Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Problem 1: Geometric Modification

1. Abstract and Motivation

One of the most common image processing operations is geometric modification in which an image is spatially translated, rotated, scaled in size, warped non linearly or viewed from a different perspective. In this technique of modification, pixel locations are changed rather than the pixel value themselves. Pixel values remain the same. These kinds of transformations are useful for following reasons:

- Create special effects to the images
- To register two images taken at same scene at different times.
- To morph one image to another.

Defining a geometric transformation: Let (u,v) be represent the image coordinate in original image and (x,y) in a transformed image. Function pairs can be used to relate corresponding pixels in two images:
Forward mapping:

$$x = x(u,v)$$
$$y = y(u,v)$$

Reverse mapping:

$$u = u(x,y)$$
$$v = v(x,y)$$

Now, if f(u,v) denote the original image and g(x,y) represent transformed image, then the relation is:

$$g(x,y) = f(u(x,y) , v(x,y))$$
$$f(u,v) = g(x(u,v),y(u,v))$$

The above equations can be illustrated with the figure 1 as shown below:

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

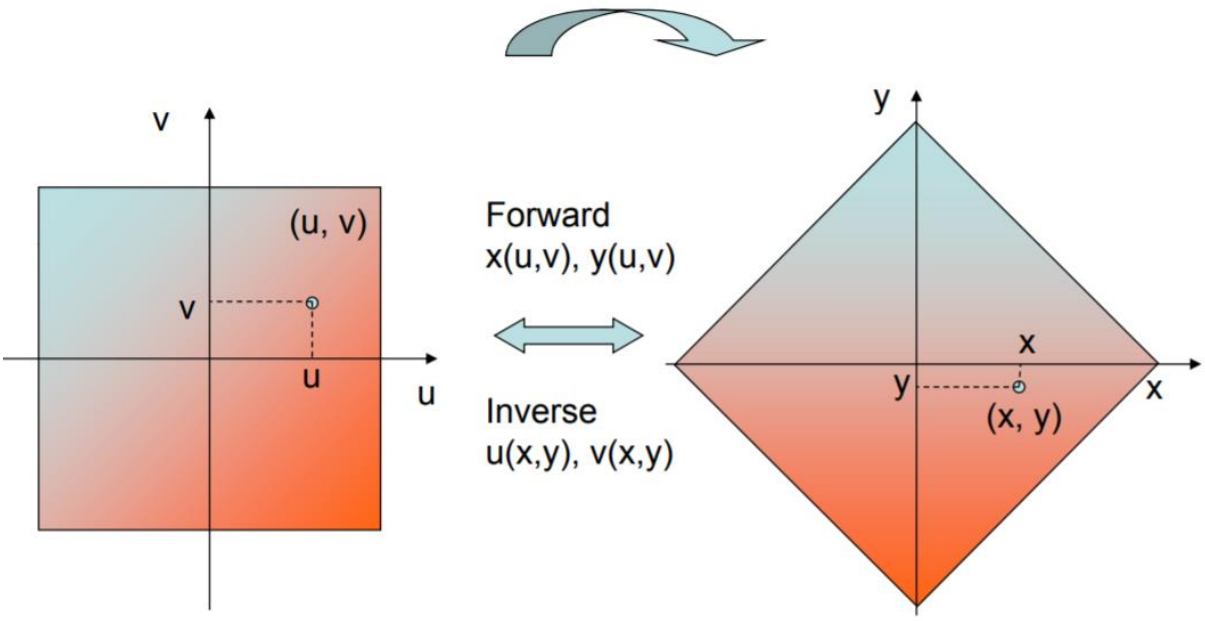


Figure 1: Forward and Reverse mapping illustration

(a) Geometric Transformation

Three kinds of geometric transformations are discussed here. All the techniques defined below works in Cartesian coordinate system and hence care should be taken for appropriate conversion to and from Cartesian to Image Coordinate systems.

(i) Translation

Translation of an image is associated with shifting the image to a specific offset without any other modification to it. The general expression for image translation can be shown as in eqn 1 where t_x and t_y denote translation in x and y directions respectively.

$$x = u + t_x$$

$$y = v + t_y \qquad \text{---- (1)}$$

Consequently, for reverse mapping, the equation 2 holds good.

$$u = x - t_x$$

$$v = y - t_y \qquad \text{----(2)}$$

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

In matrix form, the translation matrix can be shown as

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

Translation can be illustrated using fig 2

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

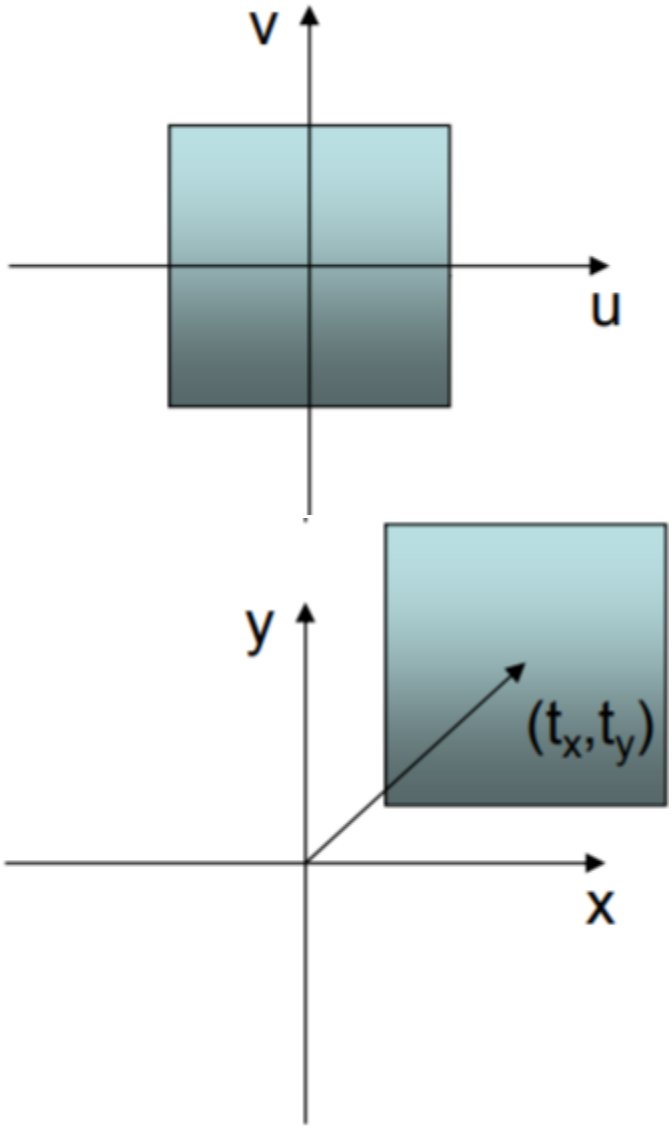


Figure 2: (a) Original image (b) Translated image

(ii) Scaling

Scaling of an image is associated with changing the size of image by a specific value without any other modification to it. The general expression for image scaling can be shown as in eqn 3 where s_x and s_y denote translation in x and y directions respectively.

$$x = s_x * u$$

$$y = s_y * v \qquad \text{---- (3)}$$

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Consequently, for reverse mapping, the equation 4 holds good.

$$u = s_x^{-1} * x$$

$$v = s_y^{-1} * y \qquad \text{----(4)}$$

In matrix form, the translation matrix can be shown as

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

Scaling can be illustrated in fig 3:

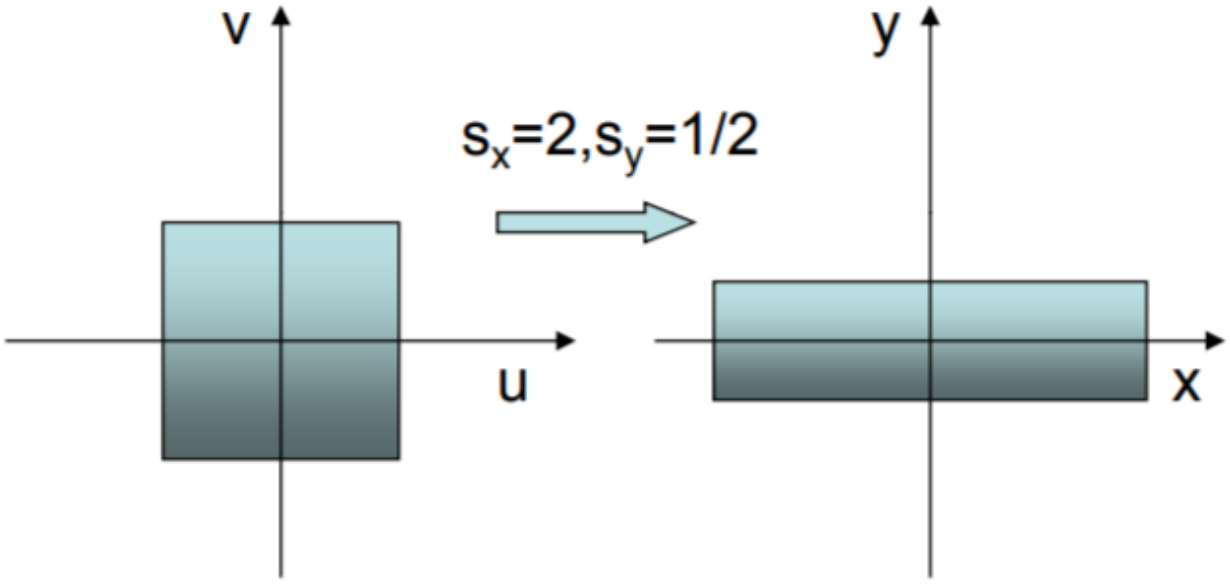


Figure 3 : (a) Original image (b) Scaled image

If s > 1, then the scaling is called magnification or zooming whereas if s < 1, the scaling is called minification or shrinking.

(iii) Rotation

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Rotation of an image is associated with rotating the image by a specific angle about its axis without any other modification to it. The general expression for image rotation can be shown as in eqn 5 where θ denotes the angle of rotation for the image in counterclockwise direction.

$$x = u*\cos\theta - v*\sin\theta$$

$$y = u*\sin\theta + v*\cos\theta \qquad \text{---- (5)}$$

Consequently, for reverse mapping, the equation 4 holds good.

$$u = x*\cos\theta + y*\sin\theta$$

$$v = -x*\sin\theta + y*\cos\theta \qquad \text{----(6)}$$

In matrix form, the translation matrix can be shown as

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Rotation can be illustrated in fig. 4:

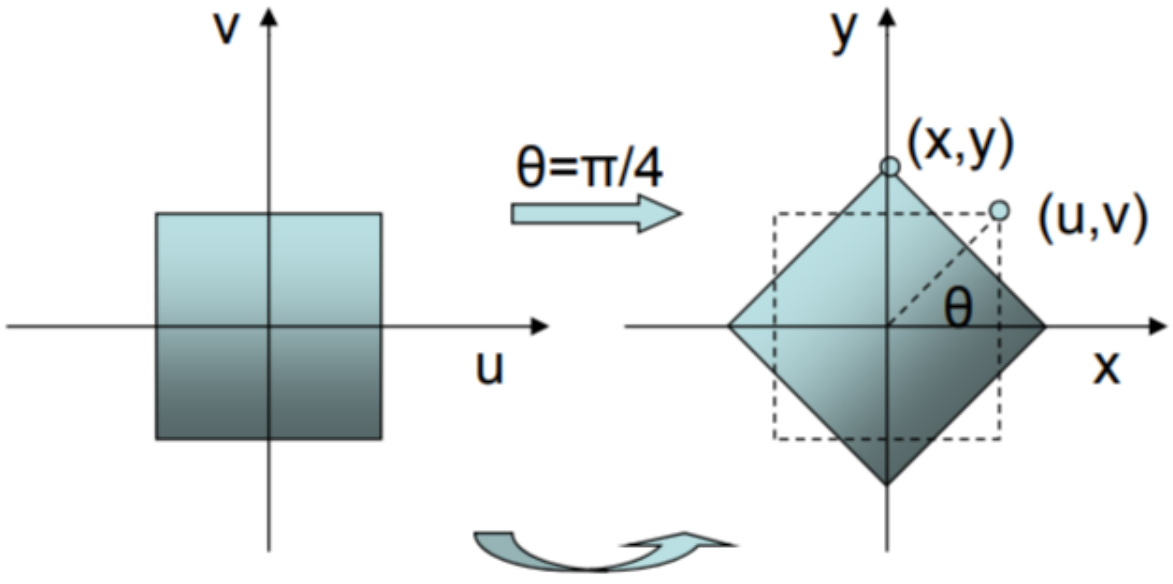
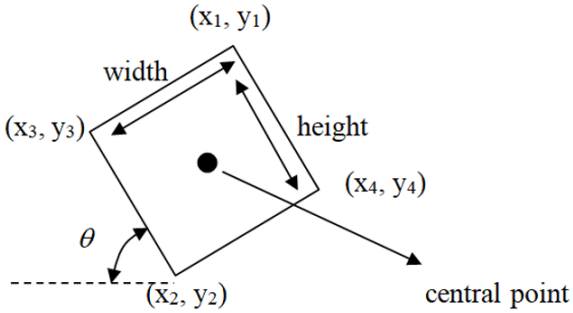


Figure 4: (a) Original Image (b) Rotated Image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

For each of translation, rotation and scaling transformations discussed above we use reverse mapping in our problem to get the transformed image and use bilinear interpolation to extract the image pixel value. In bilinear interpolation, we take the four nearest neighbors and interpolate the pixel values in the output as shown in fig 5 below.

For the given problem, we first find the translation and rotation parameters by first finding the corners and then center of the image and hence the angle by which it is rotated using the equation shown below.



$$width = \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2}$$

$$height = \sqrt{(x_1 - x_4)^2 + (y_1 - y_4)^2}$$

$$\theta = \arctan(\frac{y_2 - y_3}{x_2 - x_3})$$

$$center \quad point = \left(\frac{x_3 + x_4}{2}, \frac{y_3 + y_4}{2}\right) or \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

$$
\begin{aligned}
f(x,y) &= (1-a)(1-b)\, g_s(l,k) + a\,(1-b)\, g_s(l+1,k) \\
&+ b\,(1-a)\, g_s(l,k+1) + ab\, g_s(l+1,k+1)\,,
\end{aligned}
$$

$$
\begin{aligned}
l &= \mathrm{ceil}\,(x)\,, \quad a = x - l\,, \\
k &= \mathrm{ceil}\,(y)\,, \quad b = y - k\,.
\end{aligned}
$$

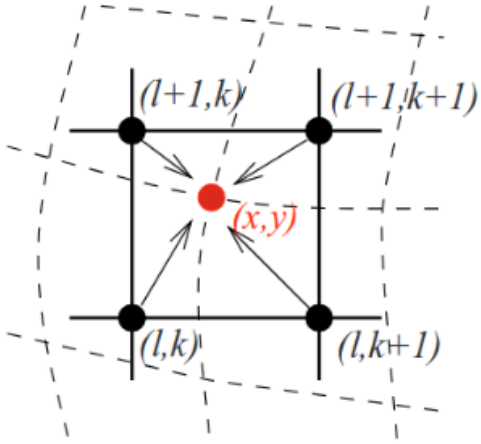


Figure 5: Bilinear Interpolation illustration

(b) Spatial Warping

The spatial warping also called rubber-sheet stretching is associated with transformation of image coordinates using the polynomial equation of higher orders. A common illustration of polynomial of order 2 is as shown in equation (7).

$$u = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2$$

$$v = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2 \qquad \text{---(7)}$$

In matrix notation it can be shown as follows:

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

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

For a pixel at location (x,y), the transformed pixel location (u,v) is calculated using the function defined by equation (7) and we then assign pixel value g(u,v) = f(x,y) using bilinear interpolation. The illustration for forward and reverse mapping is as shown in fig 6 and 7 respectively.

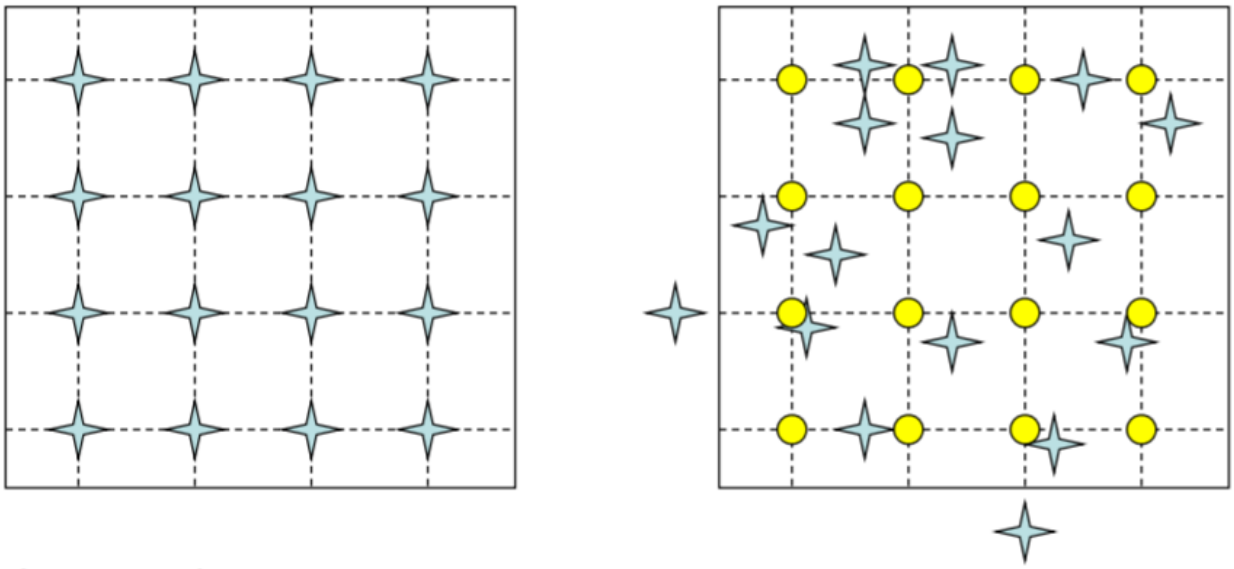


Figure 6: Forward mapping for spatial warping

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

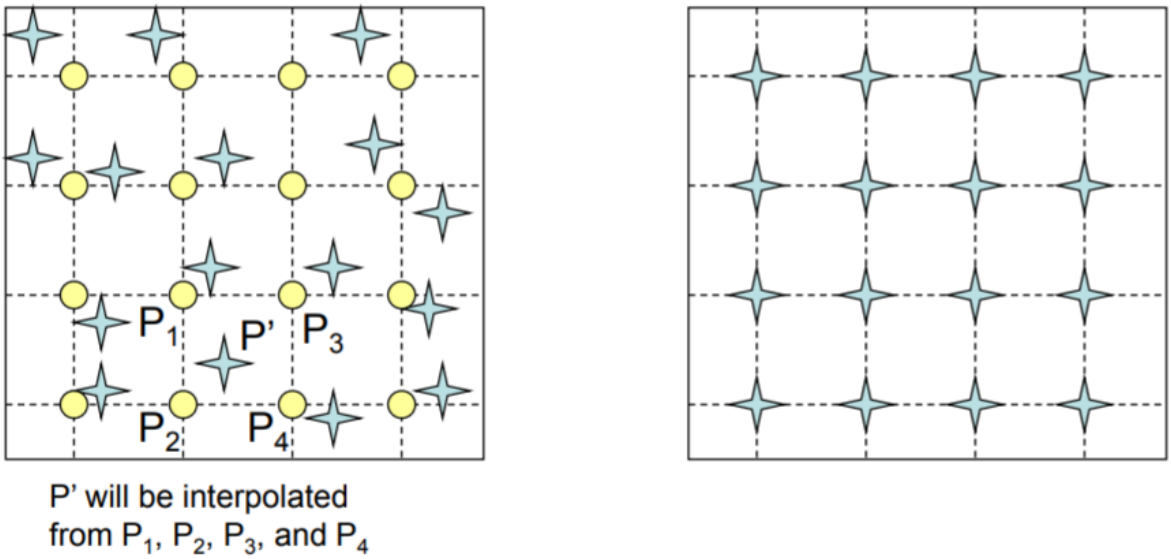


Figure 7: Reverse mapping for Spatial Warping

In our problem, we are given with a parabolic warping in four regions of the image. Four regions can be divided into four triangles as shown in figure 8. By knowing the values of what each of six input values is mapped into output as shown in in figure 8, we can find out the coefficient values for the polynomial equation defined in equation (7) by using the matrix form shown below the figure 8. Then, we use reverse mapping to find out the input pixel corresponding to the output pixel location and assign the pixel values to the output using bilinear interpolation.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

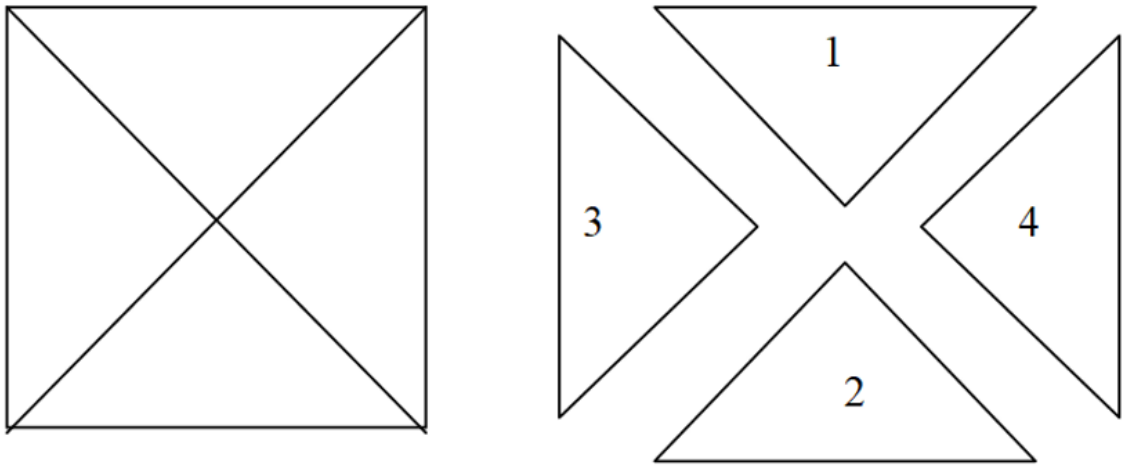


Figure 8: Four regions separated from the image for four different polynomial equations

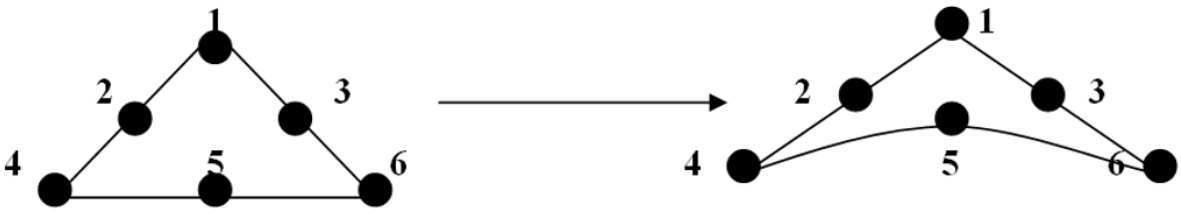


Figure 9: Input to output coordinate values to find coefficentsof the polynomial equation

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & u_5 \\ v_0 & v_1 & v_2 & v_3 & v_4 & v_5 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\ y_0 & y_1 & y_2 & y_3 & y_4 & y_5 \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 & x_4^2 & x_5^2 \\ x_0y_0 & x_1y_1 & x_2y_2 & x_3y_3 & x_4y_4 & x_5y_5 \\ y_0^2 & y_1^2 & y_2^2 & y_3^2 & y_4^2 & y_5^2 \end{bmatrix}^{-1}$$

(c) Lens Distortion

Cameras have been around for a long-long time. However, with the introduction of the cheap pinhole cameras in the late 20th century, they became a common occurrence in our everyday life. Unfortunately, this cheapness comes with its price: significant distortion. Luckily, these are constants and with a calibration and some remapping we can correct this. Furthermore, with calibration you may also determine the relation between the camera's natural units (pixels) and the real world units (for example millimeters).
The different kinds of distortion in lens can be shown I figure below.

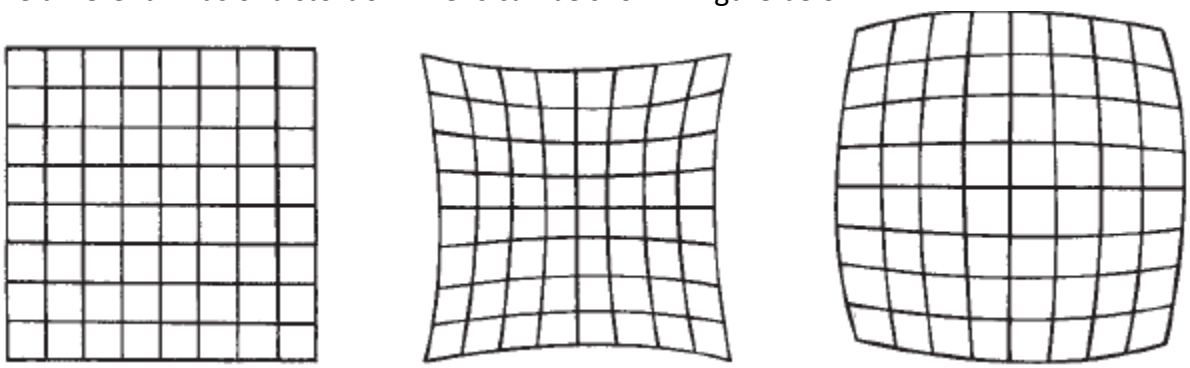


Figure 10: (a) Original Image (b) Pincushion distortion (c) Barrel distortion

The relationship between the actual image and distorted image coordinates is as shown below:

$$x_d = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$
$$y_d = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

where k1, k2 and k3 are called distortion coefficients and $r^2 = x^2 + y^2$. To find undistorted coordinate values (x,y) we find a function that maps x_d and y_d to x , y as shown below:

$$x = f(x_d, y_d)$$
$$y = g(x_d, y_d)$$

However, the exact inverse for this can not be found as the mapping is not linear. Hence, we first project each of x,y to a third dimension xd and yd separately to find the projection in 3D (x,y,x_d) and (x,y,y_d) as shown in figure 11.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

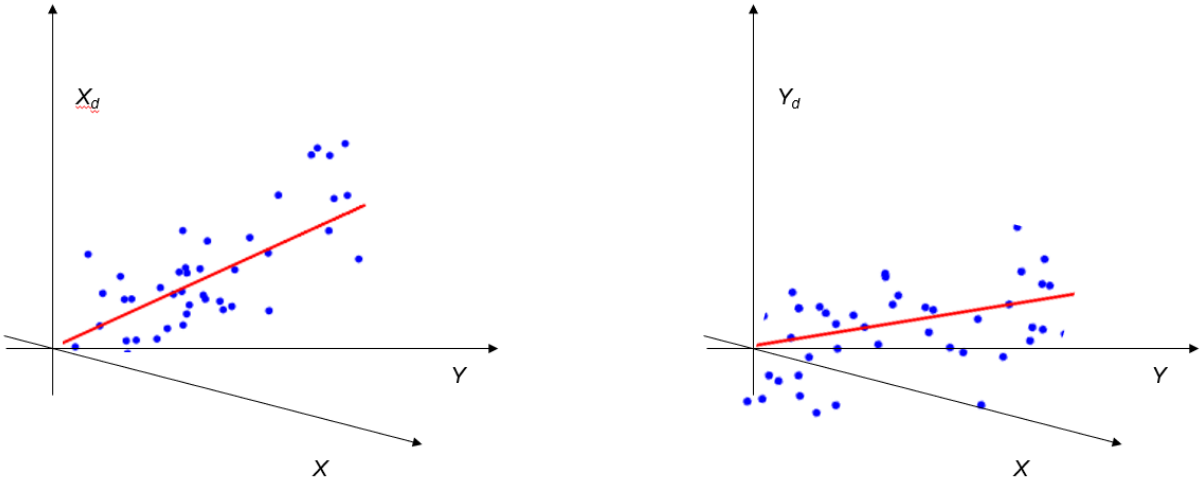


Figure 11: Projection of x,y into x_d and y_d and finding best line that approximates these points

Now, we use techniques like pseudo inverse or least square fit to find the best line that approximates the given points and use the reverse mapping to find the undistorted pixel coordinates and then use bilinear interpolation to find the pixel values at the given location.

2. Approach and Procedure

(a) Geometric Transformation

- The given raw image 'lighthouse1.raw' of size 256x256 was read byte by byte as a gray scale image.
- Iterate through the image in four directions: from top, left, bottom and right. Find the first non-white pixel in each scan which forms the four corners of the image.
- Convert the obtained co ordinates for the corners from image coordinate to cartesian coordinate using the formula:

  x = i - 128

  y = 255 - j – 128

  where i,j are image coordinates and x,y are cartesian coordinates

- The four corner coordinates in cartesian system now gives the center coordinate of the region of interest and the angle with which it is rotated by using the formula explained in the abstract and motivation of Geometric transformation. The center coordinate becomes the translation parameter and the angle will be the rotation parameter.
- Find the input coordinate (u,v) from each of the output coordinate (x,y) using reverse mapping of the function described above by taking inverses each of positive translation, rotation and negative translation.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The coordinate found is changed to image coordinate system as (p,q).
- The output coordinate (i,j) is assigned the pixel value of the coordinate (p,q) found in the previous step explained by using bilinear interpolation of the neighboring four pixels.
- The output image obtained now has its sides parallel to x and y axes.
- Find the four corners of this image by again scanning through all four directions for first non-white pixel. Crop this roi to form new image.
- Check for the size of cropped image if equal to 160x160 or not.
- If the condition in previous step fails, scale the image to 160x160.
- Write the image into a file. Repeat the above procedure for the other two images 'lighthouse2.raw' and 'lighthouse3.raw'.
- Iterate through the original image 'lighhouse.raw' and search for blocks of 160x160 white pixels. When found, fill each with the scaled images found in previous steps to obtain the complete image.
- Write the image as a raw file. Observe the outputs and analyze.

(b) Spatial Warping

- For the given information about the height of the arc, six points corresponding to the polynomial equation (7) mentioned above is found in each of the four triangular regions shown in figure 8 above.
- For each region, the coefficient values are found using the equation mentioned above.
- The input image 'hat.raw' is read byte by byte of size 512x512.
- Create a black output image of size 512x512.
- Iterate through each region of the triangles in the image and convert the image coordinates to cartesian coordinates using the formula:

  $x = i - 128$

  $y = 255 - j - 128$

- For each cartesian coordinate find the new cartesian coordinate (u,v) by using the coefficients found for the respective region.
- Convert the coordinate found to image coordinate system (p,q).
- Assign the pixel value of (i,j) in the original image to the pixel value of (p,q) in the output image in each region.
- Write the image as a raw file. Observe the outputs and analyze.

(c) Lens Distortion Correction

- The given input image 'classroom.raw' of size 1072x712 was read byte by byte into an image object.
- An empty image of 1072x712 was created for the output image.
- Using the given value for constants k1, k2 and k3 the equation mentioned in the abstract and motivation part of Lens distortion was setup

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- Iterate through the input distorted image, converting it to cartesian coordinate and normalizing by the lens radius given as 600.
- The corrected pixel is calculated using the given equation:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

- The output image pixel value was assigned the value of input image pixel of coordinate found in previous step.
- The output image was written and images were observed and analyzed.

3. Observation and Discussions:

(a) Geometric Modification

- For the first subimage, 'lighthouse1.raw', first corner coordinates were found in cartesian coordinate system with origin at center of the image (128,128). The corners are:
  (i) (x1, y1) = (-13,47)
  (ii) (x2, y2) = (-50, -55)
  (iii) (x3, y3) = (54, -96)
  (iv) (x4, y4) = (93, 9)
- Reverse mapping was used to find the translation parameter (x_c,y_c) which is the center of the region of interest using corner points and rotation parameter theta using the formula mentioned in the abstract section. The values are:
  (i) (x_c,y_c) = (20.5, -24.5)
  (ii) Theta = 289.938 (in degrees) to be rotated counterclockwise
- Using the parameters found above, the reverse mapping was applied to get rotated image parallel to x and y axes as shown in figure 13.
- From the found rotated image, corners of roi were extracted and scaled to 160x160 to obtain the scaled sub image ready to be filled as shown in figure 14. Bilinear interpolation was used in each of the steps while obtaining pixel values.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

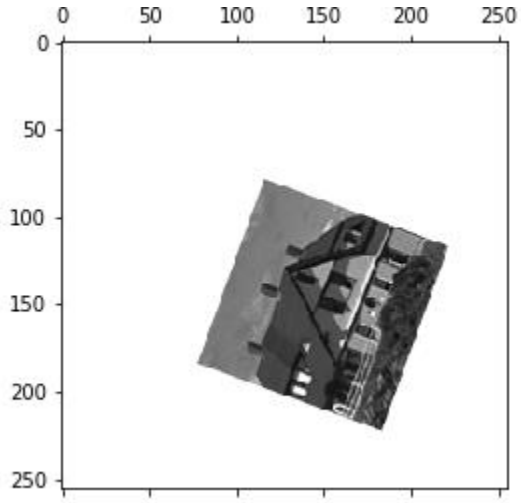


Figure 12 : Original SubImage lighthouse1.raw

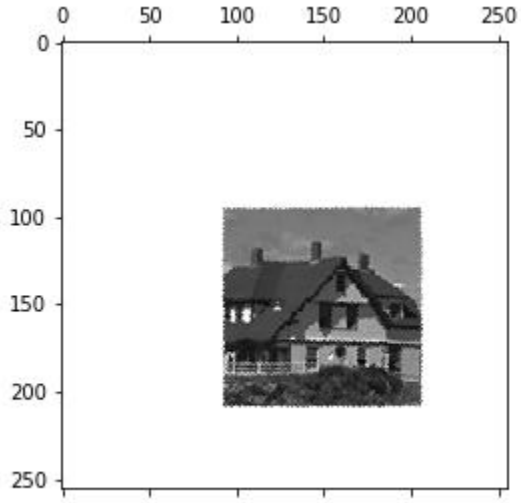


Figure 13: Translation + Rotation + Translation back

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

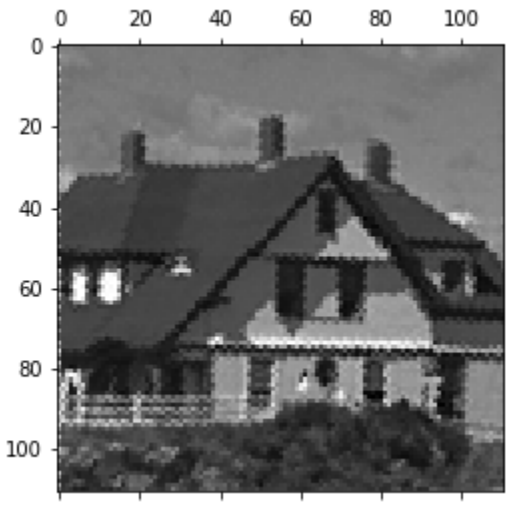


Figure 14: Scaled Image

- For the first subimage, 'lighthouse2.raw', first corner coordinates were found in cartesian coordinate system with origin at center of the image (128,128). The corners are:
  (v) (x1, y1) = (-3,119)
  (vi) (x2, y2) = (-113, 27)
  (vii) (x3, y3) = (-20, -85)
  (viii) (x4, y4) = (90, 11)
- Reverse mapping was used to find the translation parameter (x_c,y_c) which is the center of the region of interest using corner points and rotation parameter theta using the formula mentioned in the abstract section. The values are:
  (iii) (x_c,y_c) = (-11.5, 17.0)
  (iv) Theta = 230.092 (in degrees) to be rotated counterclockwise
- Using the parameters found above, the reverse mapping was applied to get rotated image parallel to x and y axes as shown in figure 16.
- From the found rotated image, corners of roi were extracted and scaled to 160x160 to obtain the scaled sub image ready to be filled as shown in figure 17. Bilinear interpolation was used in each of the steps while obtaining pixel values.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

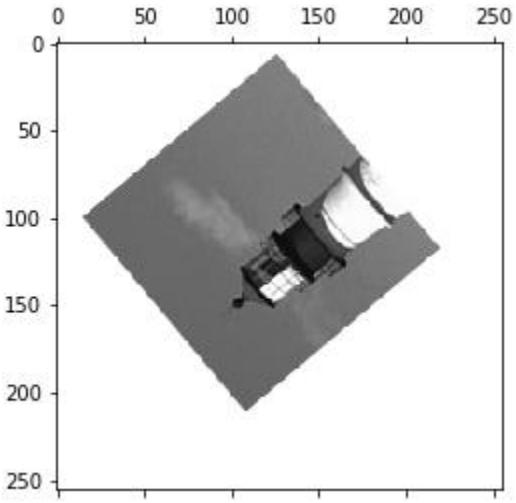


Figure 15: Original Subimaae 'lighthouse2.raw

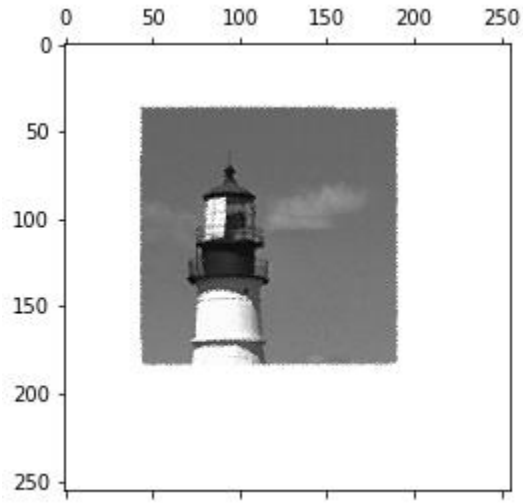


Figure 16: Translation + Rotation + Translation back

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

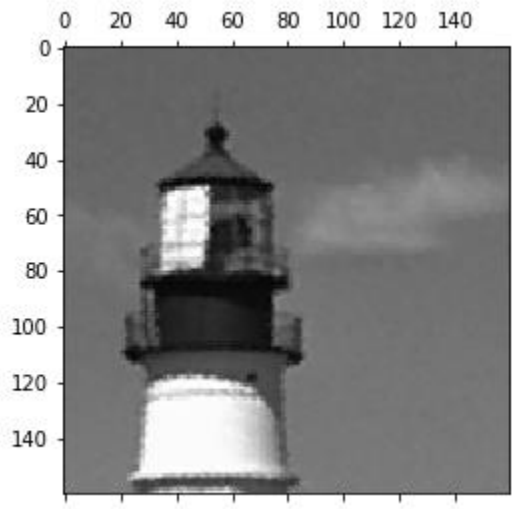


Figure 17: Scaled Image

- For the first subimage, 'lighthouse3.raw', first corner coordinates were found in cartesian coordinate system with origin at center of the image (128,128). The corners are:
  (ix) (x1, y1) = (-88,124)
  (x) (x2, y2) = (-125, -82)
  (xi) (x3, y3) = (82, -124)
  (xii) (x4, y4) = (123, 87)
- Reverse mapping was used to find the translation parameter (x_c,y_c) which is the center of the region of interest using corner points and rotation parameter theta using the formula mentioned in the abstract section. The values are:
  (v) (x_c,y_c) = (-3.0, 0.0)
  (vi) Theta = 370.182 (in degrees) to be rotated counterclockwise
- Using the parameters found above, the reverse mapping was applied to get rotated image parallel to x and y axes as shown in figure 19.
- From the found rotated image, corners of roi were extracted and scaled to 160x160 to obtain the scaled sub image ready to be filled as shown in figure 20. Bilinear interpolation was used in each of the steps while obtaining pixel values.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

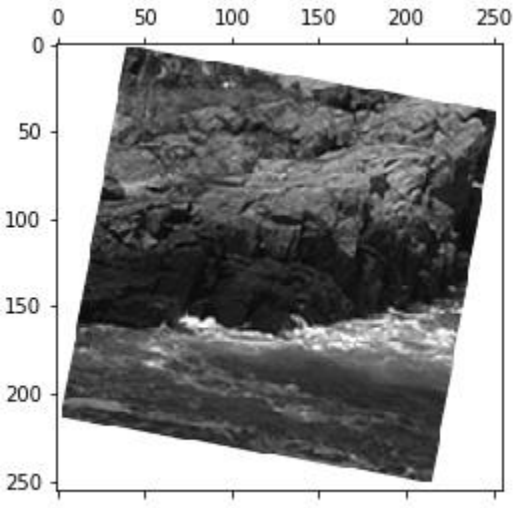


Figure 18: Original Subimaae 'lighthouse2.raw

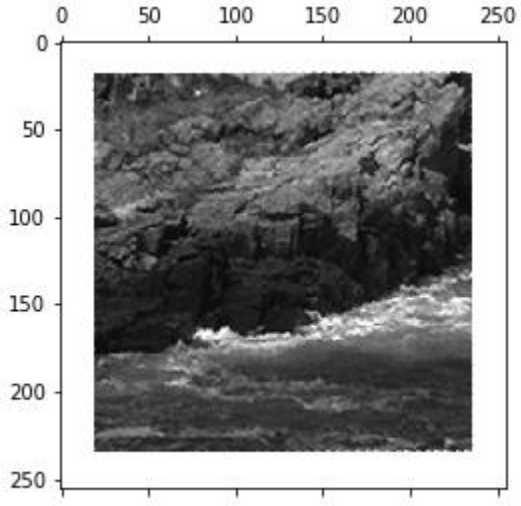


Figure 19: Translation + Rotation + Translation back

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

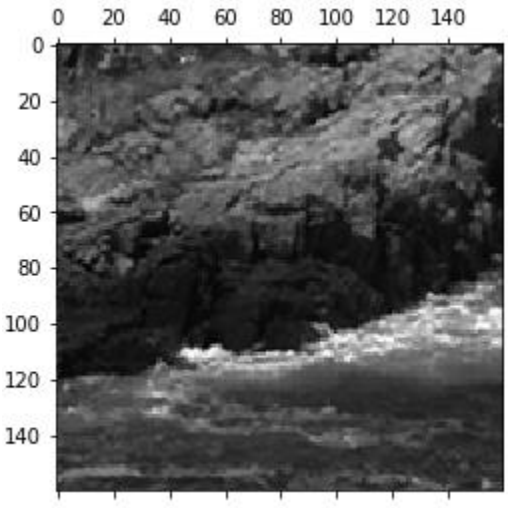


Figure 20: Scaled Image

- Now, iterating through the original 'lighthouse.raw' image, holes and its corners were found in image coordinates are:
  (i) Hole1: Corner1=(31,278)
  Corner2=(31,438)
  Corner3=(191,278)
  Corner4=(191,438)
  (ii) Hole2: corner1= (157,62)
  Corner2=(157,222)
  Corner3=(317,62)
  Corner4=(317,222)
  (iii) Hole3: corner1= (328,326)
  Corner2=(328,486)
  Corner3=(488,326)
  Corner4=(488,486)
- The holes were filled with the obtained scaled image to obtain the complete image shown in figure 21.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

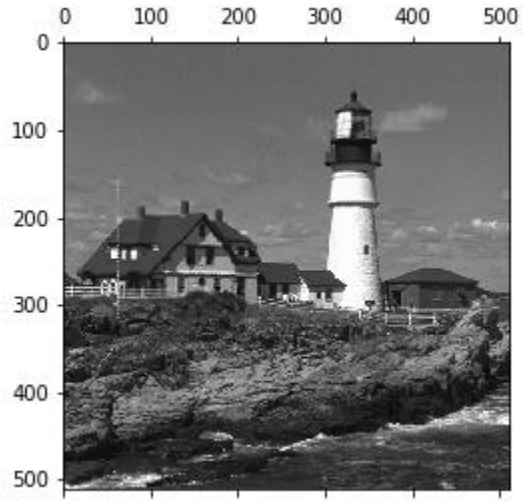


Figure 21: Complete Image

(b) Spatial Warping

- The reference points used in top region in image coordinates are: (0,0),(0,256),(0,511),(128,128),(256,256),(384,128) mapped to (0,0),(256,128),(0,511),(128,128),(256,256),(384,128)
- The reference points used in left region in image coordinates are: (0,0),(256,0),(511,0),(128,128),(256,256),(128,384) mapped to (0,0),(256,128),(511,0),(128,128),(256,256),(128,384)
- The reference points used in bottom region in image coordinates are: (511,0),(384,128),(256,256),(384,384),(511,511),(511,256) mapped to (511,0),(384,128),(256,256),(384,384),(511,511),(384,256)
- The reference points used in bottom region in image coordinates are: (0,511),(128,384),(256,256),(384,384),(511,511),(256,511) mapped to (0,511),(128,384),(256,256),(384,384),(511,511),(256,384)
- The coefficient values for polynomial equation in top region is:

```
array([[  0.00000000e+00,   1.00000000e+00,  -1.11022302e-16,
          1.30104261e-18,   8.67361738e-19,   4.33680869e-19],
       [  0.00000000e+00,  -3.33066907e-16,   1.00000000e+00,
          1.96078431e-03,  -7.68935025e-06,  -1.96847366e-03]])
```

- The coefficient values for polynomial equation in left region is:

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

```
array([[  0.00000000e+00,   1.00000000e+00,  -1.11022302e-16,
          1.96847366e-03,   7.68935025e-06,  -1.96078431e-03],
       [  0.00000000e+00,  -3.33066907e-16,   1.00000000e+00,
         -4.33680869e-19,   0.00000000e+00,  -4.33680869e-19]])
```

- The coefficient values for polynomial equation in bottom region is:

```
array([[  0.00000000e+00,   1.00000000e+00,   0.00000000e+00,
          0.00000000e+00,   0.00000000e+00,   0.00000000e+00],
       [  0.00000000e+00,   0.00000000e+00,   1.00000000e+00,
         -1.96078431e-03,  -7.65931373e-06,   1.95312500e-03]])
```

- The coefficient values for polynomial equation in right region is:

```
array([[  0.00000000e+00,   1.00000000e+00,   0.00000000e+00,
         -1.95312500e-03,   7.65931373e-06,   1.96078431e-03],
       [  0.00000000e+00,  -5.55111512e-17,   1.00000000e+00,
          2.16840434e-19,   0.00000000e+00,   0.00000000e+00]])
```

- The output obtained for warping for 'hat.raw' is shown in figure 23.

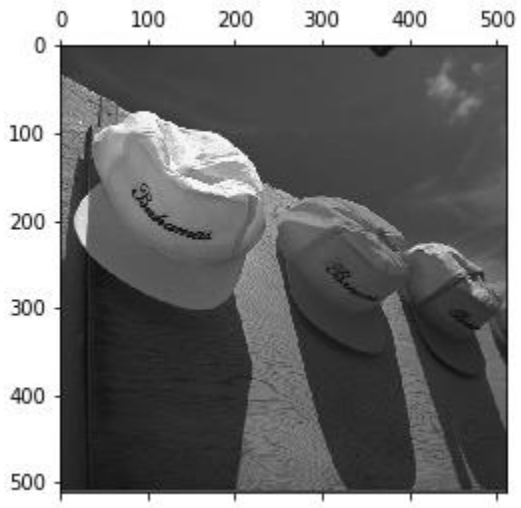


Figure 22: Original hat.raw image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

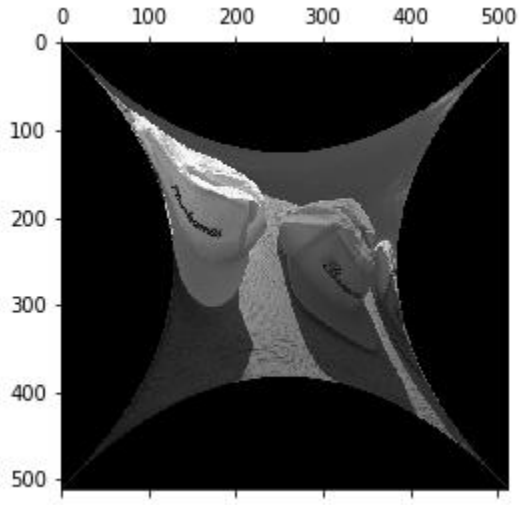


Figure 23: Warped image for hat.raw

(c) Lens Distortion Correction

- I have not used linear regression for this problem.
- The output coordinates were found using the equation mentioned in the approach. The resulting image obtained is shown in figure 25.

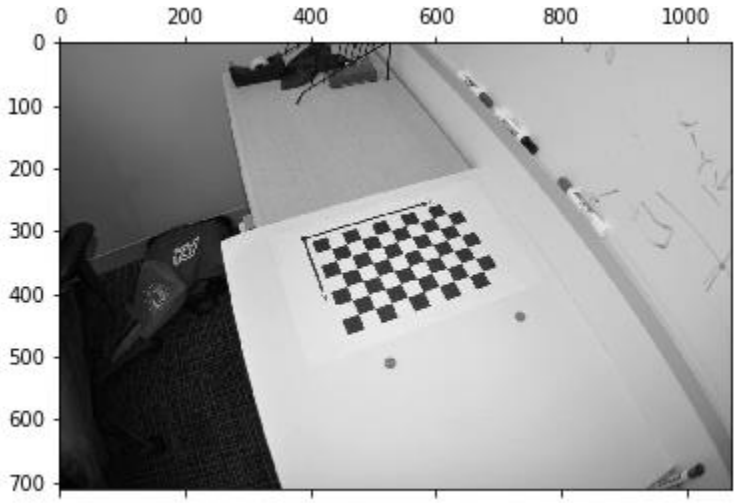


Figure 24: Original classroom.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

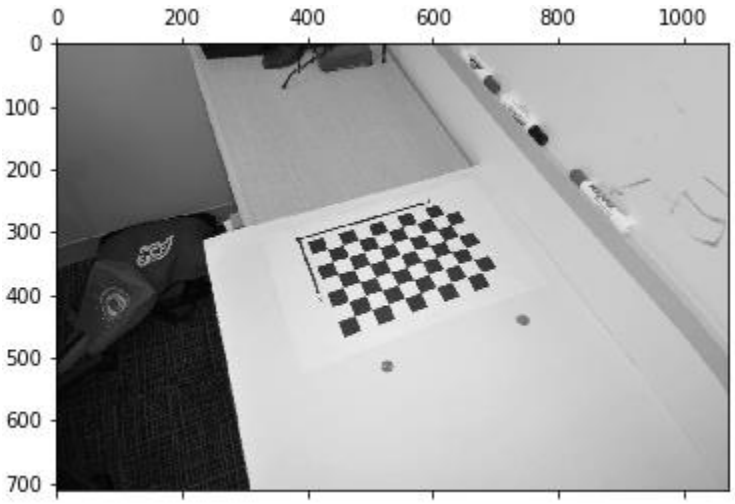


Figure 25: undistorted classroom.raw

- As can be seen in the figure 25, the edge of the table which had a curvature in the original distorted image is changed to a straight line. But, because of the correction the image is magnified and the image in the given dimension loses the boundary objects.
- This is because of the radial effect of the lens as it moves from center towards the edge.
- To include the whole image the dimensions were extended for each height and width. The observed output is shown in figure 26.

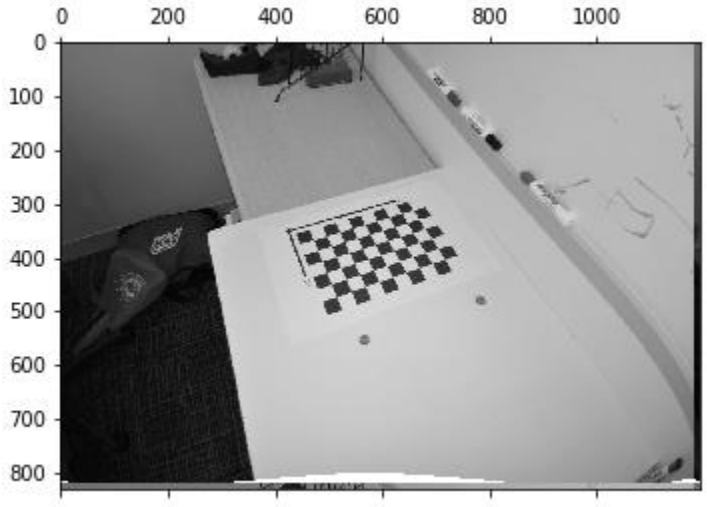


Figure 26: Undistorted Image with increasing size

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- As can be seen in figure 26, extending gives radial distortion in the image on all the four directions. The distortion in the bottom and right can be evidently seen and by further extending the dimensions for height and width, distortion in top and left regions can be seen.

Problem 2: Morphological Processing

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. The basic idea behind several morphological processing makes use of what is called hit or miss transformations. Several morphological operations like dilation, erosion, shrinking, thinning, skeletonizing etc works on the basis of hit or miss transformation. This concept makes use of a small odd sized mask, typically 3x3 scanned through a binary image. If the pixels in the mask matches exactly with the pixels in the image, the output value for the pixel is set to a desired value, usually the color of the foreground. Otherwise, it is set to background value. Setting the output pixel value to foreground color is a hit and background is a miss. The conditions for a hit can also be represented using a logical relationship between the pixel and its neighborhood. The masks for morphological processing can be categorized into additive and subtractive operators. Additive operators cause the center pixel of a 3x3 pixel window to change from background to foreground value whereas the subtractive filters change the pixel value from background to foreground value. In this section, we discuss about few morphological processing such as Shrinking, Thinning, Skeletonizing, Dilation, Erosion etc.

(a) Basix Morphological Processes Implementation

(i) Shrinking

Shrinking is a technique which erases black pixels such that an object without holes erodes to a single pixel at or near its center of mass, and an object with holes erodes to a connected ring lying midway between each hole and its nearest outer boundary. This center of mass depends also on the number of pixels around, for instance a 3X3 pixel object is shrunk to its center whereas 2x2 pixel object is shrunk arbitrarily, by definition, to a single pixel at its lower right corner. A single 3x3 mask can not guarantee a complete shrinkage of the object. A 5x5 mask could be deployed for this task as it has enough information about the neighborhood to shrink the object. But, such a 5x5 mask would need comparison of nearly 2^{25} pattern combinations which would computationally be very expensive. Hence, a two-stage technique can be deployed for each of Shrinking, Thinning and Skeletonizing morphological implementations. The input data is first examined with what is called conditional masks (M) in the first stage and the output of this stage is then examined with unconditional masks (P) as shown in figure below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

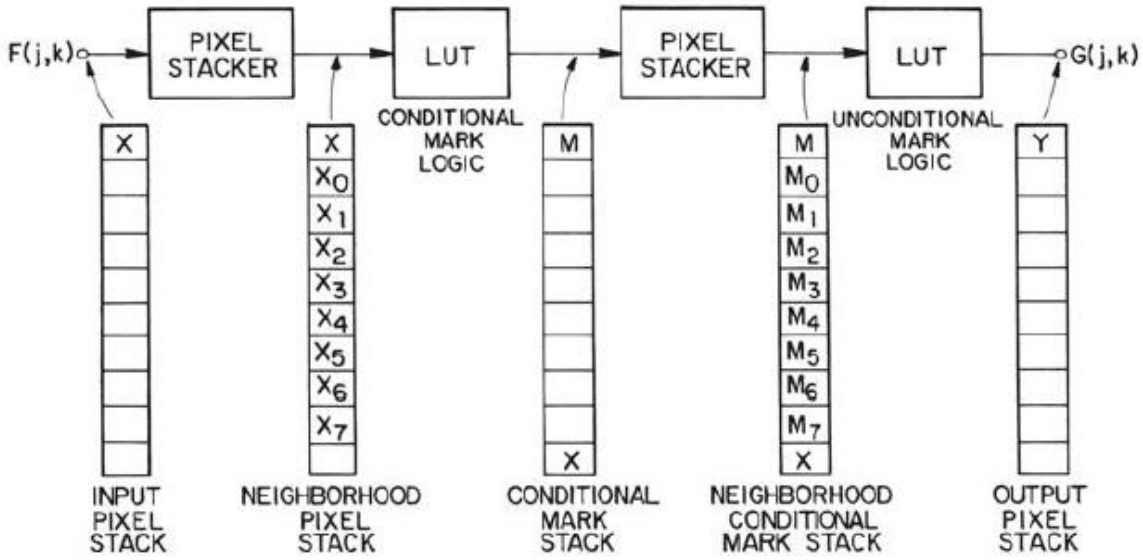


Figure 27: Lookup table for two stage implementation of Shrinking, Thinning and Skeletonizing

In the first stage, the states of nine neighboring pixels are gathered together by a pixel stacker and lookup table generates a conditional mask M for the possible erasures. The patterns to be compared is shown in the figure. In the second stage, the center pixel X and the conditional masks for 3x3 neighborhood centered about X are examined to create an output pixel. The logical relation to obtain shrinking is shown in the equation below.

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

where $P(M,M_0,\ldots,M_7)$ is an erasure inhibiting logical variable as defined in table shown in figure below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

TABLE 14.3-1. Shrink, Thin and Skeletonize Conditional Mark Patterns [$M = 1$ if hit]

Table	Bond	Pattern							
		0 0 1	1 0 0	0 0 0	0 0 0				
S	1	0 1 0	0 1 0	0 1 0	0 1 0				
		0 0 0	0 0 0	1 0 0	0 0 1				
		0 0 0	0 1 0	0 0 0	0 0 0				
S	2	0 1 1	0 1 0	1 1 0	0 1 0				
		0 0 0	0 0 0	0 0 0	0 1 0				
		0 0 1	0 1 1	1 1 0	1 0 0	0 0 0	0 0 0	0 0 0	0 0 0
S	3	0 1 1	0 1 0	0 1 0	1 1 0	1 1 0	0 1 0	0 1 0	0 1 1
		0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	1 1 0	0 1 1	0 0 1
		0 1 0	0 1 0	0 0 0	0 0 0				
TK	4	0 1 1	1 1 0	1 1 0	0 1 1				
		0 0 0	0 0 0	0 1 0	0 1 0				
		0 0 1	1 1 1	1 0 0	0 0 0				
STK	4	0 1 1	0 1 0	1 1 0	0 1 0				
		0 0 1	0 0 0	1 0 0	1 1 1				
		1 1 0	0 1 0	0 1 1	0 0 1				
ST	5	0 1 1	0 1 1	1 1 0	0 1 1				
		0 0 0	0 0 1	0 0 0	0 1 0				
		0 1 1	1 1 0	0 0 0	0 0 0				
ST	5	0 1 1	1 1 0	1 1 0	0 1 1				
		0 0 0	0 0 0	1 1 0	0 1 1				
		1 1 0	0 1 1						
ST	6	0 1 1	1 1 0						
		0 0 1	1 0 0						
		1 1 1	0 1 1	1 1 1	1 1 0	1 0 0	0 0 0	0 0 0	0 0 1
STK	6	0 1 1	0 1 1	1 1 0	1 1 0	1 1 0	1 1 0	0 1 1	0 1 1
		0 0 0	0 0 1	0 0 0	1 0 0	1 1 0	1 1 1	1 1 1	0 1 1

(Continued)

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

		1 1 1	1 1 1	1 0 0	0 0 1						
STK	7	0 1 1	1 1 0	1 1 0	0 1 1						
		0 0 1	1 0 0	1 1 1	1 1 1						
		0 1 1	1 1 1	1 1 0	0 0 0						
STK	8	0 1 1	1 1 1	1 1 0	1 1 1						
		0 1 1	0 0 0	1 1 0	1 1 1						
		1 1 1	0 1 1	1 1 1	1 1 1	1 1 1	1 1 0	1 0 0	0 0 1		
STK	9	0 1 1	0 1 1	1 1 1	1 1 1	1 1 0	1 1 0	1 1 1	1 1 1		
		0 1 1	1 1 1	1 0 0	0 0 1	1 1 0	1 1 1	1 1 1	1 1 1		
		1 1 1	1 1 1	1 1 1	1 0 1						
STK	10	0 1 1	1 1 1	1 1 0	1 1 1						
		1 1 1	1 0 1	1 1 1	1 1 1						
		1 1 1	1 1 1	1 1 0	0 1 1						
K	11	1 1 1	1 1 1	1 1 1	1 1 1						
		0 1 1	1 1 0	1 1 1	1 1 1						

Figure 28: Conditional mask patterns for Shrinking, Thinning, Skeletonizing

Here, S corresponds to patterns for Shrinking, T corresponds to pattern for Thinning and K corresponds to that of Skeletonizing. The unconditional mask for shrinking and thinning is shown in the table of figure shown below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Pattern							
Spur		Single 4-connection					
0 0 *M*	*M*0 0	0 0 0	0 0 0				
0 *M*0	0 *M*0	0 *M*0	0 *MM*				
0 0 0	0 0 0	0 *M*0	0 0 0				
L Cluster							
0 0 *M*	0 *MM*	*MM*0	*M*0 0	0 0 0	0 0 0	0 0 0	0 0 0
0 *MM*	0 *M*0	0 *M*0	*MM*0	*MM*0	0 *M*0	0 *M*0	0 *MM*
0 0 0	0 0 0	0 0 0	0 0 0	*M*0 0	*MM*0	0 *MM*	0 0 *M*
4-Connected offset							
0 *MM*	*MM*0	0 *M*0	0 0 *M*				
*MM*0	0 *MM*	0 *MM*	0 *MM*				
0 0 0	0 0 0	0 0 *M*	0 *M*0				
Spur corner cluster							
0 *A M*	*MB* 0	0 0 *M*	*M*0 0				
0 *MB*	*A M*0	*A M*0	0 *MB*				
*M*0 0	0 0 *M*	*MB* 0	0 *A M*				
Corner cluster							
MMD							
MMD							
DDD							
Tee branch							
*DM*0	0 *MD*	0 0 *D*	*D*0 0	*DMD*	0 *M*0	0 *M*0	*DMD*
MMM	*MMM*	*MMM*	*MMM*	*MM*0	*MM*0	0 *MM*	0 *MM*
*D*0 0	0 0 *D*	0 *MD*	*DM*0	0 *M*0	*DMD*	*DMD*	0 *M*0
Vee branch							
MDM	*MDC*	*CBA*	*ADM*				
DMD	*DMB*	*DMD*	*BMD*				
ABC	*MDA*	*MDM*	*CDM*				
Diagonal branch							
*DM*0	0 *MD*	*D*0 *M*	*M*0 *D*				
0 *MM*	*MM*0	*MM*0	0 *MM*				
*M*0 *D*	*D*0 *M*	0 *MD*	*DM*0				

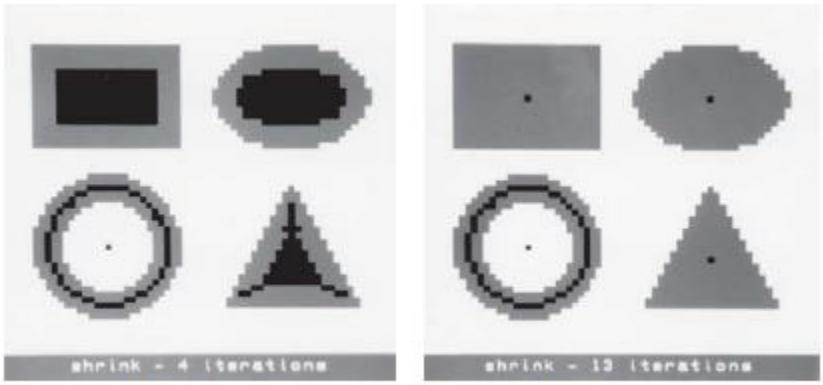
[a] $A \cup B \cup C = 1$ $D = 0 \cup 1$ $A \cup B = 1$.

Figure 29: Unconditional mask patterns for Shrinking and Thinning

The process described above is repeated until convergence. The criteria for convergence, here, is when G(j,k) equals X. For each iteration, G for the next stage is assigned X of the previous stage. The effects of Shrinking can be as shown in the figure below. Note that the background has a white color

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

and foreground has black color pixel values in the examples shown below. However, the implementation for this problem deploys white for foreground and black for background.

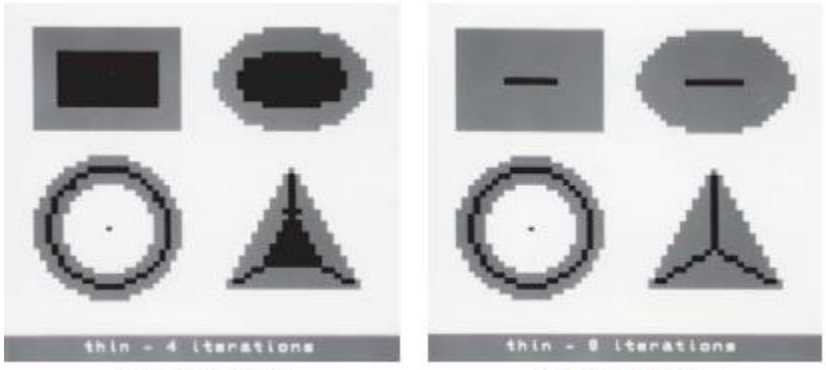


(*a*) Four iterations (*b*) Thirteen iterations

Figure 30: Illustration for shrinking of objects

(ii) Thinning

Thinning is a technique which erases 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 to a minimally connected ring midway between each hole and its nearest outer boundary. Thinning follows the exact same procedure explained for Shrinking. The conditional masks correspond to the letter T in the first column of table shown in figure. The logical equation mentioned above for the shrinking holds for thinning as well. Hence, the implementation of thinning is analogous to shrinking. The effects of thinning is as shown in the figure below.



(*a*) Four iterations (*b*) Eight iterations

Figure 31: Illustration for thinning of objects

(iii) Skeletonizing

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Skeletonizing of an object can be used to describe its structure. Although thinning of objects sometimes have appearances of a skeleton, they are not always uniquely defined. For example, both rectangle and ellipse thin to a horizontal line at its center. The intuitive explanation for skeletonizing based on prairie fire analogy follows that the medial axis skeleton consists of the set of points that are equally distant from two closest points of an object boundary. The minimal distance function is called the quench distance of the object. It is possible to reconstruct the object boundary using the medial axis skeleton of an object and its quench distance. The object boundary is determined by the union of a set of circular disks formed by circumscribing a circle whose radius is the quench distance at each point of the medial axis skeleton. The unconditional masks to be examined for the skeletonizing technique is different from those of shrinking and thinning as shown in the table of figure 17 below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Pattern												
Spur												
0	0	0		0	0	0		0	0	*M*		*M*	0	0
0	*M*	0		*0*	*M*	0		0	*M*	0		0	*M*	0
0	0	*M*		*M*	0	0		0	0	0		0	0	0
Single 4-connection												
0	0	0		0	0	0		0	0	0		0	*M*	0
0	*M*	0		0	*M*	*M*		*M*	*M*	0		0	*M*	0
0	*M*	0		0	0	0		0	0	0		0	0	0
L corner												
0	*M*	0		0	*M*	0		0	0	0		0	0	0
0	*M*	*M*		*M*	*M*	0		0	*M*	*M*		*M*	*M*	0
0	0	0		0	0	0		0	M	0		0	*M*	0
Corner cluster												
M	*M*	*D*		*D*	*D*	*D*						
M	*M*	*D*		*D*	*M*	*M*						
D	*D*	*D*		*D*	*M*	*M*						
Tee branch												
D	*M*	*D*		*D*	*M*	*D*		*D*	*D*	*D*		*D*	*M*	*D*
M	*M*	*M*		*M*	*M*	*D*		*M*	*M*	*M*		*D*	*M*	*M*
D	*D*	*D*		*D*	*M*	*D*		*D*	*M*	*D*		*D*	*M*	*D*
Vee branch												
M	*D*	*M*		*M*	*D*	*C*		*C*	*B*	*A*		*A*	*D*	*M*
D	*M*	*D*		*D*	*M*	*B*		*D*	*M*	*D*		*B*	*M*	*D*
A	*B*	*C*		*M*	*D*	*A*		*M*	*D*	*M*		*C*	*D*	*M*
Diagonal branch												
D	*M*	0		0	*M*	*D*		*D*	0	*M*		*M*	0	*D*
0	*M*	*M*		*M*	*M*	0		M	M	0		0	*M*	*M*
M	0	*D*		*D*	0	*M*		0	*M*	*D*		*D*	*M*	0

[a] $A \cup B \cup C = 1 \quad D = 0 \cup 1.$

Figure 32: Unconditional masks for Skeletonizing

The implementation for skeletonizing is also similar to shrinking and thinning except that there is an additional step of bridging after the convergence. Bridging creates a foreground pixel values if creation results in connectivity of previously unconnected neighboring foreground pixels. The logical implementation for skeletonizing can be shown as below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

$$G(j,k) = X \cup [P_1 \cup P_2 \cup \cdots \cup P_6]$$

where

$$P_1 = \bar{X}_2 \cap \bar{X}_6 \cap [X_3 \cup X_4 \cup X_5] \cap [X_0 \cup X_1 \cup X_7] \cap \bar{P}_Q$$

$$P_2 = \bar{X}_0 \cap \bar{X}_4 \cap [X_1 \cup X_2 \cup X_3] \cap [X_5 \cup X_6 \cup X_7] \cap \bar{P}_Q$$

$$P_3 = \bar{X}_0 \cap \bar{X}_6 \cap X_7 \cap [X_2 \cup X_3 \cup X_4]$$

$$P_4 = \bar{X}_0 \cap \bar{X}_2 \cap X_1 \cap [X_4 \cup X_5 \cup X_6]$$

$$P_5 = \bar{X}_2 \cap \bar{X}_4 \cap X_3 \cap [X_0 \cup X_6 \cup X_7]$$

$$P_6 = \bar{X}_4 \cap \bar{X}_6 \cap X_5 \cap [X_0 \cup X_1 \cup X_2]$$

and

$$P_Q = L_1 \cup L_2 \cup L_3 \cup L_4$$

$$L_1 = \bar{X} \cap \bar{X}_0 \cap X_1 \cap \bar{X}_2 \cap X_3 \cap \bar{X}_4 \cap \bar{X}_5 \cap \bar{X}_6 \cap \bar{X}_7$$

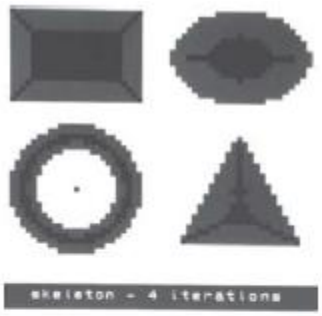
$$L_2 = \bar{X} \cap \bar{X}_0 \cap \bar{X}_1 \cap \bar{X}_2 \cap X_3 \cap \bar{X}_4 \cap X_5 \cap \bar{X}_6 \cap \bar{X}_7$$

$$L_3 = \bar{X} \cap \bar{X}_0 \cap \bar{X}_1 \cap \bar{X}_2 \cap \bar{X}_3 \cap \bar{X}_4 \cap X_5 \cap \bar{X}_6 \cap X_7$$

$$L_4 = \bar{X} \cap \bar{X}_0 \cap X_1 \cap \bar{X}_2 \cap \bar{X}_3 \cap \bar{X}_4 \cap \bar{X}_5 \cap \bar{X}_6 \cap X_7$$

This step is additionally implemented after the two-stage implementation of skeletonizing using conditional and unconditional masks. The illustration for skeletonizing can be as shown in the figure below.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019



(a) Four iterations

(b) Ten iterations

Figure 33: Illustration for skeletonizing objects

(b) Defect detection and correction

One of the applications of morphological processing is defect detection. Various additive or subtractive filters can be used to detect different kinds of noise present in the image. For the deer image given in the problem, the noise in the image looks like an isolated noise pixel. This kind of noise can be detected using a mask explained with logical relation shown below:

$$M = \bar{x} \cap x_0 \cap x_1 \cap x_2 \cap x_3 \cap x_4 \cap x_5 \cap x_6 \cap x_7$$

where $x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7$ are pixel location that follows the following neighborhood locations.

x_3	x_2	x_1
x_4	x	x_0
x_5	x_6	x_7

PIXEL NEIGHBORHOOD

If M is hit, then there is a defect that is directed otherwise there is no defect. The above logical implementation can be visualized as a 3x3 mask as shown below.

1	1	1

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

1	0	1
1	1	1

The mask is examined for every pixel location in the given image and the counter is increased every time there is a hit. Thus, the total defect is noted. For visualization of the noise pixels, the noise pixels are made white and the rest of the image is made black so that defect is noticeable.

Now, to fill the defect we can use the same mask and fill with white pixels whenever there is a hit i.e. whenever there is a defect detected.

(c) Object Analysis

Rice grain inspection is a procedure to define rice quality in the marketplace. The task to find out the number of rice grains was carried out with number of morphological operations. Since, the given image is of three channels, it needs to be converted to the grayscale image. Since, rice varieties are distributed over a range of gray scale values from dark to bright, direct thresholding of grayscale doesn't yield proper binarization of the rice grains. Hence, closing operation was performed initially on the grayscale image.

Closing operation is obtained by dilation followed by erosion. Generalized dilation is defined as

$$G(j,k) = F(j,k) \oplus H(j,k)$$

where F(j,k) for 1<j, k<N is a binary valued image and H(j,k) for 1<j, k<L where L is an odd integer, is a binary valued array called a structuring element. Dilation is associated with filling of holes of certain shape and size given by Structural element.
For the grayscale image, dilation can be defined as

$$G(j,k) = \text{MAX}\{F(j,k), F(j,k+1), F(j-1,k+1), \ldots, F(j+1,k+1)\}$$

where MAX defines the largest amplitude of pixels from the nine neighboring pixels of the pixel under inspection.

Generalized erosion is defined as

$$G(j,k) = F(j,k) \ominus H(j,k)$$

where F(j,k) for 1<j, k<N is a binary valued image and H(j,k) for 1<j, k<L where L is an odd integer, is a binary valued array called a structuring element. Erosion is associated with removal of holes of certain shape and size given by Structural element.
For the grayscale image, erosion can be defined as

$$G(j,k) = \text{MIN}\{F(j,k), F(j,k+1), F(j-1,k+1), \ldots, F(j+1,k+1)\}$$

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

where MIN defines the minimum amplitude of pixels from the nine neighboring pixels.

After the closing operation was done, the grayscale image was thresholded to binary image using two thresholds upper and lower. This method was followed by shrinking to get the number of white dots corresponding to the number of rice grains in the image.
For detecting, the size of the rice grain, the color rice grain image was read and operations like dilation, erosion and filling for the canny edges of the image was performed on matlab. Further the area and centroid of each of the filled region was found. The grains were separated into 11 clusters by separating them by their centroid locations. Maximum area of grain in each of the 11 clusters were printed out and analyzed for ranking their sizes.

2. Approach and Procedure

(a) Basic Morphological Process Implementation

- The input image 'pattern1.raw' of size 375x375 was read byte by byte.
- The Shrinking conditional masks were implemented using the table above corresponding to the letter 'S' in the first column.
- The unconditional masks for shrinking as shown in the table above were also implemented.
- A function to compare the masks with the given array of nine elements was written.
- Iterating through the pixel coordinates of input X, the 9x9 neighborhood of each pixel was formed and compared with the conditional masks using the function mentioned above.
- If there is a hit, the intermediate output M is made 1 otherwise made 0.
- The intermediate output M is then iterated through the values and compared with unconditional masks using the function mentioned above.
- If there is a hit, the unconditional output P is made 1 otherwise made 0.
- Now, using the logical relationship between X, M and P mentioned in the abstract part above, the final output G is found.
- The criteria for G = X was checked. If the criteria is True, the convergence has occurred. Otherwise, X = G and the whole procedure above for shrinking is repeated until convergence.
- The Thinning conditional masks were implemented using the table above corresponding to the letter 'T' in the first column.
- The unconditional masks for thinning as shown in the table above were also implemented.
- A function to compare the masks with the given array of nine elements was written.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- Iterating through the pixel coordinates of input X, the 9x9 neighborhood of each pixel was formed and compared with the conditional masks using the function mentioned above.
- If there is a hit, the intermediate output M is made 1 otherwise made 0.
- The intermediate output M is then iterated through the values and compared with unconditional masks using the function mentioned above.
- If there is a hit, the unconditional output P is made 1 otherwise made 0.
- Now, using the logical relationship between X, M and P mentioned in the abstract part above, the final output G is found.
- The criteria for G = X was checked. If the criteria is True, the convergence has occurred. Otherwise, X = G and the whole procedure above for shrinking is repeated until convergence.
- The Skeletonizing conditional masks were implemented using the table above corresponding to the letter 'K' in the first column.
- The unconditional masks for skeletonizing as shown in the table above were also implemented.
- A function to compare the masks with the given array of nine elements was written.
- Iterating through the pixel coordinates of input X, the 9x9 neighborhood of each pixel was formed and compared with the conditional masks using the function mentioned above.
- If there is a hit, the intermediate output M is made 1 otherwise made 0.
- The intermediate output M is then iterated through the values and compared with unconditional masks using the function mentioned above.
- If there is a hit, the unconditional output P is made 1 otherwise made 0.
- Now, using the logical relationship between X, M and P mentioned in the abstract part above, the final output G is found.
- The criteria for G = X was checked. If the criteria is True, the convergence has occurred. Otherwise, X = G and the whole procedure above for shrinking is repeated until convergence.
- After the convergence has occurred, one final step of bridging is done according to the relation mentioned in the abstract part.
- The above three procedures for Shrinking, Thinning and Skeletonizing was performed for each of 'pattern2.raw', 'pattern3.raw' and 'pattern4.raw'.
- The outputs were saved, observed and analyzed.

(b) Defect Detection and Correction

- The input image 'deer.raw' of size 691x550 was read byte by byte.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The mask for detecting the isolated noise for 4 connectivity and 8 connectivity was implemented based on the equation mentioned in the abstract part.
- The input image was iterated throughout all the pixel locations, and examined with the masks implemented in the previous step.
- If there was a hit, the pixel is the noise. To highlight the noise only the noise pixels were made white to observe the number of defect in the image.
- The total defect region in the output is then observed and analyzed.
- For this defect regions, the defect is corrected by making it a white pixel.
- The output is saved, observed and analyzed.

(c) Object Analysis

- The input image 'rice.raw' of size 500x690x3 was read byte by byte.
- The color image is converted to grayscale image using weighted conversion formula.
- The grayscale image is used to perform closing operation. Closing is defined as dilation followed by erosion.
- For dilation, the maximum amplitude of the neighborhood pixels is found and replaced for the current pixel location.
- For erosion, the minimum amplitude of the neighborhood pixels is found and replaced for the current pixel location.
- Each of dilation and erosion is performed for 3 iteration in cascaded way.
- The output of the cascade is then binarized using two threshold values: for the pixel values >=128 and <=25 the binary value of 1 was given otherwise 0 was given.
- For the obtained binary image, shrinking was performed by the same procedure described above.
- The output gives the number of dots corresponding to the number of rice grains in the image. The observations were noted and analyzed.
- Further, the input image 'rice.raw' was read byte by byte in matlab.
- Canny edge was detected for the color image.
- Dilation was performed for the edge image with line shaped structure element in horizontal and vertical directions.
- The holes in the dilated image was then filled using algorithm such as watershed used by matlab.
- The filled image was then eroded with diamond shaped structure element.
- Regionprops function was then used to find the area and the centroid of the objects in the eroded image.
- The given image was divided into regions of 11 clusters corresponding to 11 categories of rice grain.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The area of each rice grain was associated with corresponding clusters.
- The max area of rice grain in each of the clusters were utilized to compare the grain sizes and rank them based on their sizes.

3. Observation and Discussions

Shrinking results:

Pattern1.raw

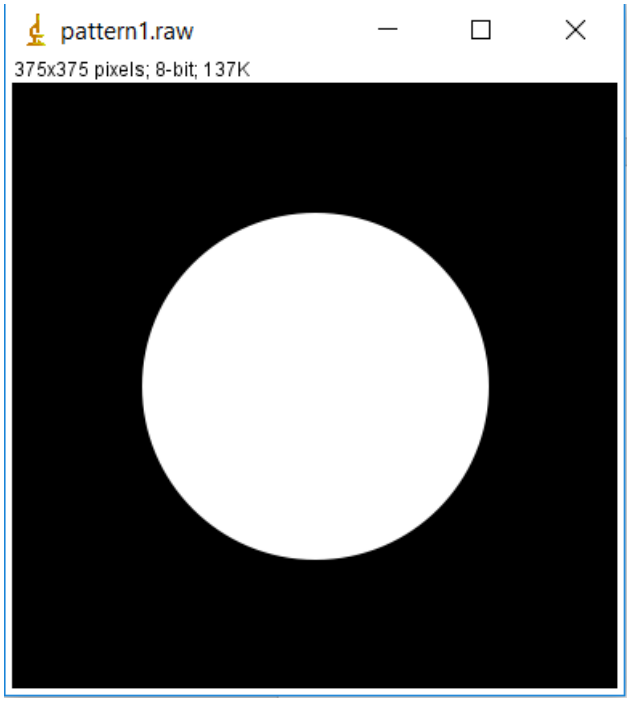


Figure 34: Pattern1.raw image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

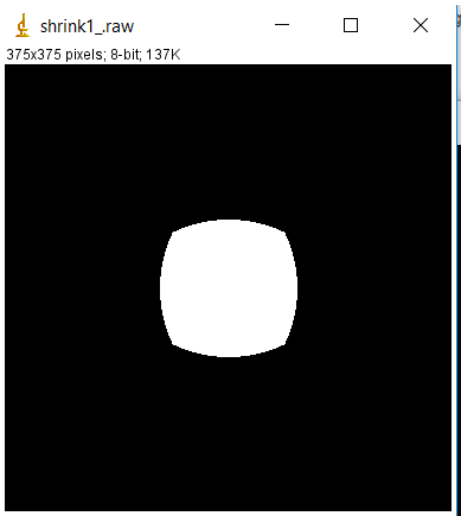
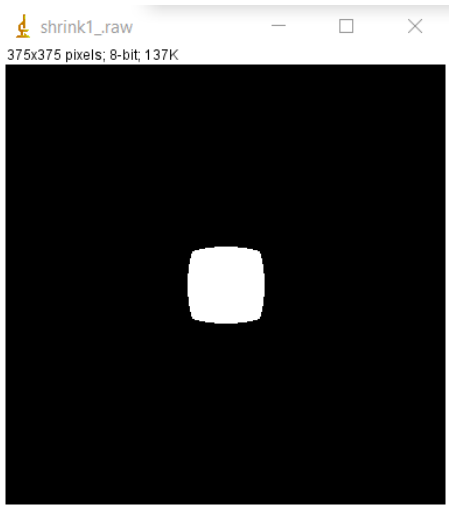


Figure 35: (a) Shrinking after 50 iterations



(b) Shrinking after 75 iterations

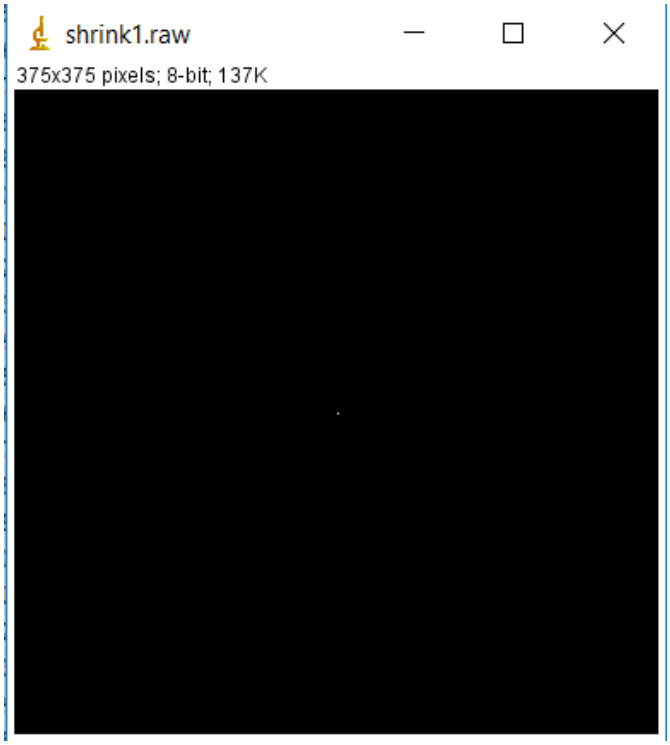


Figure 36: Shrinking of Pattern1.raw

- The pattern1.raw is the circle without a hole and the shrinking will shrink the image to a dot at its center of mass.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The center of mass for the circle is the center of the circle. Hence, the output with a dot at the center of the circle is observed.
- Total number of iterations to converge is 109
- The shrinking for different iterations is shown.

Pattern2.raw :

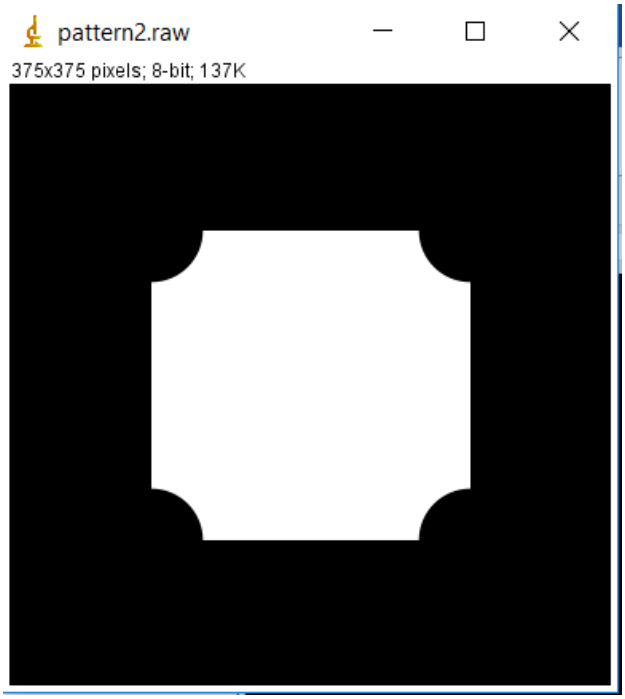


Figure 37: Pattern2.raw Image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

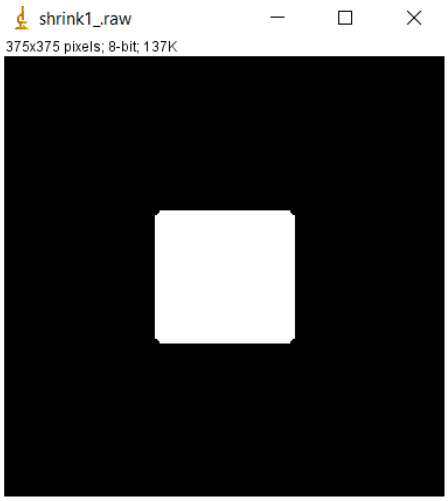
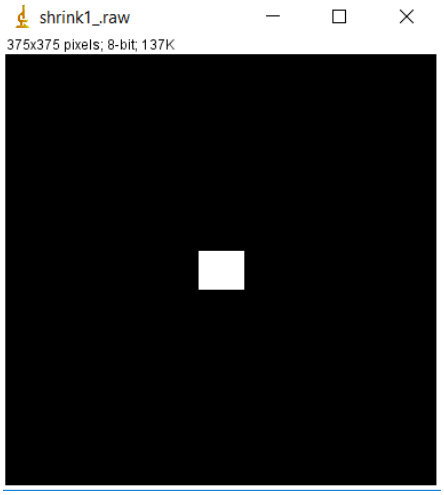


Figure 38: (a) Shrinking after 40 iterations



(b) Shrinking after 80 iterations

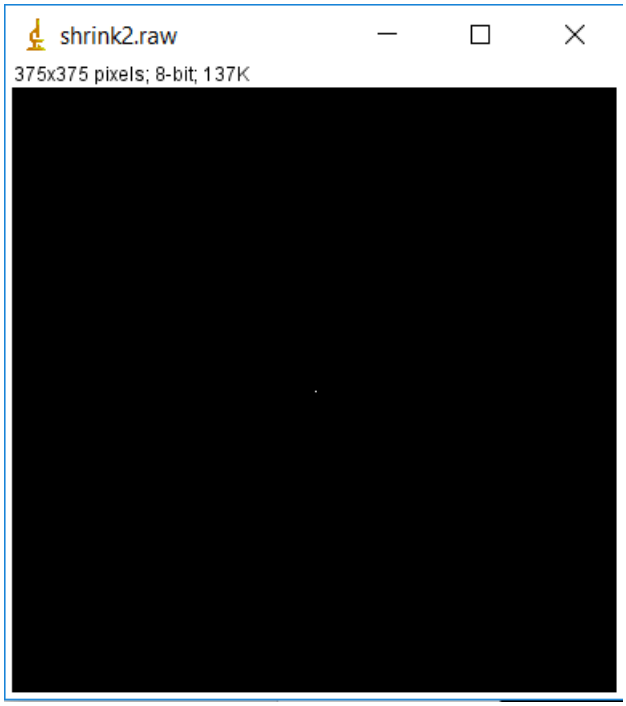


Figure 39: Shrinking of Pattern2.raw

- The pattern2.raw is a solid object without a hole and the shrinking will shrink the image to a dot at its center of mass.
- The center of mass for the object is the center its center. Hence, the output with a dot at the center of the object is observed.
- Total number of iterations to converge is 101

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The shrinking for different iterations is shown.

Pattern3.raw

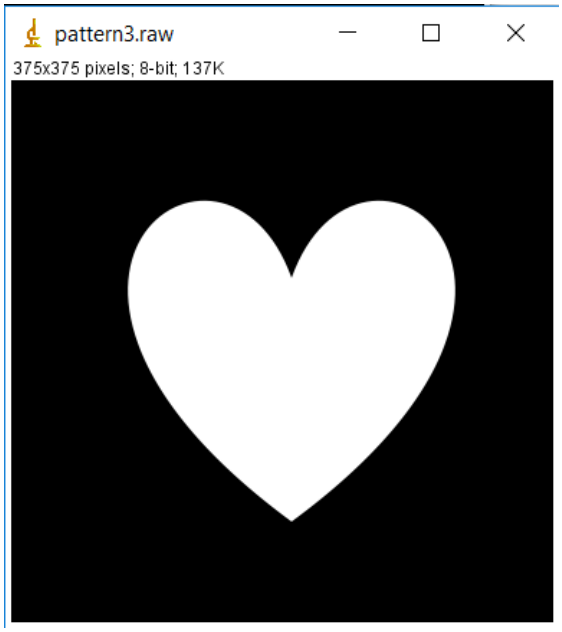


Figure 40: Pattern3.raw image

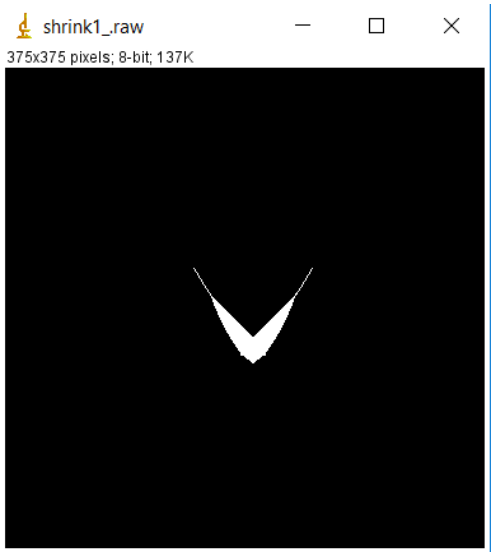
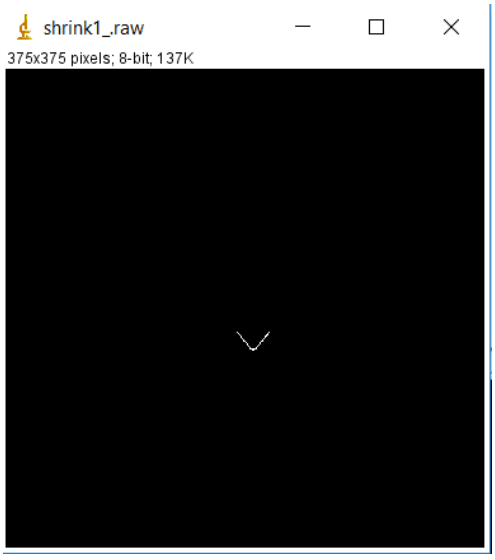


Figure 41 : (a) Shrinking after 75 iterations



(b) Shrinking after 125 iterations

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

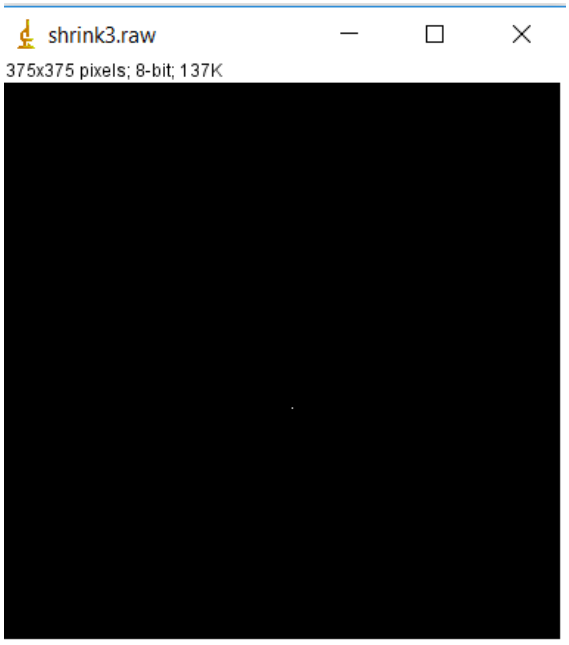


Figure 42: Shrinkind for pattern3.raw

- The pattern3.raw is a solid object without a hole and the shrinking will shrink the image to a dot at its center of mass.
- The object appears to be of even dimension i.e. the line corresponding to center of mass of heart has even white pixels. By default, the algorithm chooses the one at the right side of the center for its center of mass.
- The depth of dot is more compared to pattern1 and pattern2 because the heart doesn't have equal distribution of pixels in the top region and majority is concentrated at the bottom region.
- Total number of iterations to converge is 143
- The shrinking for different iterations is shown.

Pattern4

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

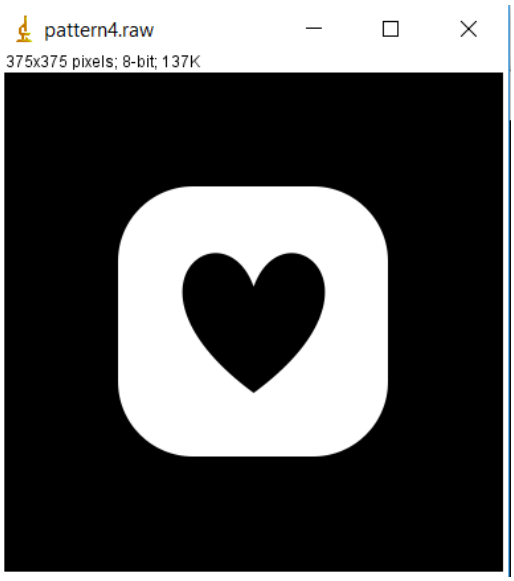


Figure 43: Pattern4.raw Image

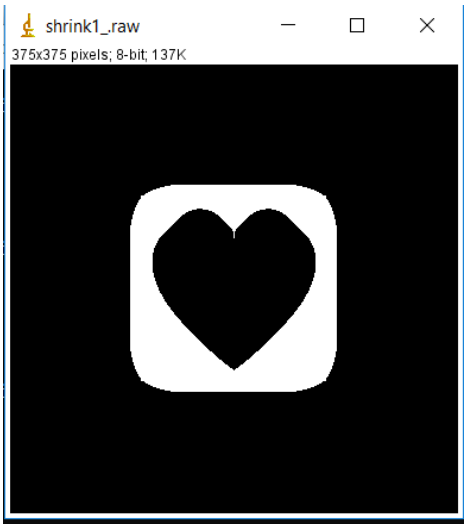


Figure 44: (a) Shrinking after 15 iterations

(b) Shrinking after 30 iterations

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

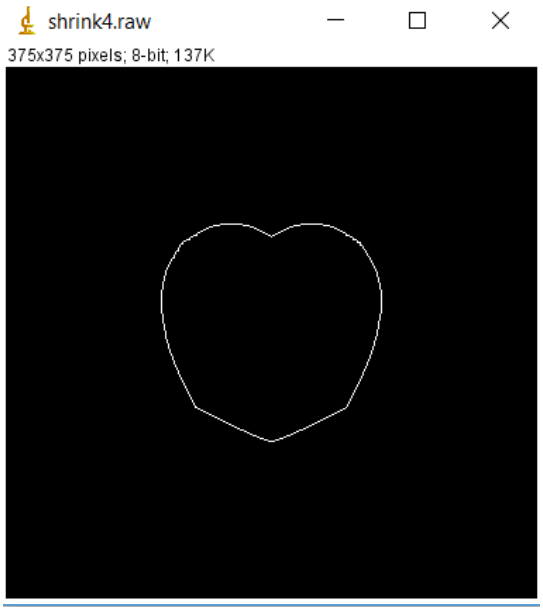


Figure 45: Shrinking for Pattern4.raw

- The pattern4.raw is a solid object with a hole and the shrinking will shrink the image to a line midway from its boundaries.
- The object appears to be of even dimension i.e. the line corresponding to center of mass of heart has even white pixels. By default, the algorithm chooses the one at the right side of the center for its center of mass.
- For the reasons specified above, the lines of shrinked image are not smooth and curve like.
- Total number of iterations to converge is 49
- The shrinking for different iterations is shown.

Thinning results:

Pattern1.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

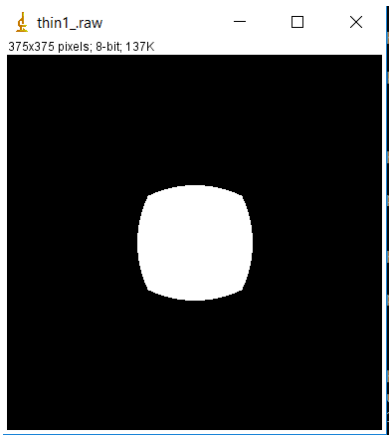


Figure 46(a) :Thinning after 50 iterations

(b) Thinning after 80 iterations

Figure 47: Thinning for pattern1.raw

- The pattern1.raw is a solid circle object without a hole and the shrinking will shrink the image to its center.
- Also, its center of mass is equidistant from the boundary pixels. Hence, thinning will also result in a dot.
- We see that thinning and shrinking has same effect on the object.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- Total number of iterations to converge is 109.
- The thinning for different iterations is shown.

Pattern2.raw

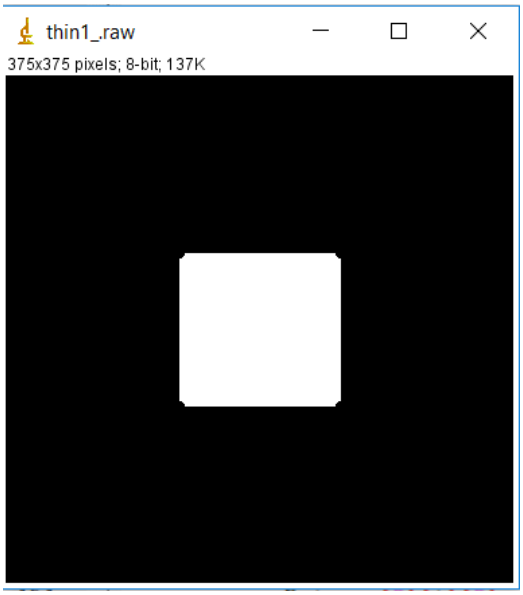
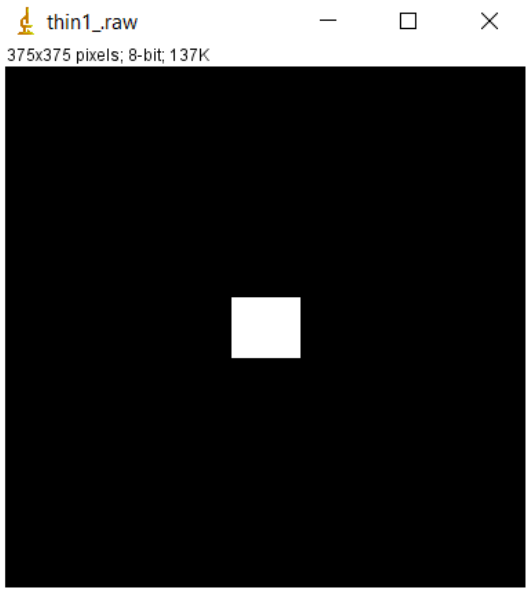


Figure 48: (a) Thinning after 40 iterations



(b) Thinning after 75 iterations

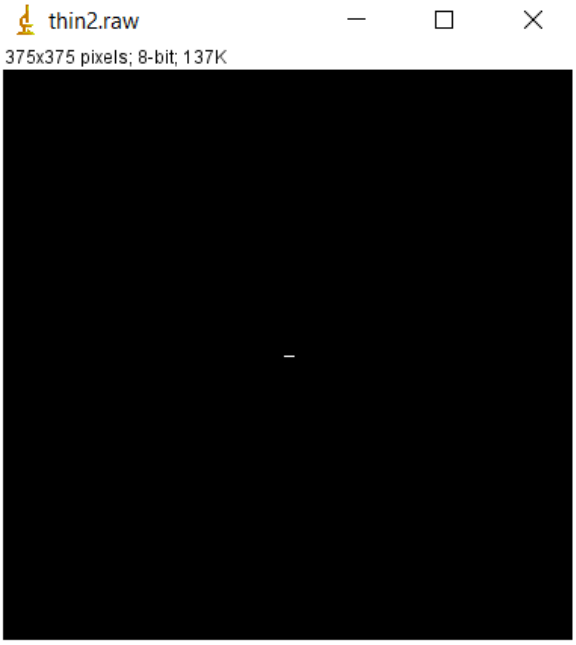


Figure 49: Thinning for pattern2.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The pattern2.raw is a solid circle object without a hole and the shrinking will shrink the image to its center.
- However, it has a line which is equidistant from its boundaries which become the thinning for the image.
- We see that thinning and shrinking do not have same effect on the object. However, the thinning is in the same line s that of the shrinked pixel for this image.
- Total number of iterations to converge is 98.
- The thinning for different iterations is shown.

Pattern3.raw

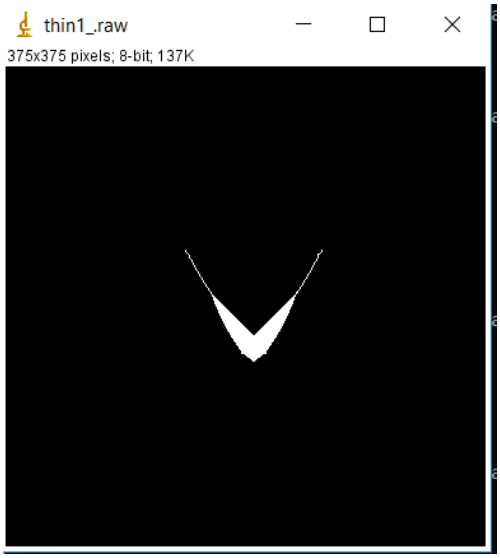


Figure 50: (a) Thinnig after 75 iterations

(b) Thinning after 40 iterations

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Figure 51: Thinning for pattern3.raw

- The pattern3.raw is a solid circle object without a hole and the shrinking will shrink the image to its center. The object appears to be of even dimension i.e. the line corresponding to center of mass of heart has even white pixels. By default, the algorithm chooses the one at the right side of the center for its center of mass.
- However, it has a line which is equidistant from its boundaries which become the thinning for the image.
- We see that thinning and shrinking do not have same effect on the object. However, the thinning is a non-smooth line for the reasons explained above of even dimensions.
- Total number of iterations to converge is 88.
- The thinning for different iterations is shown.

Pattern4.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

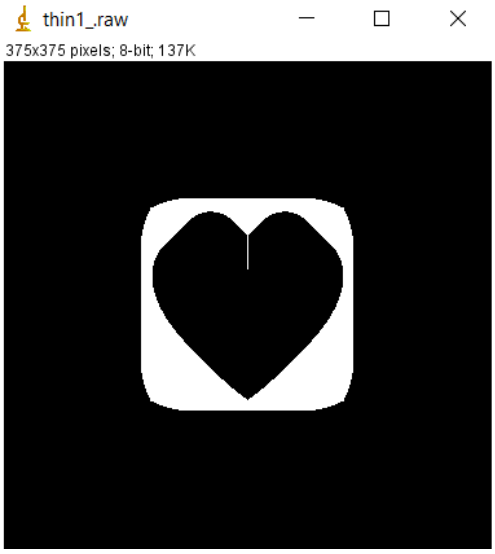
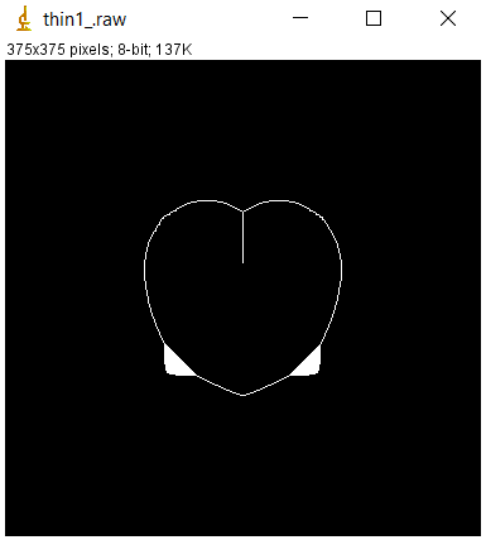


Figure 52: (a) Thinning after 20 iterations



(b) Thinning after 40 iterations

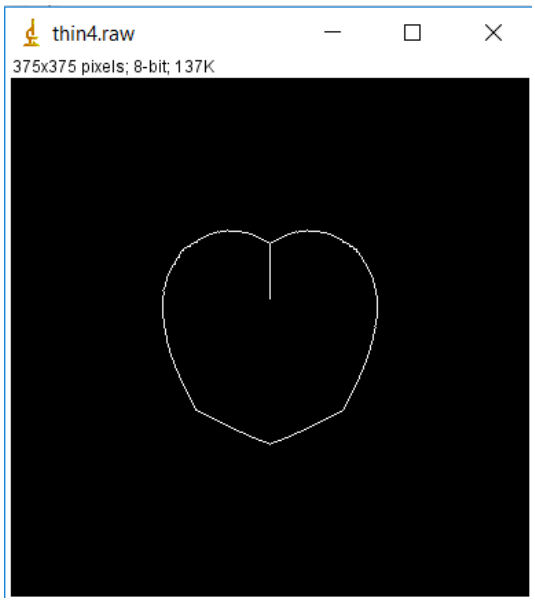


Figure 53: Thinning for pattern4.raw

- The pattern4.raw is a solid circle object with a hole. The object appears to be of even dimension i.e. the line corresponding to center of mass of heart has even white pixels. By default, the algorithm chooses the one at the right side of the center for its center of mass.
- However, it has a line which is equidistant from its boundaries which become the thinning for the image.
- We see that thinning and shrinking do not have same effect on the object. However, the thinning is a non-smooth line for the reasons explained above of even dimensions.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- Total number of iterations to converge is 49.
- The thinning for different iterations is shown.

Skeletonizing Results

Pattern1.raw

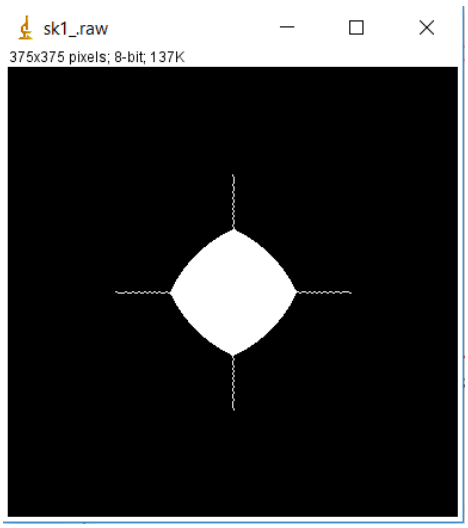
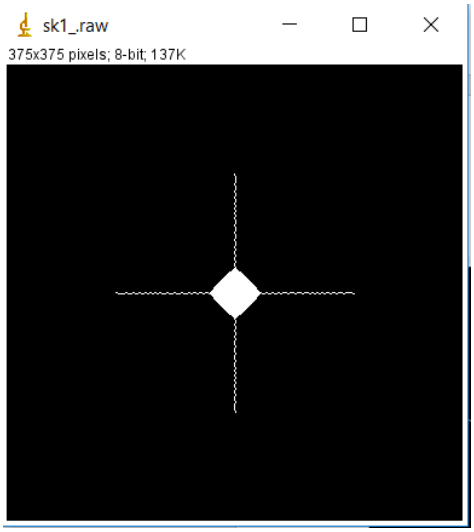


Figure 54(a): Skeletonizing for 45 iterations



(b) Skeletonizing for 65 iterations

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

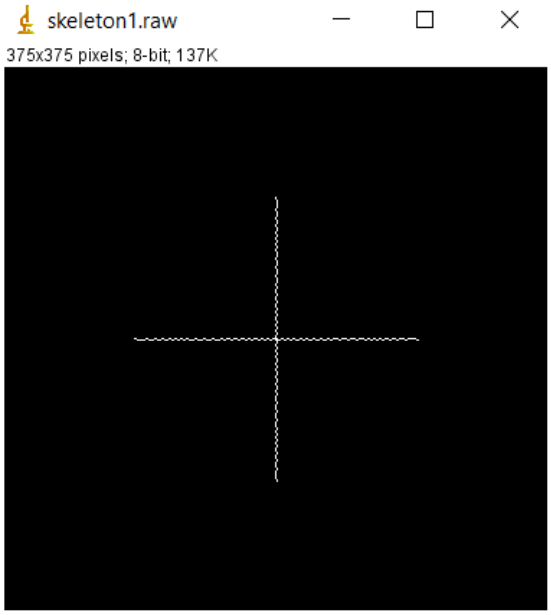


Figure 55: Skeletonizing for pattern1.raw

- The pattern1.raw is a solid circle object without a hole and the shrinking will shrink the image to its center.
- Also, its center of mass is equidistant from the boundary pixels. Hence, thinning will also result in a dot. However, skeletonizing will connect the center to its boundaries
- Total number of iterations to converge is 77.
- The skeletonizing for different iterations is shown.

Pattern2.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

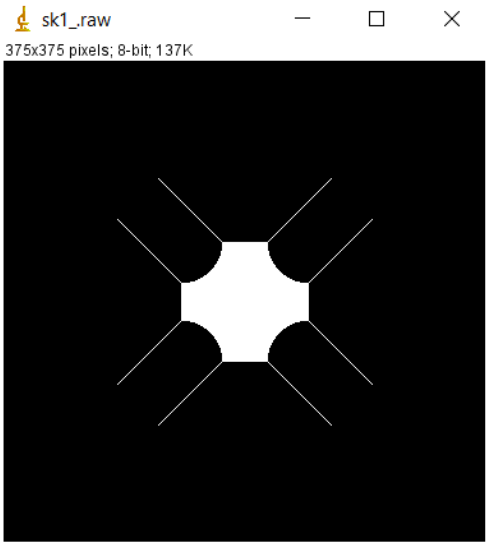
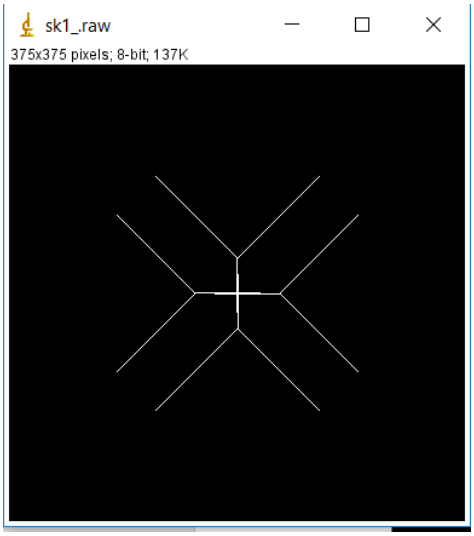


Figure 56(a): Skeletonizing after 50 iterations



(b) Skeletonizing after 80 iterations

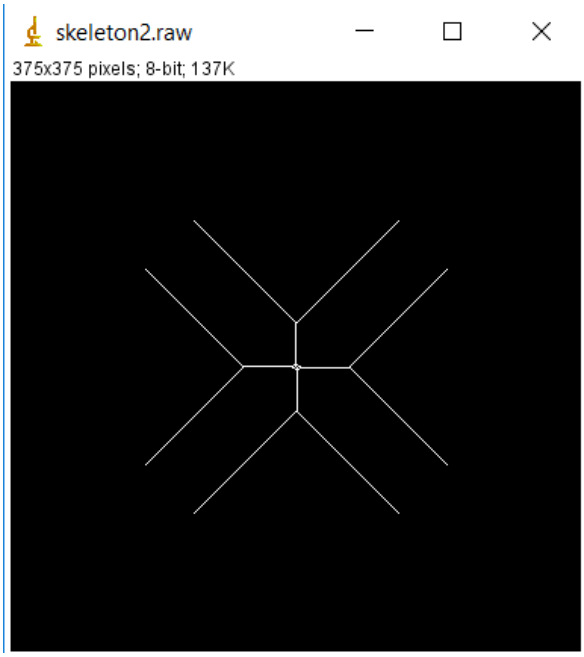


Figure 57: Skeletonizing for pattern2.raw

- The pattern2.raw is a solid object without a hole and the shrinking will shrink the image to its center.
- However, skeletonizing connects all the equidistant pixels from the boundaries wit its corners. It is basically thinning in two directions and connecting the corners.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- Total number of iterations to converge is 97.
- The skeletonizing for different iterations is shown.

Pattern3.raw

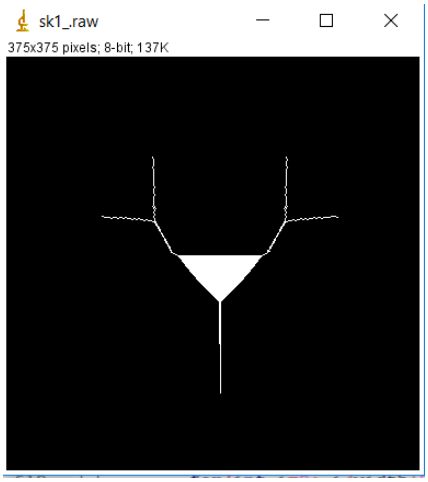
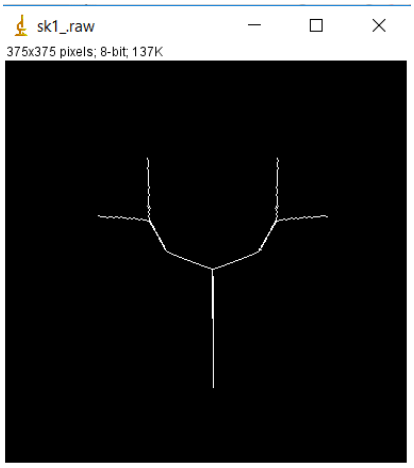


Figure 58(a): Skeletonizing after 45 iterations



(b) Skeletonizing after 65 iterations

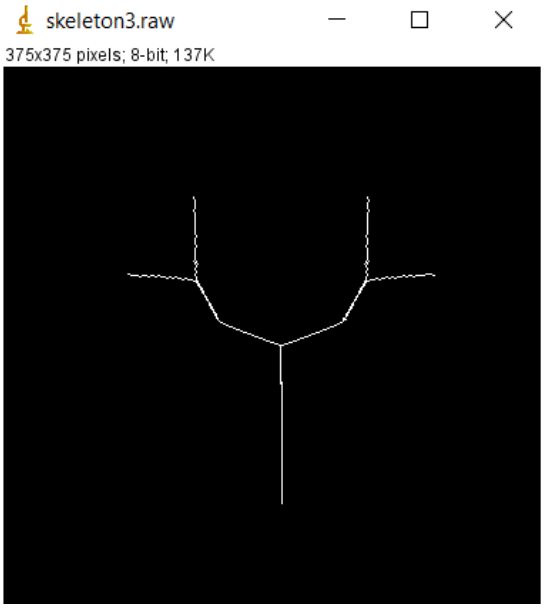


Figure 59: Skeletonizing for pattern3.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- The pattern3.raw is a solid object without a hole and the shrinking will shrink the image to its center.
- However, skeletonizing connects all the equidistant pixels from the boundaries with its corners.
- Because of even dimensions, the lines are non-smooth. The outer two lines corresponds to the boundary on the either sides of the heart which is equidistant from the center of mass.
- Total number of iterations to converge is 85.
- The skeletonizing for different iterations is shown.

Pattern4.raw

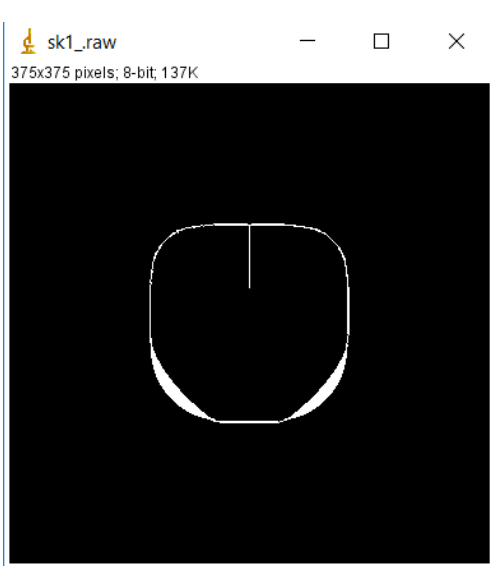
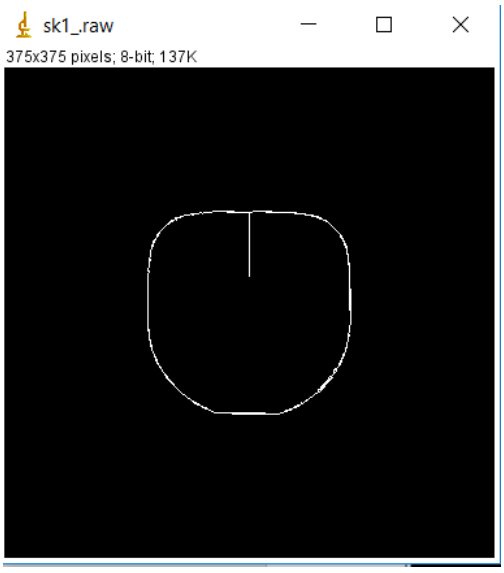


Figure 60: (a) Skeletonizing after 25 iterations



(b) Skeletonizing after 35 iterations

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

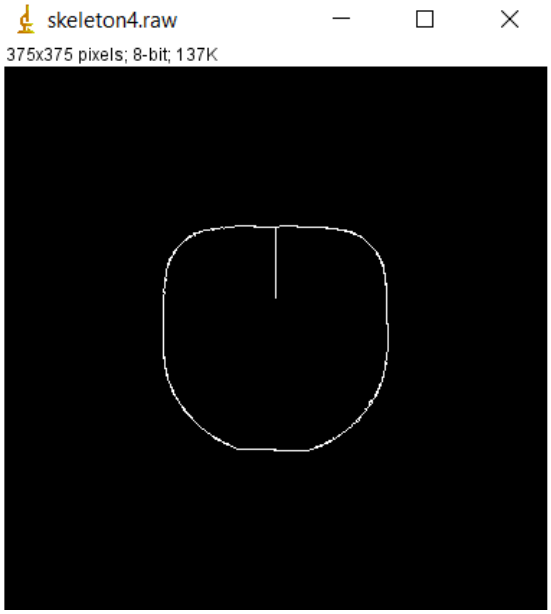


Figure 61: Skeletonizing for Pattern4.raw

- The pattern4.raw is a solid object with a hole and the shrinking will shrink the image to a line midway between the boundaries.
- However, skeletonizing connects this line to the outer corners which is also associated with thinning.
- Because of even dimensions, the lines are non-smooth. The outer two lines corresponds to the boundary on the either sides of the heart which is equidistant from the center of mass.
- Total number of iterations to converge is 48.
- The skeletonizing for different iterations is shown.

General comparison between shrinking, thinning and skeletonizing

- The unconditional masks for skeletonizing is much more than shrinking and thinning. The computation complexity is greater than shrinking and thinning.
- Shrinking and thinning can be same for objects having lines of symmetry in all the angles.
- Thinning and skeletonizing can be same but shrinking and skeletonizing can never be same.

(b) Defect detection and correction

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

- For defect detection as explained in the approach the mask for isolated pixel using four connectivity and eight connectivity was used.
- The output showing only the defect pixels are shown in the figure 62. Total of five defects were found and its coordinates when printed were:
  (i) Noise1(i,j) = (207,498)
  (ii) Noise2(i,j) = (280,93)
  (iii) Noise3(i,j) = (335,334)
  (iv) Noise4(i,j) = (352,331)
  (v) Noise5(i,j) = (284,275)
- The defects can also be said to be present by thinning the deer image and and observing the effects of holes in thinning the image as shown in previous section. The thinning of deer image before and after defect detection and correction is shown in figure 64 and 65 respecively.
- We observe that the number of loops corresponding to the thinning image of defect deer corresponds to number of defects.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Figure 62: Defect detection highlighted by white pixels

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

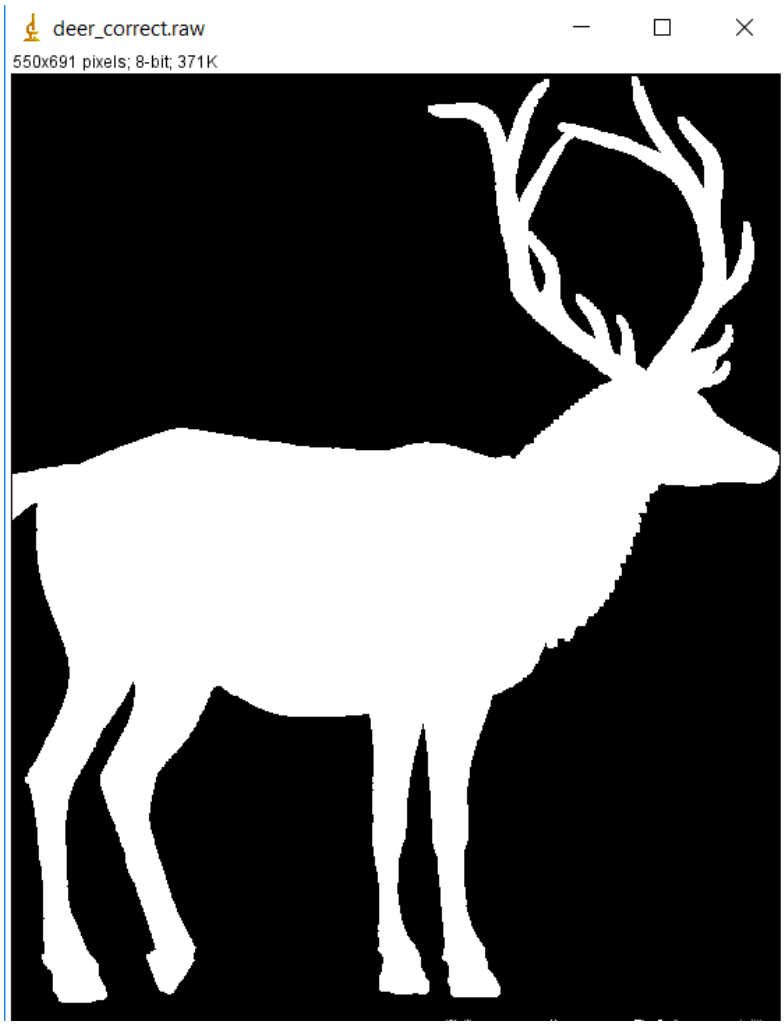


Figure 63: Defect correction for deer.raw

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

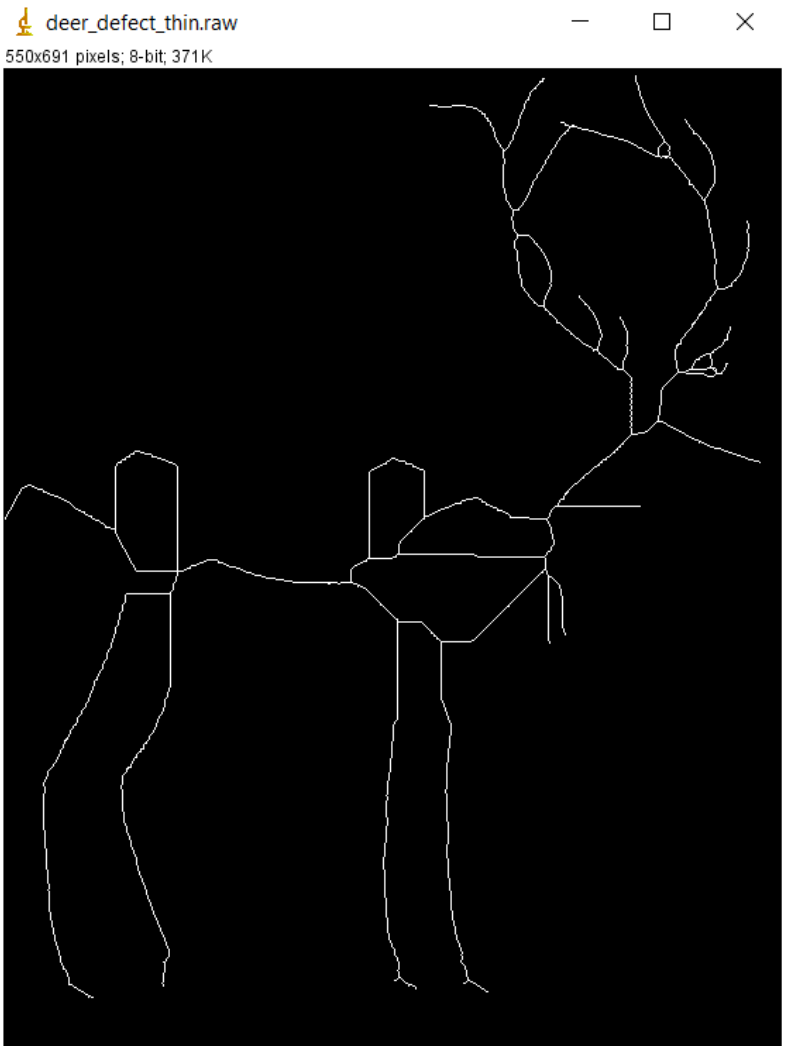


Figure 64: Thinning for defect image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

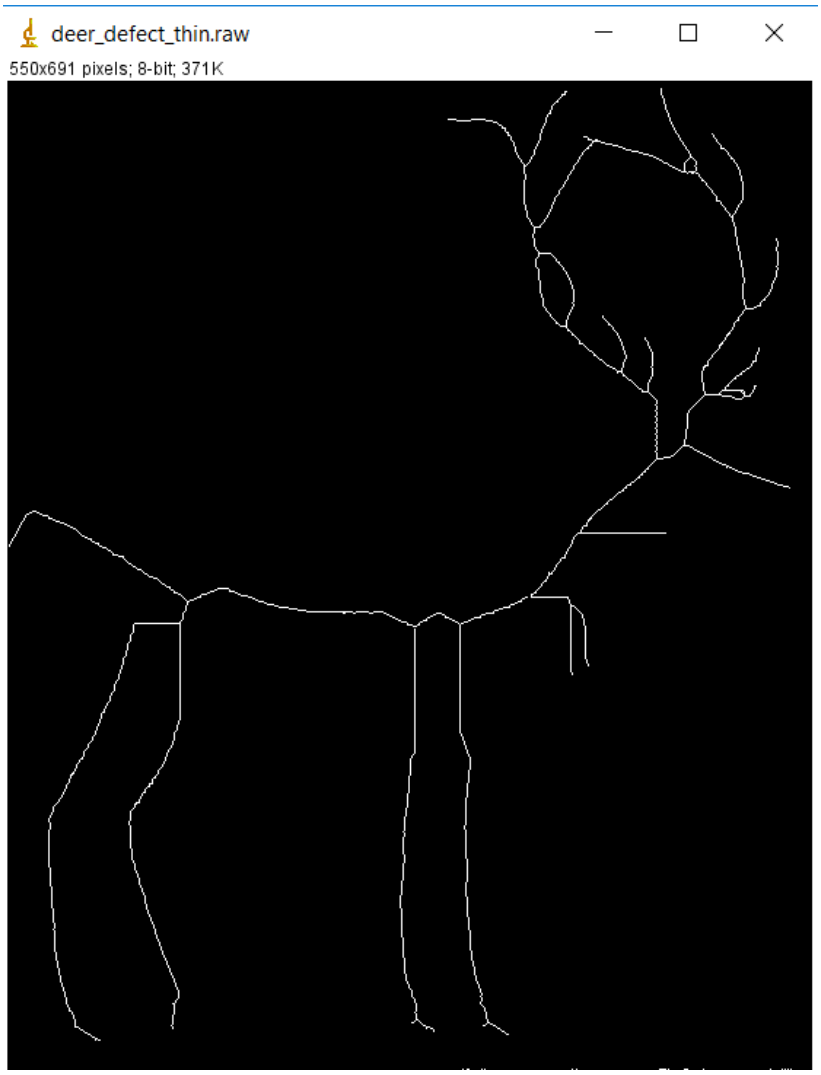


Figure 65: Thinning for defectless deer

(c) Object Analysis

(i) For the first part of counting the number of rice grains, I haven't used inbuilt functions and used my own implementations for finding the number of rice grains.

- The original rice grain image was converted to grayscale and dilated for four iterations and eroded for two iterations and then binarized for threshold >=128 and <=25 and then shrinking was applied.
- The intermediate binary output is as shown in figure 67 and the shrinked output is shown in figure 68. When printed for the number of white dots in the output image, the result was 55.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

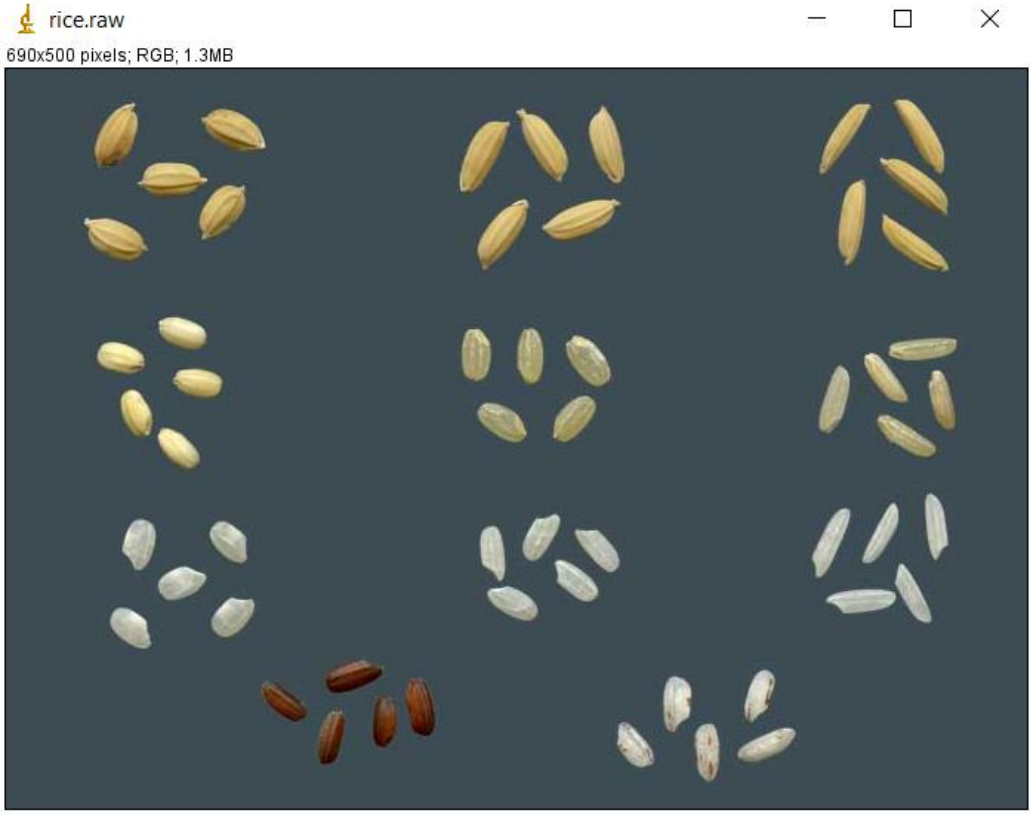


Figure 66: Original rice.raw image

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

Figure 67: Binary output for rice.raw after closing operation and thresholding

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

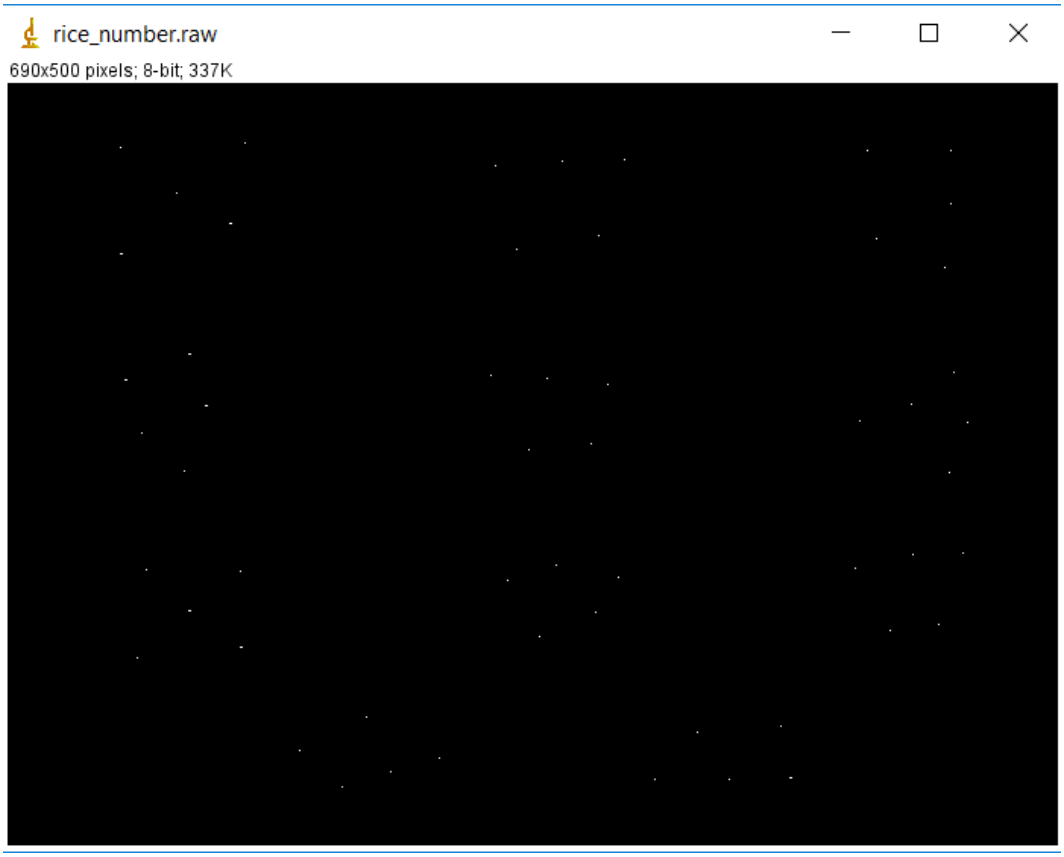


Figure 68: Shrink image for rice.raw

(ii) For the second part, I have used inbuilt matlab functions

- The rice.raw image was converted to gray and edges were extracted using canny function.
- Further, dilation was performed with structure element 'cross' and then the image was filled inside the edges.
- This image was eroded and regionprops was used to find the area and centroid of the rice images. Depending on the centroids, the whole image was divided into 11 clusters.
- Max area of rice grains in each of the clusters was found and compared to rank the sizes.
- The output of each step is shown.

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

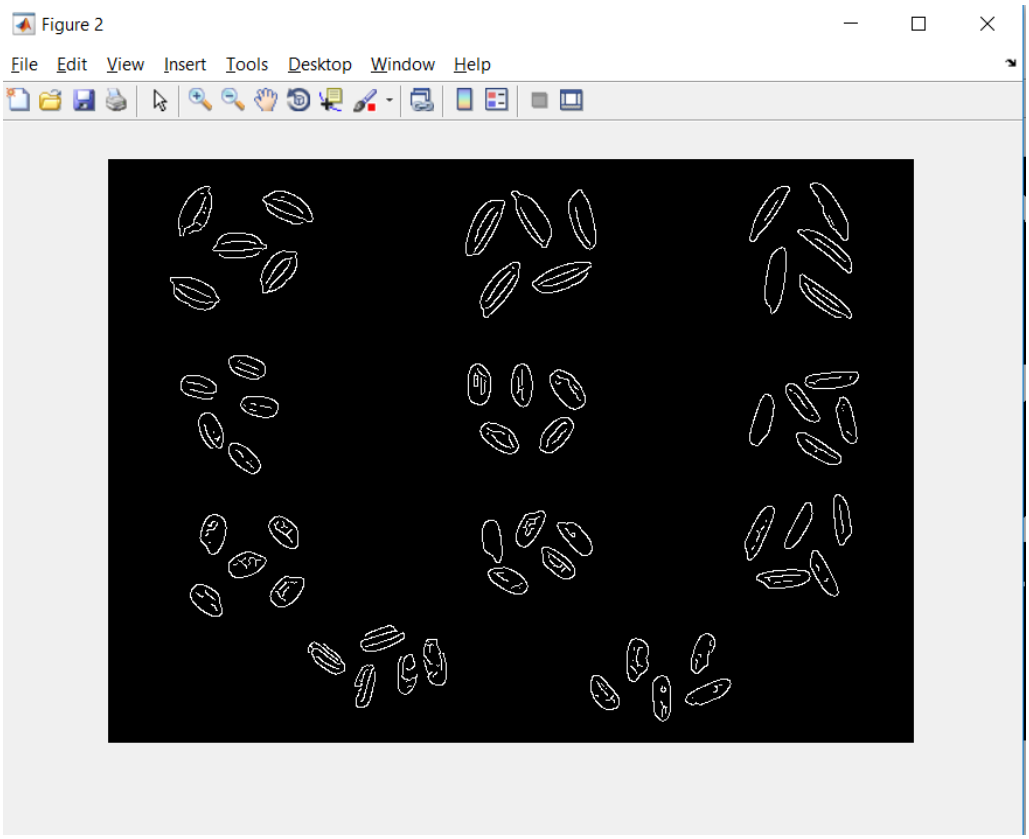


Figure 69: Output after the edge detection

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

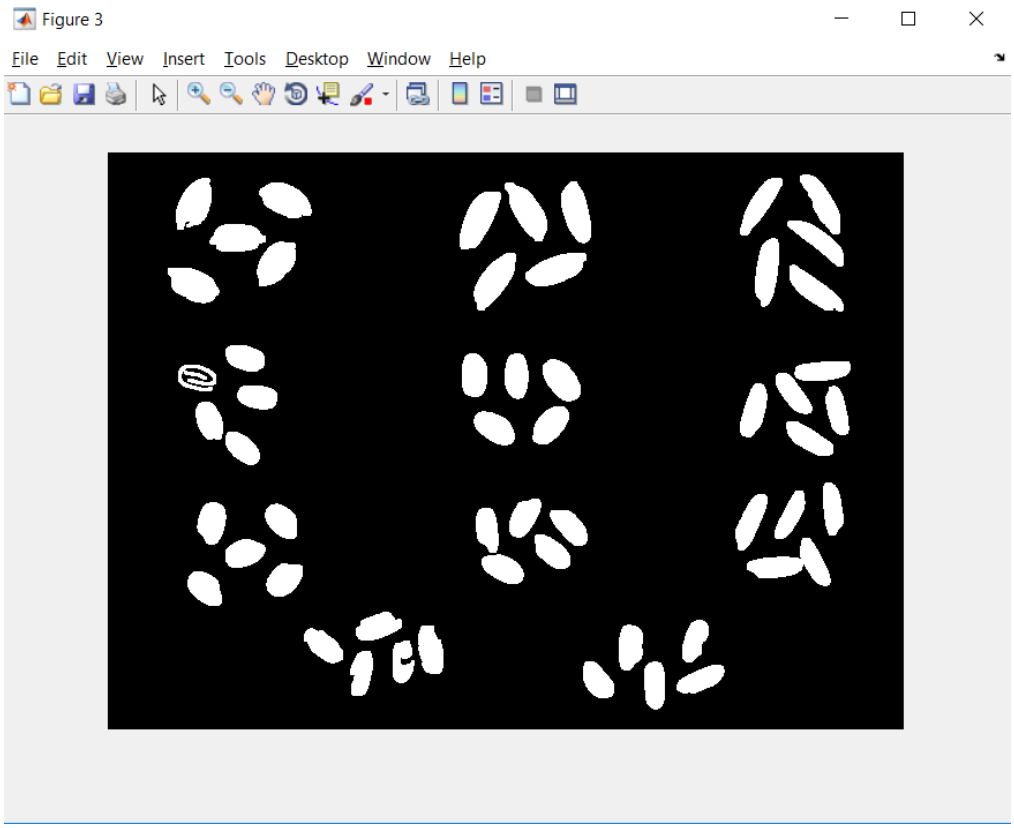


Figure 70: Output after filling the edges

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

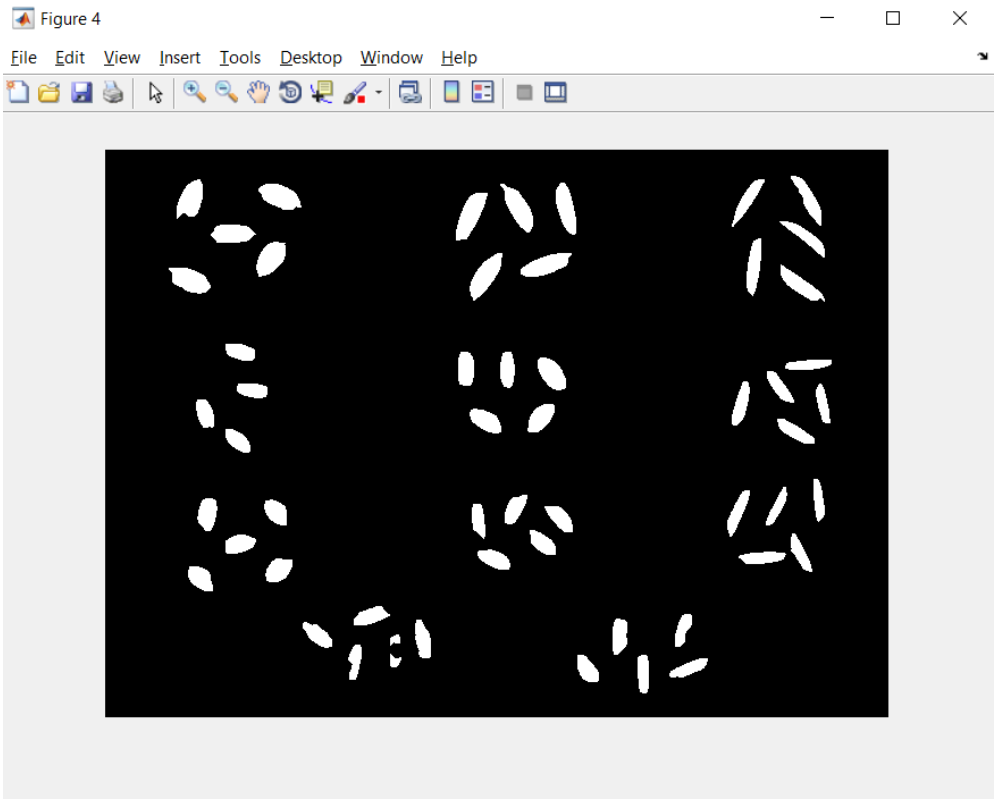


Figure 71: Output after erosion

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

```
cluster1_size =

   588

>> cluster2_size

cluster2_size =

   641

>> cluster3_size

cluster3_size =

   482

>> cluster4_size

cluster4_size =

   312

>> cluster5_size

cluster5_size =

   490
```

```
cluster6_size =

   365

>> cluster7_size

cluster7_size =

   387

>> cluster8_size

cluster8_size =

   351

>> cluster9_size

cluster9_size =

   361

>> cluster10_size

cluster10_size =

   361
```

```
>> cluster11_size

cluster11_size =

   352
```

- Based on the cluster sizes, cluster2 > cluster1 > cluster5 > cluster3 > cluster 7 > cluster 6 > cluster 9 > Cluster 10 > cluster 11 > cluster8

Akash Mohan Das
USC ID: 7247503828
USC email: amohanda@usc.edu
Submission date: 03/03/2019

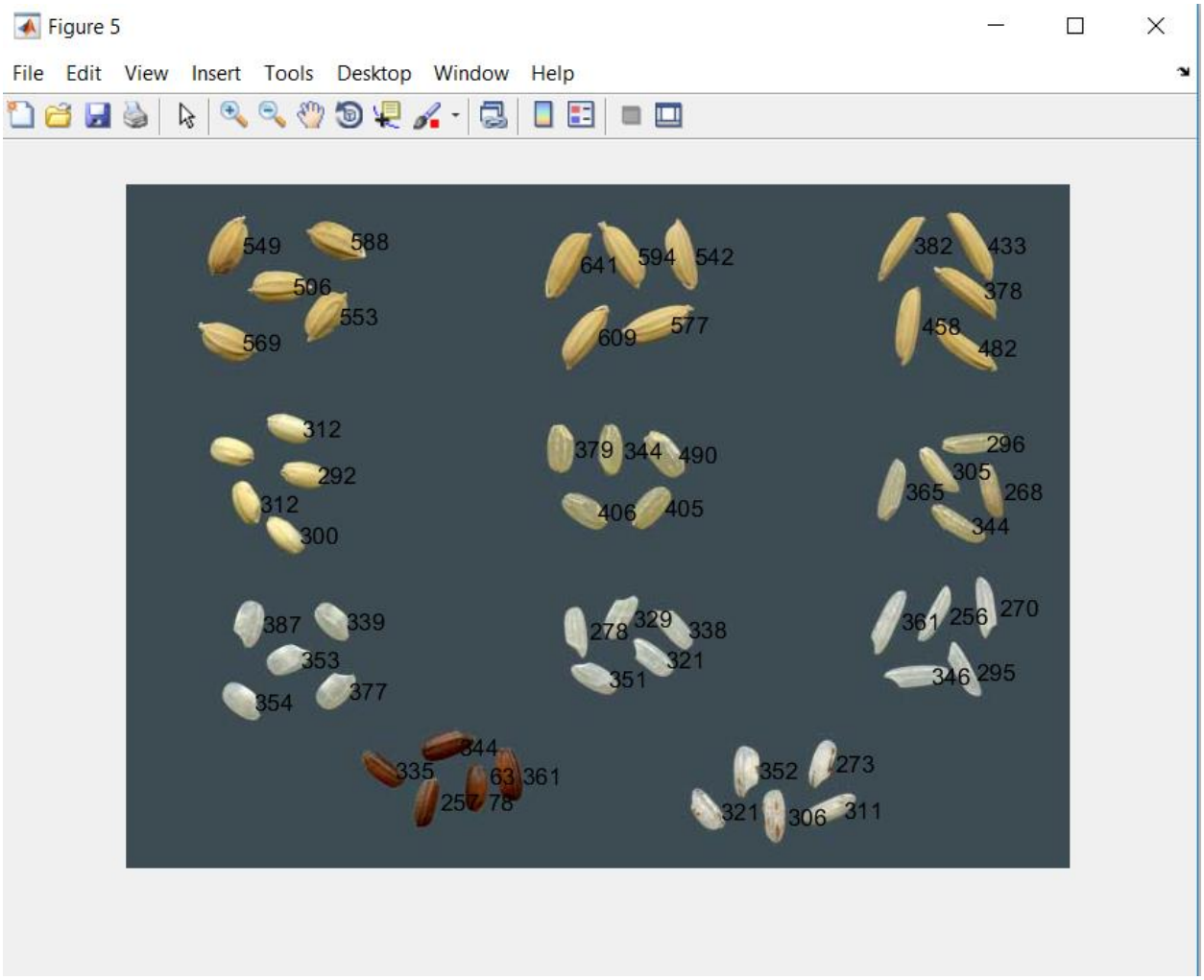


Figure 72: Output for the area of rice grains using regionprops