Digital Image Processing (2021 spring)
HOMEWORK ASSIGNMENT #2 Edge Detection, Geometrical Modification
environment: C++, opencv4
Law Po Ying b06902091

Problem 1: EDGE DETECTION
Part a

(1) Perform 1st order edge detection and output the edge maps as result1.jpg

i. Calculate the row and column gradient respectively.
ii. Calculate the gradient of the pixel and check if it is larger than the threshold value. If true, mark it as an edge pixel.

Roberts

$$G_1(j,k) = F(j,k) - F(j+1,k+1)$$
$$G_2(j,k) = F(j,k+1) - F(j+1,k)$$

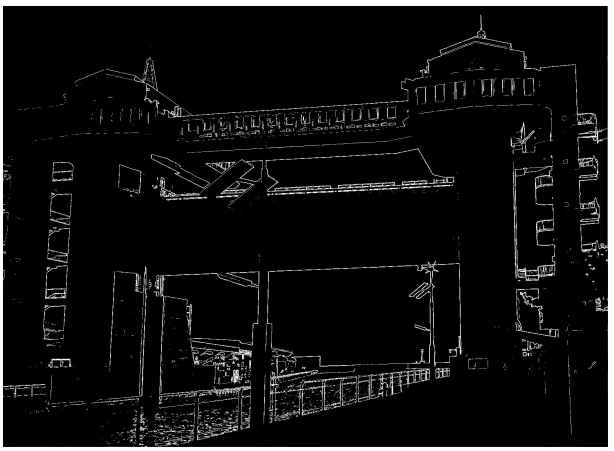
Prewitt

$$G_R(j,k) = \frac{1}{1+2}[(A_2 + 1A_3 + A_4) - (A_0 + 1A_7 + A_6)]$$
$$G_C(j,k) = \frac{1}{1+2}[(A_0 + 1A_1 + A_2) - (A_6 + 1A_5 + A_4)]$$

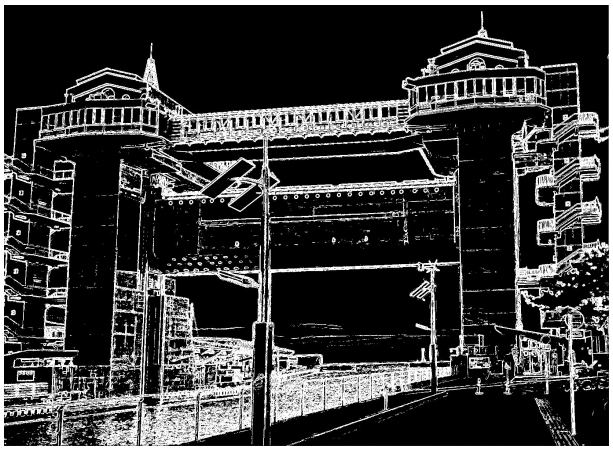
Sobel

$$G_R(j,k) = \frac{1}{2+2}[(A_2 + 2A_3 + A_4) - (A_0 + 2A_7 + A_6)]$$
$$G_C(j,k) = \frac{1}{2+2}[(A_0 + 2A_1 + A_2) - (A_6 + 2A_5 + A_4)]$$

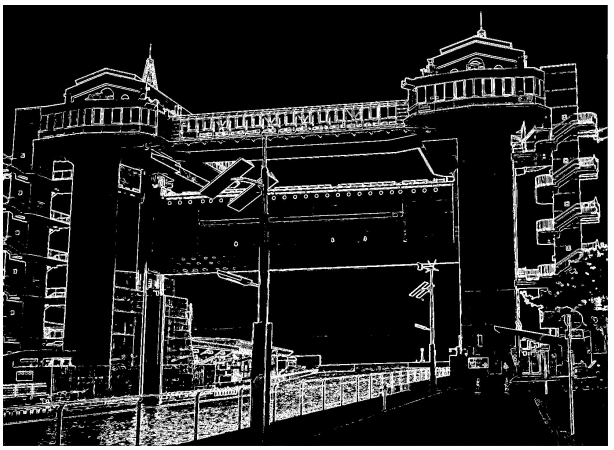
roberts t=40



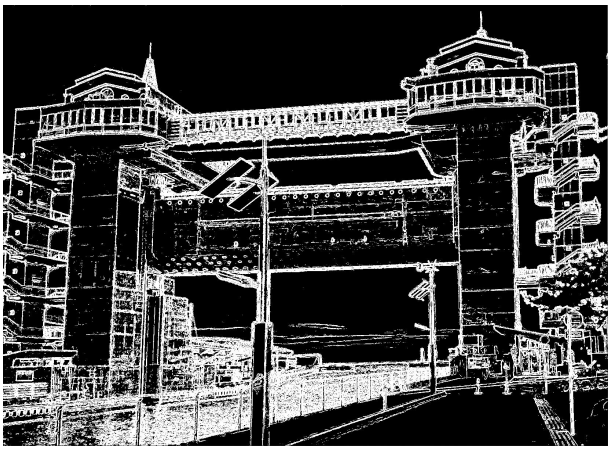
roberts t=70



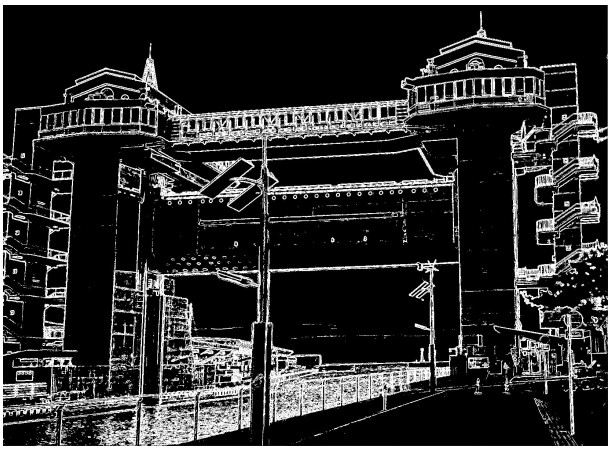
prewitts t=40



prewitts t=70



sobel t=40



sobel t=70

We can observe that the edge map is much detailed when the threshold value is low. However, the edge map and the contour of the object are much clearer when the threshold value is high. If the threshold value is too high, the edges may be disconnected. Since **Roberts** edge detection with the threshold value **40** is clear and the edges are well linked up, I have chosen it to be result1.jpg.

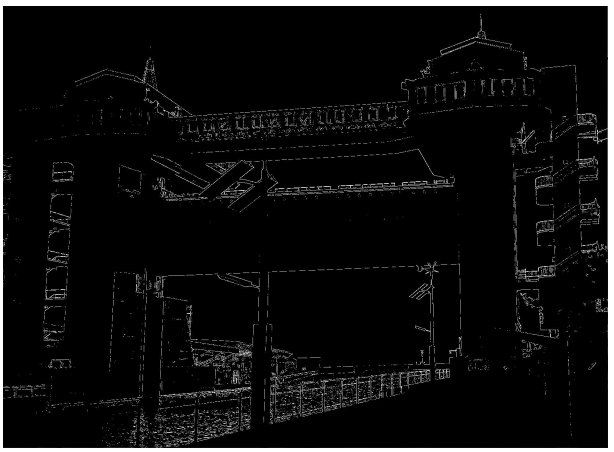
(2) Perform 2nd order edge detection and output the edge maps as result2.jpg

i. 3x3 Low-pass filter
ii. 2nd-order Derivative (Laplacian)
iii. apply thresholding to separate zero and non-zero point
iv. Decide whether the current pixel is a zero-crossing point

$$H = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$H = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

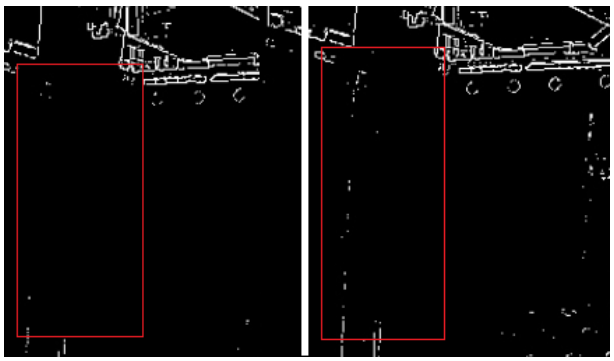
4-neighbor t=2



4-neighbor t=4

8-neighbor t=2

8-neighbor t=4



Comparing the results of the same kernel with different threshold values, using a higher threshold value can compute a clearer edge map. And we can obtain a more detailed edge map with a lower threshold value. And the difference between 4-neighbour and 8-neighbor kernels is not significant. Since the result of **8-neighbour with t=2** is clear, detailed and the edges are well linked up, I have chosen it to be result2.jpg.

(3) Perform Canny edge detection and output the edge maps as result3.jpg

i. Noise reduction - low-pass filter

ii. Calculate gradient magnitude and orientation - sobel's

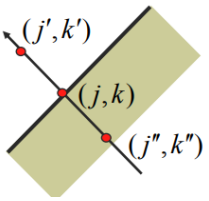
$$G(j,k)=\sqrt{G_R^2(j,k)+G_C^2(j,k)}$$

$$\theta(j,k)=\tan^{-1}\left(\frac{G_C(j,k)}{G_R(j,k)}\right)$$

iii. Non-maximal suppression

Search the nearest neighbors (j',k') and (j'',k'') along the edge normal

$$G_N(j,k)=\begin{cases} G(j,k) & \text{if } G(j,k)>G(j',k') \\ & \text{and } G(j,k)>G(j'',k'') \\ 0 & \end{cases}$$



iv. Hysteretic thresholding

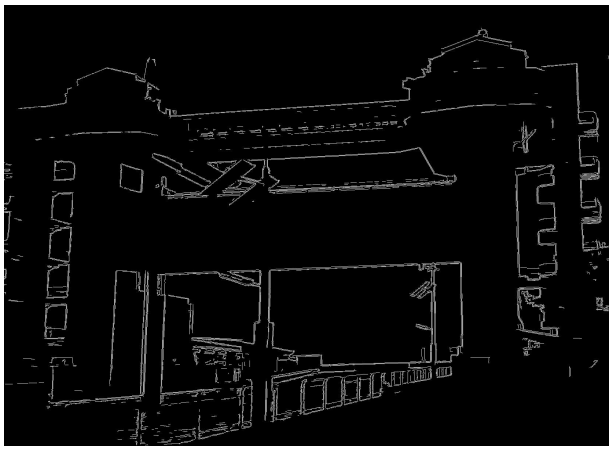
Label each pixels according to two threshold: T_H, T_L

$G_N(x,y)\geq T_H$	Edge Pixel
$T_H>G_N(x,y)\geq T_L$	Candidate Pixel
$G_N(x,y)<T_L$	Non-edge Pixel

v. Edge tracking

- Candidate pixel turns into an edge pixel, if and only if at least one of the pixels around the current pixel is an edge pixel.

Result



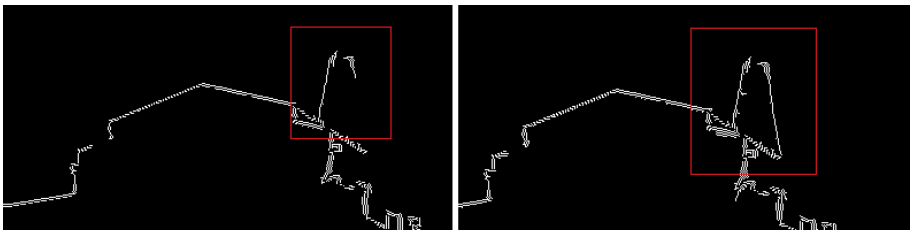
7x7 low-pass filter T_h= 110 T_L= 55

3x3 low-pass filter

7x7 low-pass filter

T_h= 110 T_L= 22

T_h= 70 T_L= 55

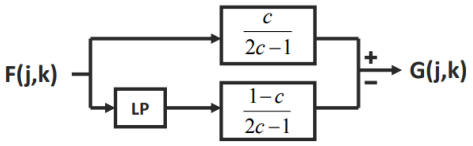


We can see that the structure of the edge map is much detailed when using a smaller size low-pass filter.

For the setting of the threshold value, we can obtain a well-connected edge map when we decrease the lower threshold. However, if we increase the higher threshold, the structure of the edge map is much more detailed. I have chosen a 7x7 low-pass filter, Th= 110 TL= 22 to be result3.jpg as it is clear and well-connected.

(4) Apply an edge crispening method to the given image, and output the result as result4.jpg. Please also generate an edge map of result4.jpg as result5.jpg.

i. Compute with all-pass filtering
ii. Compute with low-pass filtering
iii. Combine all-pass and low-pass filtering (Unsharp Masking)



$$G(j,k)=\frac{c}{2c-1}F(j,k)-\frac{1-c}{2c-1}F_L(j,k),\quad where\quad \frac{3}{5}\le c\le\frac{5}{6}$$

$$H=\begin{bmatrix}0&-1&0\\-1&5&-1\\0&-1&0\end{bmatrix}$$ filter1

$$H=\begin{bmatrix}-1&-1&-1\\-1&9&-1\\-1&-1&-1\end{bmatrix}$$ filter2

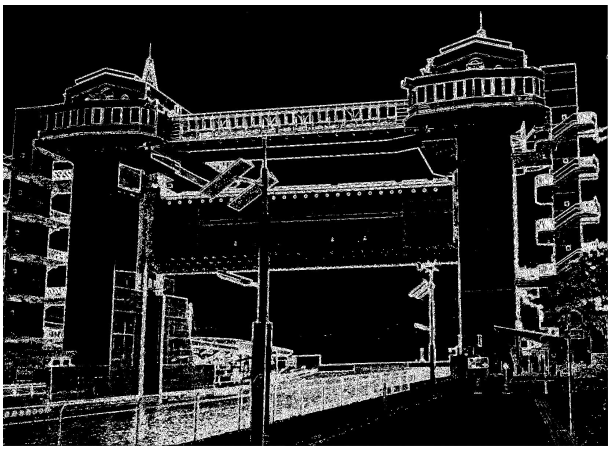
sample1.jpg

all-pass filter1

all-pass filter2

unsharp masking

roberts t=70 (original)



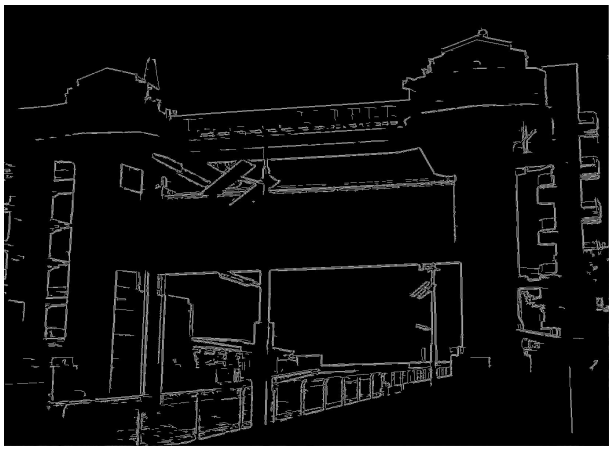
roberts t=70 (after edge crispening)

After edge crispening, we can obtain a stronger, clearer edge map with the same edge detection method and the same threshold value.

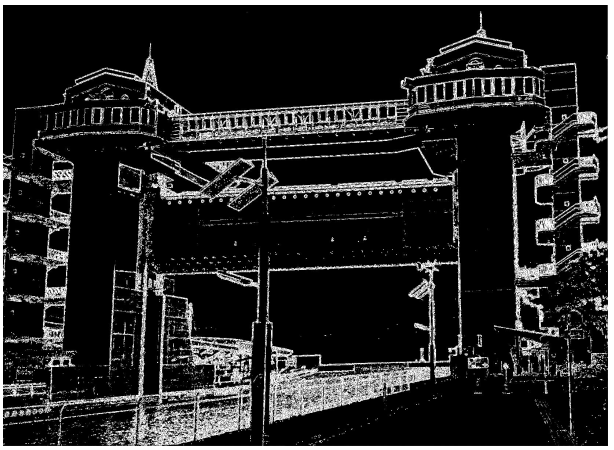
(5) Compare result1.jpg, result2.jpg, result3.jpg, and result5.jpg. Provide some discussions and findings in the report.

result1.jpg

result2.jpg



result3.jpg



result5.jpg

To conclude, the advantages and disadvantages of these edge detection methods are shown below.

pros	**cons**
<u>1st order edge detection</u>	
- simple to implement	- sensitive to noise (amplifies noise)
- detect edges and orientations	- not accurate
<u>2nd order edge detection</u>	
- detect edges and orientations	- very sensitive to noise (amplifies noise)
- fixed characteristics in all directions	- sophisticated
	- dependent on some of the existing edges
	- Malfunctioning at the corners, curves etc.
<u>Canny edge detection</u>	
- relatively unaffected by noise	- complex computation
- good localization and response	- time consuming (**x** real-time application)

Before implementing edge detection, we can amplify the edges by edge crispening. It can turn the edges of the image to become more distinct and easy to be detected.

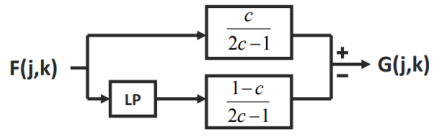
Canny edge detection can produce a very clear edge map and discard useless information of the image. For 2nd order edge detection, it produces a detailed edge map with thinner edges. If we want to implement edge detection in a shorter computation time, we can choose 1st order edge detection since it is a simple and not complicated algorithm. We can see that different edge detection methods have their advantages and disadvantages. We can choose a suitable method for different conditions.

Part b

Please design an algorithm to obtain the edge map of sample2.jpg as best as you can. Describe the steps in detail and provide some discussions.

Steps

1. Edge cripsening
   - 3x3 all-pass filter 1
   - 3x3 low-pass filter
   - c = 5 / 6



$$G(j,k)=\frac{c}{2c-1}F(j,k)-\frac{1-c}{2c-1}F_L(j,k),\quad where\quad \frac{3}{5}\le c\le\frac{5}{6}$$

$$H=\begin{bmatrix}0 & -1 & 0\\ -1 & 5 & -1\\ 0 & -1 & 0\end{bmatrix}$$

2. Edge detection
   - Prewitt operator
   - threshold = 11

   i. Calculate the row and column gradient respectively.
   ii. Calculate the gradient of the pixel and check if it is larger than the threshold value (t=11). If true, mark it as an edge pixel.

Discussion

I have tried all of the edge detection methods above and the result of the Prewitt operator is better than the others. Since the edges of the result of the Prewitt operator are simple and well-connected, the contour of the building is clear. The noise in the edge map is much less than the others.

When applying different edge detection methods to the sample2.jpg, we found that the obstacle of edge detection is adjusting the threshold value since the visibility of sample2.jpg is low and the edges are unclear.

For 1st order edge detection, it is much easier and faster to implement as the threshold value is easy to adjust.

However, the threshold value is very difficult to be determined when implementing 2nd order edge detection.

For canny edge detection, the threshold values are 22,110 in sample1.jpg. Nevertheless, I have to let 5,7 to be the threshold values to obtain a clear edge map in sample2.jpg. It is very difficult to choose a suitable and accurate threshold value for edge detection.

Problem 2: GEOMETRICAL MODIFICATION

(a) Please design an algorithm to make sample3.jpg become sample4.jpg. Output the result as result6.jpg. Please describe your method and implementation details clearly. (hint: you may perform rotation, scaling, translation, etc.)

Method

Use the following formula of geometrical modification to calculate the pixel value of the output image by the input image. Also, I use backward treatment to implement the geometrical modification.

- **translation**

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} u_q \\ v_p \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

- **scaling**

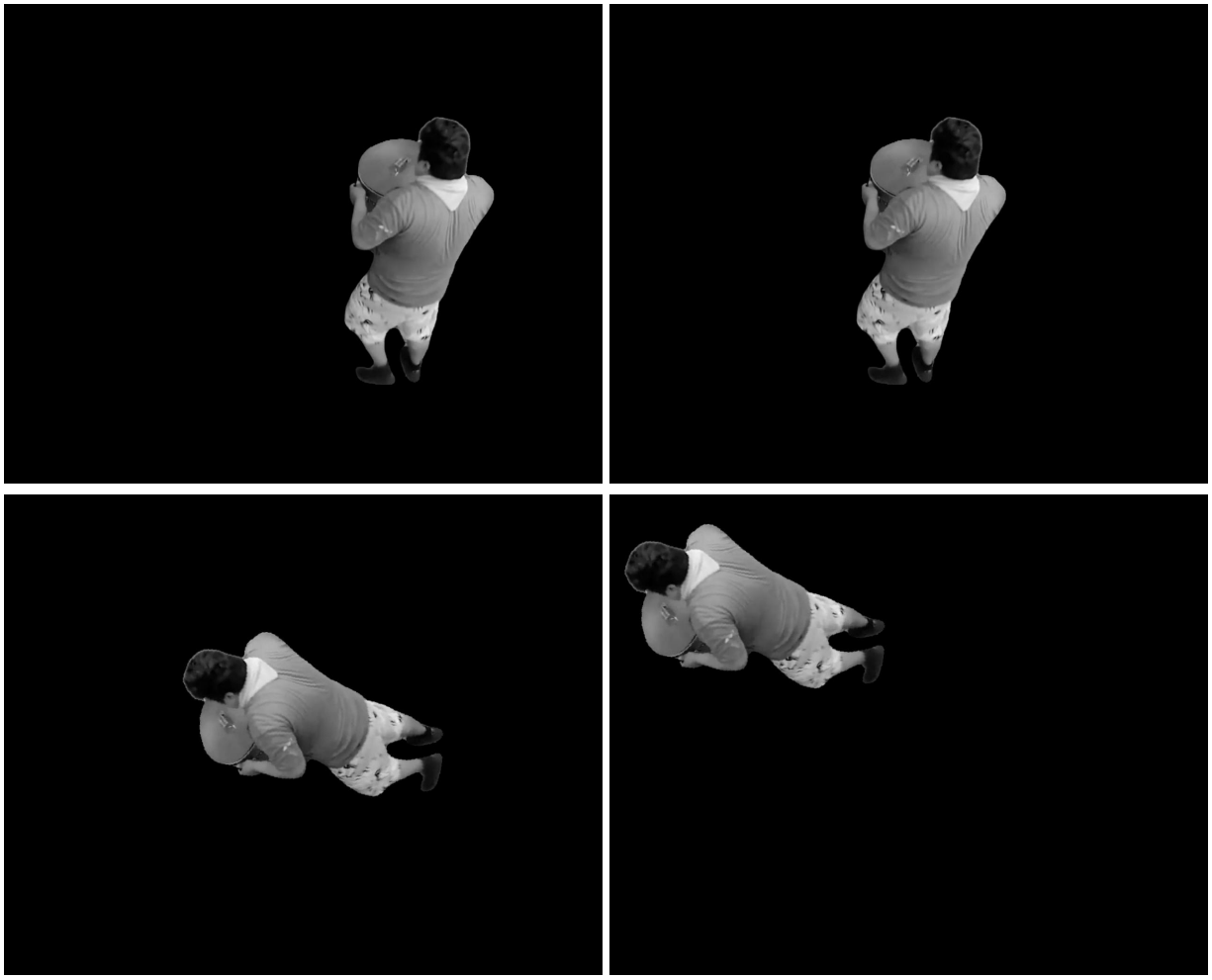
$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} u_q \\ v_p \end{bmatrix}$$

- **rotation**

$$\begin{bmatrix} x_k \\ y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_q \\ v_p \end{bmatrix}$$

Implementation details

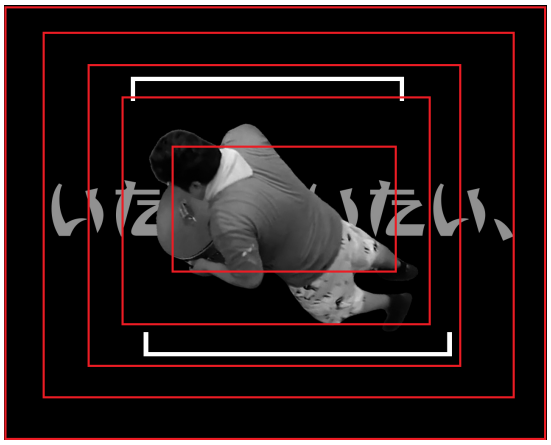
1.	translate the object to center of image	dx = -200, dy = 0
2.	rotate the object around the center of image	angle = -83 degree
3.	translate the object to upper-left region	dx=-350, dy=-230
4.	scale the object	ratio = 1.5



(b) **Imagine that there is a black hole in the center absorbing sample5.jpg. Please design a method to make sample5.jpg look like sample6.jpg as much as possible and save the output as result7.jpg. Please describe your method and implementation in detail and also provide some discussions about the designed method, the result, and the difference between result6.jpg and sample6.jpg, etc.**

Method1

Rotate the image in a specific region and keep shrinking the region at decreasing speed. The angle of rotation keeps increasing to implement the black-hole-like effect.



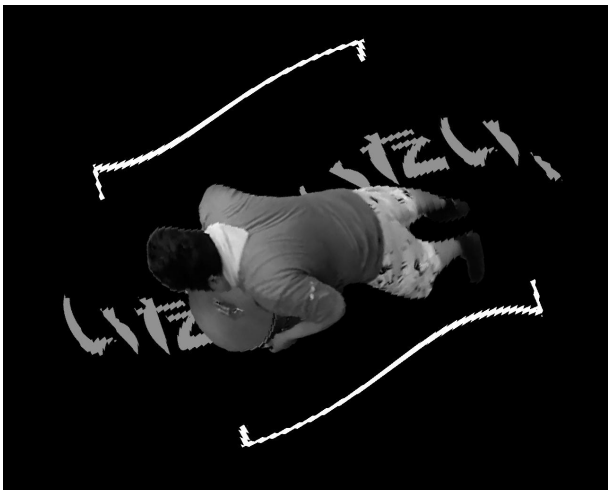
The resultant image may contain grids or slices since the rotation is region by region and the edges are not smooth enough. Moreover, over-rotated may occur in the resultant image.

Method2

Scan the image and calculate the euclidean distance between the center of the image and the current pixel. Rotate the pixel with a larger angle if the euclidean distance is small, vice versa.

It is difficult to set the ratio between the angle of rotation and euclidean distance. The whole resultant image may be over-rotated if the ratio is not suitable.

Method1



Method2

The result of the method 1 is very different from sample6.jpg since the rotation degree near the center is not enough. And the result of the method 2 is similar to sample6.jpg but over-rotated exists.

Reference / supplement:

1. https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
2. https://www.researchgate.net/post/What_are_the_differences_in_first_order_derivative_edge_detection_algorithms_and_second_order_edge_detection_algorithms
3. https://www.researchgate.net/figure/Some-advantages-and-disadvantages-of-edge-detectors_tbl1_266287401