EE569 – HW3

Mozhdeh Rouhsedaghat

2726554211 rouhseda@usc.edu

Problem 1

Abstract and Motivation

Geometrical image manipulation is a concept related to computer graphics and means transforming the coordinates of pixel values for a specific purpose. In this transformation the pixel values won't change but their location changes and make the image look different. Some examples of geometrical image manipulation are shifting, rotation, scaling, warping, etc.

Approach and Procedures

The first step for any transformation is to map the image coordinate to the cartesian coordinate to be able to simply do the transformations based on the cartesian coordinate. The formula for this transformation is given below:

$$x = k - 0.5$$

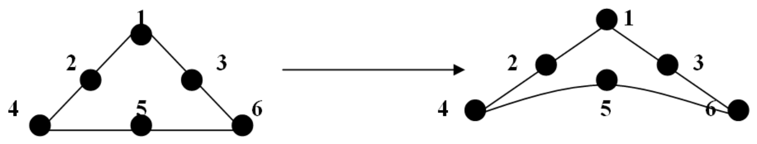
$$y = -j + J + 0.5$$

Then for applying the below transformations we adapt the reverse address mapping. It means that we don't calculate the new coordinate of each pixel of the given image, but we find the corresponding input pixel of every pixel location in the output image and copy its pixel value to the new pixel location so that we make sure that a value is assigned to any pixel location in the output.

Geometric Transformation: We can do any rotation, scaling and translation by doing a triangle to triangle mapping from the input image to output. In this approach we consider the whole transformation as a 3*3 matrix with unknown first and second row elements and we can find them by mapping three vertices of triangles. You can see such a transformation matrix below:

$$\begin{bmatrix} x_k \\ y_j \\ 1 \end{bmatrix} = \begin{bmatrix} C_1 & C_2 & C_3 \\ d_1 & d_2 & d_3 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u_q \\ v_p \\ 1 \end{bmatrix}$$

Spatial Warping: For doing spatial warping one approach is to divide the input image into 4 triangles and define a one to one mapping from each triangle to its corresponding shape. For the specific spatial warping of the homework we can use a mapping of degree 2 between 6 main points of the triangle and the output shape as bellow:



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

The mapping gives us the transformation matrix and we can use it to find the corresponding input pixel to each output pixel.

Lens Distortion Correction: Cameras may capture images in a distorted way in which the relationship between the actual image and the distorted one is as follows:

$$x_d = \mathrm{x}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_d = \mathrm{y}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

where x, y are undistorted pixel locations, K_1, K_2, K_3 are radial distortion coefficients of the lens, and $r^2 = x^2 + y^2$. To recover the undistorted image, we need to find the inverse function of the above functions, but there is no exact inverse function of the above formula so using linear regression with can find a linear approximate of the inverse function. Linear regression is a method which tries to find a linear approximate of a distribution of points by minimizing the mean square distance of the approximated line and the given input points.

Results

a)

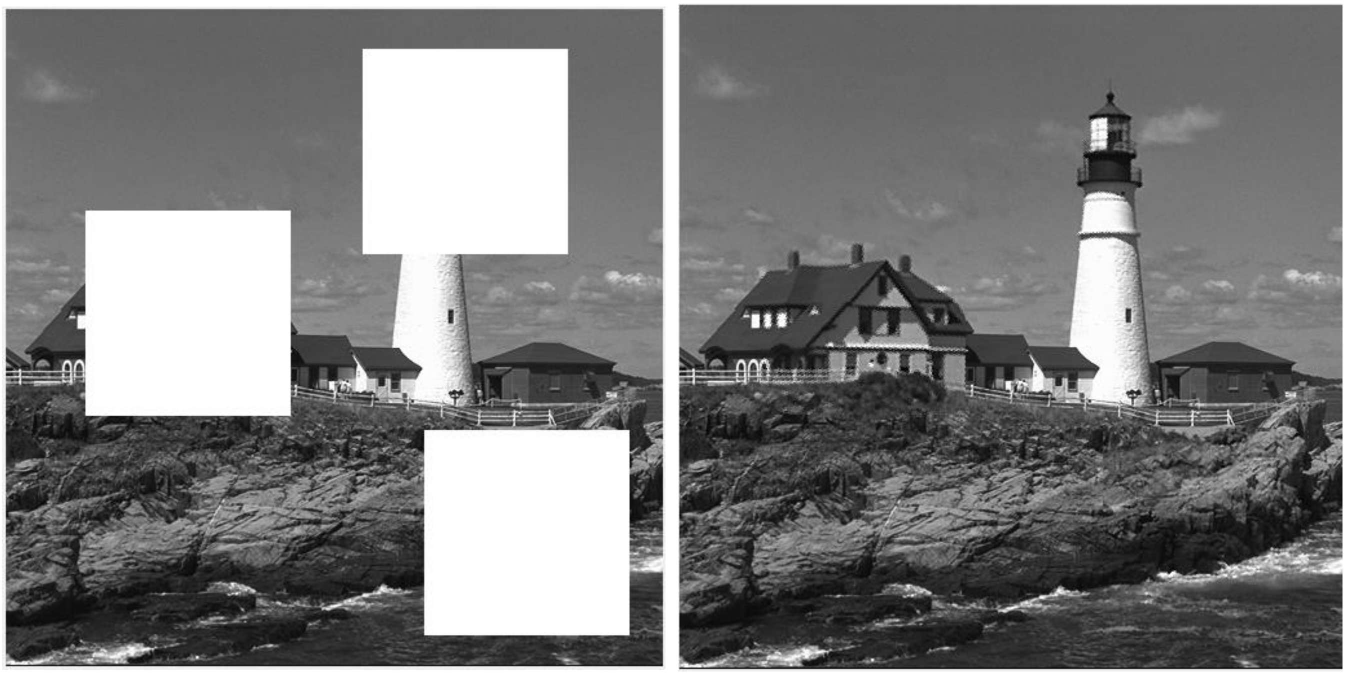


Fig1: lighthouse image before (left) and after (right) applying the transformation

I found the location of corner of rectangular holes by searching the specific pattern of each corner (for example $\begin{bmatrix} d & d & d \\ d & 0 & 0 \\ d & 0 & 0 \end{bmatrix}$ for the upper left corner) in the lighthouse image. I also found the corners of each sub image by searching the first non-background pixel from each of its four sides. Then computed the elements of the transformation matrix by triangle to triangle mapping method and at last found the pixel values by reverse address mapping. (I have copied one row or column of surrounding pixels to remove some white boundaries from the final image and reach a perfect result.)

b)

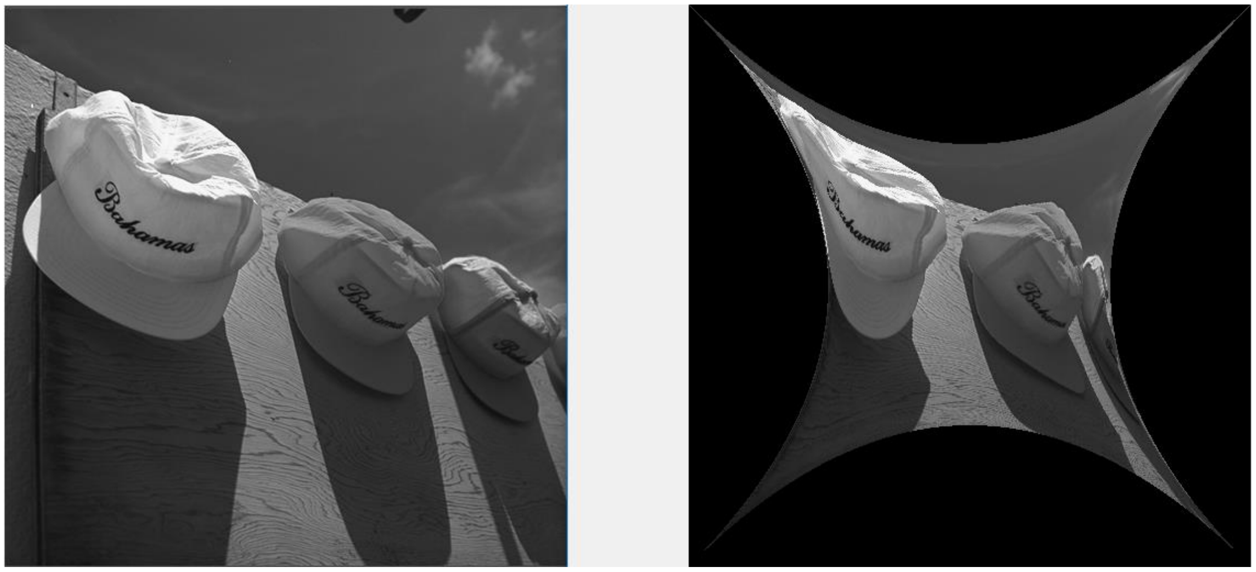
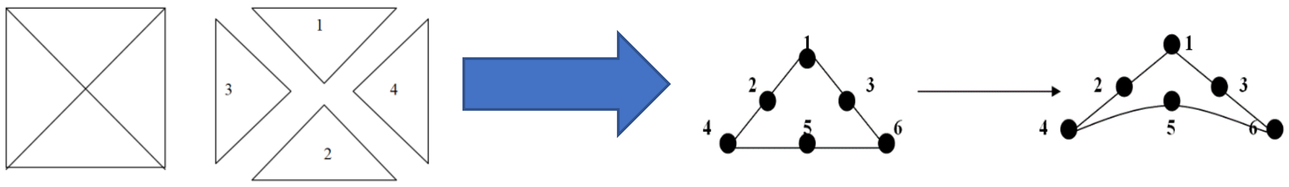


Fig2: hat image before (left) and after (right) applying the spatial warping

For finding the reference points, I have divided the image into four triangles and selected vertices and midpoint of each side as the reference point. Bellow you can see the mentioned points:



c)

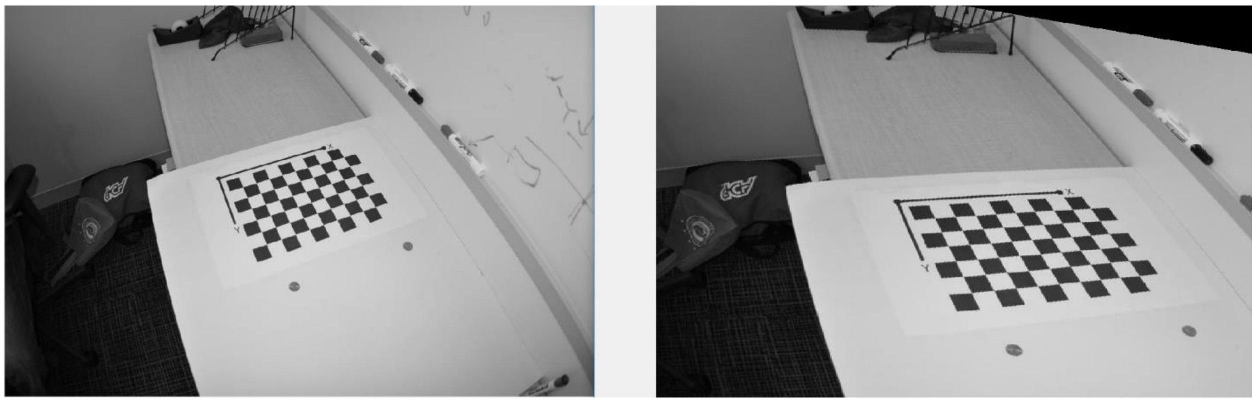


Fig3: classroom image before (left) and after (right) removing distortion

As a linear regression method, I have used the least square method to find a linear inverse function. I have used about 12000 x and y pairs to compute the least square linear approximation. They shouldn't be chosen from central pixels as they have the minimum distortion and the model won't be able to remove the distortion effectively. I have used pixels which are relatively closer to sides to find a model which is able to remove the distortion.

Discussion

As you know rotation, scaling and translation transformations can be represented as 3*3 matrices $\begin{bmatrix} cos & -sin & 0 \\ sin & cos & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{bmatrix}$, and $\begin{bmatrix} 1 & 0 & xc \\ 0 & 1 & yc \\ 0 & 0 & 1 \end{bmatrix}$ respectively. They are linear transformations and we can use their product as a single matrix which does all of them. The triangle to triangle mapping (using such a matrix $\begin{bmatrix} C_1 & C_2 & C_3 \\ d_1 & d_2 & d_3 \\ 0 & 0 & 1 \end{bmatrix}$) uses this property and by finding its elements through mapping corresponding vertices we can use the matrix as a combination of the required rotation, scaling and translation transformations.

To do the distortion correction, we need to choose a set of pixels to compute the linear approximation using them. They shouldn't be chosen from central pixels as they have the minimum distortion and the model won't be able to remove the distortion effectively. Using boundary pixels also exaggerates the amount of distortion in the approximation and doesn't give us a good model. I have used pixels which are relatively closer to sides to find a model which is able to remove the distortion.

Problem 2

Abstract and Motivation

Morphological image processing aims to analyze the shape. The input and output of such processes are binary images. Simple morphological filters are 3*3 filters, for example additive and subtractive filters are used to convert some background points to foreground points and vice versa. We also have advanced morphological filters which consist of two stages of 3*3 filters. Some examples of such filters are shrinking, thinning, and skeletonizing.

Approach and Procedures

advanced morphological filters consist of two stages of 3*3 filters. The first stage is called conditional mask and used for detecting pixels which should be changed. The second stage is called unconditional mask and used to confirm whether the detected pixels must be changed or ignored. There are a set of patterns for each stage and if the input matches these filter patterns we say it is a hit and otherwise it is a miss. In the first stage a hit means the pixel should be selected for removal and in the second stage a hit means ignoring the chosen pixels and a miss confirms the removal.

These two set of patterns are different for shrinking, thinning, and skeletonizing. There is also an extra stage of bridging for skeletonizing which reconnects some disconnected pixels.

In general shrinking gives the central pixel, thinning gives the central line and skeletonizing gives the central line with considering the shapes and corner effects.

Results

a)

I have stopped the procedure when after an iteration no changes is made to the image.

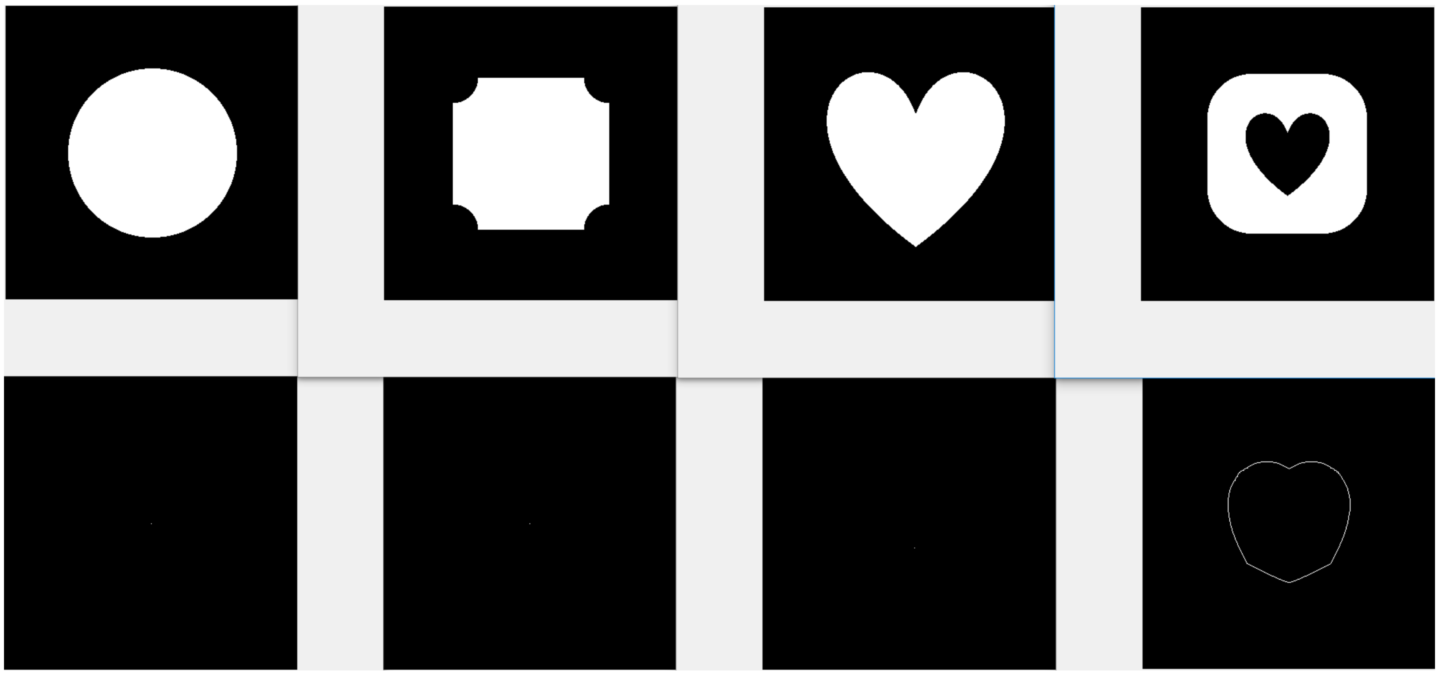


Fig4: patterns 1-4 (up) and their shrinking result (down)

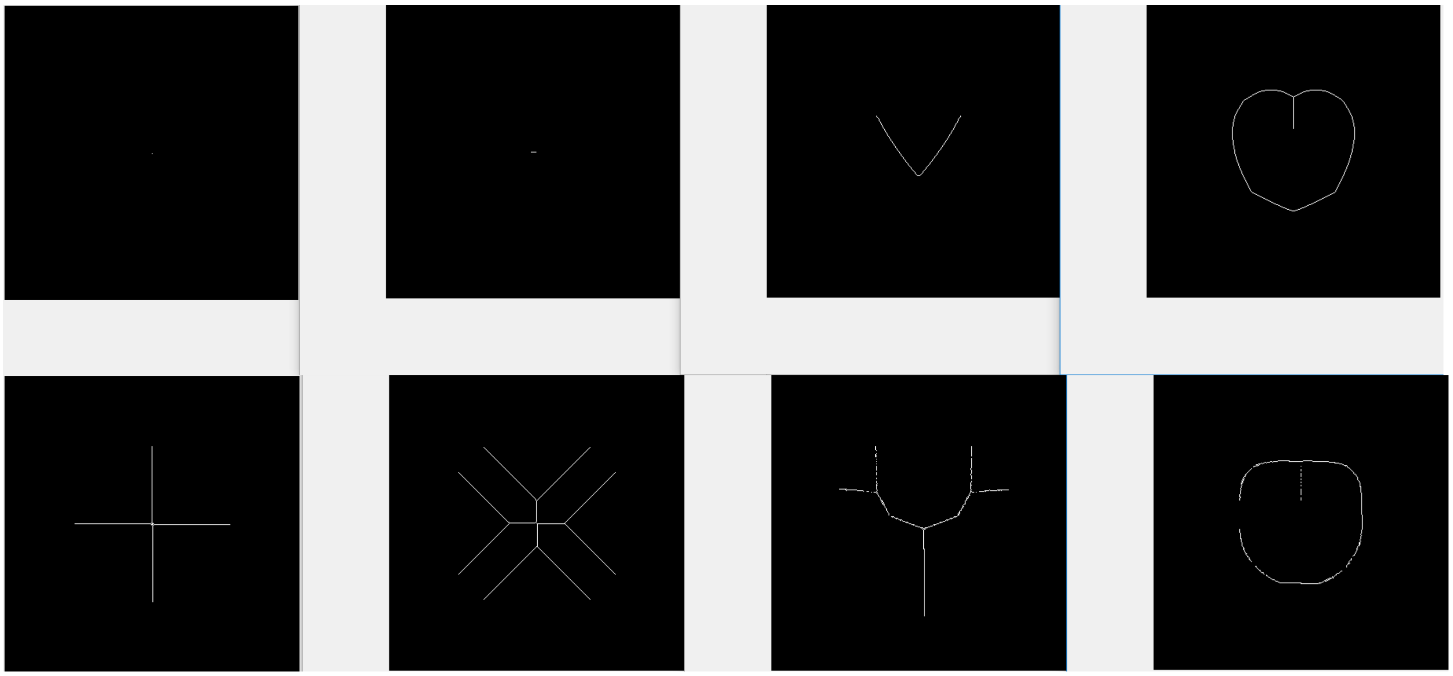


Fig5: thinning (up) and skeletonizing (down) result of patterns 1-4

As you see if the shape of the image doesn't have a hole, the result of shrinking will be the central pixel, and the result of thinning is the central line. At each step, thinning and shrinking reduces the size of the shape by one pixel from all sides, and when it becomes a line (with the width of one pixel) the thinning stops, but shrinking continues to reach a shape with one pixel. The result of skeletonizing is highly related to corners and vertices of the image and the output lines shows the structure of the corners of the initial image.

b)

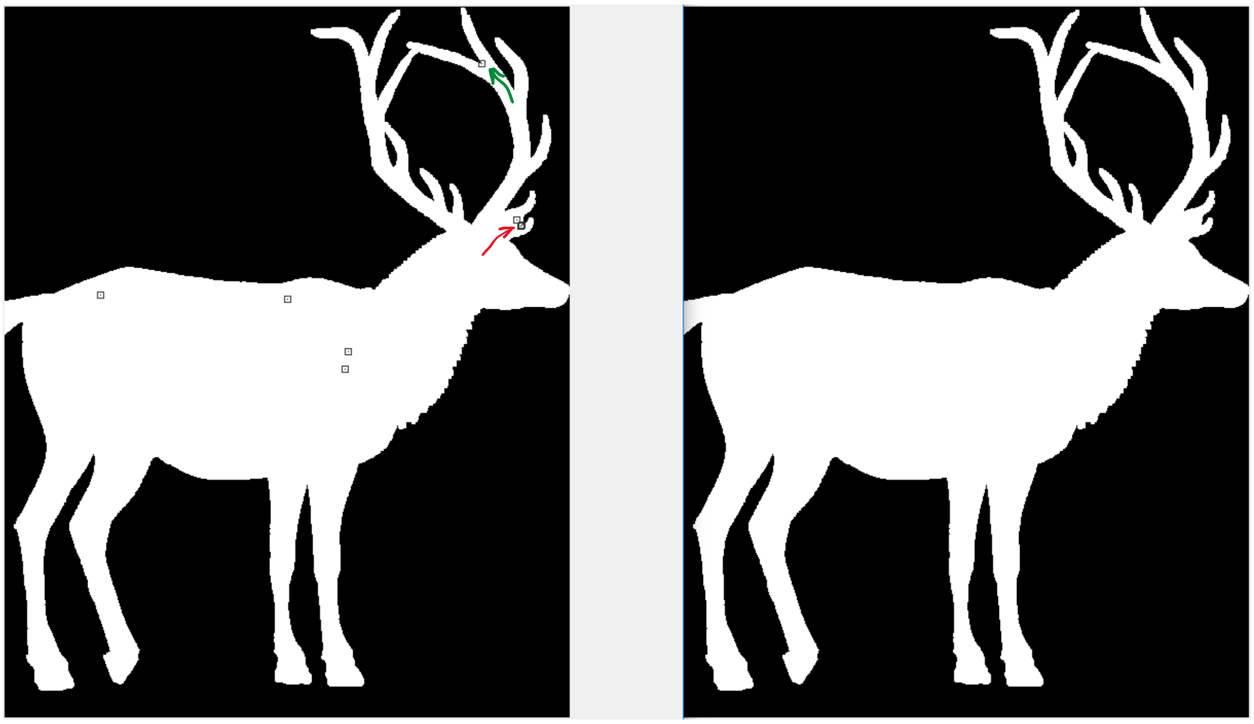


Fig6: The deer image with borders around defects (left) and the corrected image (right)

1) No, there are 8 defects in the image. (considering also the defects which have 8 connectivity with background.)
2) As you see there are 8 defects in the image. I have drawn borders around them in the left image and corrected them in the right image. 2 of them are very close, I have drawn a red arrow to show them, these 2 points and another point which I have shown with a green arrow have 8 connectivity with background.

I have used the simple 3*3 isolated edge point detector filter to find the defects, as the defects are single points, but if the defects are bigger, we can use shrinking algorithm and then find the number of closed areas in the final output..

c)

There are 55 rice grains in the image. You can see the step by step procedure below:

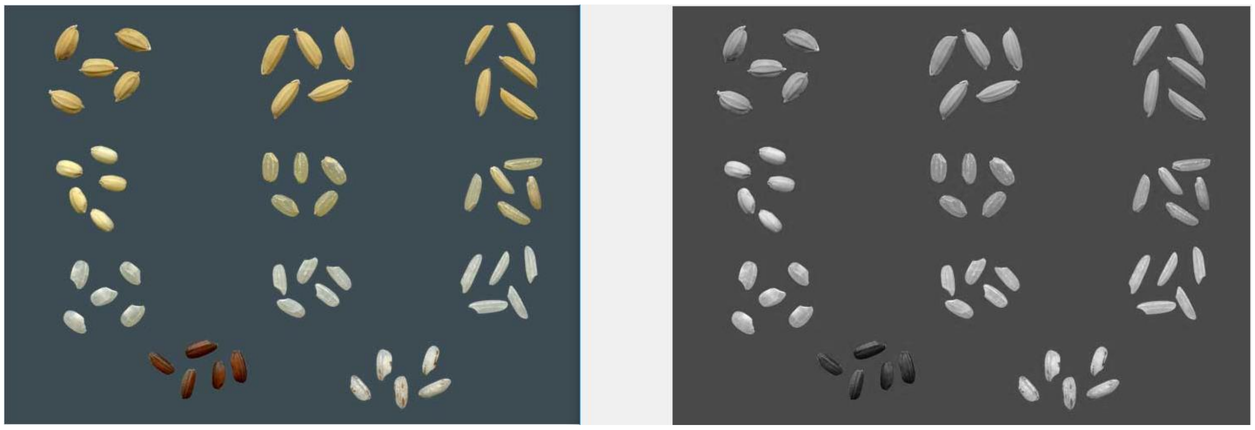


Fig7: the input image and the gray-scale version of it

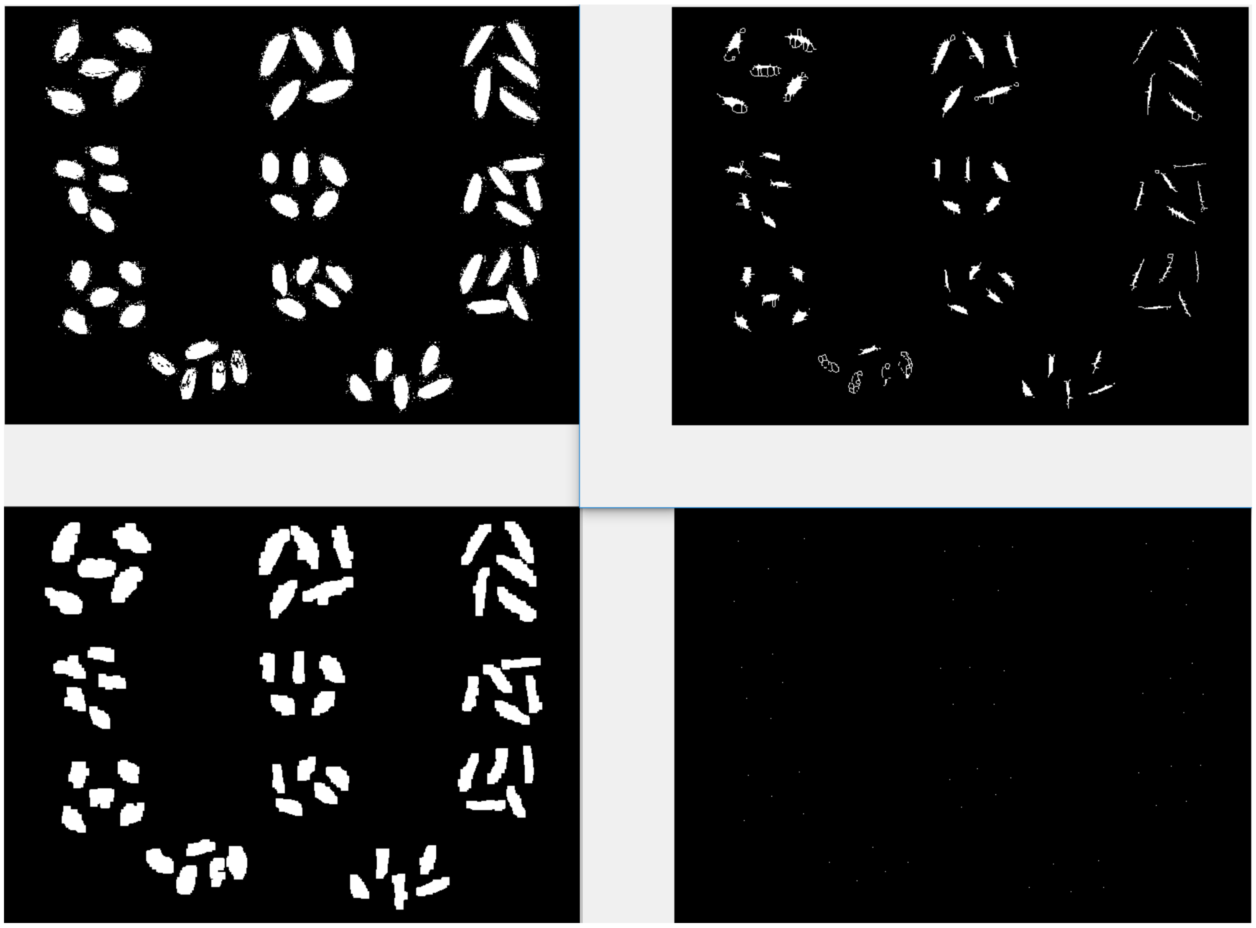


Fig8: the image after turning it to binary image by setting pixels with values between 68 and 81 to zero and others to one (upper left) the binary image after 7 iterations of shrinking (upper right) after isolated point removal plus enlarging each pixel to fill the holes (lower left) the result after shrinking the image to reach 55 dots (lower right)

First, I turned the color image to gray-scale image and for binarizing it observed that background pixels can be mainly removed by setting pixels with values between 68 and 81 to zero and others to one. But this caused some holes inside the grains, so I used a 3*3 interior fill filter to remove some holes and then applied seven iterations of shrinking to it to make other white distortions small and increase the distance between rice grains. Then I used one iteration of isolated edge point removal, enlarged the rice pixels to fill the remaining holes inside rice grains, and finally, used shrinking to reach the number of rice grains.

To compare the size of the grains I continued to shrink the output of the upper right image of Fig8. The sooner a grain becomes a pixel, the smaller it is before shrinking. You can see the result bellow:

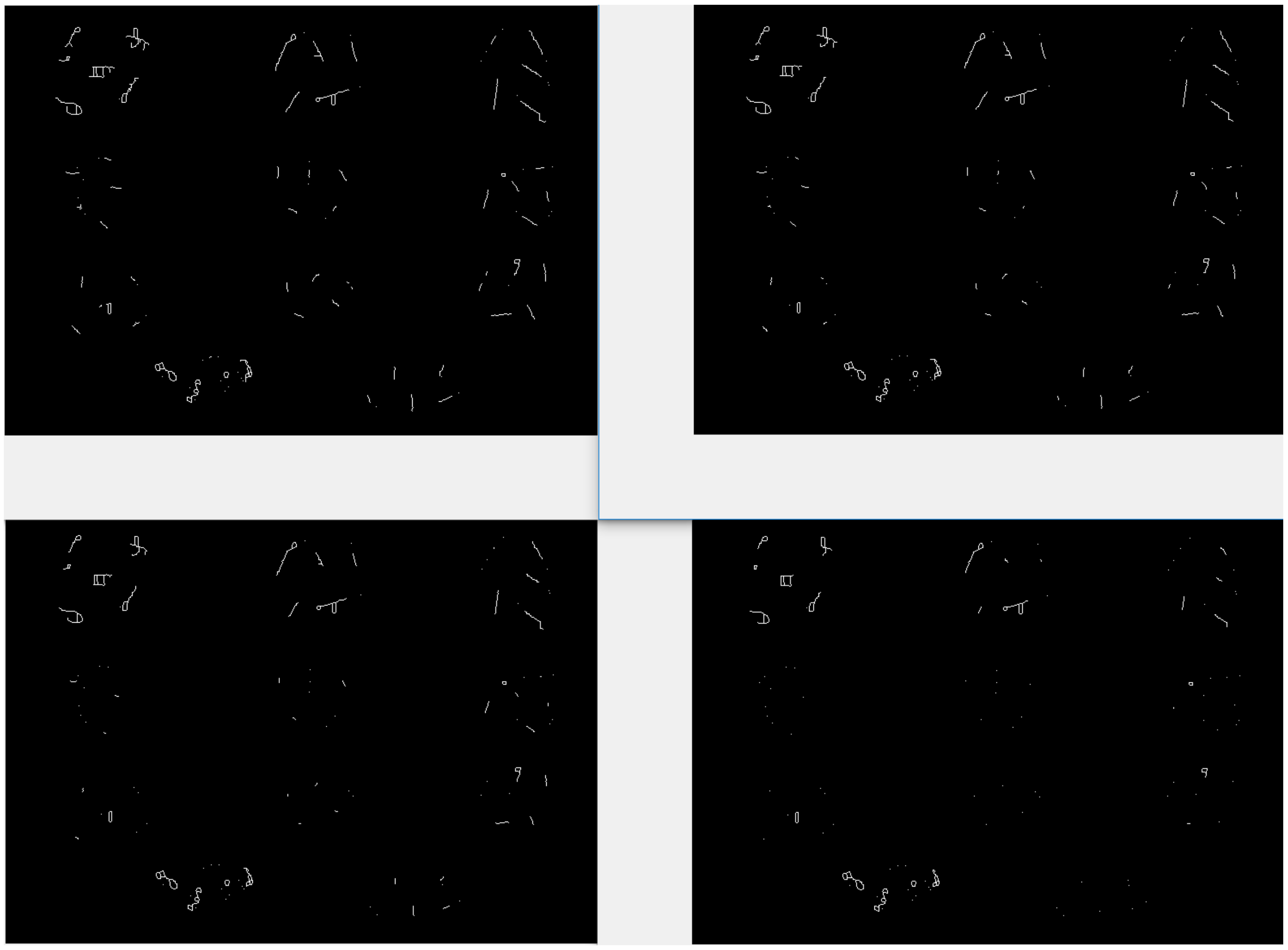


Fig9: The result of several iteration of shrinking (upper left -> upper right -> lower left -> lower right respectively)

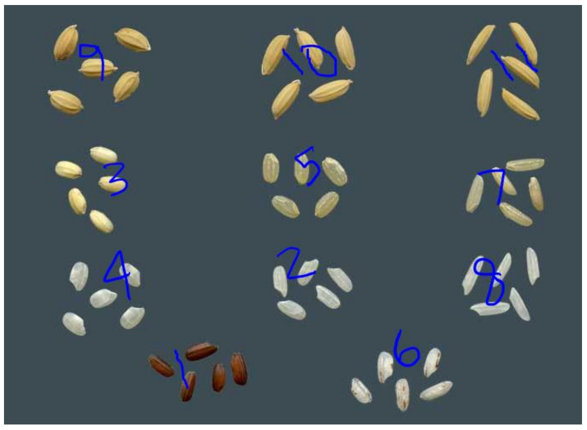


Fig10: the ranking of grain's sizes from small (1) to large (11) according to results of shrinking

Discussion

Part c was a challenging part, as there are distortions around rice grains and without using professional tools of MATLAB it wasn’t easy to remove them. For ranking their size, in a perfect binary image shrinking is a great tool, as it removes one pixel of boundary of each white grain at each iteration. But the current result may not be 100% correct as there are some holes in each grain which change the speed of shrinking for each grain. Foe example a line with no holes shrinks from both sides in an iteration while a line with a hole in one side only shrinks from its other side at each iteration.

For this reason, I only noticed a rice grain of each type with the least holes, but some types of the rice (for example the upper left type) there is no such defect-less grain, so the ranking result of such types may be incorrect.