EE569 Introduction to Digital Image Processing

Homework Report #3

Name: Boyang Xiao **USC ID:** 3326730274 **Email:** boyangxi@usc.edu

I. Problem 1: Geometric Image Modification

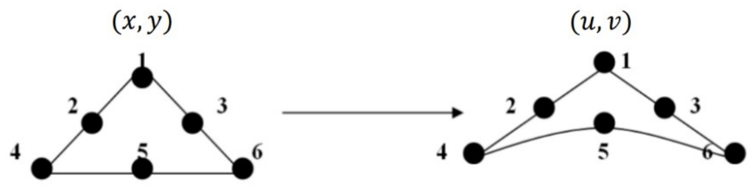
1.1. Abstract and motivation

Geometric image modification is a kind of technique which modifies the shape and position of the image spatially. It provides the methods to reposition pixels within the image, relocating the pixels from their original coordinates to new coordinates to obtain the new image. Based on geometric image modification, many higher-level techniques can be implemented, such as features extraction, image augmentation and also image encoding.

Basic manipulation methods of geometric image modification include image translation, rotation, scaling and affine transformation. And with these basic manipulation methods, some advanced manipulation operations can be achieved, such as Image warping and 3D object warping. In this part of homework, one operation of image warping will be introduced and implemented.

1.2. Approaches and procedures

The goal of this problem is to convert the provided images into star-shaped images shown in the homework requirements and then convert the images back to the normal shape. The warping operation is based on an assumed model shown in Figure 1.1 below.



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

Figure 1.1: Image warping coefficient matrix model

In Figure1.1, (x,y) stands for the pixel coordinates on the original image and (u,v) stands for the corresponding pixel coordinates on the warped image. To do the image warping, a coefficient matrix is multiplied to the matrix made up with (x,y) coordinates. There are 12 unknown coefficients in the coefficient matrix and 6 pairs of control points have to be picked to calculate the coefficient matrix. The picked control points are also shown in the Figure 1.1. After the coefficient matrix is calculated, it can be applied to every pixel in the image to calculate their coordinates after the warping. And the output image can be rendered according to the coordinate mapping relation.

Since the warping patterns for the four sides of the original image rectangle are similar to each other except for the warping direction, the original image is segmented into 4 regions as shown in Figure 1.2 below. The coefficient matrices for each region are calculated individually from the control points picked from each region.

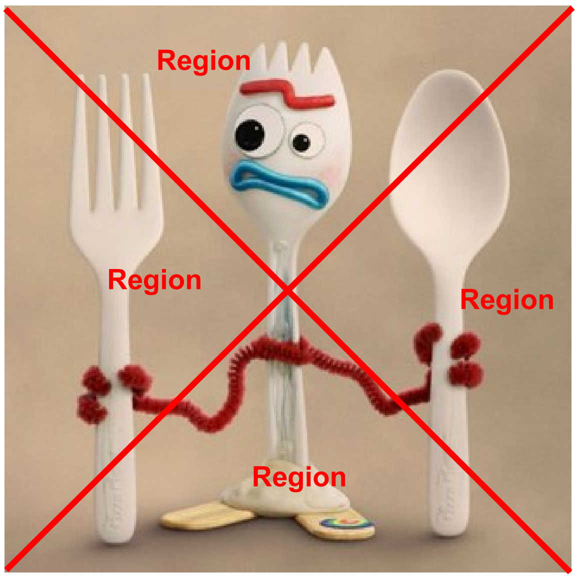


Figure 1.2 Segmentation of the image for warping according to each region

Therefore, the specific steps to do the warping algorithm **from a normal shape image to a star-shaped image** will be:

1) Segment the whole original image into four regions as shown in Figure1.2 and pick the control points for each region as shown in Figure 1.1.

2) Calculate the coefficient matrices for each region and apply the different coefficient matrices respectively to all the pixels in the four regions. A new matrix (let's call it the ***position matrix***) should be obtained, which records the coordinates mapping relation from the original image to the warped image.

3) Every single coordinate pair (***coordinate 1***) saved in the *position matrix* are the pixel coordinates after the warping, and the index for this coordinates pair in the *position matrix* is this pixel's original position (***coordinate 2***) in the original image. Therefore, to render the warped image, for every pixel that is located at the coordinate 1 should be filled with the gray-scale value of the pixel located at the coordinate 2 in the original image. After traversing the whole position matrix and the whole warped image should be obtained.

To do the **reverse spatial warping**, that is to warp an image from the star-shape back to the normal shape, the warping steps will be as following:

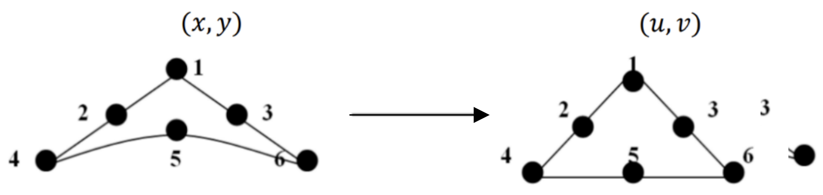


Figure 1.3 Control points selection in reverse warping operations.

1) Segment the whole original image into four regions as shown in Figure1.2. The control points selection should be inverse, as shown in Figure 1.3 below. The input points should be picked from the star-shaped image and the output points should be selected from the normal-shaped image.
2) The coefficient matrices are calculated for each region of the image. And then apply the coefficient matrices to all the pixel coordinates in the same regions respectively. A position matrix holding all the coordinates after the warping is obtained.
3) Since the reverse warping makes the points looser, if simply rendering the output image like what we did before, there will be many black dots gaps lying on the output image. Therefore, the bilinear interpolation should be applied to fill the gaps. To elaborate, if the output coordinate lies between the integer input coordinates, the output pixel gray-scale value should be calculated with bilinear interpolation from the four pixels on the four corners on the input image. After traversing all the pixels on the output image, the reversed image is obtained.

1.3. Experiment results

The warped star-shaped images are shown in Figure 1.4. below:

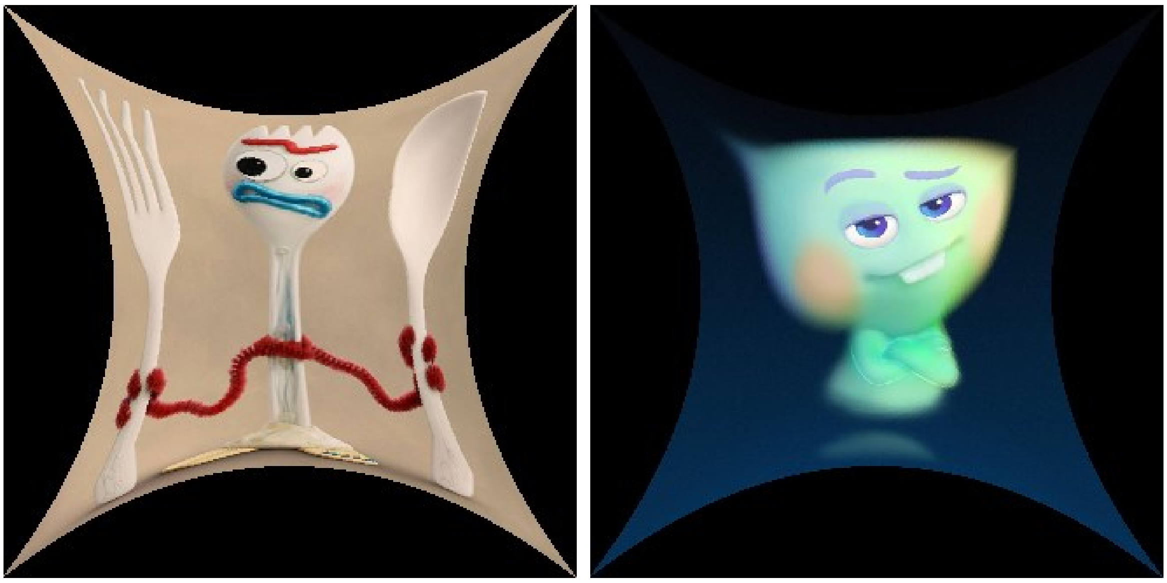


Figure 1.4 Warped star-shaped images.

The reverse warped images are shown in Figure 1.5 below:

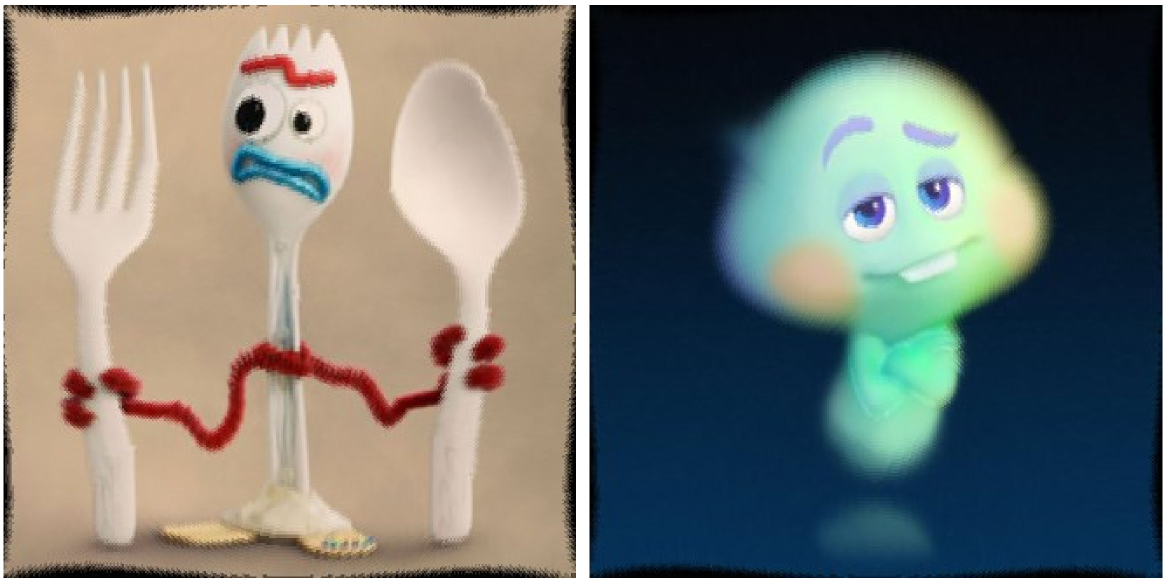


Figure 1.5 Recovered images using reverse warping.

1.4. Discussions

Comparing to the original un-warped images, the recovered images have several kinds of distortions:

1) The recovered images are bent and twisted, especially in some areas that have straight shapes (such as the fork and the spoon in the Forky.raw image).
2) The recovered images have dark areas unfilled on the boundaries. The dark areas on the four sides are of the same pattern.
3) The recovered images are blurred.

The ***reasons*** for these distortions can be:

1) Since the model we choose is not optimized and the model is not linear, either, when applying the forwarding coefficient matrix first and then applying the reverse coefficient matrix, the results of this operation cannot be strictly the same as the original coordinates. As a result, the recovered images are bent and twisted because the pixels are dislocated.
2) As we know from above, the pixels can be dislocated since the model problems, the impact on the pixels lying on the boundaries can be the most. Some dark pixels can be relocated on the output image's boundary areas after the reverse warping operation, which causes the dark areas on the recovered images.

3) When we fill in the blank gaps on the recovered images, we use the bilinear interpolation. So many pixels are actually predicted from the neighboring pixels and calculated from the averaging, but not directly obtained from the original pixel value. This step can make the output images to be blurred.

II. Problem 2: Homographic Transformation and Image Stitching

2.1. Abstract and motivation

Part 1 introduces the basic spatial image modification and one of its application, the image warping technique. In this part, another advanced application based on image modification, the Image stitching technique, will be introduced and implemented. The image stitching technique is widely used in the field of Computer Vision, such as panorama applications.

The goal of this problem is to use the image stitching technique to stitch three individual pictures taken from the same place and to form a whole panorama picture.

2.2. Approaches and procedures

The Image stitching technique is based on the basic image modification methods, especially the affine translation. One single affine translation operation has 8 degrees of freedom and 4 control points are therefore needed to be selected to calculate the translation matrix. When one of the image is compared to another image, many points can be found similar. We use this points as the control points to work out the translation matrix and apply the translation matrix to one of the image, and the translated image will be joined to another image. By operate on pairs of images one by one, the whole panorama will be generated.

The specific steps to do this algorithm will be as following:

1) Convert images into gray-scale images and use detectSURFFeatures() function in MATLAB Computer Vision Toolbox to extract feature points in each gray-scale image.

2) Match the feature points between each two images in order and find at least four matching points between each pair of images.

3) Use the control points to calculate the affine translation matrices and multiply the current translation matrix with the matrix multiplication before. If we want to put the middle image in the central place, we can also multiply each affine translation matrices with the inverse of the middle images' translation matrix.

4) Create a big canvas that is enough to hold the whole panorama and put render the translated images one by one. The images are firstly put at the center of the canvas and then uses the translation matrix to calculated the pixels new coordinates after the translation and render the target pixels with the pixel gray scale value from the original pixels (in RGB three channels). After the rendering procedure, a whole panorama will be generated on the canvas.

2.3. Experiments results

The control points selected from the Left image and the Middle image are as shown in Figure 2.1 below. 22 control points are selected to calculate the translation matrix using MATLAB function: ***estimateGeometricTransform2D(…)***.

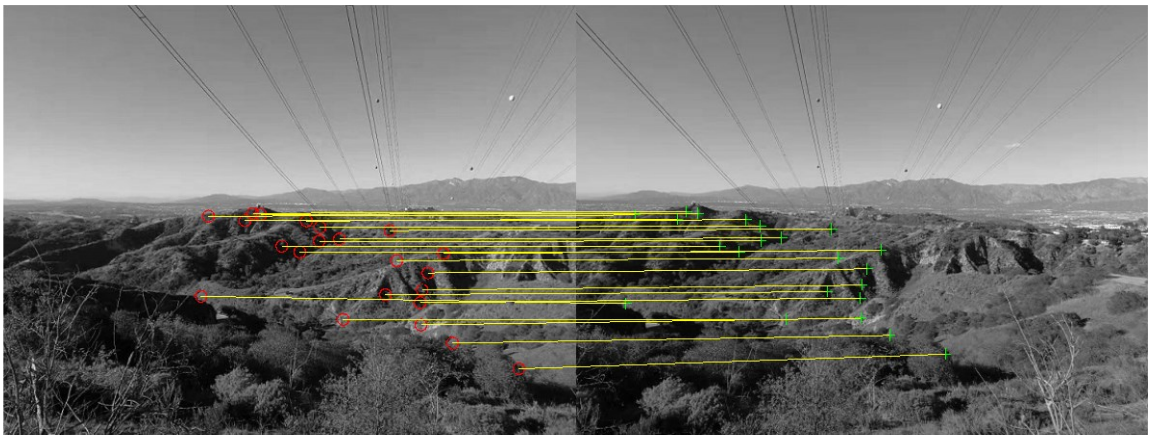


Figure 2.1 Control points selected between the Left image (left) and .the Middle image (right)

The control points selected from the Middle image and the Right image are as shown in Figure 2.2 below. 30 control points are selected to calculate the translation matrix using MATLAB function: ***estimateGeometricTransform2D(…)***.

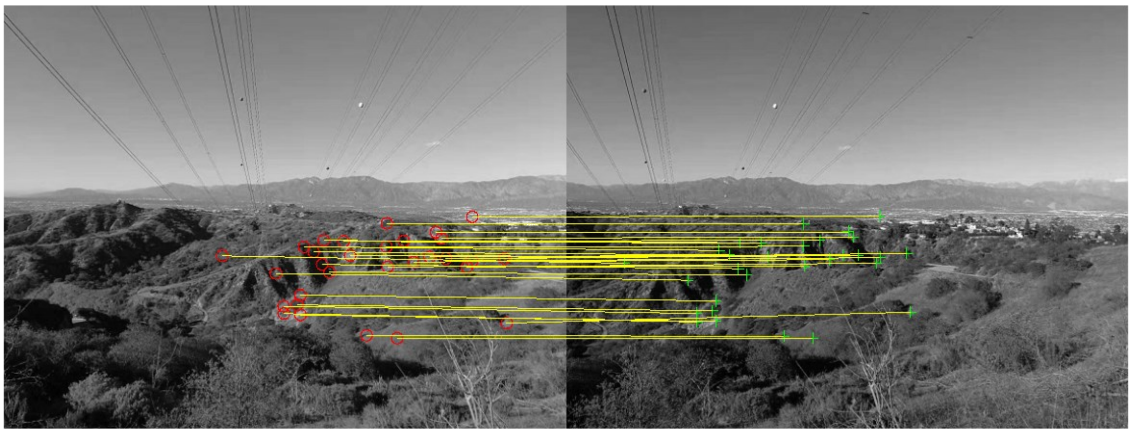


Figure 2.2 Control points selected between the Middle image (left) and .the Right image (right)

The image stitching results are shown in the Figure 2.3 below on the next page. From the resulting panorama image, we can find that the three images are stitched together and most parts of the images can match well. Except for some parts that cannot match well because of the angle and position at which the pictures are taken.

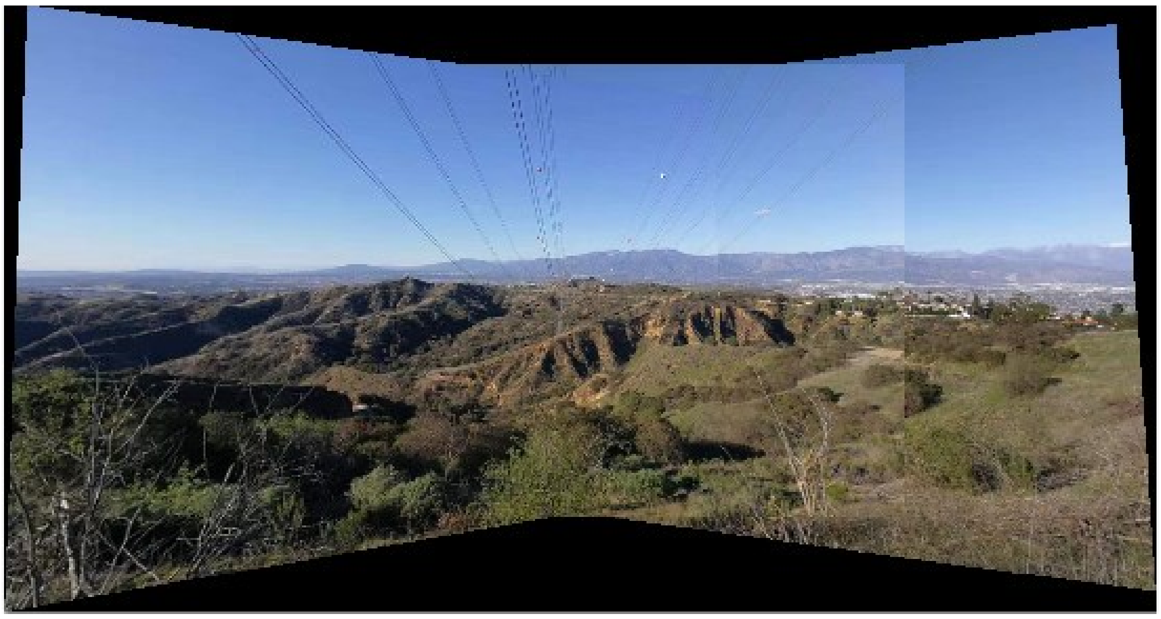


Figure 2.3 Image stitching results: The panorama

2.4. Discussions

Control points numbers:

22 control points are selected from the Left image and the Middle image, and 30 control points are selected from the Middle image and the Right image. The visualization figures are shown in Figure 2.1 and Figure 2.2.

SURF and matching points selection:

To pick the matching points, the features of the three images are firstly extracted by **detectSURFFeatures(…)** from MATLAB toolboxes. When features and their indexes are extracted, they are compared and matched using the **matchFeatures(…)** function. With different MaxRatio parameters, different matched pairs of feature points indexes are marked as the control points. And then the matching points are used to calculate the affine translation matrices using function **estimateGeometricTransform2D(…)**. The proceeding processes are elaborated in section 2.2.

III. Problem 3: Morphological Processing

3.1 Part (a): Basic morphological process implementation

3.1.1 Abstract and motivation

Morphology is a study of object shapes, and morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. When inputting a binary image, the morphological image processing will conduct multiple operations on the image such as shrinking, thinning and skeletoning etc. to serve some higher-level processing techniques.

In this part of problem, the ***Thinning*** processing algorithm will be introduced and implemented.

3.1.2 Approaches and procedures

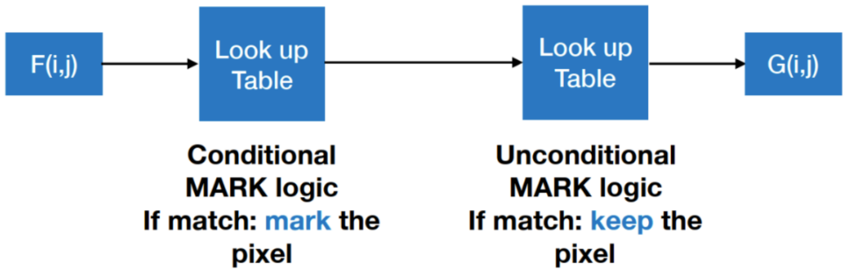

F(i,j)
Look up Table
Look up Table
G(i,j)
Conditional MARK logic If match: mark the pixel
Unconditional MARK logic If match: keep the pixel

Figure 3.1 Basic morphological processing flowchart

Following the Basic morphological processing flowchart in Figure 3.1 above, there are **four steps** in the thinning processing algorithm:

1) Binarize the input image and add a zero padding to the four boundaries of the image.
2) Use a 3*3 window to slide on the image and to match the conditional mark patterns. If there is a match, mark the central pixel of the window as “matched”.
3) Combine the matching mark and the original images. Use a 3*3 window to slide on the combined image and to compare the combined image to the unconditional patterns. If the sliding window matches any of the unconditional patterns, the central pixel of the window will be preserved in the original image, otherwise, the central pixel of the window will be set to 0 in the original image.
4) Repeat Step 1 - 3 until no more pixels are changed.

3.1.3 Experimental Results

The results and intermediate results of thinning processing are shown in Figure 3.2 below:

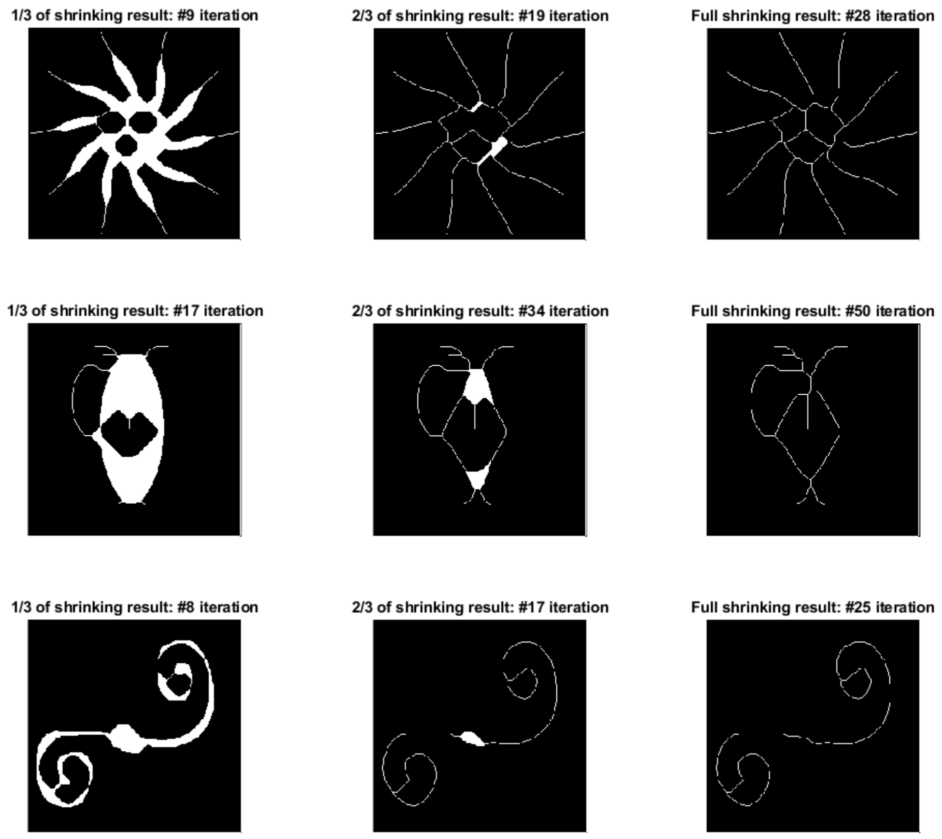

1/3 of shrinking result: #9 iteration
2/3 of shrinking result: #19 iteration
Full shrinking result: #28 iteration
1/3 of shrinking result: #17 iteration
2/3 of shrinking result: #34 iteration
Full shrinking result: #50 iteration
1/3 of shrinking result: #8 iteration
2/3 of shrinking result: #17 iteration
Full shrinking result: #25 iteration

Figure 3.2 Thinning results and intermediate results

The intermediate results are selected from the 1/3 of the whole iterations and 2/3 of the whole iterations for the object image. Also the final thinning results and the whole iteration numbers are also shown in the Figure 3.2.

From the figure, we can find that the lines of the input images are getting thinner after thinning iterations. Because the thinning processing will mark all the pixels that lie out of the line area and remove these pixels from the image. When the morphological processing cannot change any pixel in the image, the processing procedure has arrive the final status, and all the lines in the image are one-pixel wide. That is what we expect from the thinning processing.

3.1.4 Discussions

All the results and question answers are shown in section 3.1.3 above.

3.2 Part (b): Defect detection and counting

3.3.1 Abstract and discussion

An iconic application based on basic image morphological processing is Defect detection and counting. In the images, there are usually some blocks of whole patterns. However, if there some blank dots or areas, which are defined as the defects of the images, they have to be detected and removed. If detect these defects directly using the original images, it will be much more tricky, since the defects usually have different sizes and different shapes. The assumption that which areas are defects are difficult to make. But if the Images are being processed by the morphological processing and all the blank areas are shrinking to a single dot, it will be much easier to design the proceeding algorithms,

In this part of problem, Defect detection and counting method will be implemented based on morphological shrinking processing.

3.3.2 Approaches and procedures

- **Defect detection and counting**

Since in the provided image, the object is all white and the defects are all black areas, the image should be inversed first for convenience. After the inversion, the defects become white and the other parts of the object become black. Then, the shrinking processing are applied to the whole image. After iterations of shrinking processing until no more pixels are changed, all of the white areas should be converted into a single dot, where the dot is the center of the original defect areas. If we count all the white dots' number, the result should be the number of defects from the original image.

One thing that should be noticed is that although the defects areas will be defected in this way, the other areas that are not defects but are also closed areas will be detected too. Therefore, when counting the shrinking dots, we should also ***detect the size of the blank areas*** centered around these dots in the original image.

To achieve this target, my approach is to sample the areas that centered around the present shrinking dot in the original image, and then to count the black dots in this area. Because the homework requirements say that the black holes smaller than 50 pixels are defined as defects, the real defects that meet the requirements will be selected from all the shrinking dots mentioned above. Also, different sizes of defects can also be classified in this way.

- **Defect clearing**

After obtaining the defects' positions and defect's sizes, the very next step is to clear the defects. Since the object pattern in the image is all white and the defects are all black, my approach is to mark the whole area around the defects' central position and render the area all white. In this way, the defects areas are converting to white.

3.3.3 Experiments results

The shrinking procedure for the inversed original image is shown in Figure 3.3 below. The intermediate results are selected from the 20th iteration, the 50th iteration and the final iteration. You can find that with iterations going on, the white areas are becoming more and more smaller and in the final result, there are only 10 white dots in the whole image.

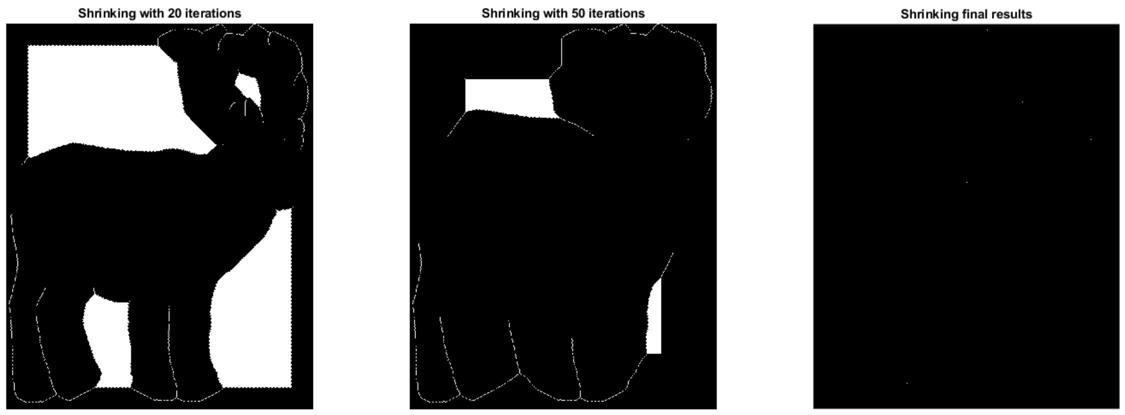


Figure 3.3 The shrinking procedure

After obtaining the shrinking image, I use the algorithm mentioned in section 3.2.2 to detect the real defect points and remove the fake defect points. The defect points sizes and their corresponding numbers are shown in Figure 3.4. From the histogram in Figure 3.4, there are 5 defects of 1-pixel size and there are 1 defect of 39-pixel size. The total number of defects are 6.

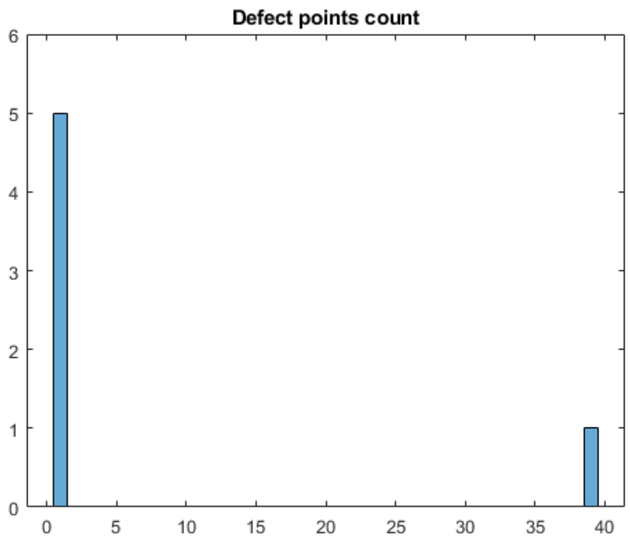


Figure 3.4 Defects sizes and numbers

And then the defects are filled and the clear image are shown in Figure 3.5 below.

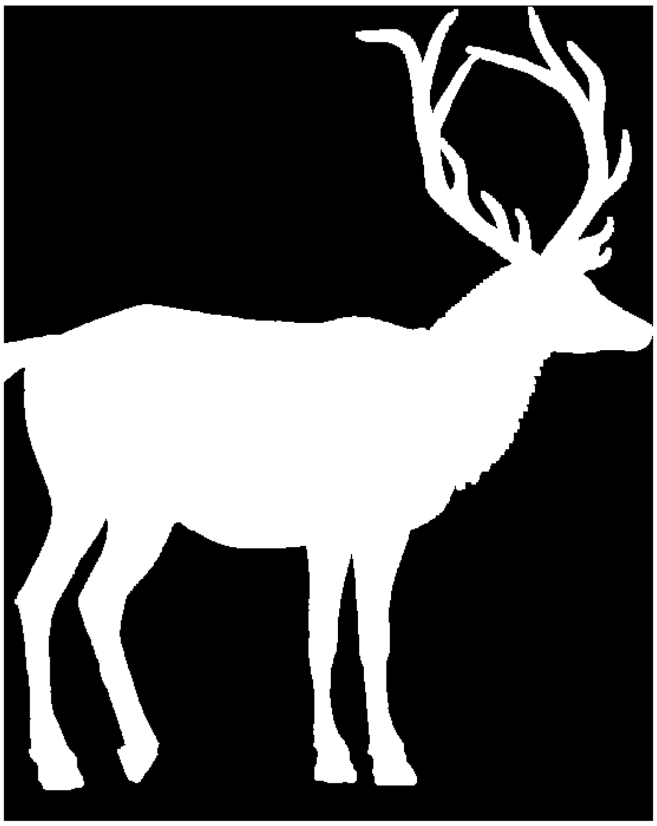


Figure 3.5 The cleared image.

3.3.4 Discussions

All the questions are answered above and all the results are also shown above.

3.3 Part (c): Object segmentation and Analysis

3.3.1 Abstract and motivation

In addition to the defect detection, another iconic application based on the basic morphological image processing is image segmentation and object analysis. For objects that have different sizes and patterns, the morphological processing algorithms can convert them into the same sizes dots and make it much easier to count and analyze.

In this part of problem, we will use the image morphological algorithm to count different beans number, segmentate them and sort them in the order of sizes.

3.3.2 Approaches and procedures

To count the number of beans, there are three steps to do this process:

1) Convert the beans image into gray-scale image, and then binarize the image. For convenience, the binary image is then inversed to make the beans' areas all white and the background all black.
2) Conduct the shrinking processing until no more pixels are changed, and all the beans will be single dots.
3) Go over all the pixels in the shrinking image and count the white dots. The number is the number of the beans in the original image.

After knowing how many beans in the image and their locations, we can segment the whole image into several individual images that only contain a single bean in the image. To sort the sizes of the images, we can conduct the shrinking processing again and count how many iterations it goes through to achieve the final status(no more pixels are changed). The more iterations it goes through, the biger the bean is .In this way, we can sort the sizes of beans by sorting the iteration loops.

3.3.3 Experiments results

The results of shrinking processing for the whole image is shown in Figure 3.6 below. We can observe that there are 5 white dots in the image, so there are 5 beans in the original image. However, this counting procedure is also implemented in the program.

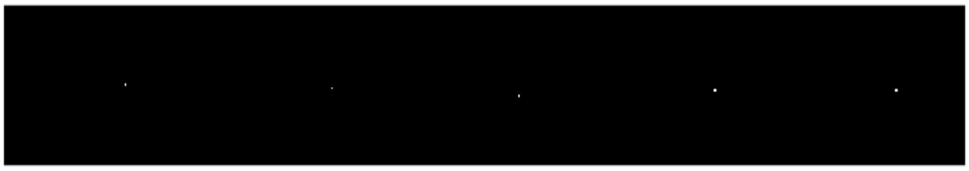


Figure 3.6 The shrinking processing final results for the whole beans image.

Figure 3.7 below shows the results of shrinking processing conduct to each segmented bean image and their intermediate results. 1/3, 2/3 and the final stage of the whole iterations are picked to displayed in the figure.

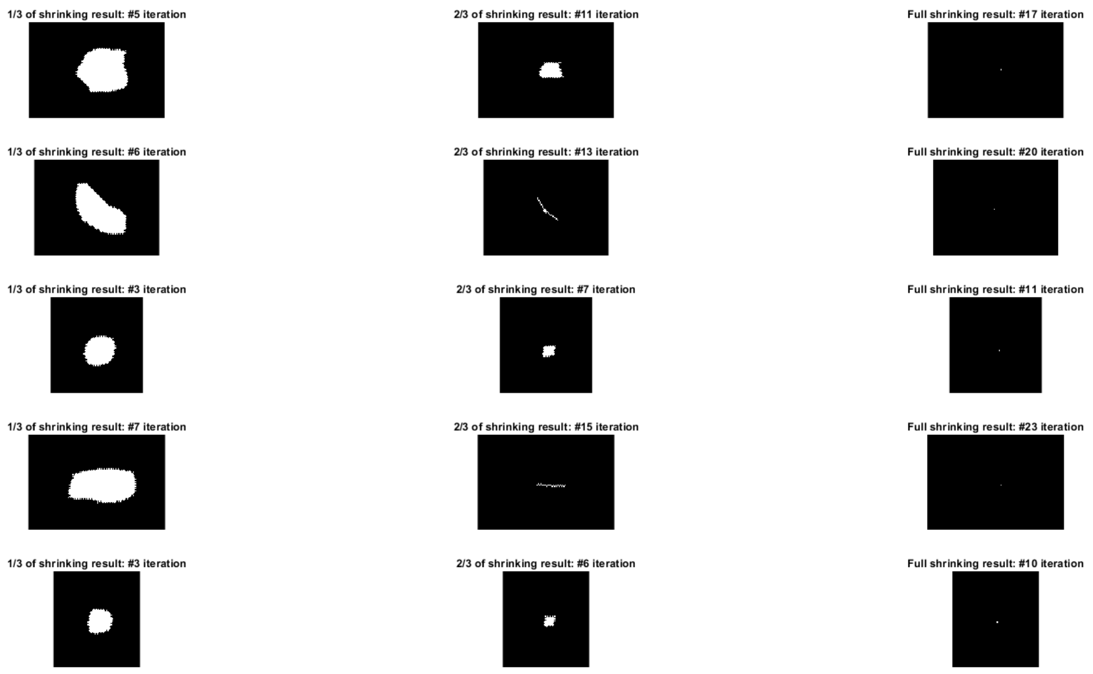

1/3 of shrinking result: #5 iteration
2/3 of shrinking result: #11 iteration
Full shrinking result: #17 iteration
1/3 of shrinking result: #6 iteration
2/3 of shrinking result: #13 iteration
Full shrinking result: #20 iteration
1/3 of shrinking result: #3 iteration
2/3 of shrinking result: #7 iteration
Full shrinking result: #11 iteration
1/3 of shrinking result: #7 iteration
2/3 of shrinking result: #15 iteration
Full shrinking result: #23 iteration
1/3 of shrinking result: #3 iteration
2/3 of shrinking result: #6 iteration
Full shrinking result: #10 iteration

Figure 3.7 The results of shrinking processing conduct to each segmented bean image and their intermediate results

The iterations numbers for each kind of beans are plotted below in figure 3.8. From the Figure, we can find that the size order for these 5 beans are:

④ >② > ①> ③ >⑤.

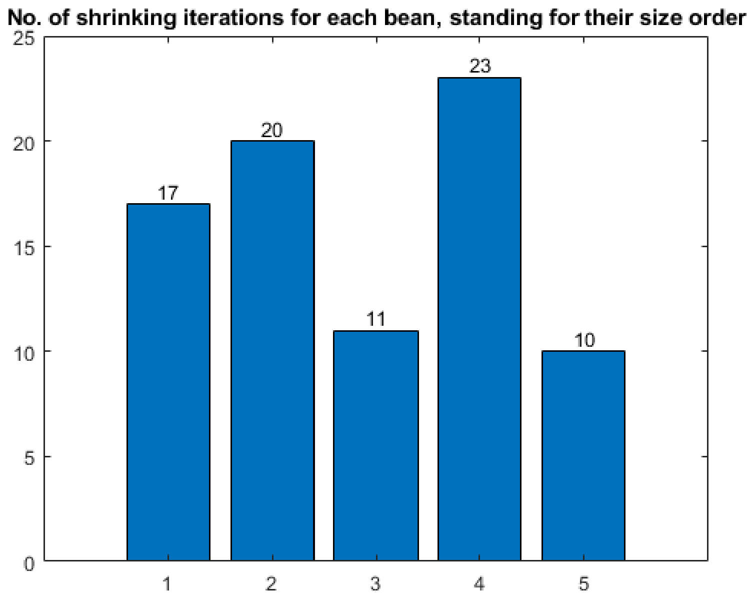


Figure 3.7 Shrinking iterations number for each kind of beans.

3.3.4 Discussions

All the questions are answered above and all the results are also shown above.