 qiruisun@usc.edu

Homework #3

Issued:2/23/2021 Due: 11:59PM,3/12/2021

Problem1. Geometric Transform

Transfer Image geometric

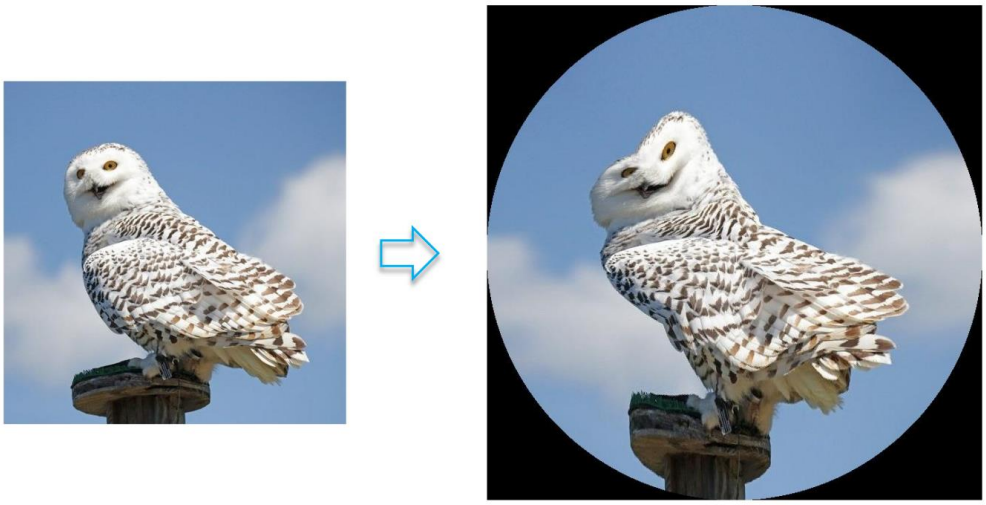


Figure1. transfer mapping

From the geometric explanation, we can extract the mathematical model for the Transform. From the diagonal axis, it is clear that the image is extended on both x-axis and y-axis. Hence, we abstract the math model,

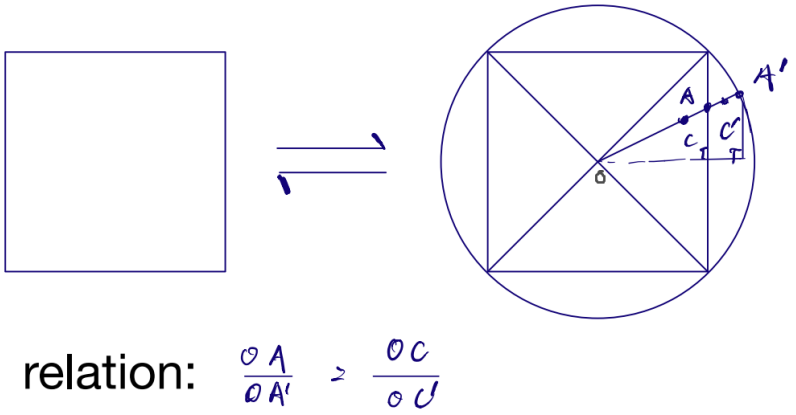


Figure2. mathematical explanation

From the pattern, we can easily obtain the radius $r = \sqrt{2} \times 1ength$, Hence we can calculate the relation of input and output.

Assuming the radius of the circle is r. From the input, we can find that the angle of pixel with center doesn't change. Hence, there is a convincing assumption that points after transfer still lie on the line between the original point and center. Here, A, C are the points on the image before transform and A', C' are the points after transform. Specifically, A is on the edge of the square and C is on the edge of the circle.

Hence, Cartesian coordinate of points, $C: (x_0, y_0), C': (x, y), A: \left(\frac{1}{\sqrt{2}} \times r, y_a\right)$

Finally, I concluded the relationship between the input coordinates and output coordinates.

$$\frac{x}{x_0} = \frac{y}{y_0} = \frac{r}{\sqrt{\frac{1}{2} \times r^2 + y_a^2}}$$

Owing to that the points are all lining on one line. Here, we introduce the by the angle of the whole image. The equation above is only on the first quadrant. Here we extended and generalized the relation to four quadrants.

$$max\{\left|\frac{x}{y}\right|, \left|\frac{y}{x}\right|\} = max\{\left|\frac{x_0}{y_0}\right|, \left|\frac{y_0}{x_0}\right|\} = max\{\left|\frac{y_a}{\frac{1}{2} \times r}\right|, \left|\frac{x_a}{\frac{1}{2} \times r}\right|\}$$

$$\left|\frac{x}{x_0}\right| = \left|\frac{y}{y_0}\right| = \frac{r}{\sqrt{\frac{1}{2} \times r^2 + min\{x_a^2, y_a^2\}}}$$

Transform Algorithms:

Step1. Transfer the image index to cartesian coordinates by center O.

Step2. Iterate the image pixel by pixel on the target canvas (inverse mapping) (x,y)

Step2.1 Calculate the resize ratio by x, y

Step2.2 Inverse mapping the pixel and find the original coordinates x_0, y_0

Step2.3 Apply Bilinear interpolation to calculate the pixel value.

Step3. Transfer the cartesian coordinates inversely to pixel index and assign the pixel value

Step4. Crop the image by the circle r

Inverse Transform Algorithms:

Step1. Transfer the image index to cartesian coordinates by center O.

Step2. Iterate the image pixel by pixel on the target canvas (inverse mapping) (x,y)

Step2.1 Calculate the resize ratio by x_0, y_0

Step2.2 Inverse mapping the pixel and find the original coordinates x, y

Step2.3 Apply Bilinear interpolation to calculate the pixel value.

Step3. Transfer the cartesian coordinates inversely to pixel index and assign the pixel value

Experiment Results:

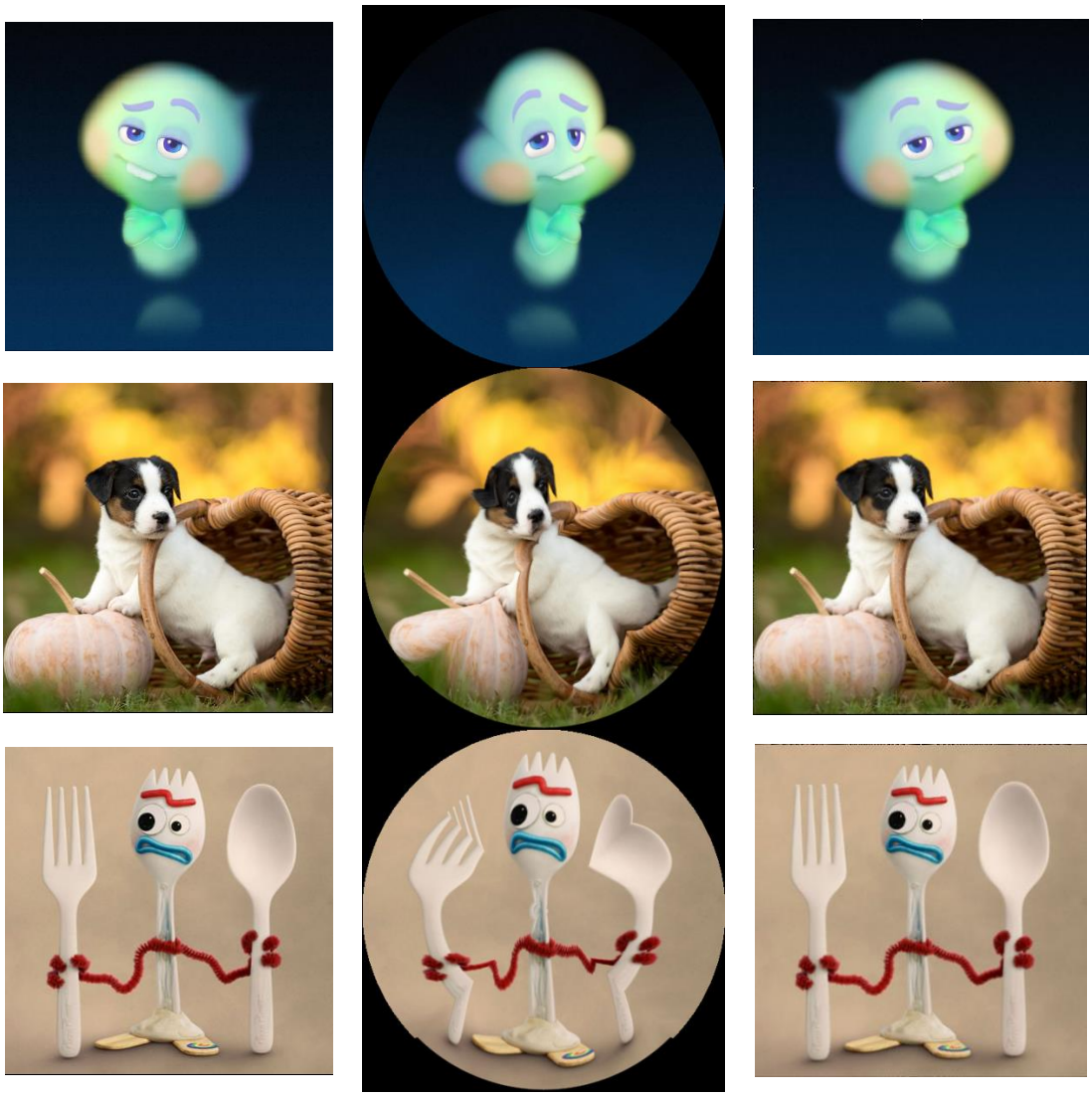


Figure3. Experiment results

Discussion:

From the image, we found that the transformed image is little blurred because the pixel value is calculated by the bilinear transform, while the original images don't need to be distributed linearly. Hence, the method may introduce some error.

Besides, From the recovered image, the edge of image is not preserved as well as the original image. This scenario is resulted from the image cropping to circle during forward transform. Some of the pixel value was assigned to be 0. The image index is integer while the cartesian coordinates are the float number. This process may assign some pixel value to 0 which is also used during inverse transform.

Problem 2: Homographic Transformation and Image Stitching

Image Stitch Algorithms:

Step1. Search form key points by SURF key point detector

Step2. Match the key points by image pair.

Step3. Select four key points manually

Step4. Transfer the image index to the cartesian coordinate on the default canvas

Step5. Calculate the projective Transform Matrix on every two images.

Step6. Inverse mapping the canvas and apply projective Transform Matrix to find points on default image

Here I selected the key points manually, because there are some wrong key points and the number of key points is not enough much. The principle of my algorithms for key point selection is to find the most diverse and clear points. It is because that when the key points are near, the manifold of matrix equation degrades and we don't have enough correct information to match the point. Hence, I found the key points mainly from diverse object in the pairs.

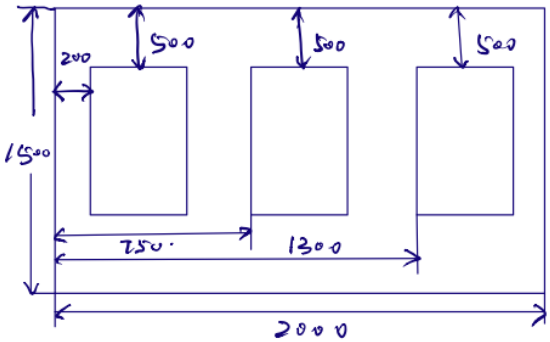
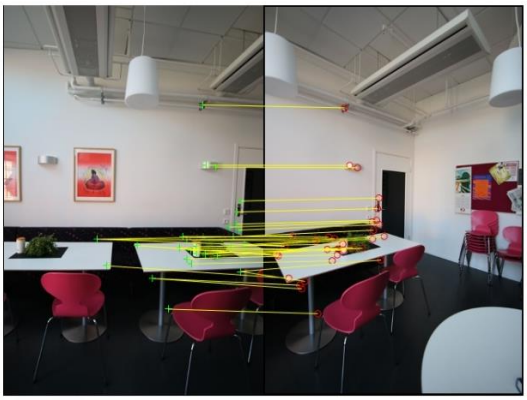
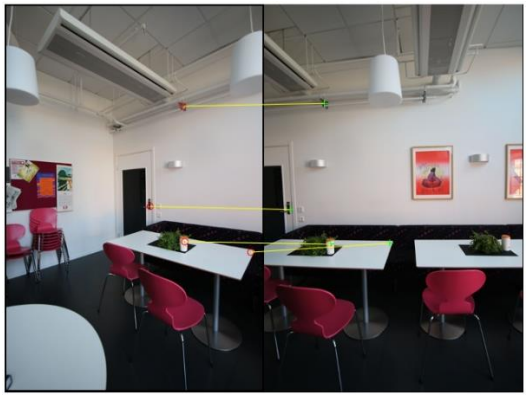


Figure4. Default canvas pattern



Matched key points left

Matched key points right

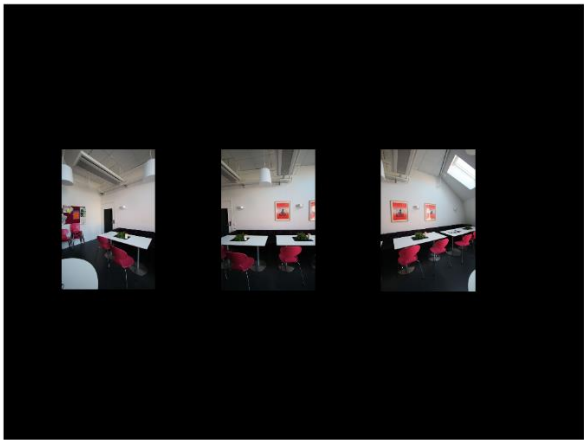


Selected key points left

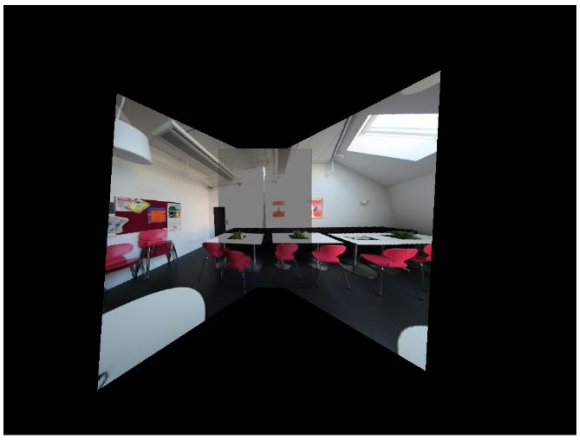
Selected key points right

Figure5. Key points Result

Experiment Results



Default Canvas



Stitching panorama

Figure6. panorama Result

Discussion:

For key point selection part, there are so many algorithms can be selected the key points selection. I have tried the RANSAC Algorithm but the result is not as selecting by hands. The core is to find the key points accurate and mor diversely. For example, the left image, I choose the key points from ceil, door handle, desk corner and bottle sign on the table. For right image pair, I select key points on the light, table, ceil and chair.

Image Stitch Algorithm

INPUT: image, KEY POINTS

OUPUT: Panorama

Step1. Calculate the Transfer matrix by key points H

$$\boldsymbol{P_{post} = H \times P_{pre}}$$

Step2. Calculate the Inverse Transfer Matrix H', by $\boldsymbol{H' \times H = I}$

Step3. Iterate the Canvas pixel and calculate the original points before the perspective transform (containing the coordinate convert).

Step4. Convey the pixel value from original canvas to the new canvas index.

Problem 3: Morphological processing

Morphological process is the process to determine the image shape information. The mainly framework of two steps. The first step is to determine whether the image patch matches the supposed pattern. The second step it to determine whether the image matches unconditional condition to avoid that miss delete or over delete.

Morphological Algorithms

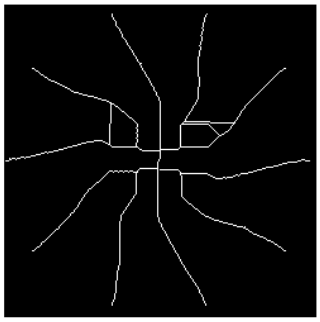
While image changes in every iteration:

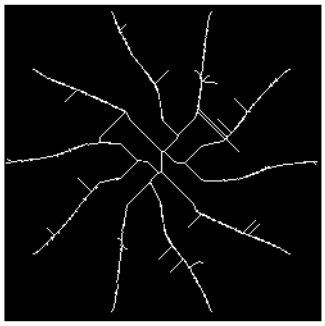
Step1. Calculate the bond matrix B, Calculate the Matrix S to record surrounding elements, and use bond matrix B and S to help with determine Matrix M.

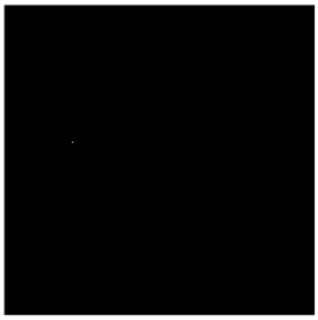
Step2. Match the Matrix M to specific patterns. Determine the image pixel to be shrink or not based on matching scenarios.

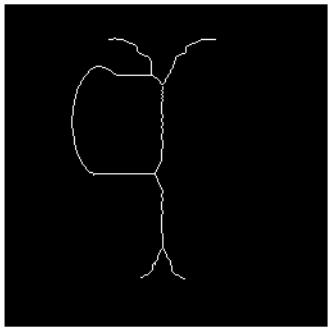
Experiment Results

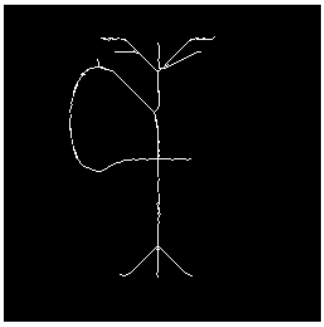

Flower shrink result

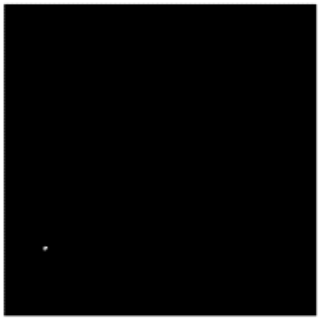

Flower thin result

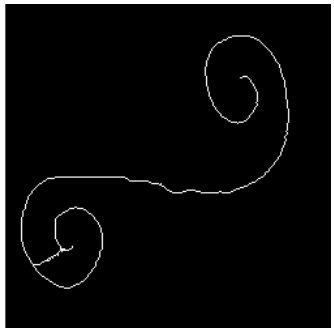

Flower skeleton result


Jar shrink result


Jar thin result


Jar skeleton result


Spring shrink result


Spring thin result

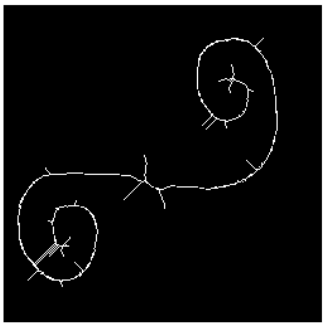

Spring skeleton result

Figure 7. Basic morphological process result

(b) maze problems

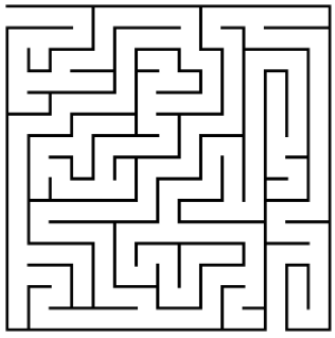


Figure8. maze image

From the maze image, we know that the boundary is black (0) and the potential path it white (1). There is 1 surrounded by the maze. Hence, if there is a path from the entrance to exit. The path cannot shrink to a point but a closed circle by the connected the path and outside. Meanwhile, the potential road is closed, shrink process will push them to be a line and finally disappeared during the process. Hence, Solution to this problem is to shrink the maze to find the final Answer.

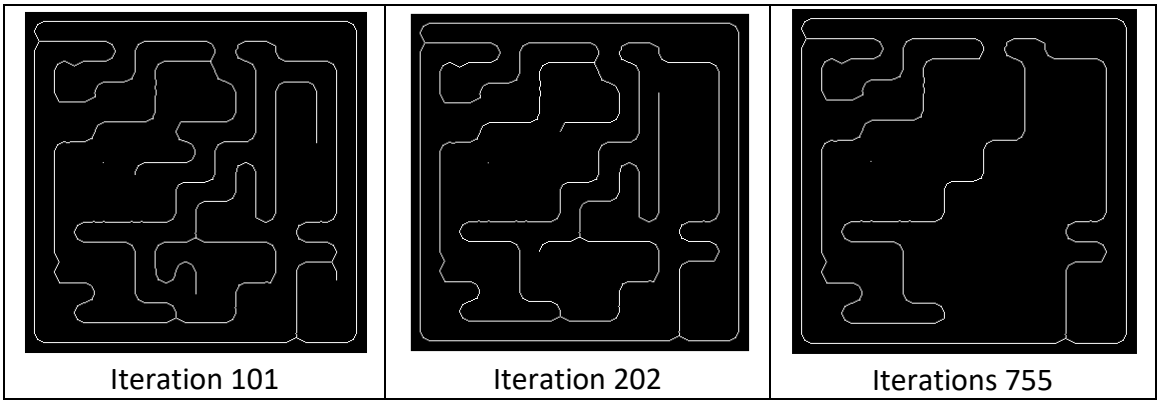


Figure9. maze shrink process output

During the experiment, we chose the threshold = 30 and the process terminate and complete at iteration 755. From the subplot Iteration 755 above, we can clearly justify the final solutions to the image output.

(c) Defect detection and counts

c.1 Count the disconnected region

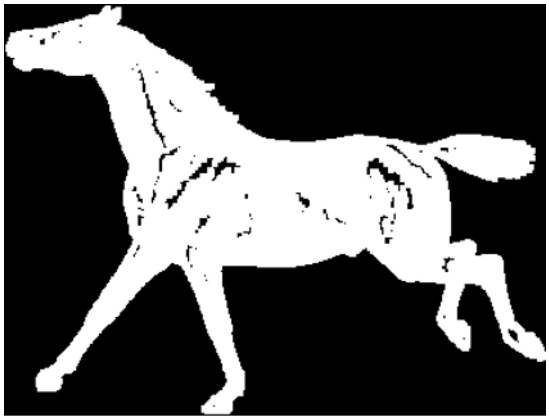
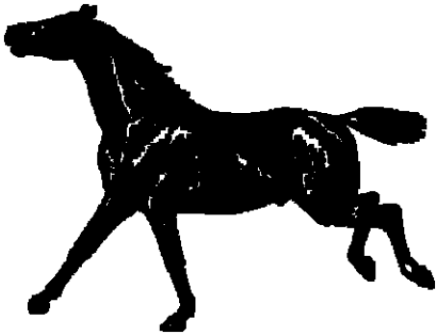


Figure10. the horse

From the image, the defects are the region that is black and isolated from the white region. Hence, in order to detect that region, we need to shrink the pre-processed image and

counts the dots in the image. Owing to the back toed of the horse is in the edge of image. Firstly, we need to do zero padding on the image and do the dithering by threshold. From now the defects and background surrounding the horse are labelled as 1.

Experiment results



pre-processed horse

Shrink horse

Figure 11. pre-processed horse and shrink points

When choosing 30 as dithering threshold, the total number of defects is 63.

c.2 Size of defect and find the frequency of size

From the discussion section, we use the number of iterations of every defect point to represent the size of defect. The core to calculate in which iteration that the defect location converts to be isolated.

Algorithm:

Step1. Shrink the image and filter the surrounding shrink points.

Step2. Record every defect location.

Step3. Shrink the image and check if the defect locations are isolated during every iteration.

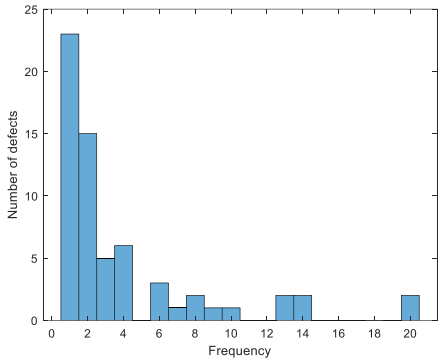


Figure13. the histogram of defect size

From the histogram, we can find that there are 13 different size of defects based on the iteration.

c.3 Clear the image

During the image clear parts I use the thin and shrink of the image to guide for detection. Normally the clear process can only be guided by thin image. However, for this image, the surrounding of image is recognized as object and will be thinned as a close loop surrounding the image. Hence, here we use the both information to use.

Algorithm:

Step1. Calculate the shrink of image $\boldsymbol{I_{shrink}}$

Step2. Calculate the thin of image $\boldsymbol{I_{thin}}$

Step3. Iterate pixel by pixel, justify the point whether defect or not

If $Image(i,j) == 0$ and $\boldsymbol{I_{shrink}} == \boldsymbol{1}$ and $\boldsymbol{I_{thin}} == \boldsymbol{1}$

Use a 5 by 5 kernel to clear the defect.

Experiment result

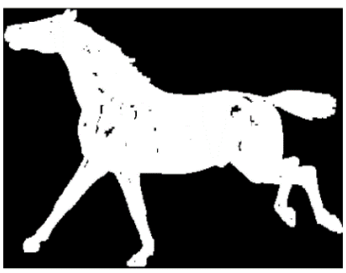


Figure14. cleared horse

This algorithm can work for clearing but the accuracy is not as good as possible. It is because that we can not clearly record the location of every defects. In order to improve the accuracy. We use Connected-Component Labelling (CCL) algorithm.

C4. Use Connected-Component Labelling to repeat

Connected-Component Labelling is an algorithm to assign different location with different object label. If the pixels are connected, Use the same label for that region.

Algorithm:

1st pass: Find the surrounding 4 pixels and justify the raw labels.

Razer scanning the image

If the pixel is background:

Continue;

If the pixel is object and surroundings are all background

Assign a new label to the pixel;

If the pixel is object and surroundings exist object

Assign the smallest label to the pixel;

Record that labels in a connected region as one similar region;

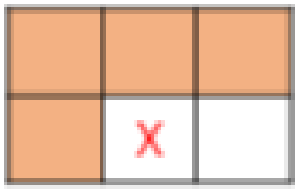


Figure15. surrounding pixels

2nd pass: Update the same label with the same label

While image == previous_image:

Find the minimum label of surrounding 8 pixels;

Update the target label with the minimum label;

Experiment Results:

Figure16. label mask by CCL Algorithm

We can use the label mask to repeat **C1 and C2.** For the C1 part, the Questions can be very straight forward. We can calculate the frequency of different labels on the image. There is a reminder that the label 0 is backgrounding and label 1 is the pixel outside the image. The defects are the other labels.

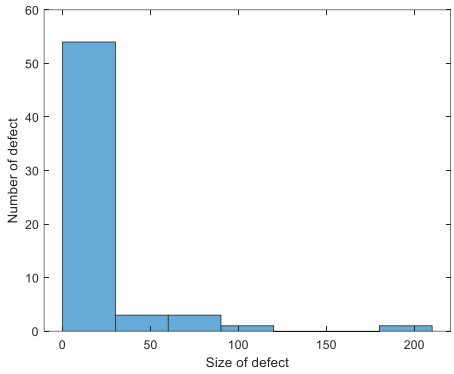


Figure17. Defects frequency by CCL

From the experiment, we use the threshold =30. there are **24** different sizes of defects in total. Besides, To use the image mask for defects. The result is perfect.

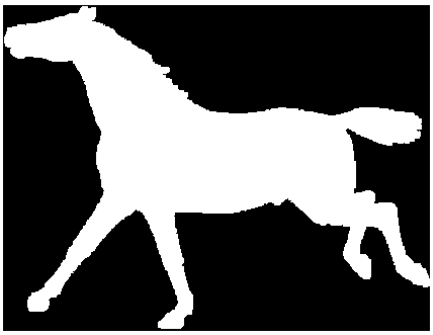


Figure18. Clear Horse by CCL

Discussion

If we don't use Connected Component Labelling, we only use the iterations of defect points to represent the size. The Size is not the true size of image, while the mask obtained by CCL can label every pixel and the size of defect and calculate by counting.