EE 569 HOMEWORK III

Royston Marian Mascarenhas

PROBLEM 1

Geometric Modification

(a)

Geometric Transformation

1. **Abstract and Motivation:**
   Geometric transformation of images is a necessary cog in the wheel of the modern image processing scenario. It is used in image filters in social media apps, missing image matching, in the advertising industry and much more. In this section, we explore the three core operations of geometric transformation: translation, rotation and scaling. For this, we use cartesian coordinate conversion and use coordinate mapping functions/ matrices and image reproduction techniques for missing pixels to manipulate images as per need.

2. **Approach and Procedures:**

   Geometric transformation is carried out with three core operations:
   1. Translation
   2. Rotation
   3. Scaling

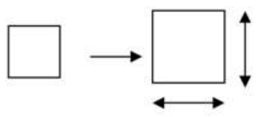
   Bilinear interpolation is used to improve the quality of the output image.

   In this case, the following relations were used in the above processes:

   1. **Translation**:
      Translated coordinates:
      Cartesian Co-ordinates (x,y)
      x = j + 0.5
      y = var - i - 0.5
      where var is the height of the image and (i,j) are image co-ordinates

   2. **Scaling**

- Scaling

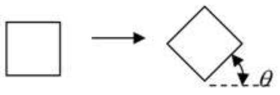


$$x = s_x u$$
$$y = s_y v$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

3. Rotation:

- Scaling
- Rotation



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

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

***Bilinear Interpolation**:*

ImageF3n[i,j] = (1-ha)*((1-hb)*ImageP3[uf,vf] + hb*ImageP3[uf,vc]) + ha*((1-hb)*ImageP3[uc,vf] + hb*ImageP3[uc,vc])

via reverse mapping where ImageF3n is the original image which is transformed back and the ImageP3n is the image which is disoriented. a and b are distances from the nearest pixels and uf,vf,uc,vc are the farthest and nearest coordinates used for reverse mapping.

Reverse Address Mapping:
Reverse address mapping is used to rework our way through the disoriented image to the original image. Co-ordinates are mapped to co-ordinates and then the pixel intensities are transferred.

Algorithm:
1. The image coordinates are converted to cartesian cooridinates based on every iteration through rows and columns.
2. Meanwhile the corners are detected (piece image), the height, width, center, and angle of rotation is calculated. Additionally, translation and scaling factors are also computed.

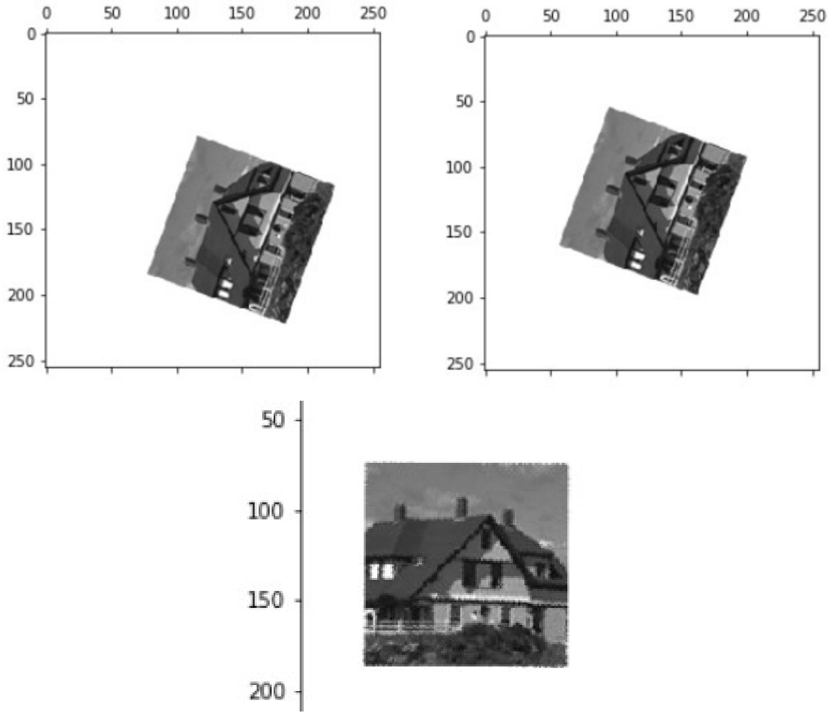
3. Then, translation is applied to take the piece to the origin
4. The image is scaled
5. The image is rotated
6. It is translated back
7. Use reverse mapping to transfer pixel intensities
8. The top left corner of the holes in the original image are detected and the piece image is filled back into the original image.

The inverse functions are taken so the coordinates for the original translation is not the center of the scaled up image.

It is applied in the following order:

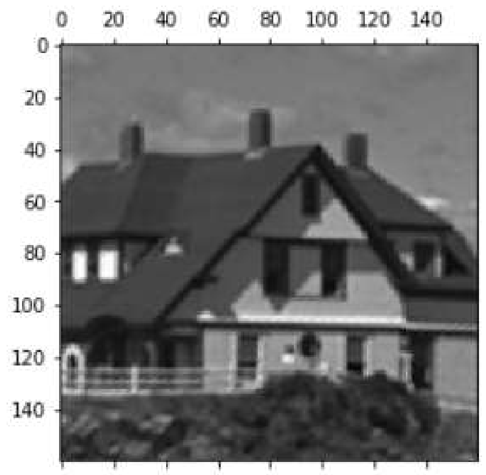
[u v 1] = (Translation)inverse * Rotation (inverse) * Scaling (inverse)*(Translation)inverse * [x y 1]

3. Results:

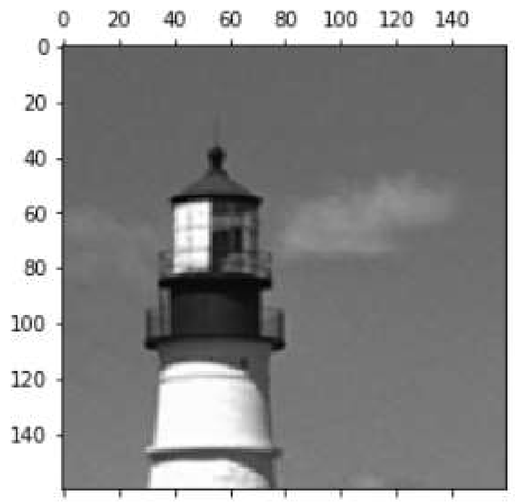


The first image piece after translation to the center and rotation

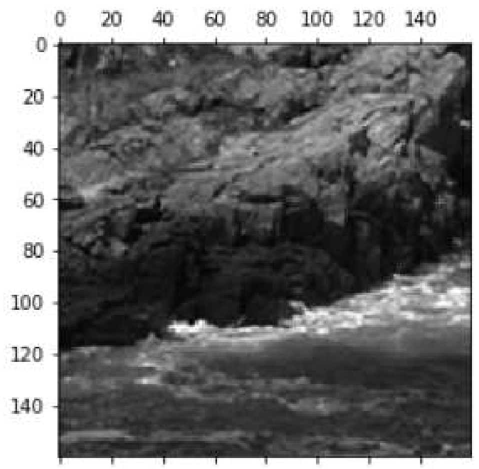
```
Out[358]: <matplotlib.image.AxesImage at 0x2818e6fd4e0>
```



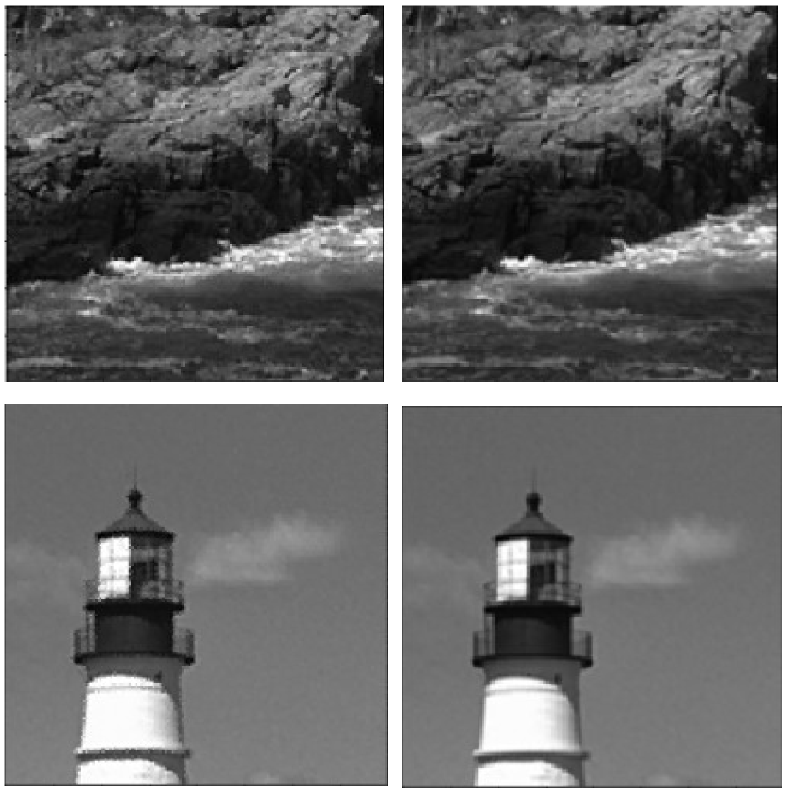
```
Out[377]: <matplotlib.image.AxesImage at 0x2818ea3d438>
```

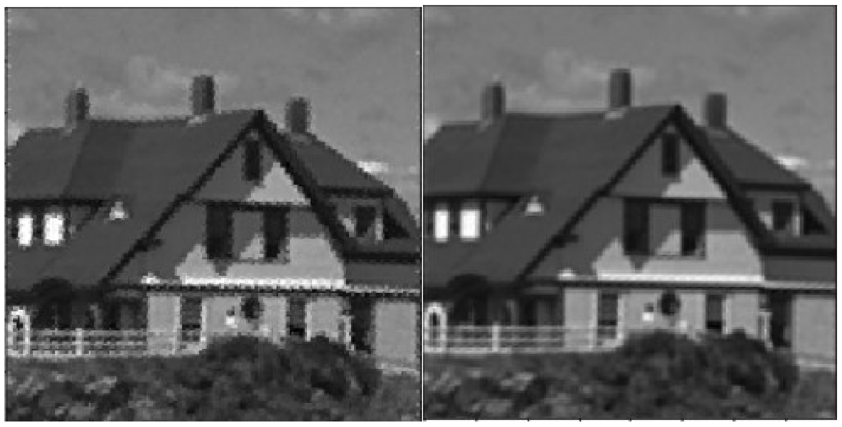


```
Out[314]: <matplotlib.image.AxesImage at 0x2818c5b5c88>
```



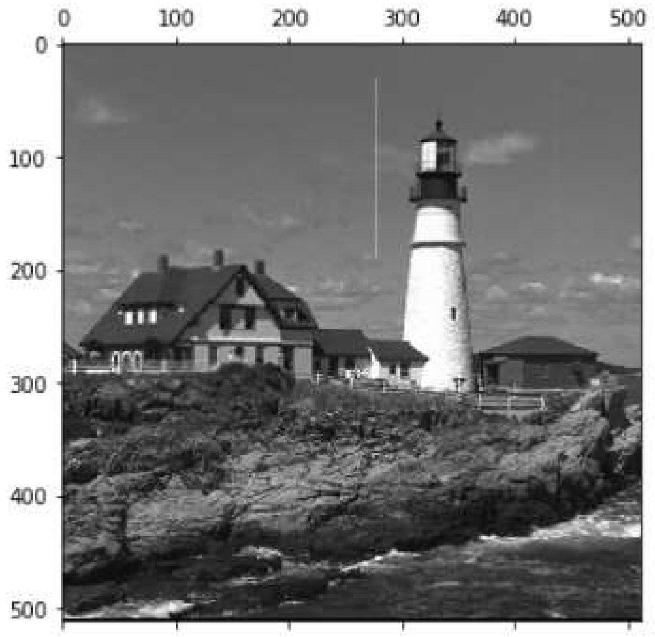
The piece images after rotation, translation and scaling operations





Before and after bilinear interpolation

```
Out[378]: <matplotlib.image.AxesImage at 0x2818e7909b0>
```



Filling the pieces into the main image.

4. Discussion

	Size	Height (Cartesian)	Width (Cartesian)	Theta (in rad) and center in cartesian	Scaling factor [sx,sy]
Piece 1 (house)	256 x 256	112.60550608207397	111.32834320154055	−π/2 − (−0.3472407828961109) and (150.0, 104.0)	[1.4358896666507757, 1.4521901655838703]
Piece 2 (tower)	256 x 256	144.69277798148738	144.04513181638595	−π/2 − 0.7175413405411444 and (117.0, 145.5)	[1.1057911958845044, 1.110762980896513]
Piece 3(river)	256 x 256	214.21951358361358	214.21951358361358	π/2 -1.3972057931300859	[0.7468974106205747, 0.7468974106205747]

Bias used:

1. Piece 1: [sx,sy] + 0.015
2. Piece 2: theta + 0.02 rad
3. Piece 3: no bias

Co ordinates of corners of holes in the main image: (top left pixel for each hole)

```
house : [157,62]

tower: [31,278]

river: [328,326]
```

Co-ordinates of corners of the pieces images (detected by program):

Piece 1: (Image Coordinates)

```
[ 80 115]
[223 183]
[185  78]
[118 221]

In Cartesian:
[115.5 175.5]
[183.5  32.5]
[78.5 70.5]
[221.5 137.5]
```

Piece 2: (Image Coordinates)

```
[102  15]
[116 218]
[212 108]
[  8 125]
```

In Cartesian:

```
[ 15.5 153.5]
[218.5 139.5]
[108.5  43.5]
[125.5 247.5]
```

Piece 3: (Image Coordinates)

```
[ 40 251]
[214   3]
[ 3 40]
[251 214]

Cartesian:
[251.5 215.5]
[ 3.5 41.5]
[ 40.5 252.5]
[214.5   4.5]
```

Additional Points:

- Bias is used because the calculation might rotate, translate or scale the images by little less or more amount since cartesian estimation is automated. The bias values are very small.
- The lines in the image while fitting it back into the main image can be overcome by either using bilinear interpolation on the boundaries of the holes or iterating throughout the boundaries using a median filter.

(b)

Spatial Warping

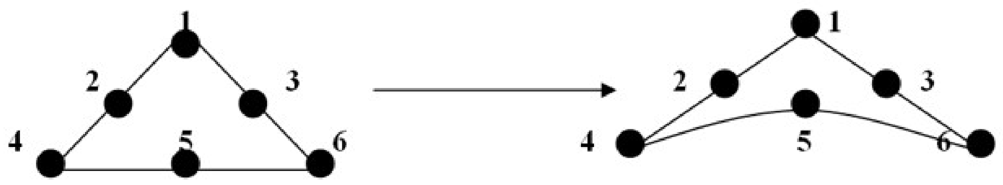
1. **Abstract and Motivation:**
Spatial warping is an image processing technique which is used to contort images to produce fantastical effects. Warping introduces aesthetic value to the image and is used as a tool of amusement as well by companies like Snap Inc for their image filters. In this section, we transform coordinates in the input to the coordinates of the required warping boundary and then apply the warping coefficients to all pixels which are supposed to be warped.

2. **Approach and Procedures:**

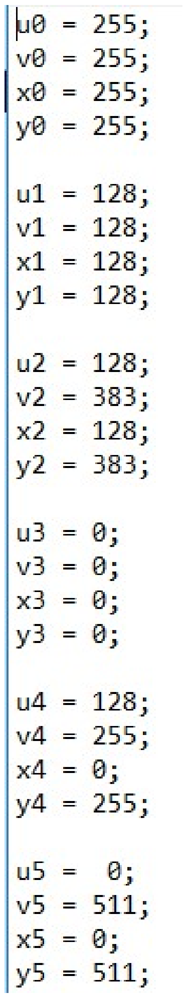
Algorithm:
1. Calculate the a and b coefficients by inputting the reference coordinates
2. Get the warped coordinates by using the coefficients and the input
3. Perform transfer of image intensities to corresponding points

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

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



Reference points for upper triangle:



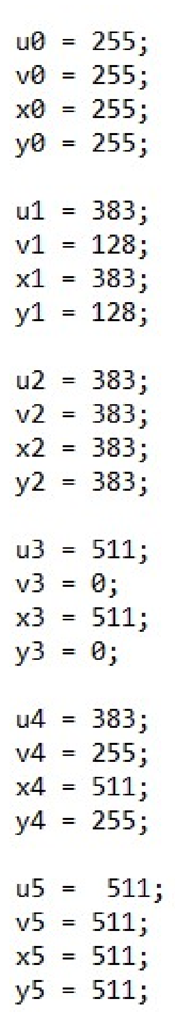
```
u0 = 255;
v0 = 255;
x0 = 255;
y0 = 255;

u1 = 128;
v1 = 128;
x1 = 128;
y1 = 128;

u2 = 128;
v2 = 383;
x2 = 128;
y2 = 383;

u3 = 0;
v3 = 0;
x3 = 0;
y3 = 0;

u4 = 128;
v4 = 255;
x4 = 0;
y4 = 255;

u5 =  0;
v5 = 511;
x5 = 0;
y5 = 511;
```

where U, V are the warped co- ordinates

Coefficients for upper triangle:

```
array([[ 0.    , -0.002,  1.002,  0.002,  0.    , -0.002],
[ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ]])
```

Reference points for lower triangle:



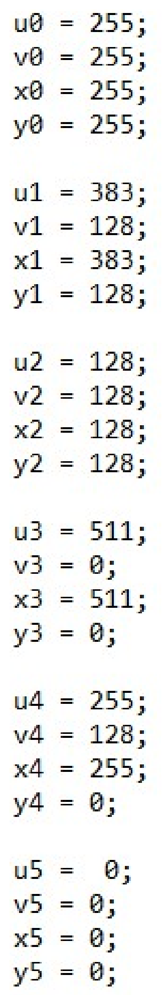
```
u0 = 255;
v0 = 255;
x0 = 255;
y0 = 255;

u1 = 383;
v1 = 128;
x1 = 383;
y1 = 128;

u2 = 383;
v2 = 383;
x2 = 383;
y2 = 383;

u3 = 511;
v3 = 0;
x3 = 511;
y3 = 0;

u4 = 383;
v4 = 255;
x4 = 511;
y4 = 255;

u5 =  511;
v5 = 511;
x5 = 511;
y5 = 511;
```

where U and V are the input to the desired warped coordinates

Coefficients for lower triangle:

```
array([[ 0.000e+00,  2.002e+00, -1.002e+00, -2.000e-03,  0.000e+
00,
  2.000e-03],
[ 0.000e+00,  0.000e+00,  1.000e+00,  0.000e+00,  0.000e+00,
  0.000e+00]])
```

Reference points for left triangle:



```
u0 = 255;
v0 = 255;
x0 = 255;
y0 = 255;

u1 = 383;
v1 = 128;
x1 = 383;
y1 = 128;

u2 = 128;
v2 = 128;
x2 = 128;
y2 = 128;

u3 = 511;
v3 = 0;
x3 = 511;
y3 = 0;

u4 = 255;
v4 = 128;
x4 = 255;
y4 = 0;

u5 =  0;
v5 = 0;
x5 = 0;
y5 = 0;
```

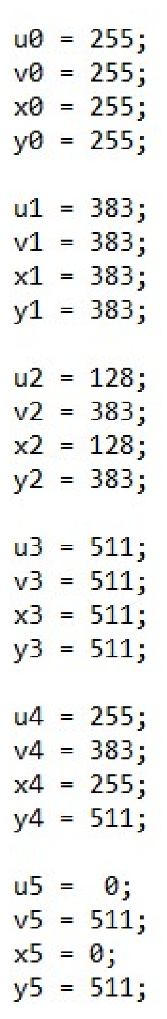
U, V are the desired warped coordinates

Coefficients for left triangle:

```
 array([[ 0.   ,  1.   ,  0.   ,  0.   ,  0.   ,  0.   ],
 [ 0.   ,  1.002, -0.002, -0.002,  0.   ,  0.002]])
```

Reference points for right triangle:

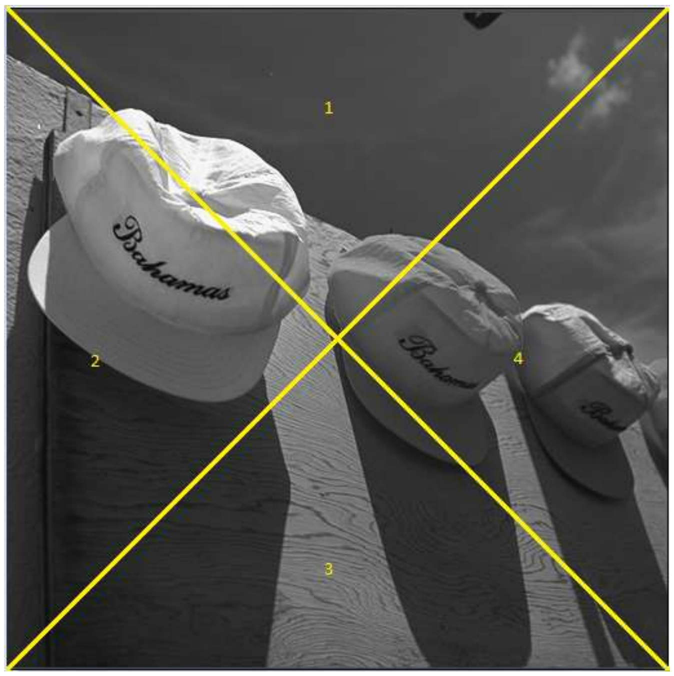
```
 array([[ 0.000e+00,  1.000e+00,  0.000e+00,  0.000e+00,  0.000e+
00,
   0.000e+00],
 [ 0.000e+00, -1.002e+00,  2.002e+00,  2.000e-03,  0.000e+00,
  -2.000e-03]])
```



```
u0 = 255;
v0 = 255;
x0 = 255;
y0 = 255;

u1 = 383;
v1 = 383;
x1 = 383;
y1 = 383;

u2 = 128;
v2 = 383;
x2 = 128;
y2 = 383;

u3 = 511;
v3 = 511;
x3 = 511;
y3 = 511;

u4 = 255;
v4 = 383;
x4 = 255;
y4 = 511;

u5 =  0;
v5 = 511;
x5 = 0;
y5 = 511;
```

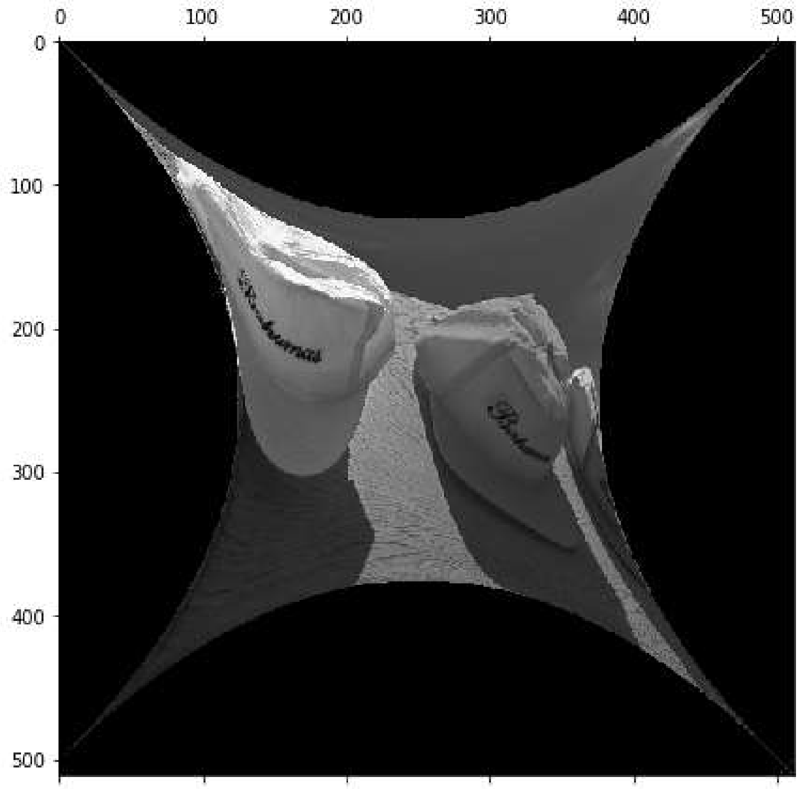
Coefficients for right triangle:

```
array([[ 0.000e+00,  1.000e+00,  0.000e+00,  0.000e+00,  0.000e+
00,
  0.000e+00],
[ 0.000e+00, -1.002e+00,  2.002e+00,  2.000e-03,  0.000e+00,
 -2.000e-03]])
```



3. Results:

```
Out[6]: <matplotlib.image.AxesImage at 0x205cfd00080>
```



4. **Discussion**
    - Iteration of the triangular coordinates is one possible cause of missing pixels while warping
    - That is fixed by either rounding off the final coordinates or redesigning the iteration algorithm
    - The warped image is a non linear transformation of the original image
    - Forward mapping is used to get warped image
    - Reverse mapping is used only if the image has to be unwarped.

(c)

Lens Distortion Correction

1. **Abstract and Motivation:**
   The curvature of camera lens generates radial distortion which distorts the pixels in a way that it seems curved. Here, we use the inverse function to
2. **Approach and Procedures:**

   The algorithm followed here is as follows:

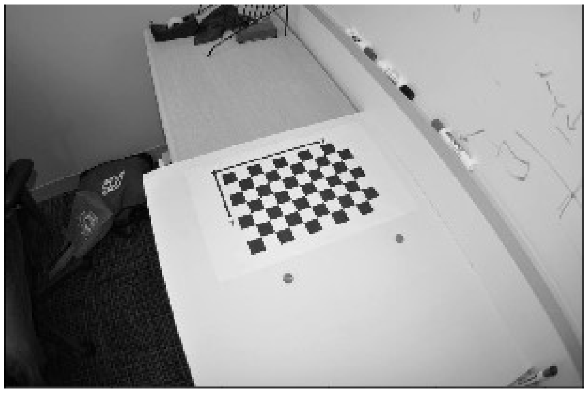
   1. Iterate through height and width number of pixels of the image
   2. Convert to camera co-ordinates using this formula

      camx = (i - center_x)/fx
      camy = (j - center_y)/fy

      where I and j are image co-ordinates and fx and fy are 600.

   3. Convert to distorted camera coordinates
   4. Reverse map back them back to original image whilst converting to image coordinates

5. **Results:**



Distorted Image

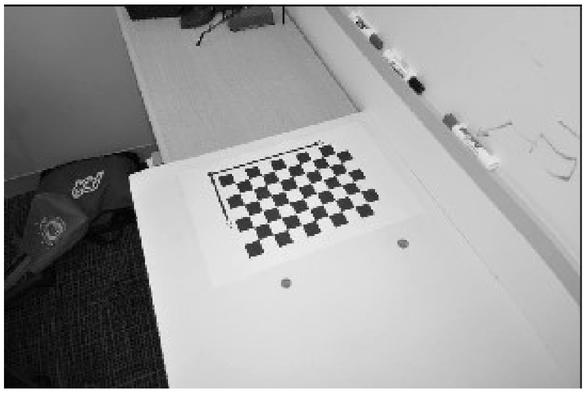
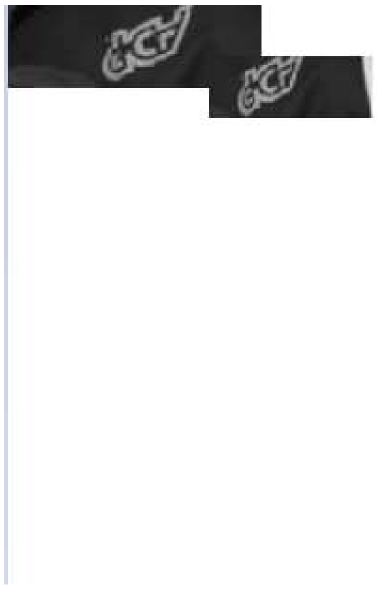
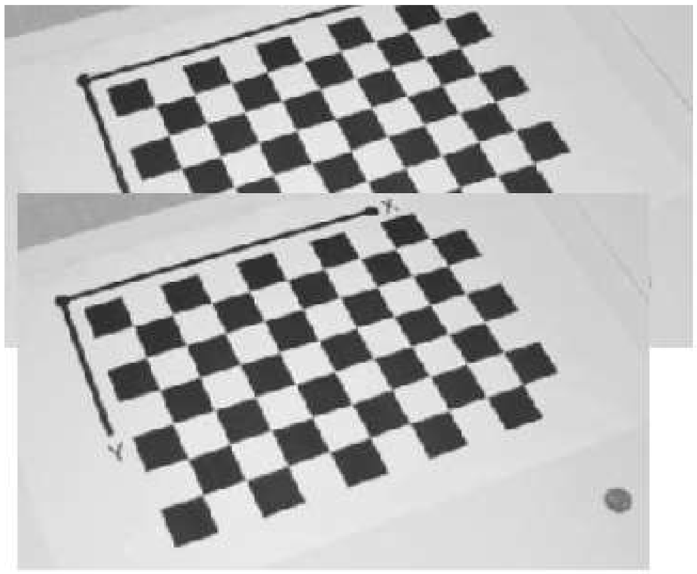


Image after reverse mapping





Aliasing effects observed (after v/s before)

4.Discussion:

- Reverse mapping was used to obtain the original image
- The non linearities in the equation were used as an inverse to obtain the original image
- **Aliasing effects** can be observed in the distorted chessboard coordinates
- These can be overcome by bilinear interpolation
- The image grows in size
- The array needs to be large enough to hold the enlarged image since it is radially undistorted

PROBLEM 2

Morphological Processing

(a)

Basic Morphological Processing Implementation

1. Abstract and Motivation:

Binary images are prone to defects caused by noise and texture in the non binarized image. Morphological processing uses a pivotal mask or structuring element and checks the similarity of the mask with the central pixel and its neighbourhood. This foundational approach is used with several masks and over several iterations often ending with a climactic logical operation combining various morphological results. Morphological processing, in this manner, helps remove the defects in the image and assists in image inspection

2. Approach and Procedures:

Shrinking:

Involves shrinking an object which is completely foreground to a single point if it does not have any holes or two a set of equidistant points otherwise.

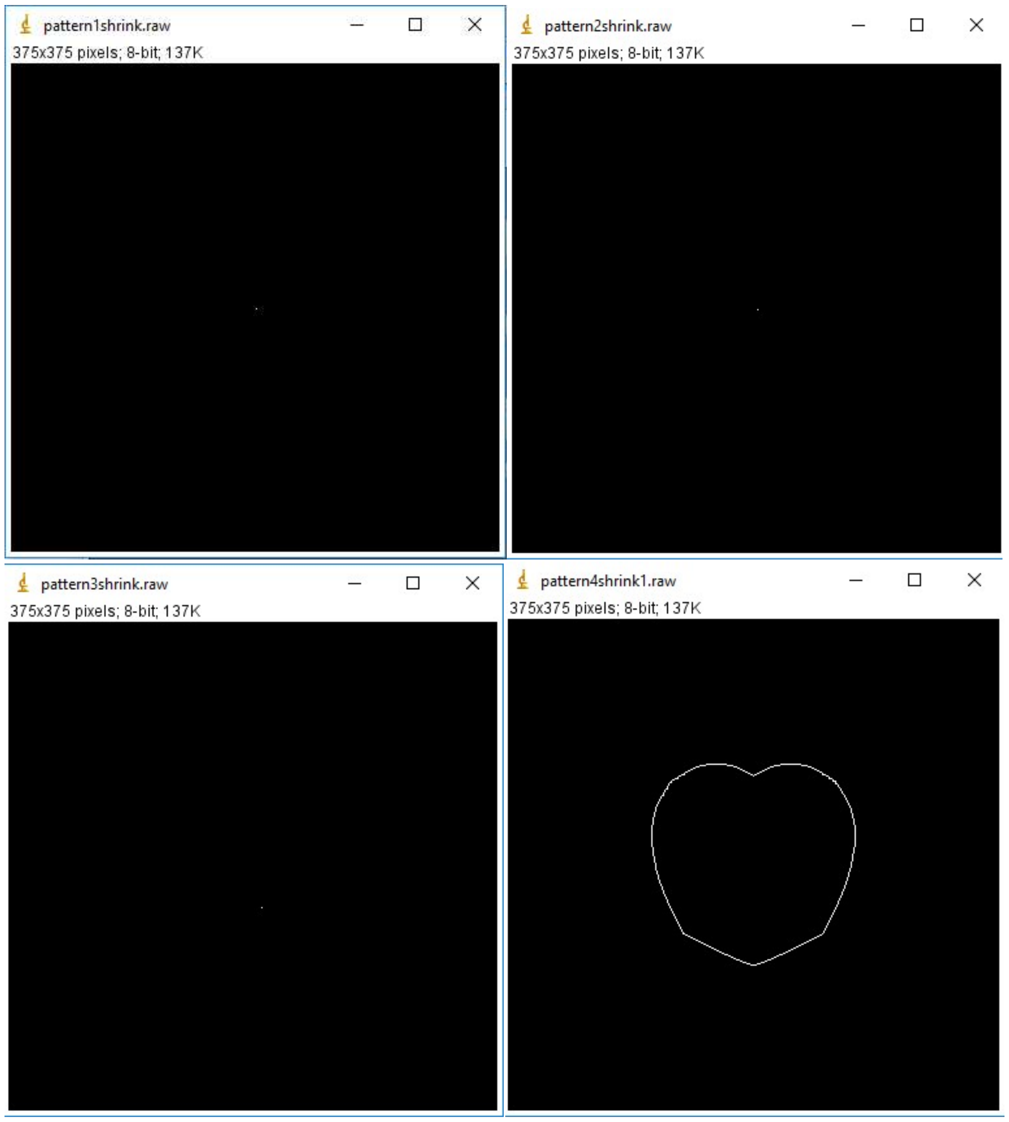
Thinning:

Thinning involves obtaining the points which are inside the object and equidistant from the boundaries.

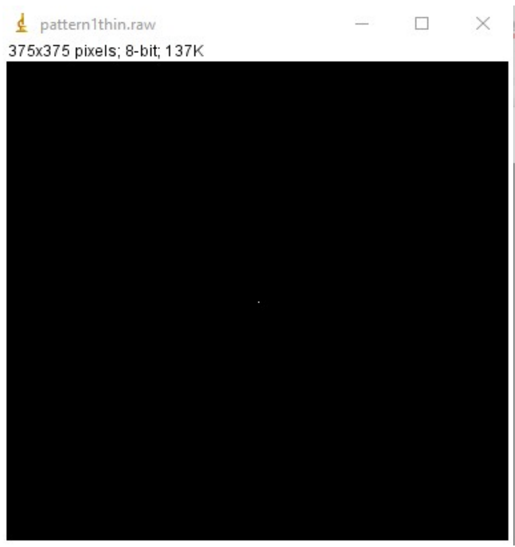
Skeletonization:

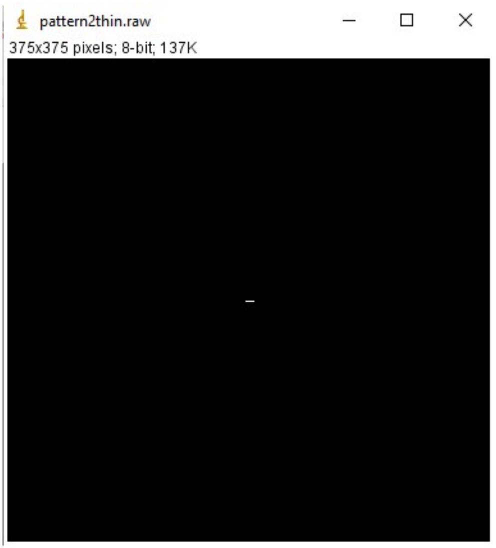
Skeletonization is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels. To see how this works, imagine that the foreground regions in the input binary image are made of some uniform slow-burning material. Light fires simultaneously at all points along the boundary of this region and watch the fire move into the interior. At points where the fire traveling from two different boundaries meets itself, the fire will extinguish itself and the points at which this happens form the so called `quench line'. This line is the skeleton. Under this definition it is clear that thinning produces a sort of skeleton.

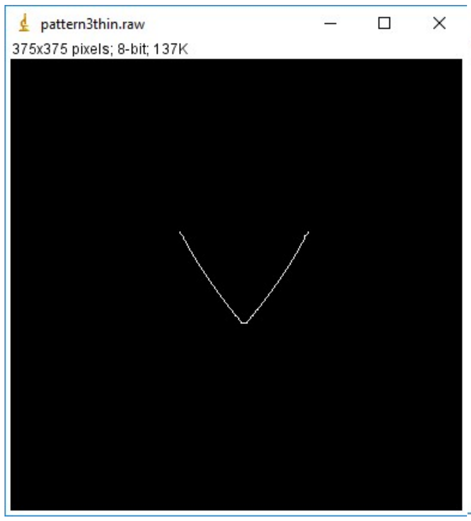
3. Results:

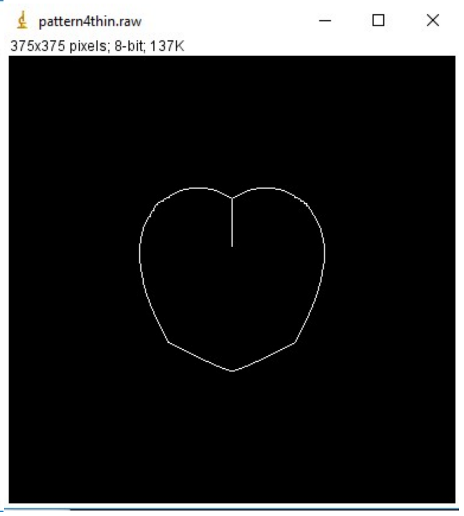

pattern1shrink.raw
375x375 pixels; 8-bit; 137K
pattern2shrink.raw
375x375 pixels; 8-bit; 137K
pattern3shrink.raw
375x375 pixels; 8-bit; 137K
pattern4shrink1.raw
375x375 pixels; 8-bit; 137K

Shrinking results 1-4 (top facing left, top facing right, bottom facing left, bottom facing right)

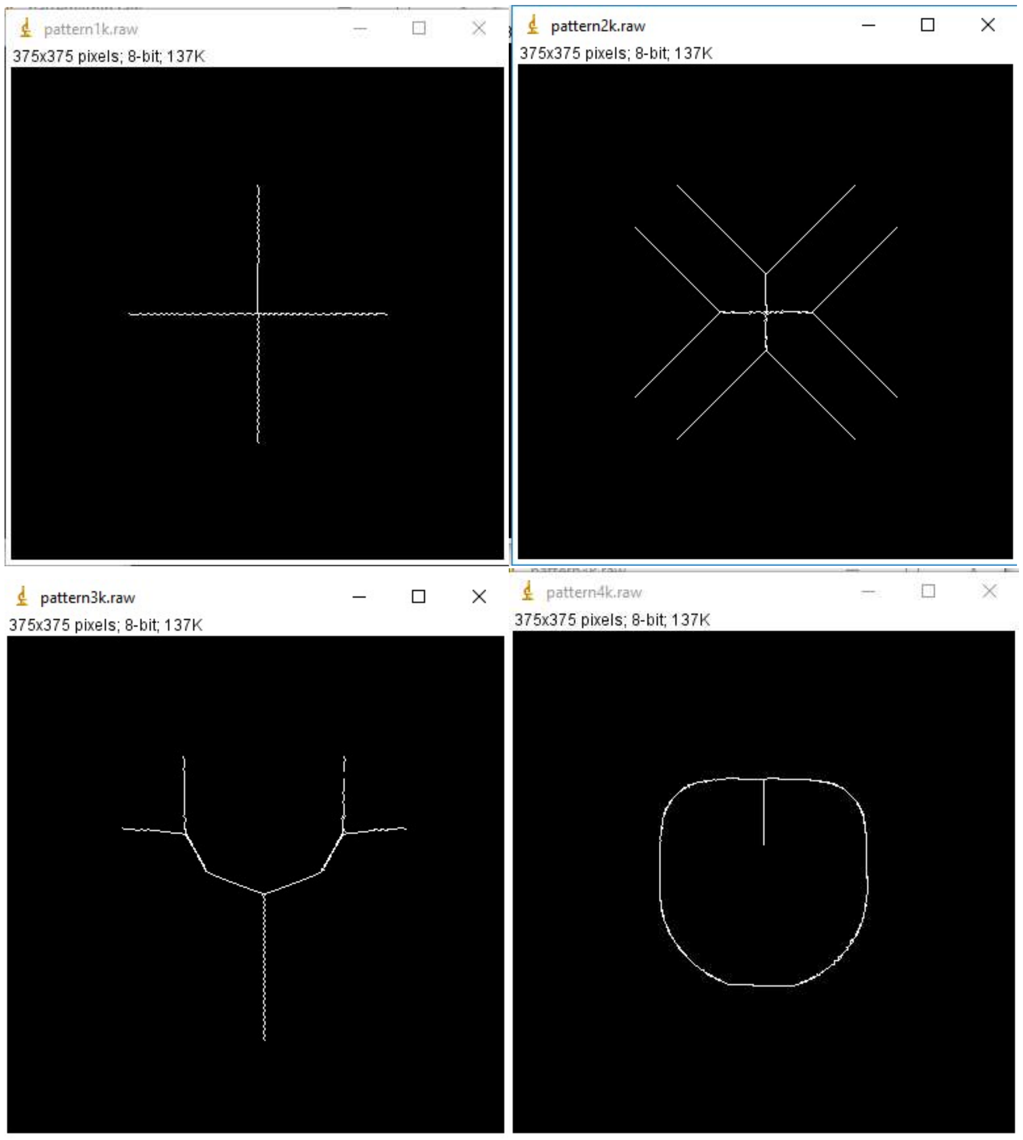




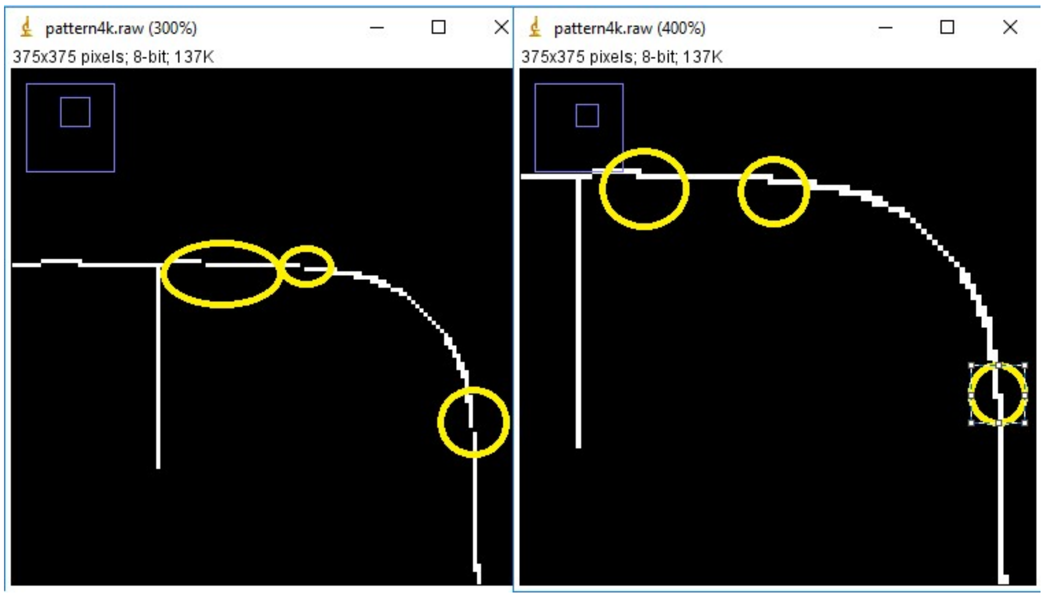




Thinning results 1-4 (top facing left, top facing right, bottom facing left, bottom facing right)


pattern1k.raw
375x375 pixels; 8-bit; 137K
pattern2k.raw
375x375 pixels; 8-bit; 137K
pattern3k.raw
375x375 pixels; 8-bit; 137K
pattern4k.raw
375x375 pixels; 8-bit; 137K

Skeletonization results 1-4 (top facing left, top facing right, bottom facing left, bottom facing right)



Pattern 4 segment before and after bridging in skeletonization

4. Discussion

Pattern 1 :

- The shrinking output of the circle is a dot since it is a complete object in the foreground without any holes
- The thinning output is also a dot since the center is the only point equidistant from all the boundaries
- The skeletonizing output is a cross since it represents the intersection of the diameters which are the backbone of the circle.

Pattern 2:

- The shrinking output of the curved edge rectangle is a dot since it is a whole object.
- The thinning output represents a thin line since those are the only points which are equidistant from the boundaries.
- The skeletonization output represents the intersection of the diagonals since it is the sole backbone on which the mass of the rectangle can be placed. The V shapes are due to the curved edges.

Pattern 3:

- The shrinking output is a misplaced dot which points to its center of mass.
- The thinning output is a V since as we traverse the curved edges from the bottom, it loses equidistance from the boundary.
- The skeletonization output represents V shape of the heart which hosts its foundation. (in a spatial perspective, it houses majority of its mass)

Pattern 4:

- The shrinking output
- The thinning output
- The skeletonization output represents the inner object which is a curved edge square with heart as background pixels. The line leads inward and stops when it meets the location of the background heart.

(b)

Defect Detection and Correction

1. **Abstract and Motivation:**
   As images in posters and advertisements are enlarged, the defects tend to become more obvious. This section involves the use of morphological masks like the hit and miss masks that help remove such defects.

2. **Approach and Procedures:**

   My approach had mainly 3 steps
   1. Morphological process to help determine whether the image has defects:
      - Iterate through the image
      - Identify stray pixels if there is a hit on the isolation mask
      - Highlight stray pixel from the background as shown in result

   2. Morphological hit and miss to remove stray pixels inside the body

   3. Morphological hit and miss with manual masks to remove stray pixels which are on a flat boundary, to improve visual opinion of the image.

   The masks I used are:

   1. Isolation mask:
      1 1 1
      1 0 1
      1 1 1

   2. Hit and miss masks (manual; for stage 2; 3x3 flattened into a 1D array):

   {1,1,0,1,0,0,1,1,0},

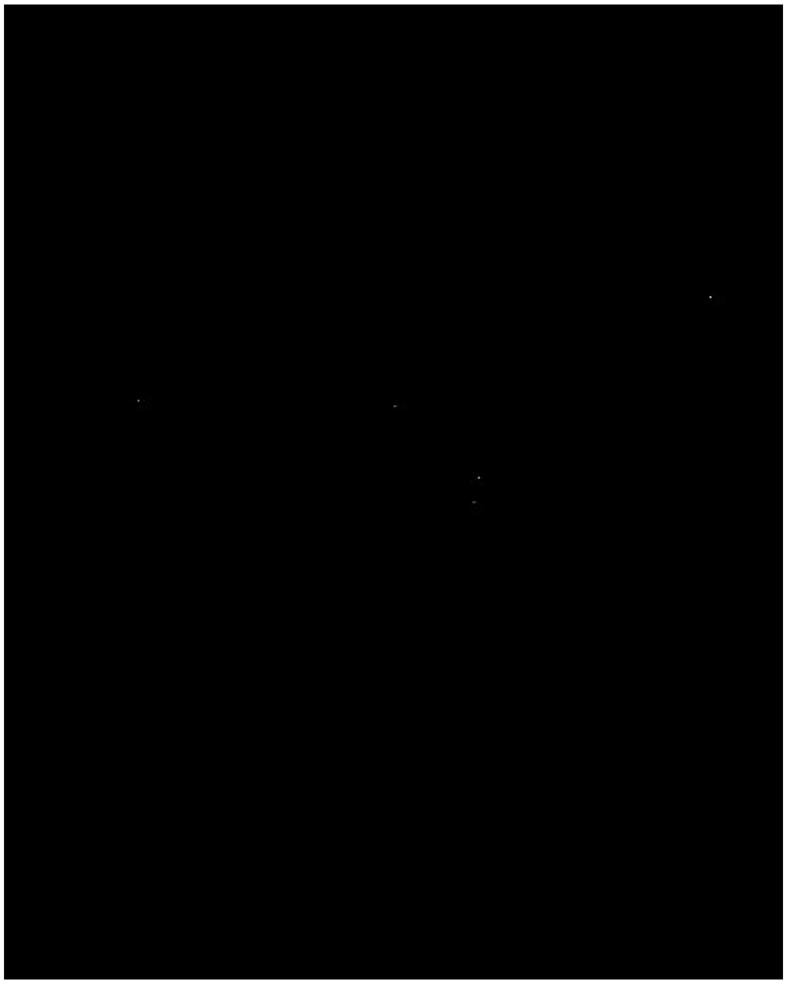
   {0,1,1,0,0,1,0,1,1},

   {0,0,0,1,0,1,1,1,1},

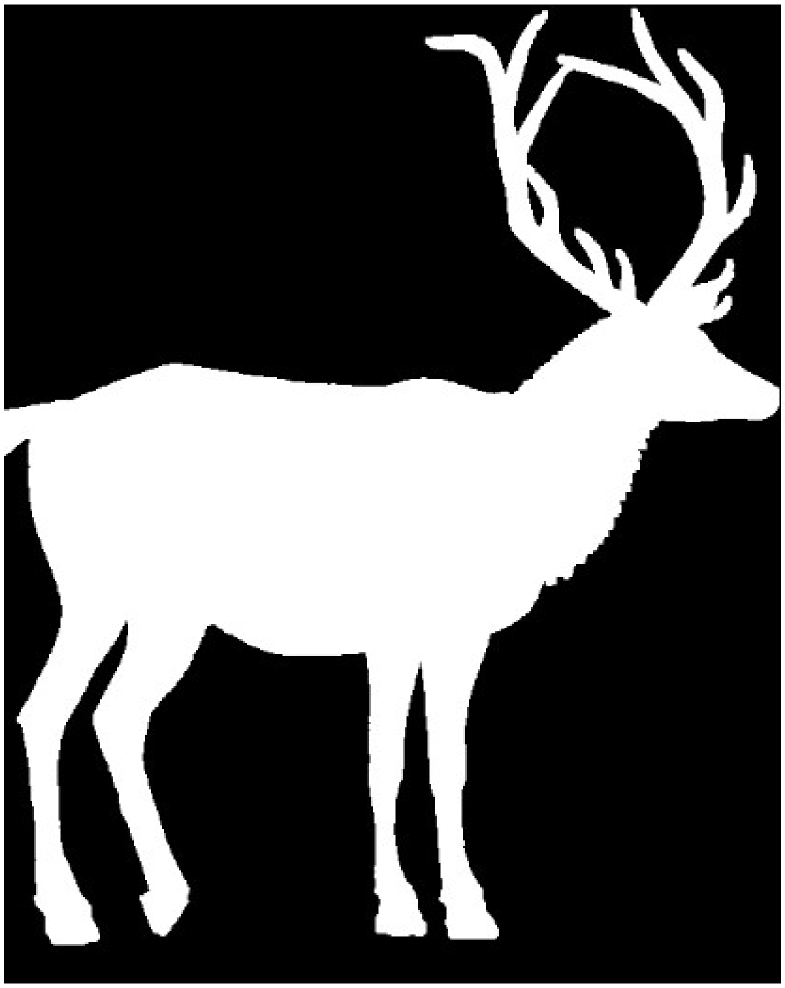
   {1,1,1,1,0,1,0,0,0}

3. **Results:**

   A. Preliminary Examination:

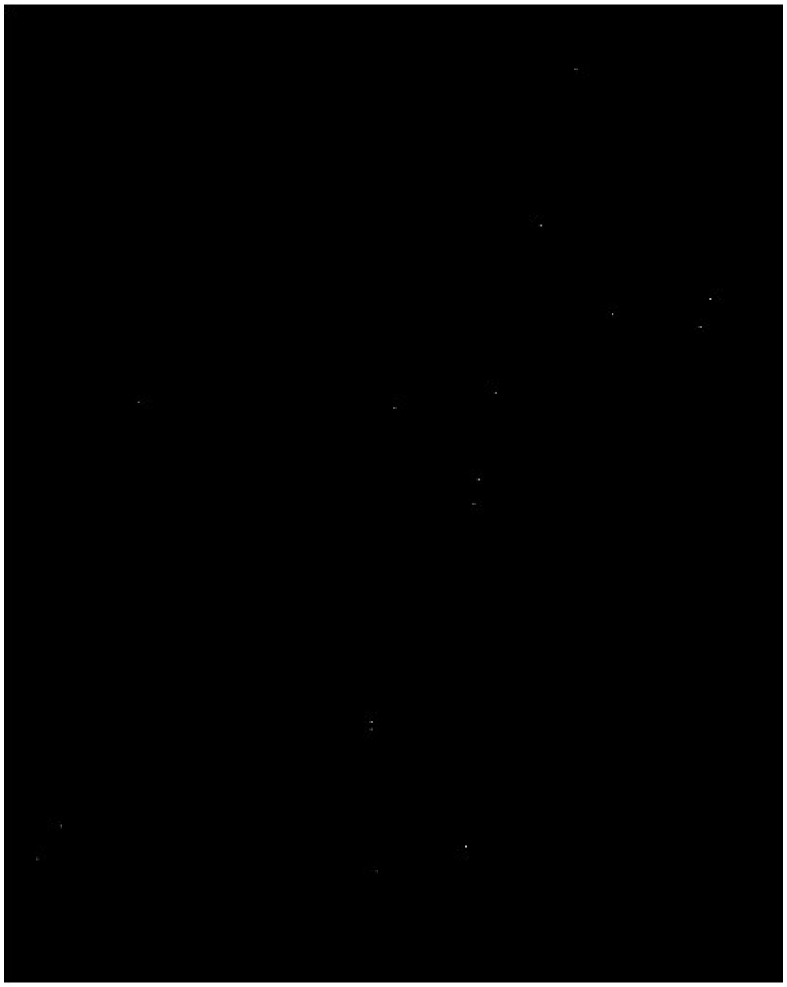


Morphological process to determine whether the image has defects. The defects have been highlighted from the background. The image has been inverted since white pixels on a black background are more noticeable visually. The white pixels are the locations of the defects. 5 such changes were made.

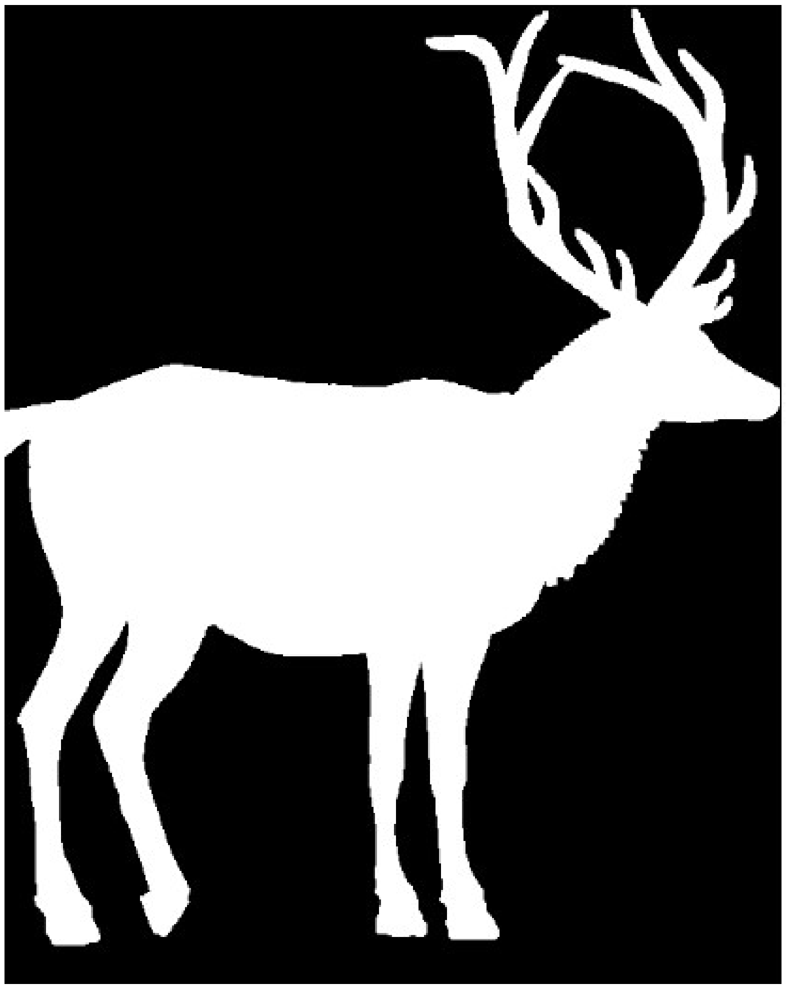


Deer Image after removing the defects inside the body

B. Further Improvement (to increase aesthetic appeal) :

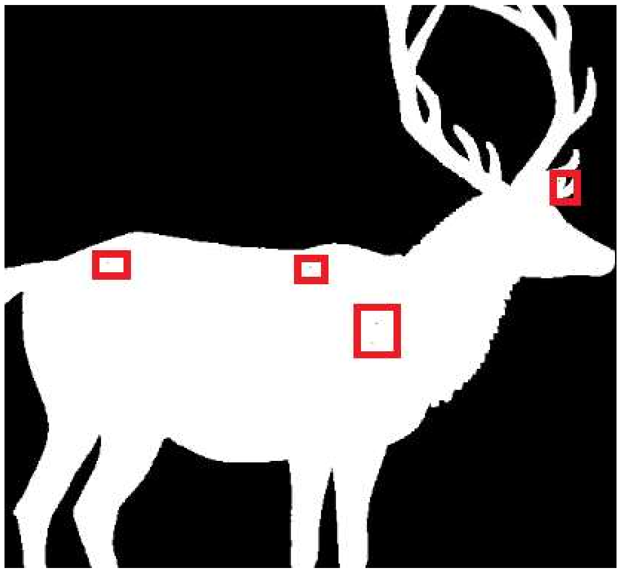


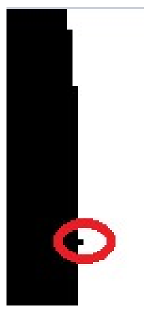
On further analysis, it was observed that in a few areas, the slope of the contour was flat but a pixel strayed inside the body of the image. The defects have been highlighted from the background (along with the defects mentioned in the preliminary examination). The image has been inverted since white pixels on a black background are more noticeable visually. The white pixels are the locations of the defects.



The deer image after removal of defects.

4. Discussion





- Hit and miss masks are a great way to ensure that smaller stray pixels can be eliminated
- The computational complexity for these masks is extremely low
- When the image is enlarged, the smaller pixels become more obvious, therefore, there is need to conduct a more detailed investigation into pixels that might affect the aesthetic appeal of the poster
- The preliminary examination involved getting rid of the obvious defects inside the main body as in the first figure
- The secondary inspection involved pixels that are stray but not more obvious
- Morphological processes like hit and miss can also be used to determine whether the image has defects as shown in the figure

(c)

Object Analysis

1. **Abstract and Motivation:**
   Object analysis is used for many application where automation is required. For example. deciding whether an X-ray indicates cancer or an eye image depicts glaucoma. This is done with the help of morphological processing where masks are run through the image to perform logical operations to extract meaning from the information in the image.

2. **Approach and Procedures:**

   In object analysis, I used several algorithms in order to preprocess the data. Thresholding the greyscale image alone did not work as the information of the brown grains at the bottom were getting lost.

   Therefore, this called for adaptive thresholding. I obtained a histogram of the greyscale image and observed that the background had a dominant number of pixels. Therefore, allowing for some variance, I tried thresholding just the grayscale image. It still turned out that information was getting lost.

Furthermore, I observed that the red channel had a good amount of information about just the brown rice grains. Still, information loss persisted when I tried to use just that threshold. In addition, I observed that the blue channel had even more information about the brown grains than the red one.

Thus, I followed the following approaches to preprocess the rice grain image. **(Extra Credit attempted)**

1. Thresholding: Thresholding was carried out across two channels: The grey channel (65 and 77 (inclusive) for 0) and the blue channel (below 30 for 1). The observed output still had a good amount of stray pixels outside boundaries since the thresholds were ORed to retain information in the brown pixels.
2. Subtle Erosion: A mask containing a clump of white pixels was iterated through the image to retain white clumps and eliminate isolated white patterns of small size.
3. Medium Erosion: Histogram Manipulation: A 5x5 window was sent across the image to further bolster the majority of white pixels in a given window by favouring the white binary value with a probability of around 0.7. (If 70 percent is white, then make the entire window white). Note that the boundary generates stray pixels due to boundary padding. They are not considered as they don't even come close to the original objects in the image and are unnecessary.
4. Hit and Miss Masks: A few pixels were still seen loitering in isolation. To eliminate them, I sent four masks across the image since the shapes were few and recurring. These four masks are:

```
0 0 0 0 0
0 0 0 0 0
0 1 1 0 0
0 0 1 0 0
0 0 0 0 0

0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 1 0 0
0 0 0 0 0

0 0 0 0 0
0 0 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0

1 1 1 1 1
1 1 1 1 1
```

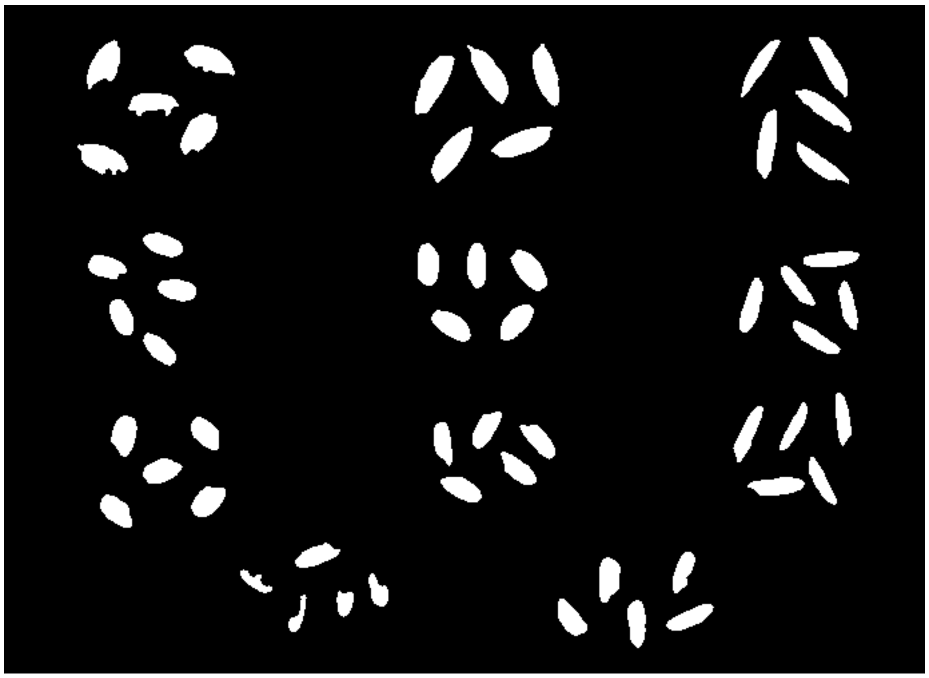
```
1 1 0 1 1
1 1 1 1 1
1 1 1 1 1
```

5. Morphological Process – Shrinking:
   Shrinking is carried out to count the number of grains (foreground objects in this case) that have been shrunk as a result of them being whole objects without holes.

6. To count the area occupied by each type of seed, I have iterated through a bounding box circumscribing the category of seeds and counted the number of foreground pixels and then divided them by 5.

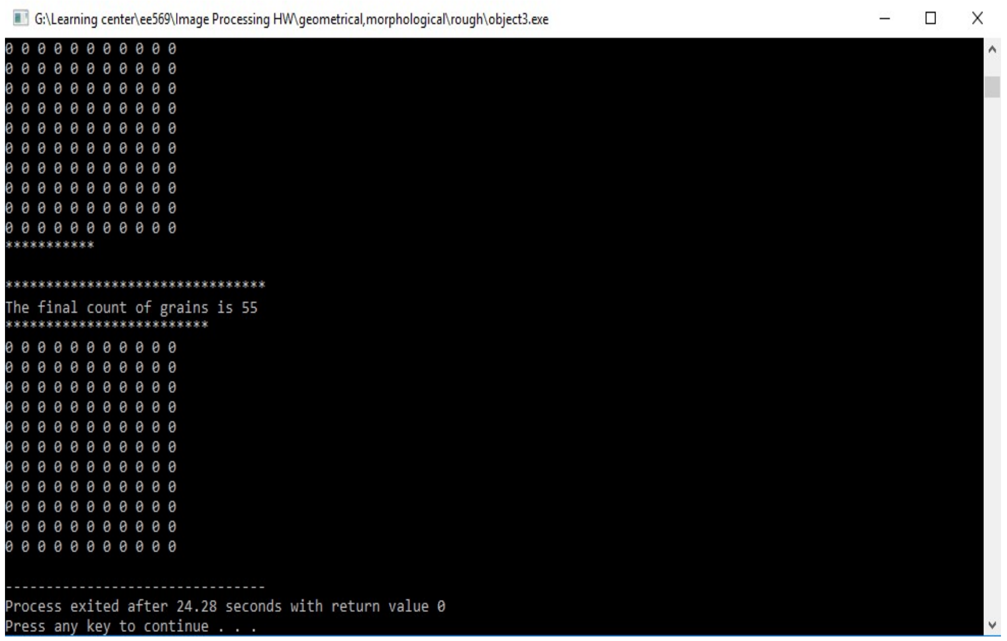
Note: Please be patient with my debug statements in my code. They were put there to clear the output buffer when runtime error was caused.

3. Results:



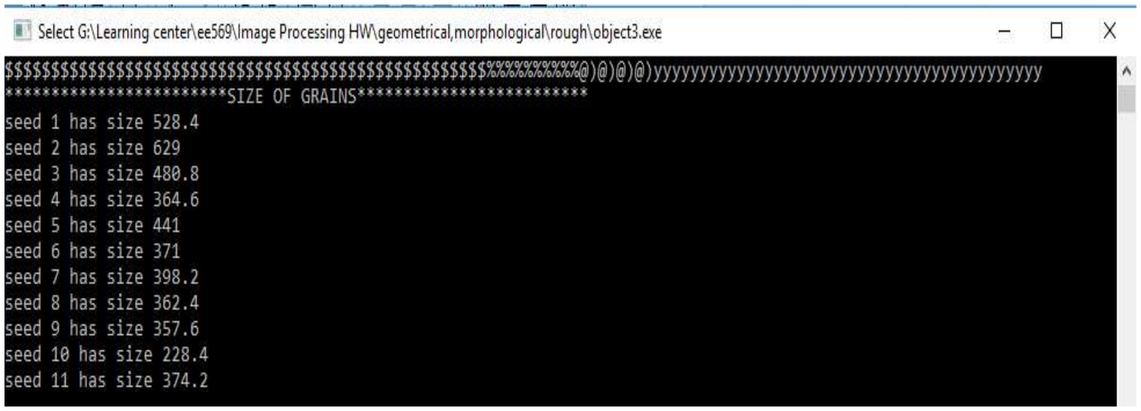
After pre-processing for this task

After shrinking



```
G:\Learning center\ee569\Image Processing HW\geometrical,morphological\rough\object3.exe
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
***********

*******************************
The final count of grains is 55
************************
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0

--------------------------------
Process exited after 24.28 seconds with return value 0
Press any key to continue . . .
```

Console showing the final count


Select G:\Learning center\ee569\Image Processing HW\geometrical,morphological\rough\object3.exe

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$%%%%%%%%%@)@)@)@)yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
***********************SIZE OF GRAINS*************************
seed 1 has size 528.4
seed 2 has size 629
seed 3 has size 480.8
seed 4 has size 364.6
seed 5 has size 441
seed 6 has size 371
seed 7 has size 398.2
seed 8 has size 362.4
seed 9 has size 357.6
seed 10 has size 228.4
seed 11 has size 374.2
```

Sizes of each type of unit grain based on an average estimate.

Seeds are labelled:

1 2 3
4 5 6
7 8 9
10 11

4. **Discussion:**
   - Thresholding the rice image depended on various factors and not just the histogram of the pixels
   - This gives rise to the fact that even while thresholding, like morphological processes, logical operations help reveal the pattern underneath the layers
   - Postprocessing also involves morphological masks that ensures that no stray pixels/ any patterns are removed
   - Erosion and dilation are used when there is a need to bias the shape of the objects into either smaller or larger objects respectively. In this case, I used masks which did the job of ensuring that the objects remained whole.
   - Hit and miss masks can help in manual deletion of unnecessary patterns that recur over a given space
   - In rice grain analysis, similar patters can be used to with images that have higher resolution and masks that are smaller so more information can be gleaned from them without losing out the details
   - Working in the RGB domain will ensure preservation of information in rice grain analysis
   - Shrinking works wonders when it comes to counting the number of objects, especially when they are whole. Although, the pre and post processing involve gruelling operations.

Note on computational complexity:

(Relative scale)

Most complex to least complex:

Morphological operations > Rice grain analysis > Geometric Transformation > Warping > Radial distortion > Image defect correction

References:

1. **http://homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm**