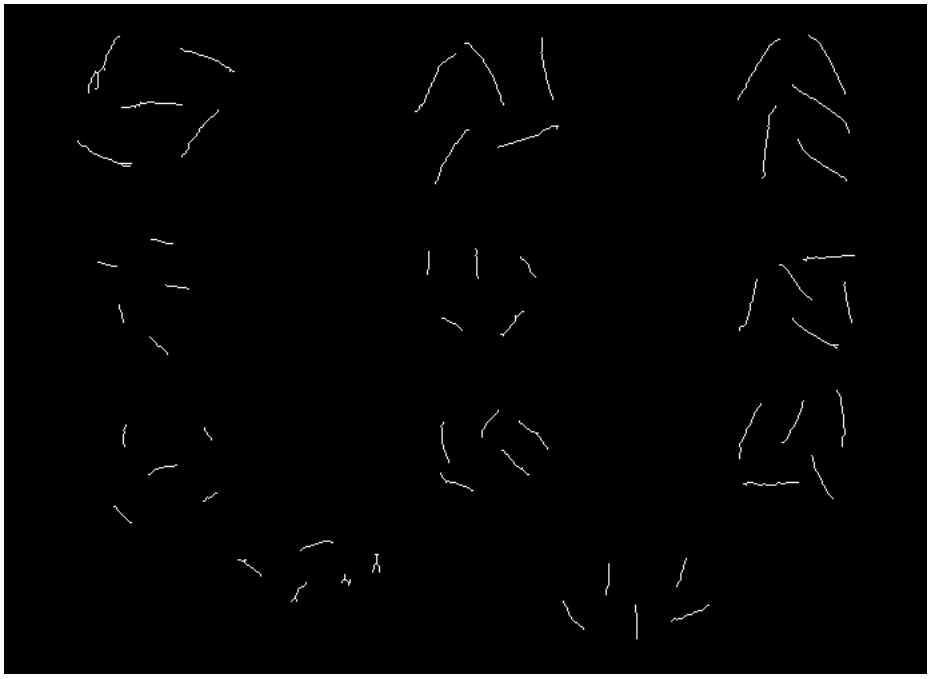


Shrinked image of pre-processed image

```
Count of rice
    55

>>
```

Pre-processed image after thinning process



2.c.4 Discussion

1. Since, the colour of the rice grain is not similar for all the types of grains, it is important to adjust the intensity of the pixel before further processing.

2. The distorted binary image is obtained from the threshold set, threshold play a vital role

3. Filling the holes is an important step to be taken before shrinking, since the hole expands when image is shrinked.