EE569 Digital Image Processing

HOMEWORK #3

Zheng Wen
zwen1423@usc.edu

Problem 1: Geometric Image Modification

(a) Geometric Warping

Geometric warping is a kind of operation on coordinates of each pixel to create a new look of the original image. In this problem, the original square image is projected into a circle. There are three ways to accommodate a square image with a circle, squeezing the square image horizontally, vertically and along with diagonal, respectively, and the output by different procedures vary. From the given example, for the ears of the panda is squeezed together, so the horizontal squeezing, also called Elliptical Grid Mapping[1], should be implemented, the demonstration of this projection is shown in Fig. 1[1].

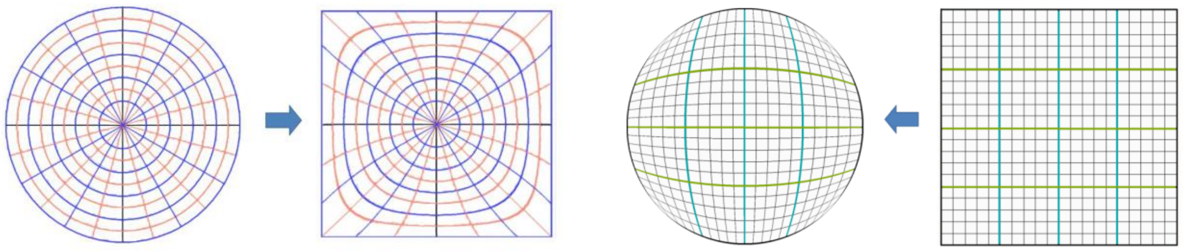

(a) From Circle to Square (b) From Square to Circle

Fig. 1 Demonstration of Elliptical Grid Mapping[1]

(1) In order to project the square image into a round one, the reverse mapping should be used, or there will be some missing points in the result image. Suppose x, y stand for the horizontal coordinates and vertical coordinates in square image, respectively, and u, v stand for the horizontal coordinates and vertical coordinates in circle image, respectively, their relationship[1] is shown below,

$$\begin{cases} x = \frac{1}{2}\sqrt{2 + u^2 - v^2 + 2\sqrt{2}u} - \frac{1}{2}\sqrt{2 + u^2 - v^2 - 2\sqrt{2}u} \\ y = \frac{1}{2}\sqrt{2 - u^2 + v^2 + 2\sqrt{2}v} - \frac{1}{2}\sqrt{2 - u^2 + v^2 - 2\sqrt{2}v} \end{cases}$$

In my approach, the image is firstly divided into four parts horizontally and vertically through the center, then each of them is flipped to the lower right part, so that the coordinates of 2-dimension array could be made use of, the flipping procedure is shown in Fig. 2. Then in the lower right part, the pixels (u, v) is used to find the corresponding position (x’, y’) in the original square image, if (x’, y’) lies in the square image, then the value of pixel is obtained by bilinear interpolation given in lecture 2 and shown below, using the pixel value in (x, y), (x + 1, y), (x, y + 1) and (x + 1, y + 1), if (x, y) lies in 2nd flipped image while (x’, y’) does not, then the pixel of (u, v) is obtained by (x, y) directly, otherwise, the pixel value is set to 0. where x, y is the integral part of x’, y’, respectively. After getting the mapped image of square image, each part is flipped back by the reverse flip procedure of Fig. 2, shown in Fig. 3, and then assigned to the right place to get the final result, shown in Fig. 4.

$$\begin{aligned} F(x', y') = F(x + \Delta x, y + \Delta y) & \\ &= (1 - \Delta x)(1 - \Delta y)F(x, y) + \Delta x(1 - \Delta y)F(x, y + 1) \\ &+ (1 - \Delta x)\Delta y F(x + 1, y) + \Delta x \Delta y F(x + 1, y + 1) \end{aligned}$$

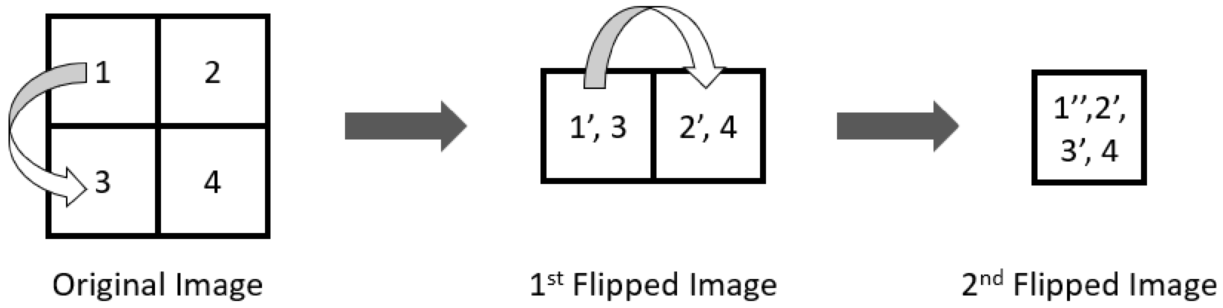


Fig 2. Flip Procedure

In the Original image, the upper left, upper right, lower left and lower right part of image is labeled as 1, 2, 3 and 4 respectively, the image is firstly flipped along horizontal direction, then the 1st flipped image is obtained, where the left part shows the position of 1' and 3, the right part shows the position of 2' and 4, then they are flipped vertically to get the 2nd flipped image, which shows the position of 1'', 2', 3' and 4.

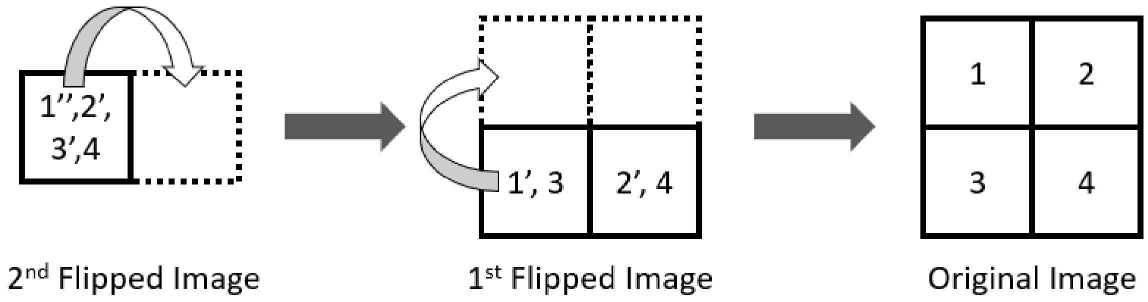
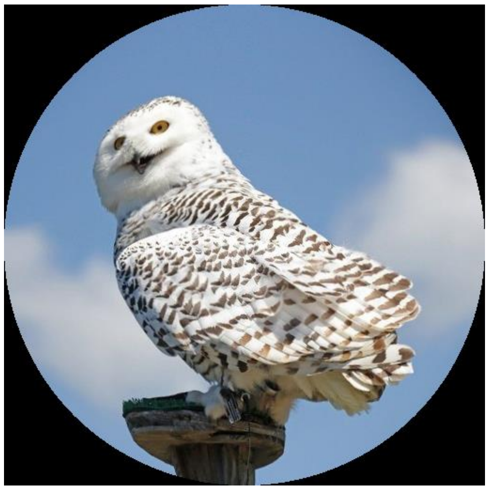


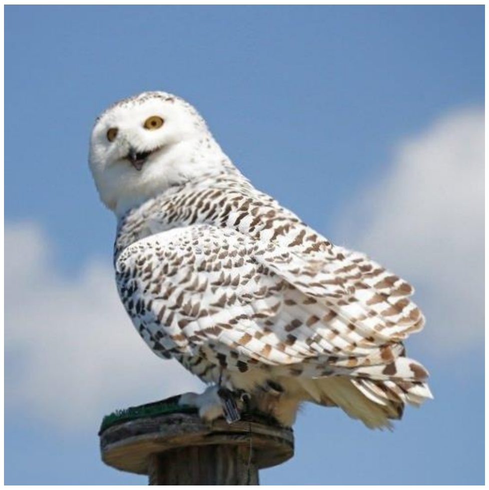
Fig 3. Reverse Flip Procedure

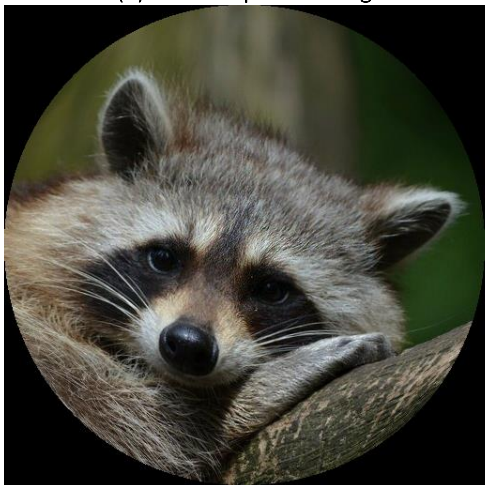
(2) To map the warped image back to original image, the warped image is also divided in four part and conquered to get the original image, and the lower right part is used to get the coordinates as well, the only difference is the transformation function. In this case, the coordinate in square image (x, y) is used to calculate the corresponding coordinate in the warped image (u', v') using the following formula[1].

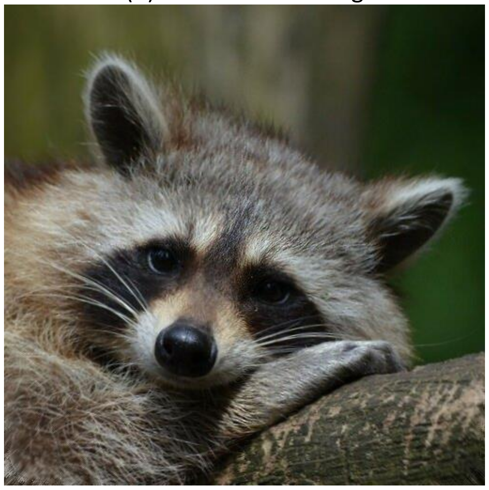
$$\begin{cases} u = x\sqrt{1 - \dfrac{y^2}{2}} \\ v = y\sqrt{1 - \dfrac{x^2}{2}} \end{cases}$$

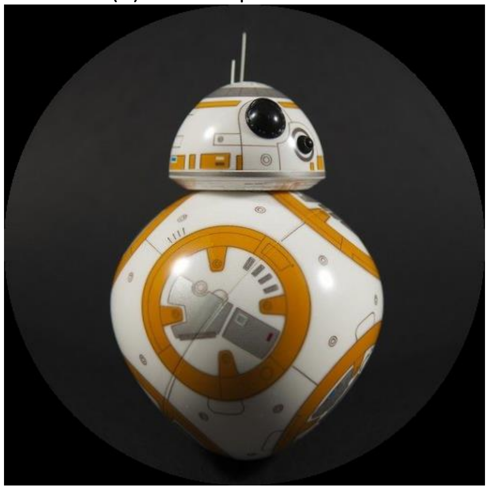
For the circle has a certain radius equal to the side length of the 2nd flipped image, so if $u'^2 + v'^2 \leq r^2$, the corresponding pixel value in (x, y) is obtained by bilinear interpolation shown above, else, because the pixel value in points (u', v') with $u'^2 + v'^2 > r^2$ is zero as illustrated above, so this the pixel value of the corresponding (x, y) is obtained by the point which corresponding to the integral part of (u’, v’), so the recovered image does not contain black pixels caused by the loss of edge information. The final results are shown in Fig. 4.


(a) Disk-Shaped Hedwig


(d) Recovered Hedwig


(b) Disk-Shaped Raccoon


(e) Recovered Raccoon


(c) Disk-Shaped bb8

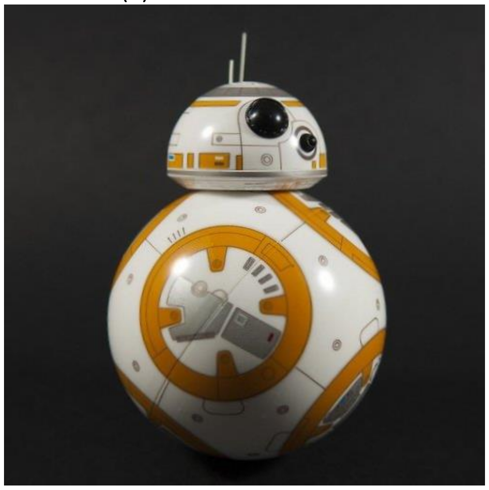

(f) Recovered bb8

Fig 4. Disk-Shaped Image (a, b, c) and Recovered Image (d, e, f)

(3) First of all, the recovered image is a little blurrier in most part than the original image and the edge of the recovered image is less sharper than that of the original image because of the pixel values in the recovered image are obtained by bilinear interpolation, while the pixel value in the original image are not necessarily distributed linearly.

Another difference between original image and recovered image could be seen clearly only around the corner of recovered Raccoon, because only this image has different pixel values around the corner of the image while the other two images are kind of pure color there. The corner of the recovered image is blurrier than that of the original image, the further from the center of the recovered image the pixel lies, the blurrier the region becomes. This kind of difference is caused by the principle of Elliptical Grid Mapping. From Fig. 1(a), the line in square shows the source of the distortion, for the counter lines in the square is not distributed evenly, the distance between two counter lines is further around the corner than that along side the edge far away from the corner. So, the pixel is stretched far away from each other around the corner, which caused the blurry pattern.

There are some ray-like patterns around the corner recovered image, this is caused by the loss of surrounding information because the squeezing transformation. When calculate the corresponding (u’, v’) of each (x, y), there are few points could not find enough information to implement bilinear interpolation for their next points lies outside the circle region with image in it. For these points, I use the integral part of (u’, v’) to find pixel values, or there will be some black rays which makes the image corner looks strange. A finer look at the recovered one with fix, the recovered one without fix and the original one is shown in Fig. 5, the result in part (2) are obtained with fix.

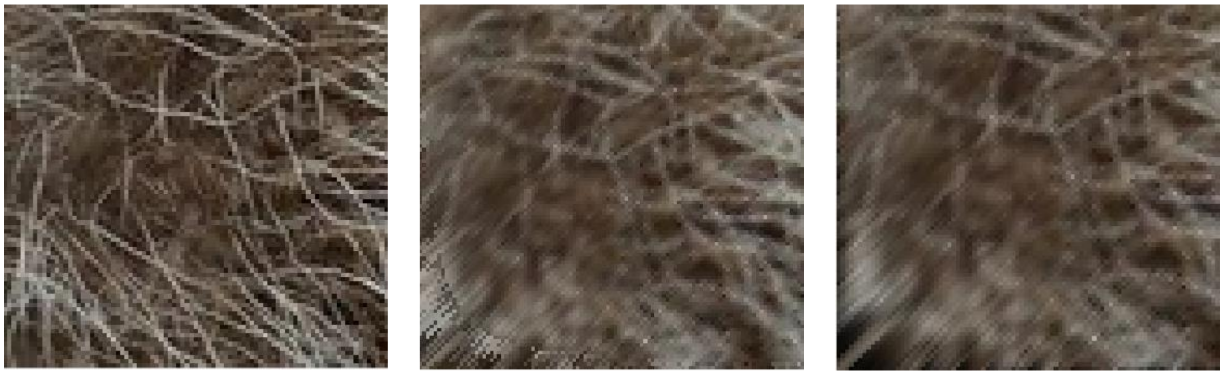

(a) Original Raccoon (b) Recovered Raccoon with Fix (c) Recovered Raccoon without Fix

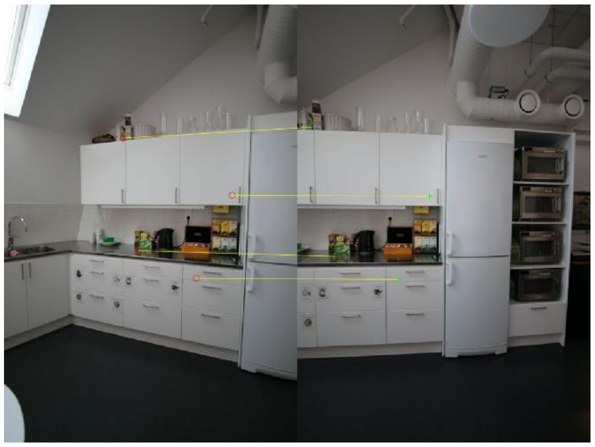
Fig. 5 A Finer Look at Original Image and Recovered Image

(b) Homographic Transformation and Image Stitching

Homographic transformation is a kind of transformation between two planes, it could align lines with different slots in the two planes by changing a quadrilateral in one plane into the shape of the corresponding quadrilateral in another plane. With image stitching technique, the image of shot by different angles could be aligned, then the panorama could be created.

(1) As shown in the homework 3 tutorial, 4 pairs of corresponding feature points are selected to estimate the 3×3 homographic transformation matrix, which means to match left.raw and middle.raw, 4 control points on left.raw and 4 control points on middle.raw corresponding to

those on left.raw are used, the same when matching middle.raw and right.raw. So, in total, there are 16 control points are used, 4 on left.raw, 8 on middle.raw and another 4 on right.raw. The selected control points are shown in Fig. 6


(a) Control Points Between Left and Middle

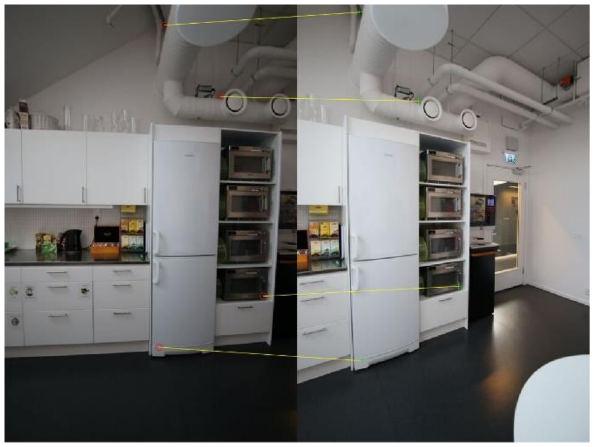

(b) Control Points Between Middle and Right

Fig. 6 Selected Control Points

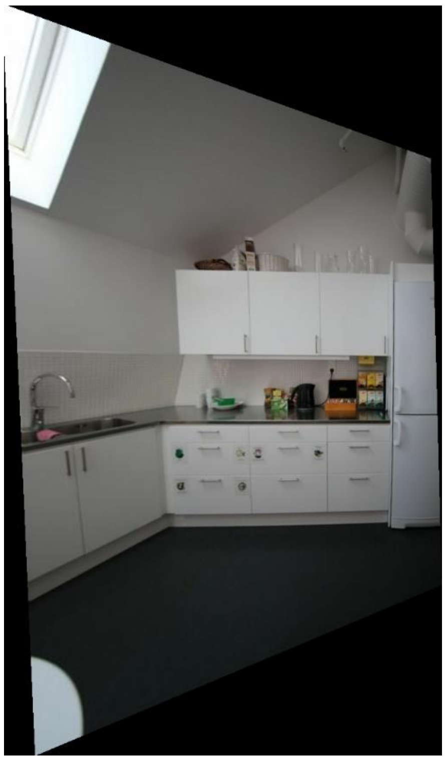
(2) To find the desired control points, the first step is to delete some wrongly paired control points between the two images from the selection poll, then the features which could keep the minimum deformation of the two left corners in left.raw or the two right corners in right.raw are chosen, because if the control points is chosen in this way, the image after morphological processing could have the most part overlapping with the middle.raw. After several experiments, the control points are shown in Fig. 6.

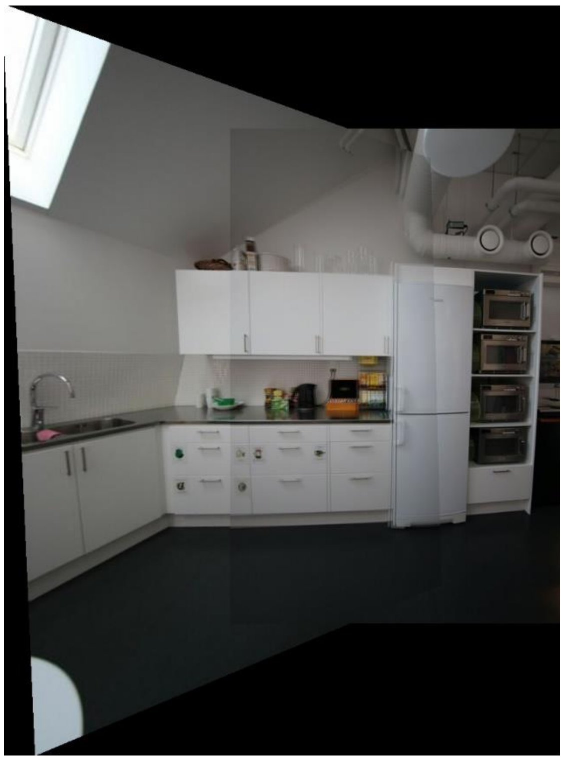
When projecting left.raw onto middle.raw, firstly suppose the original locates at the upper left corner of the left image and the middle image lies in the right of the left image, which means the x coordinates of middle.raw is added with the x length of left.raw. To make use of the information given by control points, the homographic transformation matrix H is estimated using the control points. Following the steps in homework 3 tutorial and [2], w_2'could be obtained by x_1 and y_1, then x_2' and y_2' could obtained, then with 4 pairs of points, the elements in H could be solve by the following matrix, with $H_{33} = 1$.

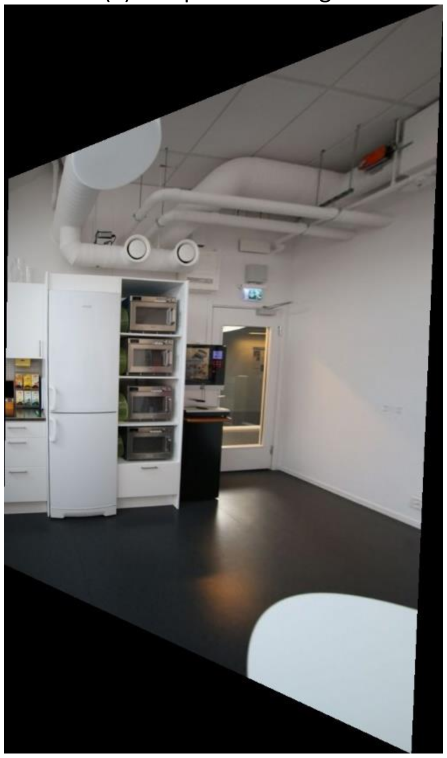
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix}$$

After getting H, each point in left.raw could be projected onto right.raw, the coordinates (x, y) in left.raw is extracted and then the corresponding coordinates (x’, y’) after projection are obtained by the following formula,

$$x' = (H_{11}x + H_{12}y + H_{13})/(H_{31}x + H_{32}y + H_{33})$$
$$y' = (H_{21}x + H_{22}y + H_{23})/(H_{31}x + H_{32}y + H_{33})$$


(a) Warped Left Image


(b) Image Concatenating Left and Middle


(c) Warped Right Image

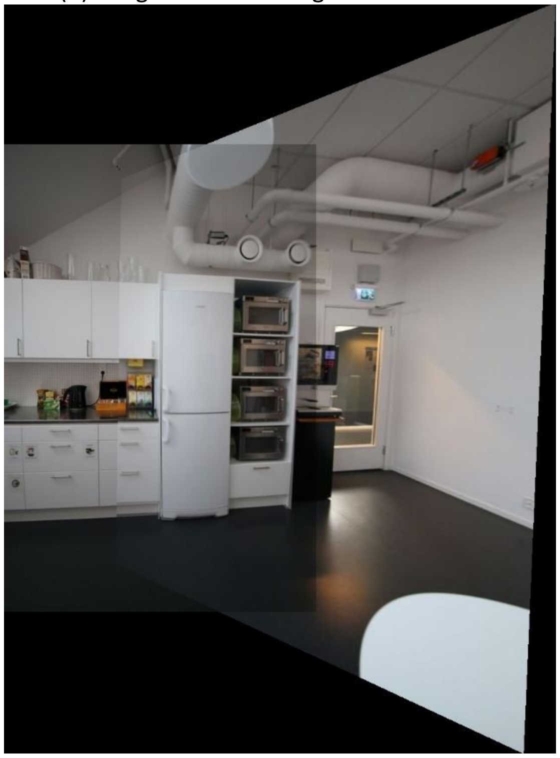

(d) Image Concatenating Middle and Right

Fig. 7 Pre-Processing of Panorama

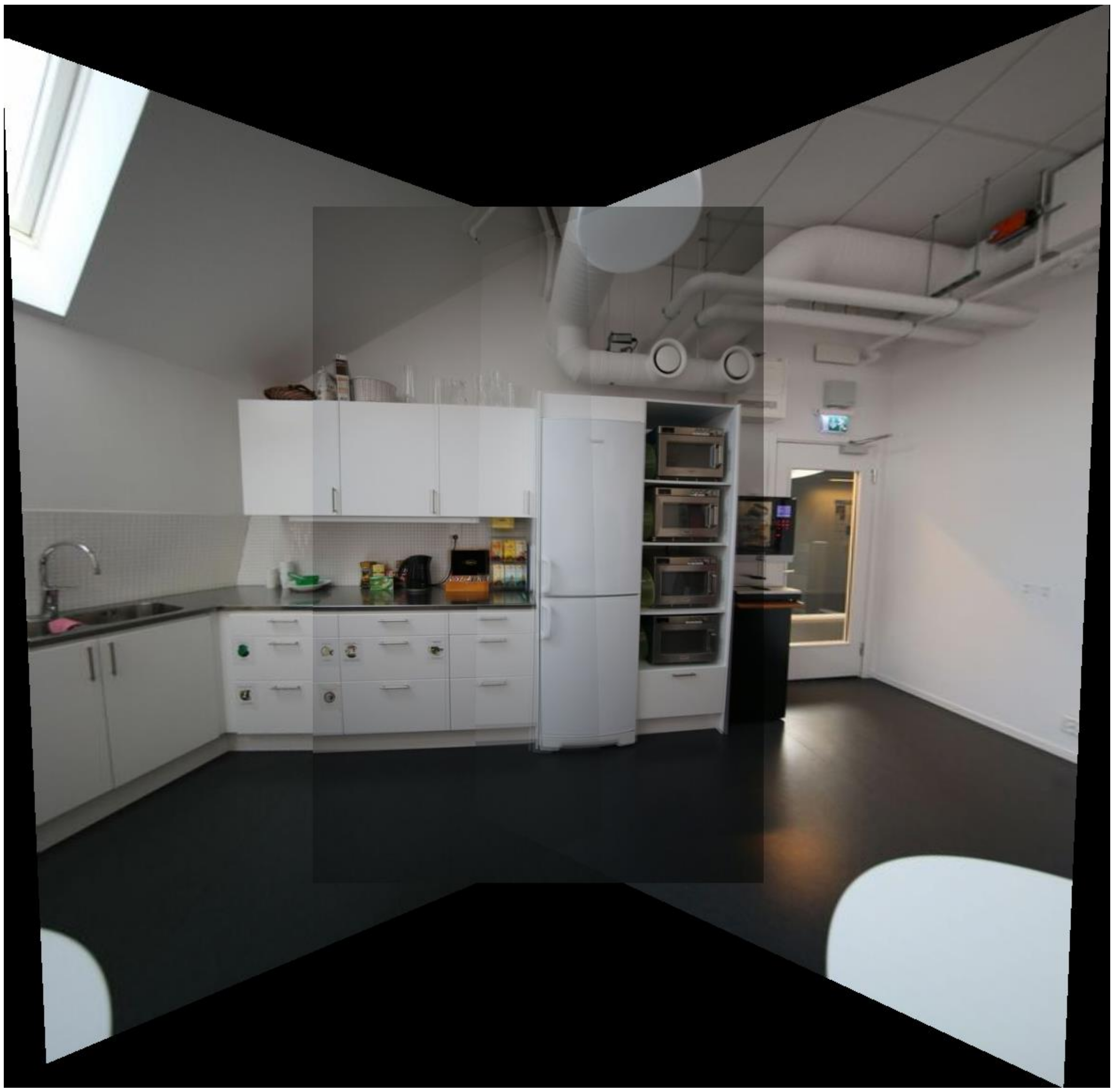


Fig. 8 Panorama

To accommodate the projected left.raw, the original point is translated by the minimum value of x' and y', denoted as x'_{min} and y'_{min}, then a new canvas is allocated to accommodate the projected image by the range of x' and y', the integral coordinates in the new canvas are called (p, q). The relationship between (p, q) and corresponding (x', y') is shown below,

$$p = x' - x'_{min}$$
$$q = y' - y'_{min}$$

For each (p, q), the pixel value is obtained by inverse transform of H, shown by the following formula, where (p', q') is the points in the original left.raw, and H_{ij}^{-1} denotes the elements in H^{-1}.

$$p' = (H_{11}^{-1}(p + x'_{min}) + H_{12}^{-1}(q + y'_{min}) + H_{13}^{-1})/(H_{31}^{-1}(p + x'_{min}) + H_{32}^{-1}(q + y'_{min}) + H_{33}^{-1})$$
$$q' = (H_{21}^{-1}(p + x'_{min}) + H_{22}^{-1}(q + y'_{min}) + H_{23}^{-1})/(H_{31}^{-1}(p + x'_{min}) + H_{32}^{-1}(q + y'_{min}) + H_{33}^{-1})$$

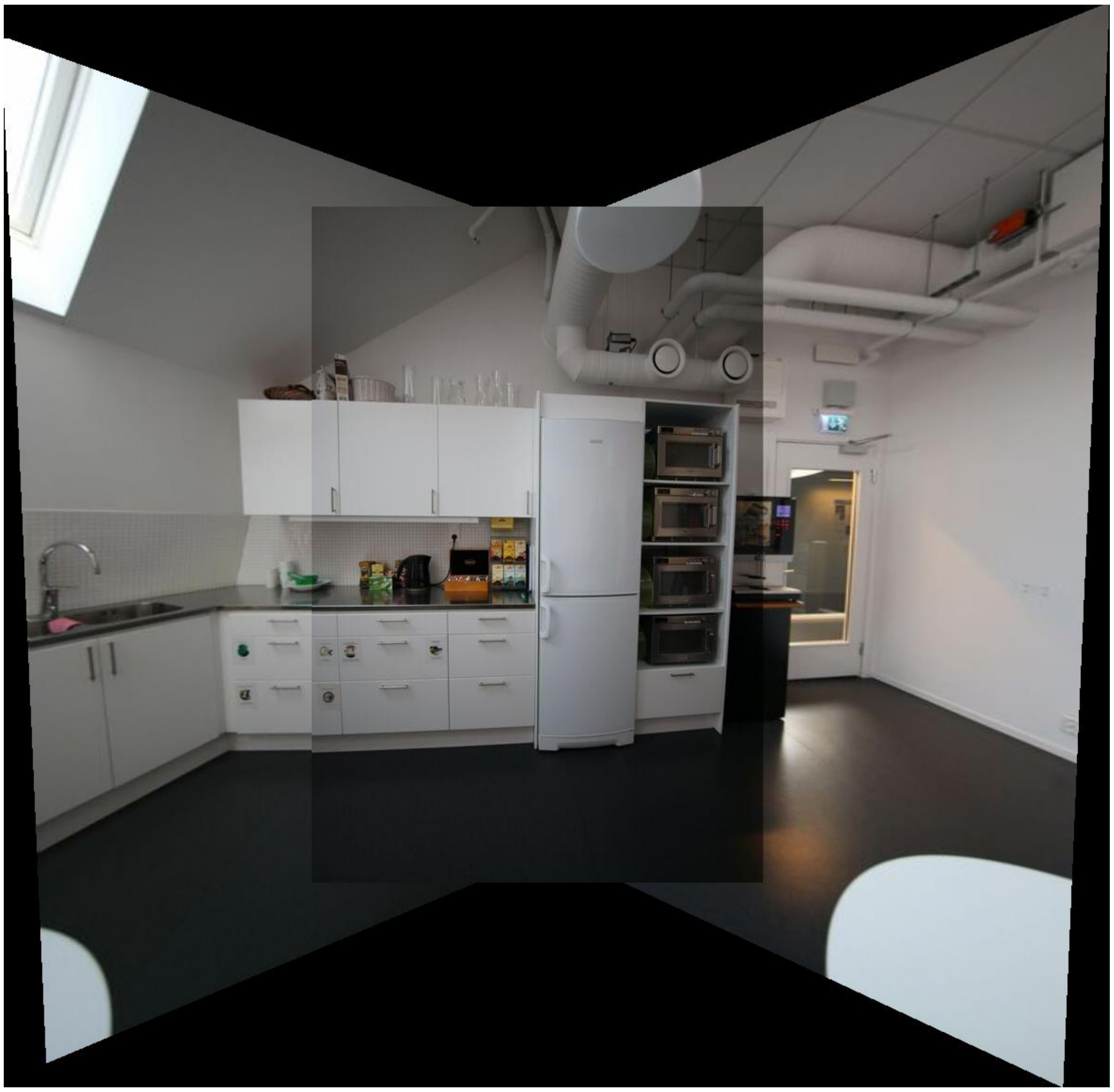


Fig. 9 A Finer Panorama

So, the pixel of (p, q) in the projected image could be calculated by binary interpolation using the point (p’, q’) in the original image, the warped left image and right image is shown in Fig. 7.

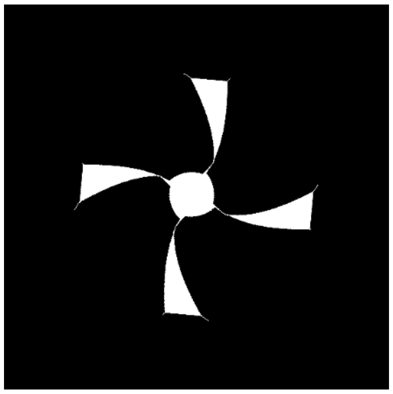
After getting the projected image in the new canvas, the next step is to concatenate the left image and the middle image together. The mean of the selected four feature points is used as the unchanged point. Suppose the mean feature point in the left image is (x_L, y_L), after the procedure described above, the corresponding feature point in the projected left image is (p_L, q_L). Suppose the mean of selected four feature points in the middle image is (p_M, q_M), so the upper left corner of middle image in the new canvas is $(p_M - p_L, q_M - q_L)$, so the two image could be concatenated together in the new canvas, as shown in Fig. 7. If there are only one of the two images, the pixel values are assigned directly by the corresponding pixel values, or the pixel values are obtained by averaging the overlapping part of the two images. The right part could be obtained by the same procedure described above, the locations of upper left corner of middle image are recorded in both images as unchanged points to concatenate Fig. 7(b) and Fig.

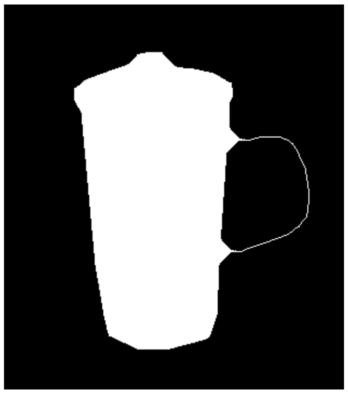
7(d) together. A new canvas is allocated and only the unchanged points are different as the concatenate step above, the result image is shown in Fig. 8.

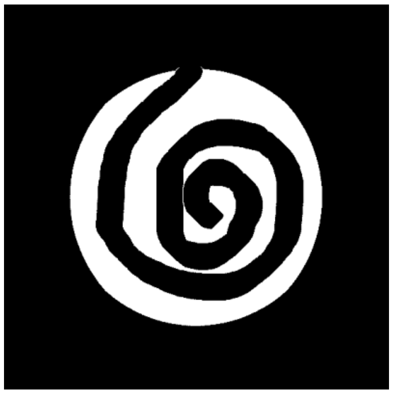
For there are some regions in the picture do not overlap well because of the noise in the chosen feature point pairs, especially in the middle part, there are some shadow like pattern. A finer approach is thus implemented, which using the original middle image as the middle part, instead of averaging the overlapping part in the middle, as shown in Fig. 9.

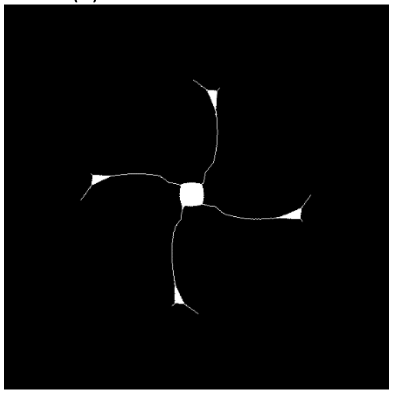
Problem 2: Morphological Processing

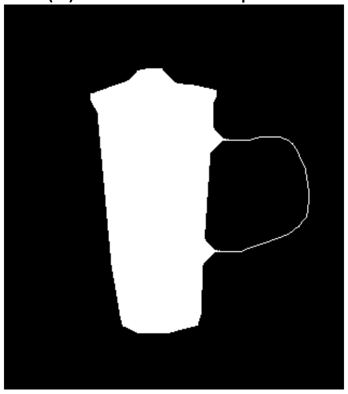
(a) Basic Morphological Process Implementation

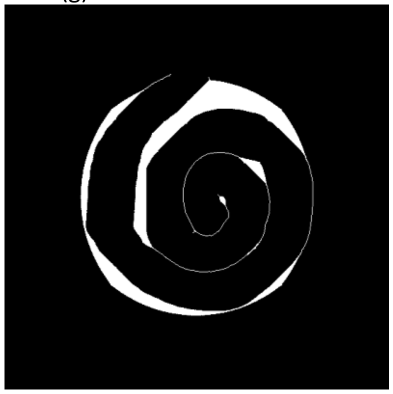

(a) 15 Iteration fan.raw

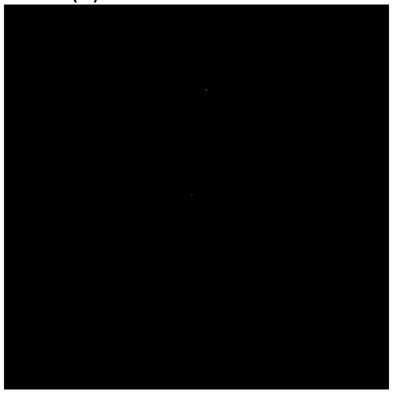

(d) 15 Iteration cup.raw

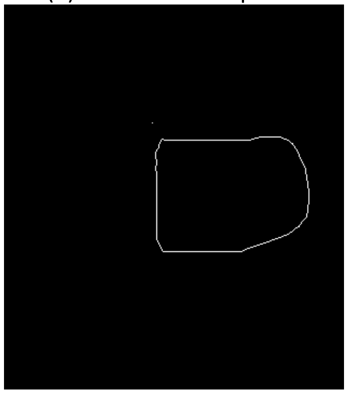

(g) 15 Iteration maze.raw


(b) 30 Iteration fan.raw


(e) 30 Iteration cup.raw


(h) 30 Iteration maze.raw


(c) Converge cup.raw (219 Iter)


(f) Converge cup.raw (106 Iter)

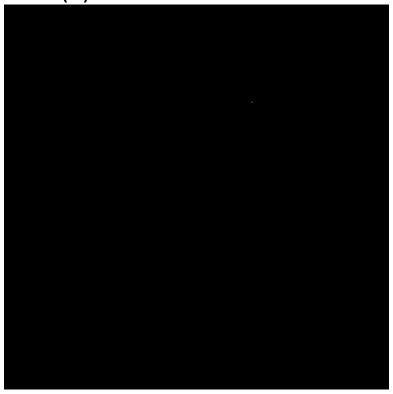
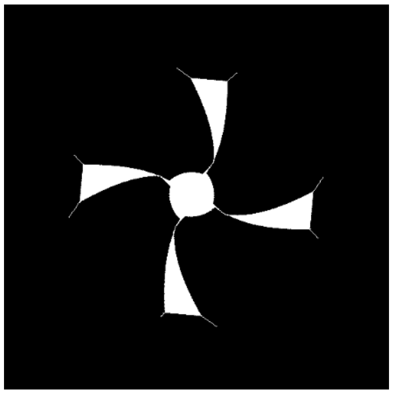
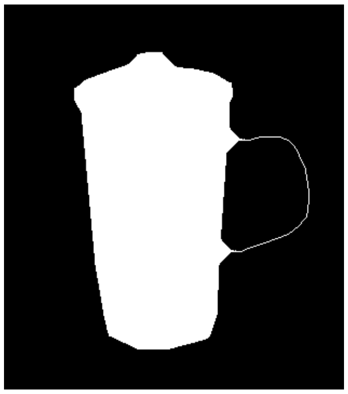

(i) Converge maze.raw (902 Iter)

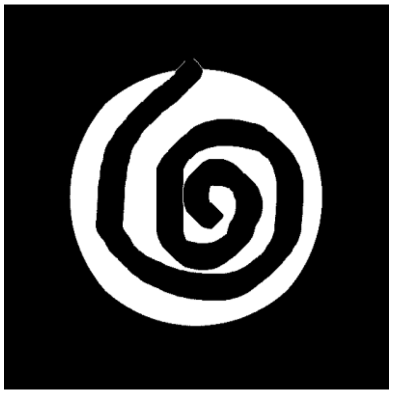
Fig. 10 Shrinking Results

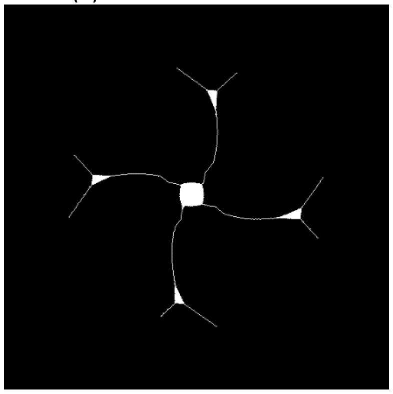
Basic morphological processing includes “shrinking”, “thinning”, “skeletonizing” and some else operations which only take consideration of the information of neighborhood of each points.

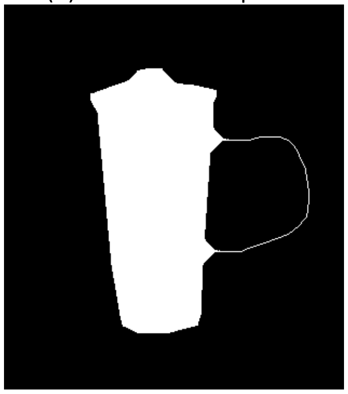
In my approach, the conditional pattern is encoded by binary numbers scanning from the upper left corner by row to the lower right corner of the patterns, the center point in every pattern is skipped for they are all 1. These patterns are classified according to the bond of the center pixel. When comparing, the image is traversed by every point, if the value of a point is 0, it will be skipped, else the bond of the point will be calculated, then the 8-neighborhood pixel values are encoded to binary number in the same sequence as the patterns, then according to the bond, the target patterns are founded and the binary number is turned into decimal number to compare. If they are matched, the pixel will be marked.

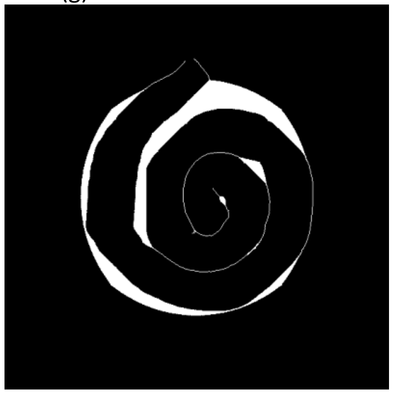

(a) 15 Iteration fan.raw

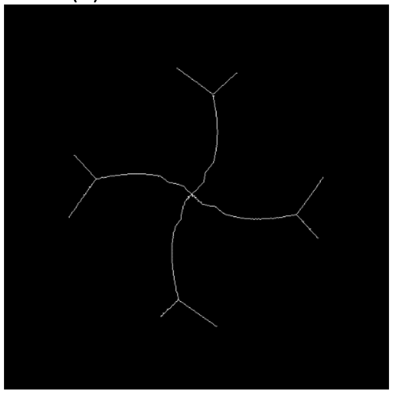

(d) 15 Iteration cup.raw

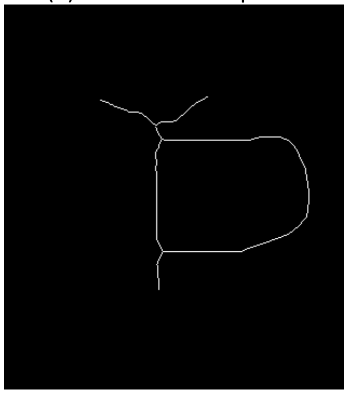

(g) 15 Iteration maze.raw


(b) 30 Iteration fan.raw


(e) 30 Iteration cup.raw


(h) 30 Iteration maze.raw


(c) Converge cup.raw (49 Iter)


(f) Converge cup.raw (90 Iter)

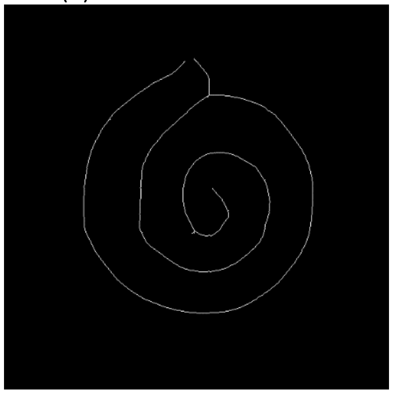
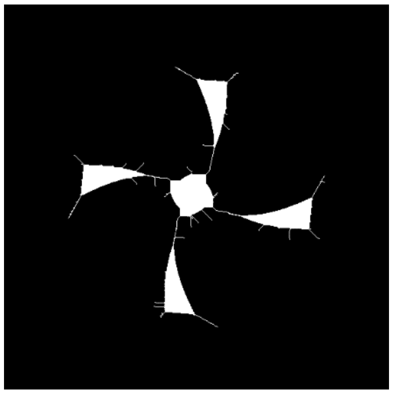
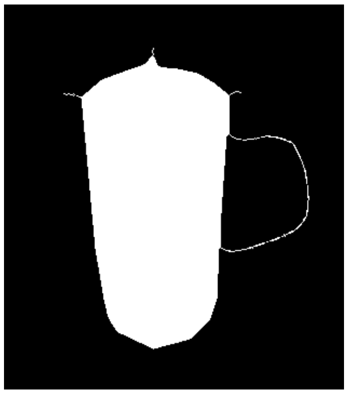
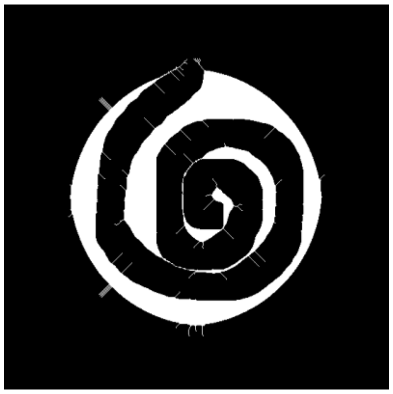

(i) Converge maze.raw (54 Iter)

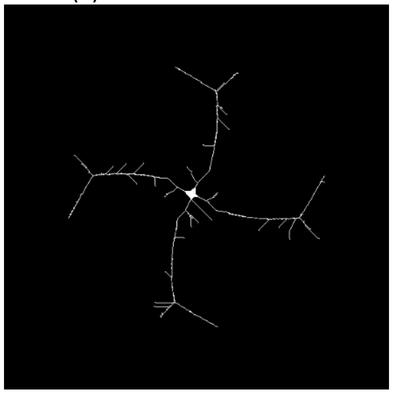
Fig. 11 Thinning Results

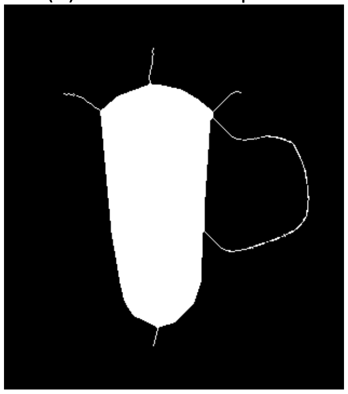
The unconditional patterns are encoded by logical expressions for there are some uncertain expressions like A, B, C, D in the patterns. When comparing, the 8-heighborhood information of a point is used to compared with all of the patterns if the value of the point is 1. After comparing with all unconditional patterns, if there are no matches, the corresponding position in the original image is set to 0, if at least one of these patterns matched, the position in the original image is kept. After the comparison with conditional patterns and unconditional patterns, one iteration is finished. To get the final result, the total number of pixels with value 1 is calculated after each iteration, if the number does not change, the loop comes to an end and the final result is obtained.

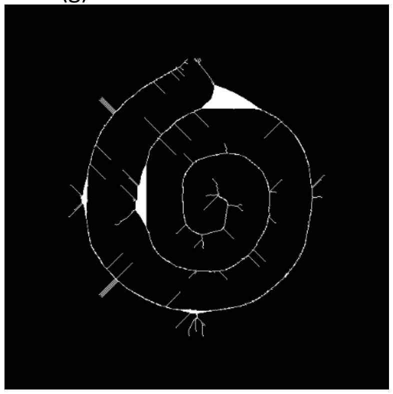

(a) 15 Iteration fan.raw

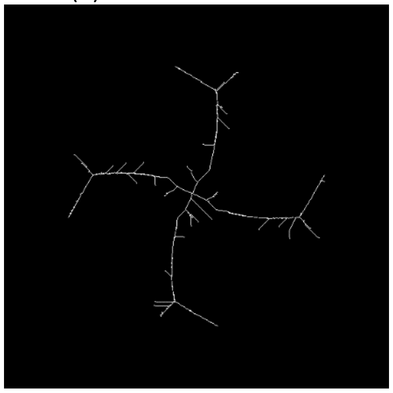

(d) 15 Iteration cup.raw

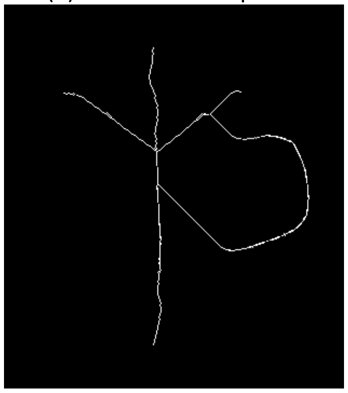

(g) 15 Iteration maze.raw


(b) 30 Iteration fan.raw


(e) 30 Iteration cup.raw


(h) 30 Iteration maze.raw


(c) Converge cup.raw (35 Iter)


(f) Converge cup.raw (80 Iter)

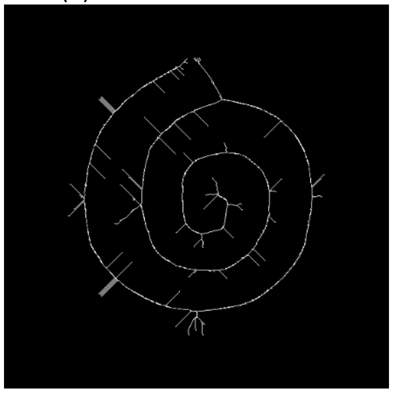

(i) Converge maze.raw (45 Iter)

Fig. 12 Skeletonizing Results

These three operations are all tend to reduce some edge points in the original image. For the procedure of these three operations are the same except that they are using different patterns.

The difference between shrinking and thinning comes from the conditional patterns. Shrinking will mark all pixels with bond 1, 2 and 3, which makes shrinking will reduce a line into a point, no matter how the pixels in a line are connected. While thinning will not mark the pixels with bond 1, 2, and 3, which will leave points at the end of some lines unmarked, thus remain. Shrinking will denote a connection area with a point or a ring, while thinning will not only denote the connection area, it will also leave some information about the length between the longer contour and the shorter contour.

Skeletonizing have the less conditional patterns and it tend to keep all the information of sharp angles of a connected area. This operation will keep the size of the connection area unchanged while reduce some outer pixels of the area, leaving the most information in these three operations.

Note: In the following part, shrinking operation is implemented by bwmorph() in MATLAB.

(b) Counting Games

Assumption:

All of the stars could be just accommodated by a horizontal or vertical diamond shaped frame, which means the maximum L2 distance of each stars could be obtained by $max(i_{max}-i_{min}, j_{max}-j_{min})$

(1) To count the total number of stars in the image, the image is first binarized to remove noise and break the connections between stars close to each other by setting threshold equals to 150. Then apply shrinking to the image until the loop end, there is no changes doing another iteration of the shrinking. Then the number of alone points, which means there are all 0 in the 8-neighborhood of a point with value 1, is counted, which equals to the number of stars. There are 112 stars in the image totally. Fig. 13 shows the result of shrinking of stars.raw

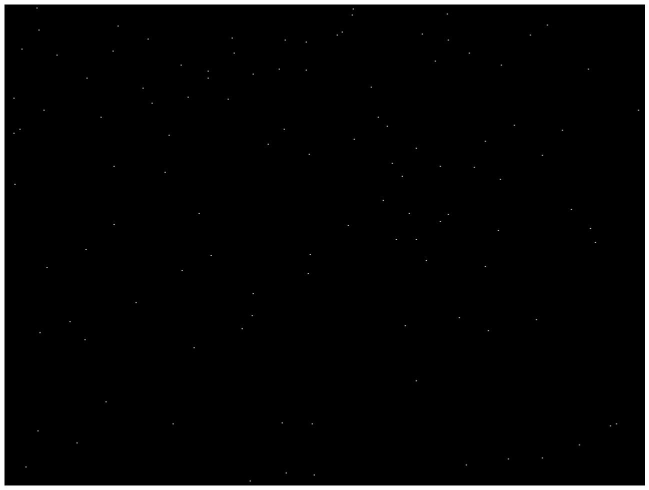


Fig. 13 Shrinking Result of stars.raw

(2) To count the size of each stars using morphological filters, the shrinking operation is used. Because stars of different size will take different iterations to finish shrinking, so after each

iteration, the number of alone points is counted, which stands for the cumulative sum of stars of different size, then take the differential of the cumulative sum, then the star size could be obtained. The result is shown in Fig. 14

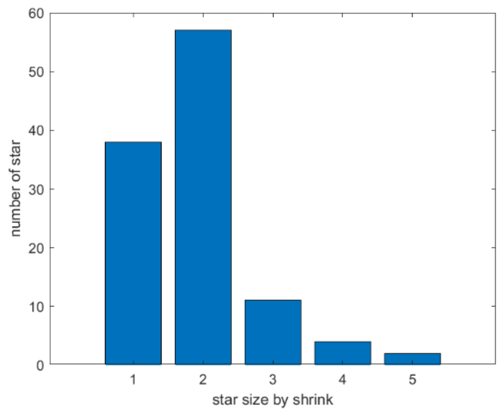


Fig. 14 Number of Stars Counted by Shrink

(3) Another method to do (1) and (2) is to use Breadth-First Search (BFS), after shrinking the original image until converge, the location of center of each star is obtained by counting the alone points, then for each center point, a stack is assigned find all connection part of a star. First, the center point is pushed into stack, while the stack is not empty, the top element of the stack is pop and all its 4-connected neighborhood points with pixel value 1 are pushed into the stack, loop till the stack is empty. During this iteration, the maximum value and minimum value of horizontal coordinates and vertical coordinates of a star is recorded, as well as the number of points with pixel value 1, as shown in Fig. 15. The final result is obtained by maximum value between difference of horizontal and vertical coordinates, as shown in Fig. 16

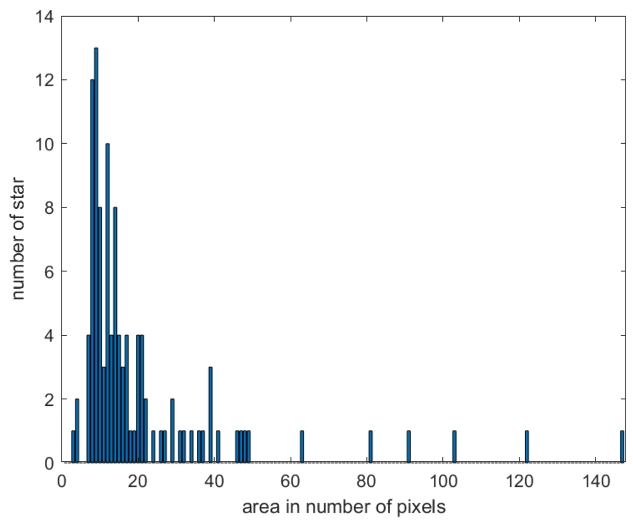


Fig. 15 Number of Stars Counted by Area

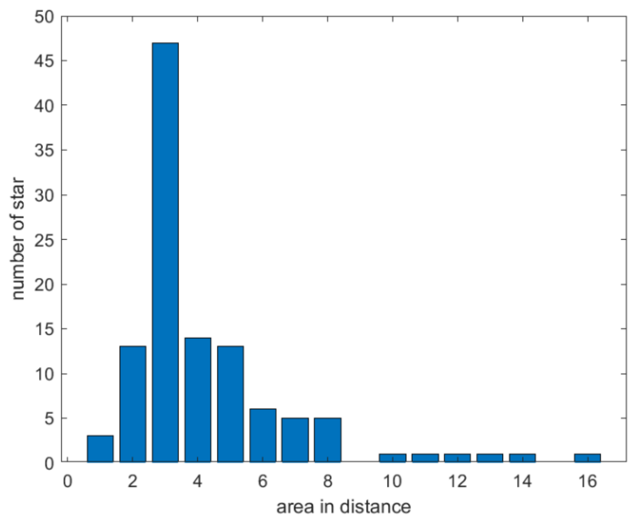


Fig. 16 Number of Stars Counted by L2 Distance

(c) PCB Analysis

Assumption: The outside is connected and counted as one path for the four edges in the original image is white and there are some black blocks stretching out of the edges.

(1) To find the number of holes in PCB.raw, the image is shrunk for several iterations, in my approach, the image is shrunk until converge. The result is shown in Fig. 17, then the alone point defined in part (b) is counted, the number of holes could be obtained. There are 71 holes in total. The location of the alone points is remembered in preparation for (2).

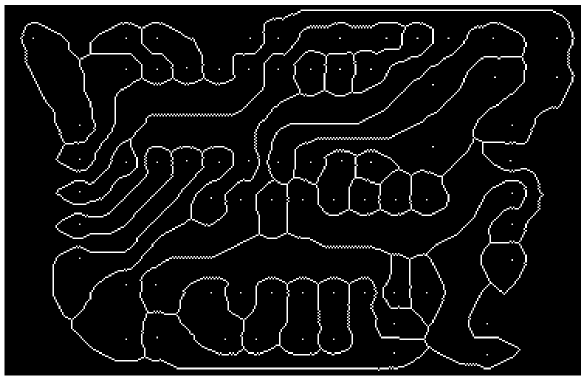
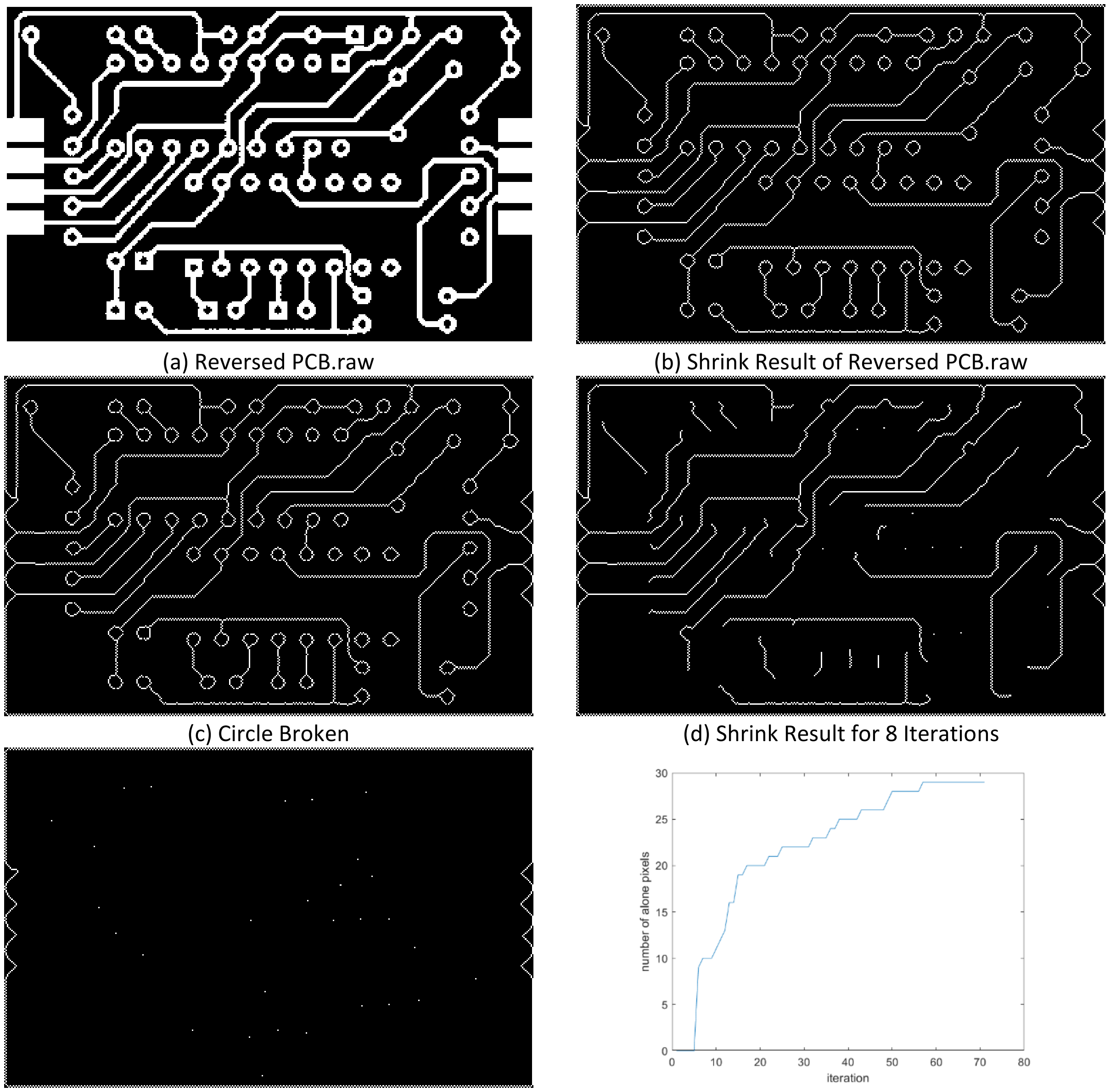


Fig. 17 2 iteration shrink of PCB.raw

(2) To find the number of pathways in PCB.raw, the image is first reversed, then shrunk to convergence, the results are shown in Fig. 18(b). From the result, we could see that there are several holes and others are pathways, if the connectivity of a hole could be cut, the image could be shrunk again. To break the connectivity of holes, the position remembered in (1) is used as the center of each hole, start from the center of each hole, find the first pixel with value 1 above it. If the bond of this point is 2, which means if this point is removed, the connectivity of a circle must loss, the point will be removed. If the bond of the point above it is not 2, find the point left to the center with pixel value 1, test whether the bond of it is 2 or not, if the left point does not



(a) Reversed PCB.raw (b) Shrink Result of Reversed PCB.raw

(c) Circle Broken (d) Shrink Result for 8 Iterations

(e) Final Shrink Result (f) Plot of Number of Alone Points per Iteration

Fig. 18 Pathway Counting Process

satisfy, find the right point or the below point. The circle could be broken, the result is shown in Fig 18(c). Next, the image could be shrunk again. While shrinking, the number of alone points is counted until converge. Because there are rings in the shrink result of reversed PCB.raw, so in the first several iterations of shrinking, the number of alone points stands for the number of rings for rings contain less pixels than any pathways. Then, the number of alone points must remain the same for the path is still shrinking for several iterations, then the number of alone points will increase until unchanged, which stands for the sum of the number of pathways and the number of rings.

From the description above, the algorithm becomes following:

1. Reverse the PCB.raw and shrink until converge.

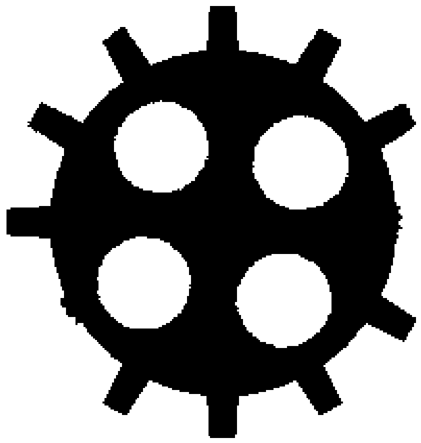
2. Using the information of the center of each circle and break the connectivity of each circle.

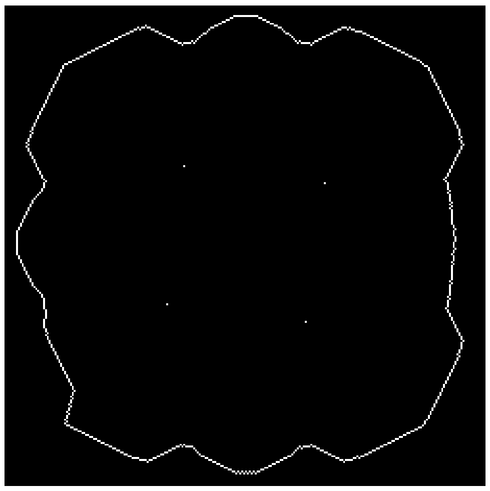
3. Shrink the image until converge, while shrinking, the number of alone points is counted, the first non-zero unchanged number is stored as the number of alone rings, then the number of pathways is obtained by the difference between the number of alone points in the final iteration and the number of alone rings. According to the assumption, the result should be added with 1 because the edge of the whole image is connected.

To prove the correction of the algorithm, the plot between number of iterations and the number of alone points is shown in Fig. 18(f). From Fig. 18(f), we found that the number around 8th iterations remains unchanged, this time, there must be no rings and the pathways must be shrinking, to test this, the result of 8th iteration is shown in Fig. 18(d). We could see that there are incompletely shrunk pathways and alone points shrunk from rings. If we continue shrinking until converge, we could get the final shrinking result containing dots shrunk from rings and pathways, the result is shown in Fig. 18(e). In the final result, the number of alone dots is 29, in the 8th iteration results, the number of alone dots is 10, so there are 19 pathways in the middle of PCB.raw. According to the assumption, the outside is considered as a whole pathway, so the number of pathways in PCB.raw is 20 in total.

(d) Defect Detection

To find the positions of missing teeth, the first thing to do is to find the center position of the big circle by averaging the center positions of four small circles. To find the center of the small circles, the image is first reversed, then shrunk, the result is shown in Fig. 19(a) and (b), respectively. Then the center of small circle is located by alone point counting, the center of big circle is obtained by the average of the centers of four small circle. After getting the center of the large circle, the next task is to estimate the radius of the big circle. Because the teeth are uniformly distributed, and the center could connect the upper, lower, left and right tooth directly by line without go through the black region, so start from the center of the large circle, then traverse to the four directions until there are points with pixel value 0, the number of pixels moved is counted, then the value of radius could be estimated by the average of them. For we do not know whether there is tooth or not in the four directions mentioned above, so the estimated radius ranges between the radius of gear itself and the sum of the radius of gear and length of tooth. So, we should add some deviation to make the radius of the circle we are going to search lies in all of the teeth. In my approach, because there is already a missing tooth in the left, so the deviation is set to 0. After get the radius of the circle we are going to search, the points around the circle is tested to find the missing points. If a point on the circle is 1 while its symmetric point verses the center of gear is zero and the bond of the symmetric point is zero, which means that the symmetric point is really a part of one tooth, instead of a point with zero pixel value near the other tooth, the symmetric point is marked as the potential location of a missing gear. After getting all of the potential location of missing gears, they are separated into different groups according to their location, in this case, the missing gear points are grouped by horizontal coordinates into left and right. Then each group is averaged then the final result is obtained. In this problem, the missing gear lies in (180, 34) and (126, 233), where the 1st element in the tuple stands for the row of the image and the 2nd element stands for the column of the image.


(a) Reversed GearTooth.raw


(b) Shrink Result of Reversed GearTooth.raw

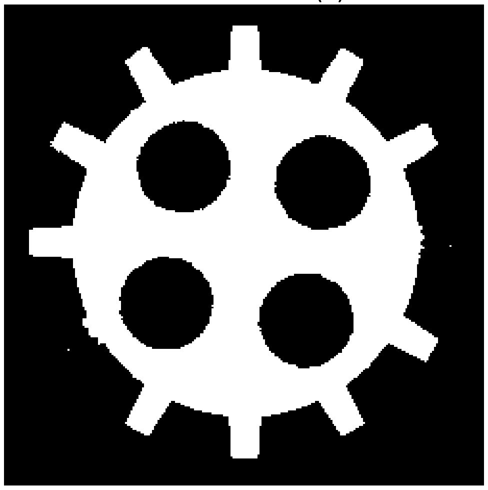

(c) Result of Missing Tooth Detection

Fig. 19 Missing Tooth Detection Process

References:

[1] Fong C. Analytical methods for squaring the disc. arXiv preprint arXiv:1509.06344. 2015 Sep 21.

[2] http://vision.cse.psu.edu/courses/CompPhoto/panorama.pdf