Homework Three Report

Zifan Wang

March 4, 2019

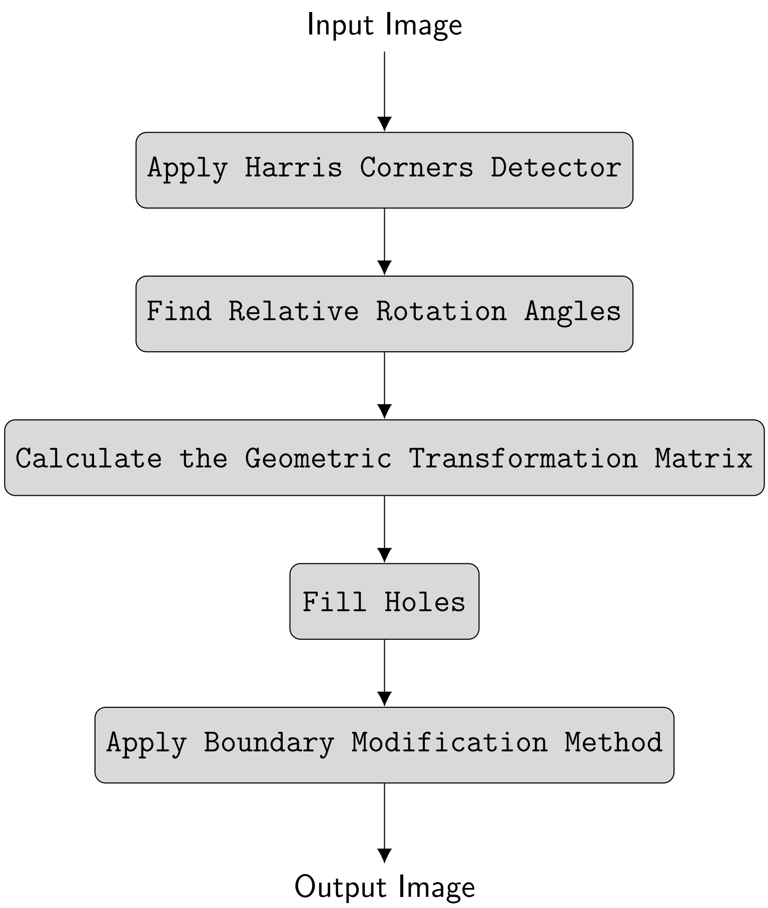
1 Problem 1: Geometric Modification

1.1 A. Geometric Transformation

1.1.1 Abstract and Motivation

Geometric modification is a commonly used method in the image processing and computer vision fields. Rotation, translation, and scaling are three operations of geometric modification to help people create desired output images. Take panoramic image as an example. There are some city view images which share some common features. Desired panoramic city view image adopts geometric transformation to relate these images together. This section contains some experiment about geometric transformation technique.

1.1.2 Approach and Procedures



In this problem, a light house image with three holes and three relative images are given. The desired output light house image is to fill three holes with these three sub-images. The gray scale given images help to reduce color conversion step. First, apply Harris Corners Detector method [2]. to locate four important corners in the relative images and 3 top-left corners in the light house original image. Harris Corner Detection Algorithm is listed below.

- Compute x and y derivatives of image

$$I_x = G_x \times I$$

$$I_y = G_y \times I$$

  where G_x and G_y are coefficient matrices in **Sobel Edge Detector**, I_x and I_y are derivatives of image in x and y directions, and I is the input image. The G_x and G_y are defined below:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

- Compute products of derivatives at every pixel

$$I_x^2 = I_x \cdot I_x$$

$$I_y^2 = I_y \cdot I_y$$

$$I_{xy} = I_x \cdot I_y$$

- Apply a 3×3 window to weight I_x, I_y, and I_{xy} to generate a M matrix which contains important information about x and y gradient in this window.

$$M = \sum_{x,y} w(x,y) \times \begin{bmatrix} I_x^2 & I_{xy} \\ I_{xy} & I_y^2 \end{bmatrix}$$

  where $w(x, y)$ are usually Gaussian weighted function or 1.

- Compute the response of the detector at each pixel

$$R = Det(M) - k(Trace(M))^2$$

  where k is usually between 0.04-0.06.

- Threshold on value of R to determine important corners.

- Apply non-maximum suppression algorithm which is discussed in the **Canny Edge Detector** in the last homework to limit neighborhood corners.

Second, finding relative rotation angle is important to fill the holes correctly in the light house image. Find the lowest mean-square error (mse) between each side of sub-image and the bottom side of original light house image. If the bottom side of relative image has the lowest mse, further rotation does not need. If the left side of relative image has the lowest mse, furthre 270 degrees rotation is needed. If the top side of relative image has the lowest mse, further 180 degrees rotation is required. If the right side of relative image has lowest mse, 90 degrees rotation is needed.
Third, use geometric transformation matrix to map each pixel to the new location in the image and use bilinear interpolation to compute the pixel value in the new location. The following representations are scaling, translation, and rotation matrices used in the geometric transformation:

- Translation:

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

- Scaling:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_x & 0 & 0 \\ 0 & \lambda_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

- Rotation:

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

where u, v is original location and x, y is the transformed location.
Fourth, apply the translation matrix to fill the holes in the original light house image. Last, the boundary modifier is apply to the filled holes' image to generate the output image. Use a 3×3 mean filter which is discussed in the homework one to decide the value at the boundary of holes. After this process, generate the output image.

1.1.3 Result

(a) (b) (c)

Figure 1: Sub-images Rotation

The figure 1 shows the relative sub-images rotate to the corrected positions.

(a) Output Image Without Boundary Modification (b) Output Image With Boundary Modification

Figure 2: Output Light House Image

The figure 2 shows the filled holes light house image after geometric transformation.

1.1.4 Discussion

Comparing the 2(a) with 2(b), the boundary modification algorithm produce the better output. However, the mean filter will also smooth the boundary of holes and does not preserve the sharp edges. In addition, using mean-square error to find the relative rotation angles of sub images is not very robust. The **Scale-invariant feature transform** (SIFT) and **Speeded-Up Robust Features** (SURF) algorithm will improve this detection.

1.2 B. Spatial Warping

1.2.1 Abstract and Motivation

People are always attracted by beautiful and special images. Warped image is a common type of special images. The geometric modification technique allows people to create special images from an original rectangular shape to any desired shape.

1.2.2 Approach and Procedures

Same to the geometric transformation in the previous section, the spatial warping method also map set of points into new location. Shape the image into four triangular regions and define the reference points which are the red dots in the following picture:

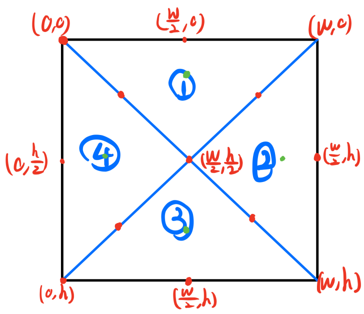


Figure 3: Spatial Warped Algorithm

The following picture shows a algorithm to map the old reference points into the new points. This algorithm can be represented by the following equation:

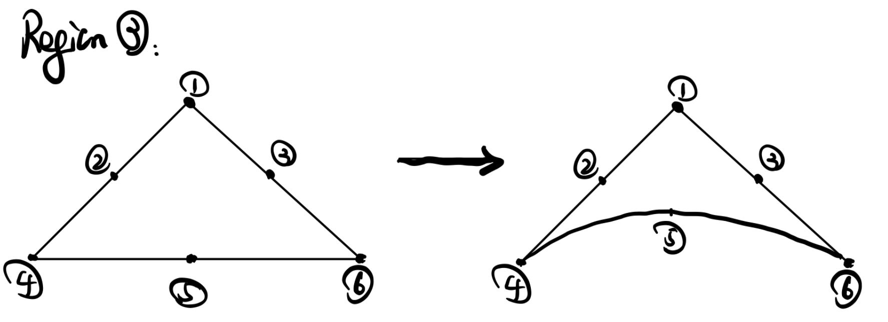


Figure 4: Spatial Warped Algorithm

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

where a, b are coefficients, u, v are old points, and x, y are mapped location.

1.2.3 Result

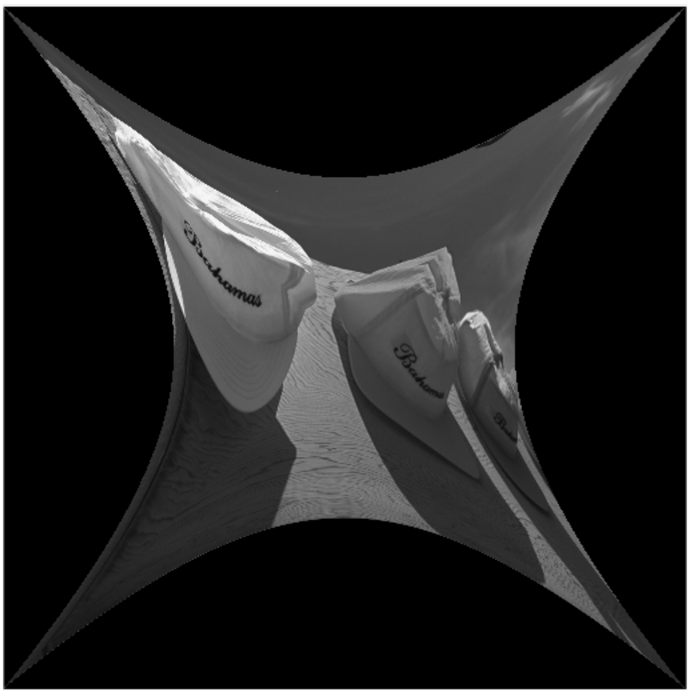


Figure 5: Spatial Warped Output Image

The figure 5 is the warped output images.

1.2.4 Discussion

Defining different arc lengths will return different warped images. By following steps of spatial warping method, the special and fantastic images.

1.3 C. Lens Distortion Correction

1.3.1 Abstract and Motivation

When taking images, people cannot make sure they are capturing on a flat camera's focal plane. This will result a radial distortion effect in images. To reduce this effect, the lens distortion correction is applied to modify images.

1.3.2 Approach and Procedures

Applying the same logical as geometric transformation in the previous sections, the lens distortion correction method also maps in the distorted pixel location to the correct location. The following equations show the relationship between the actual image and its distortion:

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

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

where x, y is undistorted pixel location, x_d, y_d is distorted pixel location , k_1, k_2, k_3 are called radial distortion coefficients of camera lens, and $r^2 = x^2 + y^2$. Apply inverse mapping to recover the undistorted pixel location by the distorted pixel location and focal length.

1.3.3 Results

The following figure 6 is the image with radius distortion effects:

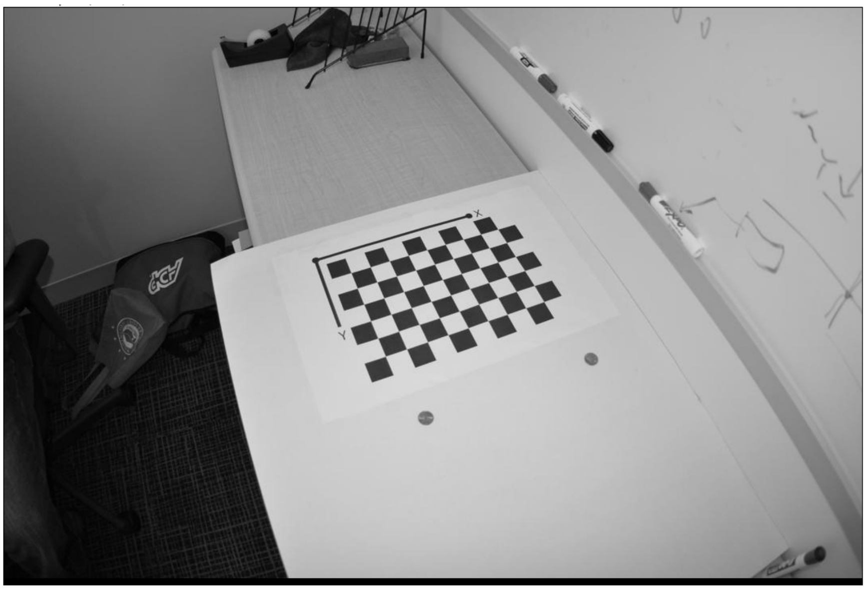


Figure 6: Lens Distorted Image

The following figure 7 shows the recovered image.

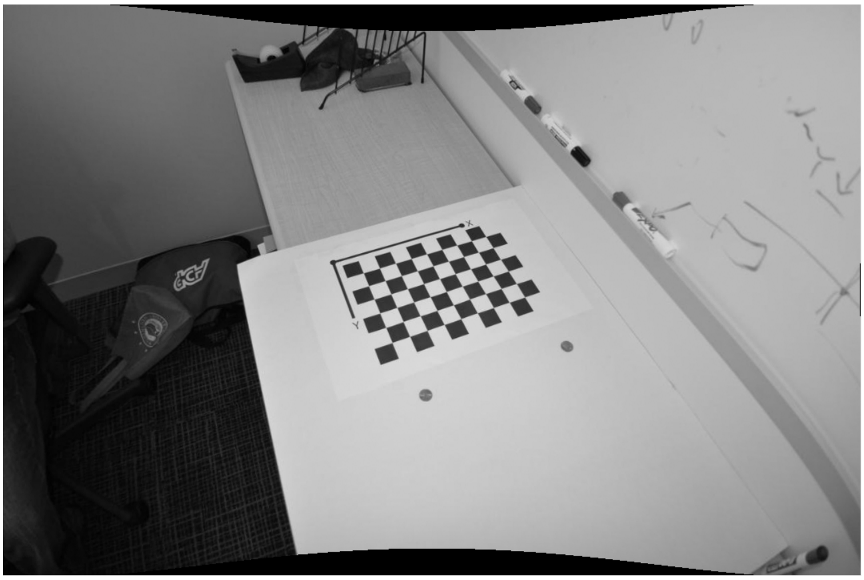


Figure 7: Recovered Image

1.3.4 Discussion

The figures 6 and 7 show differences between distorted and undistorted image. The figure 7 gives a better result. Moreover, most computer vision jobs require to calibrate the camera to reduce lens distortion effect to achieve better performance.

2 Problem 2: Morphological Processing

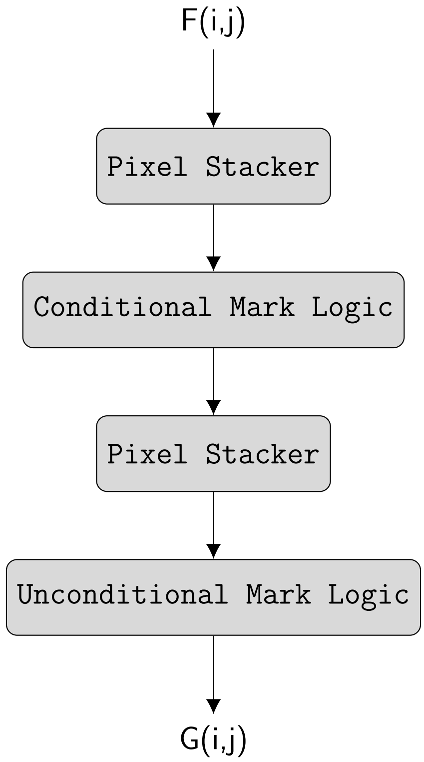
2.1 Basic Morphological Process Implementation

2.1.1 Abstract and Motivation

Morphology is a study about the form of objects. In the human vision system, the way to analyze the shape of objects is using the appearance and the shape. People also want to adopt this method in the machine vision. The morphology processing is a way to study the shape of objects. Shrinking, thinning, and skeletonization are three basic operations for morphology processing.

2.1.2 Approach and Procedures

Shrinking is the process that remove 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. **Thining** is defined as erasing black pixels such that an object without holes erodes to minimally connected stroke located equidistant from its nearest outer boundaries and an object with holes erodes to a minimally connected ring midway between each hole and its nearest outer boundary. **Skeletonization** is to extract a region-based shape feature representing the general form of an object. They are sharing the same algorithm to implement with different hit-or-miss filters. The flow chart below shows the detail implementation [1].



The first pixel stacker collects nine neighboring pixels and the following conditional mark logic determines the possible erasures. The second pixel stacker gathers nine neighboring pixels' conditional mark values and the following unconditional mark logic decides to remove this center pixel or not. In short, the following logical expression shows the output pixel value at position (i, j):

$$G(i,j) = X \cap [\overline{M} \cup P(M, M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7)]$$

where M is the value calculated by conditional mark logic defined in the figure 8 [1] and $P(M, M_0, M_1, M_2, M_3, M_4, M_5, M_6, M_7)$ is an erasure inhibiting logical variable defined in the figure 9 and the figure 10 [1]..

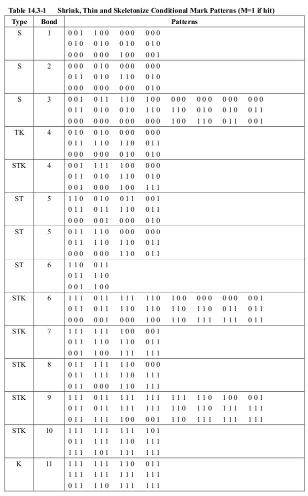


Table 14.3-1 Shrink, Thin and Skeletonize Conditional Mark Patterns (M=1 if hit)

Type	Bond	Patterns
S	1	001 100 000 000 010 010 010 010 000 000 100 001
S	2	000 010 000 000 011 010 110 010 000 000 000 010
S	3	001 011 110 100 000 000 000 000 011 010 010 110 110 010 010 011 000 000 000 000 100 110 011 001
TK	4	010 010 000 000 011 110 110 011 000 000 010 010
STK	4	001 111 100 000 011 010 110 010 001 000 100 111
ST	5	110 010 011 001 011 011 110 011 000 001 000 010
ST	5	011 110 000 000 011 110 110 011 000 000 110 011
ST	6	110 011 011 110 001 100
STK	6	111 011 111 110 100 000 000 001 011 011 110 110 110 110 011 011 000 001 000 100 110 111 111 011
STK	7	111 111 100 001 011 110 110 011 001 100 111 111
STK	8	011 111 110 000 011 111 110 111 011 000 110 111
STK	9	111 011 111 111 111 110 100 001 011 011 111 111 110 110 111 111 011 111 100 001 110 111 111 111
STK	10	111 111 111 101 011 111 110 111 111 101 111 111
K	11	111 111 110 011 111 111 111 111 011 110 111 111

Figure 8: Conditional Mark Logic

Table 14.3-2 Shrink and Thin Unconditional Mark Patterns

Spur	0	0	M	M	0	0						
	0	M	0	0	M	0						
	0	0	0	0	0	0						
Single	0	0	0	0	0	0						
4-connection	0	M	0	0	M	M						
	0	M	0	0	0	0						
	0	0	M	0	M	M	M	M	0	M	0	0
L Cluster	0	M	M	0	M	0	0	M	0	M	M	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
	M	M	0	0	M	0	0	M	0	0	M	M
	M	0	0	M	M	0	0	M	M	0	0	M
4-connected	0	M	M	M	M	0	0	M	0	0	0	M
Offset	M	M	0	0	M	M	0	M	M	0	M	M
	0	0	0	0	0	0	0	0	M	0	M	0
Spur corner	0	A	M	M	B	0	0	0	M	M	0	0
Cluster	0	M	B	A	M	0	A	M	0	0	M	B
	M	0	0	0	0	M	M	B	0	0	A	M
Corner	M	M	D									
Cluster	M	M	D									
	D	D	D									
Tee Branch	D	M	0	0	M	D	0	0	D	D	0	0
	M	M	M	M	M	M	M	M	M	M	M	M
	D	0	0	0	0	D	0	M	D	D	M	0
	D	M	D	0	M	0	0	M	0	D	M	D
	M	M	0	M	M	0	0	M	M	0	M	M
	0	M	0	D	M	D	D	M	D	0	M	0
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	D	M	0	0	M	D	D	0	M	M	0	D
Branch	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 or B or C = 1 D = 0 or 1 A or B = 1												

Figure 9: Unconditional Mark Logic for Shrinking and Thinning

Table 14.3-3 Skeletonize Unconditional Mark Patterns

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	0	0	0	0	0	0	0	0	0	0	M	0
4-connection	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	M	M	D	D	D	D						
Cluster	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	D	M	0	0	M	D	D	0	M	M	0	D
Branch	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 or B or C = 1 D = 0 or 1

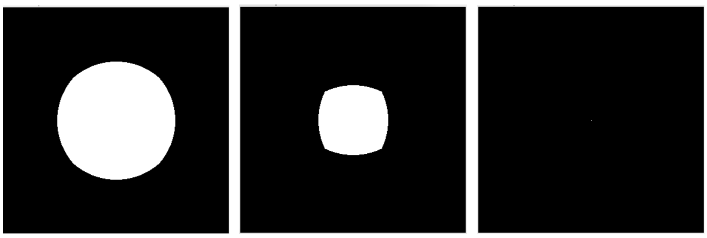
Figure 10: Unconditional Mark Logic for Skeletonizion

In the conditional mark logic, the letter S stands for shrinking patterns, the letter T indicates the thinning patterns, and the letter K is for the skeletonizion patterns.

2.1.3 Result:

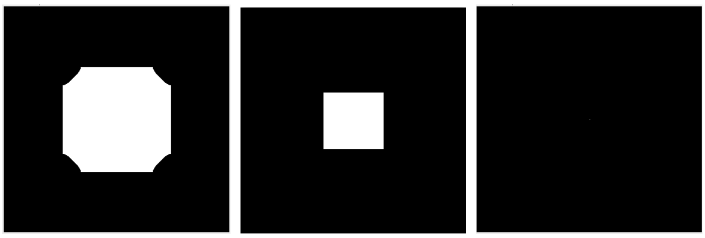
Shrinking

Showing in the figure 11(c), 12(c), and 13(c). the pattern 1-3 are a white dot near the center of images. However, the figure 14(c) shows that the pattern 4 is a contour shape after shrinking.



(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 11: Output of Shrinking Pattern 1

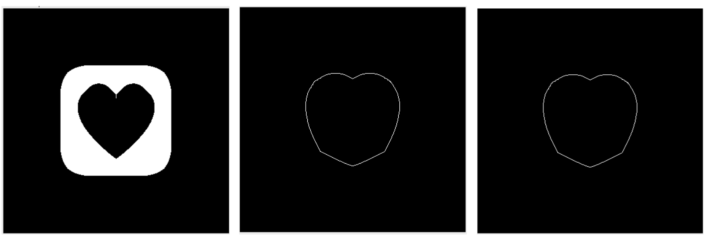


(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 12: Output of Shrinking Pattern 2

(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 13: Output of Shrinking Pattern 3

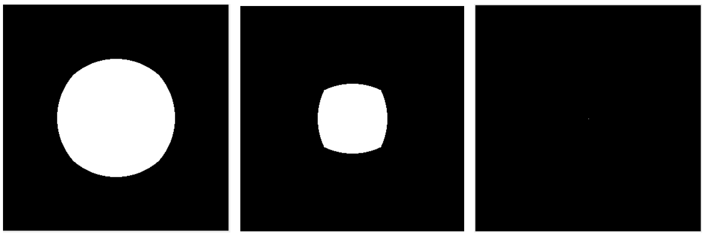


(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 14: Output of Shrinking Pattern 4

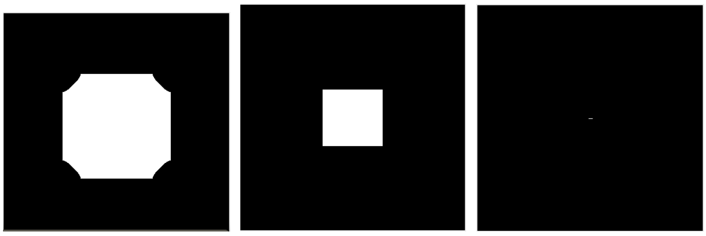
Thinning

Showing in the figure 15(c) and 16(c), the pattern 1 and 2 are a white dot near the center of images. However, the figure 17(c) and 14(c) shows that the pattern 3 and 4 are contour shapes after thinning.



(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 15: Output of Thinning Pattern 1

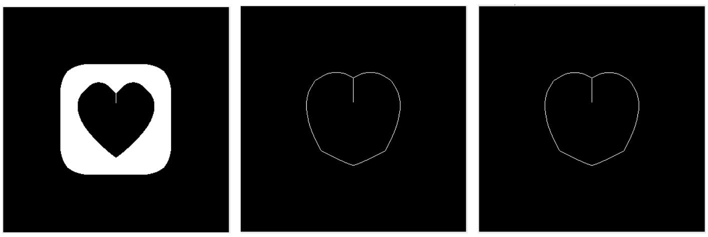


(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 16: Output of Thinning Pattern 2

(a) 10 iterations (b) 50 iterations (c) Final Output

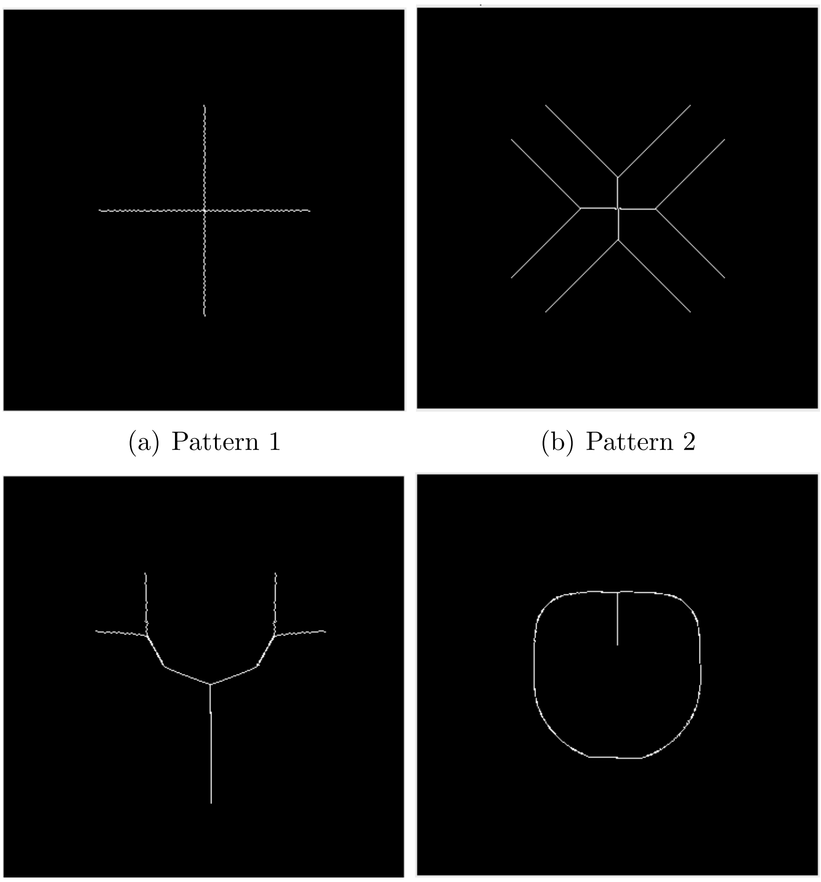
Figure 17: Output of Thinning Pattern 3



(a) 10 iterations (b) 50 iterations (c) Final Output

Figure 18: Output of Thinning Pattern 4

Skeletonizion



(a) Pattern 1 (b) Pattern 2

(c) Pattern 3 (d) Pattern 4

Figure 19: Output of Skeletonizion Process

2.1.4 Discussion

As definition before, the **Shrinking** is the process that remove 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. After shrinking process, pattern 1,2, and 3 is a white dot in the center of mass. However, the pattern 4 has a hole in it, so it reduce to a contour shape in the midway between hole and its nearest outer boundary. For **Thining**, the pattern 1 and pattern 2 should be eroded to a minimally connected stroked located equidistant from its nearest outer boundaries. This is a white dot in the middle of image. Same to the pattern 3 which reduce two connected lines. For pattern 4 which contains a hole, it is reduced to a connected ring midway between each hole and its nearest outer boundary. **Skeletonizion** should return more detail of original pattern. The figure 19 meets our perspective.

2.2 B. Defect Detection and Correction

2.2.1 Abstract and Motivation

Detecting and removing defects are applications of morphological processing. This process is able to ensure that when images with defections are enlarged, people cannot obviously see the defections.

2.2.2 Approach and Procedures

Checking the binary image connectivity is one way to detect defects. Four connectivity and eight connectivity two type of connectivity. The four connectivity is defined as any of 4 nearest neighboring pixel of center pixel is 1. The eight connectivity is defined as any of 8 nearest neighboring pixel of center pixel is 1. In this problem, the main body defects have the following pattern: center pixel's value is zero and the four or eight nearest neighboring pixels' value is one. If this pattern is detected, the center pixel can be modified to 1 in the image.

2.2.3 Result

The following figure 20 shows the deer image after thinning process.

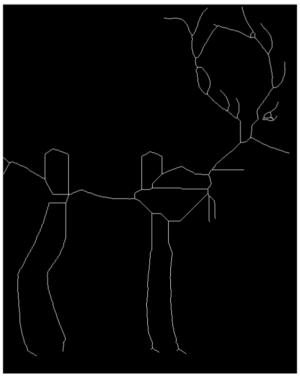


Figure 20: Thinning of Deer Image

This the body of the deer does not thin to one line shows that there are some defects in the main body of deer. The defects' finding filter is applied. The following table 1 shows the location of defections.

x	y
498	207
93	280
275	284
334	335
331	352
464	55
503	212
502	213

Table 1: Table of defects' Locations

The following figure 21 shows the corrected deer image.

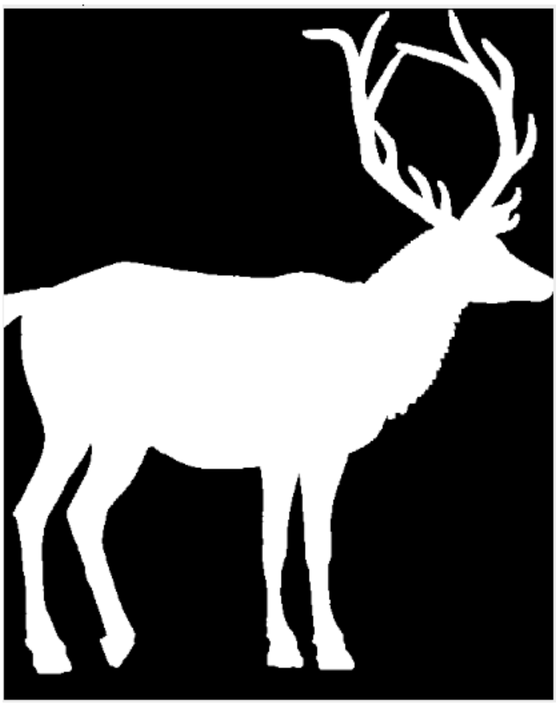


Figure 21: Corrected Deer Image

2.3 C. Object Analysis

2.3.1 Abstract and Motivation

The morphology processing can also be used in the object analysis aspect. As state before, the morphology processing is a way to recognize the shape of objects. So, one way to classify objects is to compare them by sizes.

2.3.2 Approach and Procedures

Converting a color image to a gray scale image is needed to perform morphological operations. The reason is that the color image is hard to perform morphological operations. After reading the image, apply a color conversion to make the input image into gray scale, which is done by the following equation.

$$G(i,j) = 0.2989 \times I(i,j,0) + 0.5870 \times I(i,j,1) + 0.1140 \times I(i,j,2)$$

where the $I(i,j,0)$ is red color value at the input image position (i,j), the $I(i,j,1)$ is green color value at the input image position (i,j), and the $I(i,j,2)$ is blue color value at the input image position (i,j).
However, there are some problems with this conversion. Some objects' colors are close to the background. This will cause the conversion does not work. Instead, taking each color image out is necessary. Applying a median filter to remove noise and modifying the pixel values based on the most appearance brightness in each color channel will return a better result. Then, according to pixel values in the same location of each channel to decide whether there is a pixel value in this location. For example, in the location i,j

only blue channel has a pixel value and red and green channels are 0, this location i, j has a pixel value 255 in the output image. After this process, using the hole filling algorithm and removing isolated pixel which is discussed in the previous section to fill hole in the body and remove unnecessary pixels.
Shrinking processing will eraser objects without holes to be a point, so this method can help to determine the number of objects in the image. Therefore, shrinking process is applied to determine number of rice in the image. The thinning processing will return ellipsoid shape objects to lines. These lines' sizes are determine by the size of rice. Thus, a thinning processing will help to rank the size of rice.

2.3.3 Result

The following figure 22 is the output image after preprocessing process:

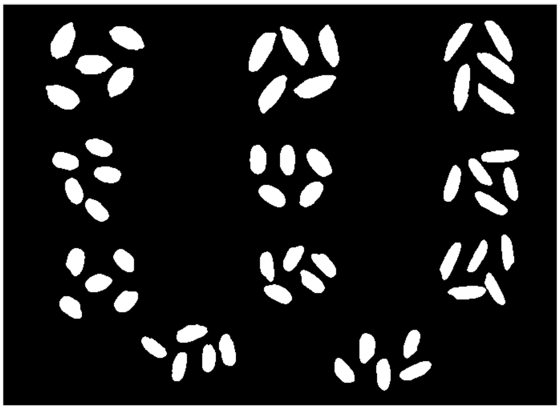


Figure 22: Rice Image After Preprocessing

The following figure 23 is the output image shrinking process and is used to count the number of rice. There are 55 rice in total in the image.

Figure 23: Rice Image After Shrinking

The following figure 24 is the output image thinning process and is used to compare the size of rice grains. The right top grain has the largest size. Second is the top middle grains. Then, the top left one has the third place. The rice in the second row and the third row in the right of the image have similar size, so they are ranked fourth. Last, the rest of rice grains' sizes also do not have much difference. Thus, they are in the last rank.

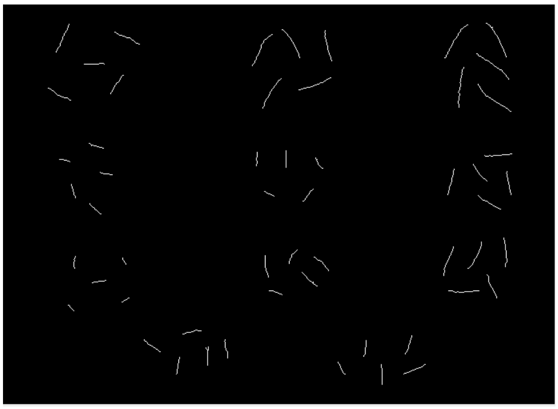


Figure 24: Rice Image After Thinning

2.3.4 Discussion

The figure 24 shows that the rice grains after thinning process is not a perfect line. Therefore, it is hard to compute the size. However, the opening, closing, and dilation process in the OpenCV library can be helped to solve this problem.

References

[1] William K. Pratt. *Digital Image Processing, 4th*. PixelSoft, Inc. Los Altos, California. 2007.

[2] Robert Collins: Lecture 06: Harris Corner Detector. `http://www.cse.psu.edu/~rtc12/CSE486/lecture06.pdf`