Image Processing HW1

電機所 N26094207

王昶文

1. System Structure

My program conducts with a control panel and result section, as shown in Fig.1. Control panel allows user to choose the image operation, load/save image, set threshold, hold image, and undo operation.

Choosing operation are shown in Fig.2, you're going to get a list of all the operation written in the program, and able to choose. Then, the boxes of chosen operation needs will automatically generate, as Fig.3 shown. After loading image, the result will automatically calculate and draw on the interface, as Fig.4, 5 shown.

If user want to do more operation on the image being calculated, there's a list in the control panel, which is able to select the image user desired to remain, respect to the id below the image, as shown in Fig.6. After changing operation, the original image well become the image you chosen, and the operation will be operate on the select image. Also, if the user wants to save image, it's the same to use this box to save the image user wants.

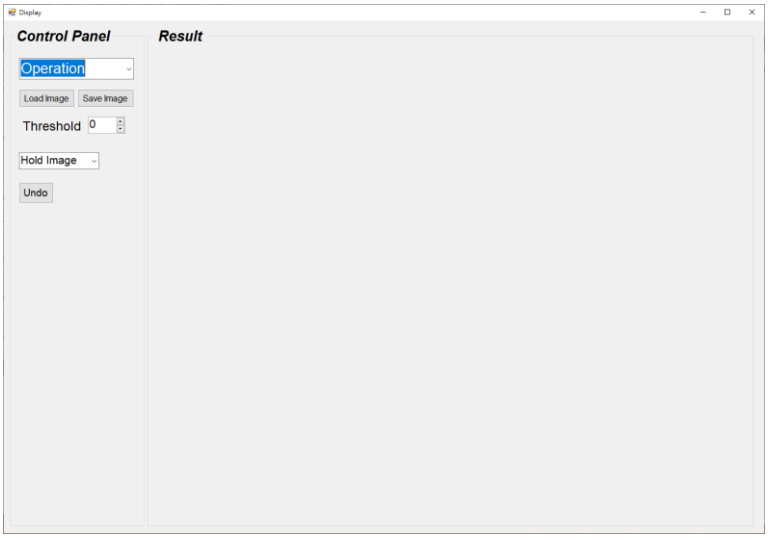


Fig.1 Interface

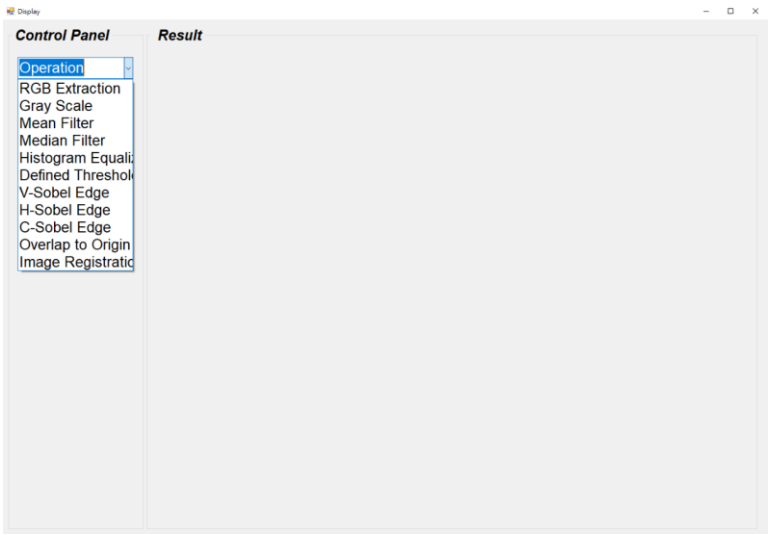


Fig.2 Operation choosing

Fig.3 Operation chosen

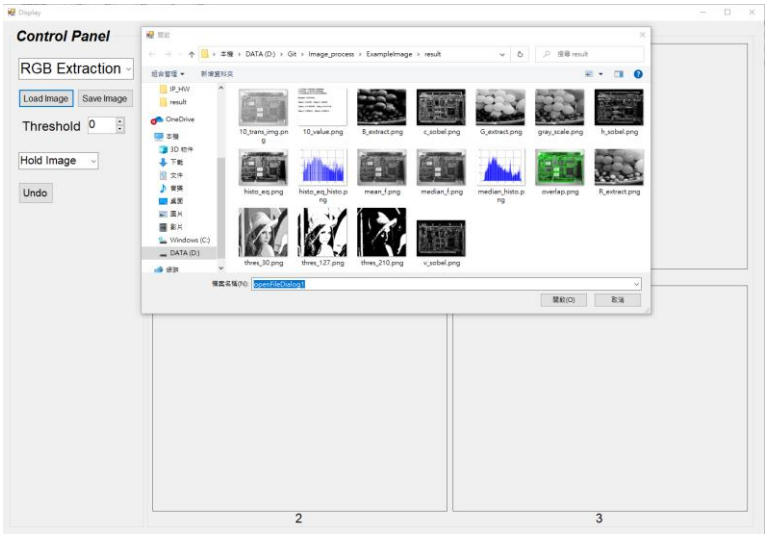


Fig.4 Select image

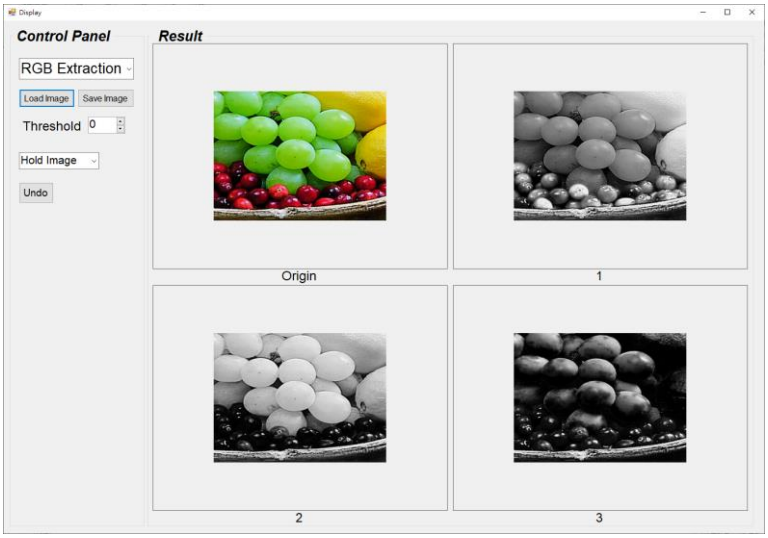


Fig.5 Result

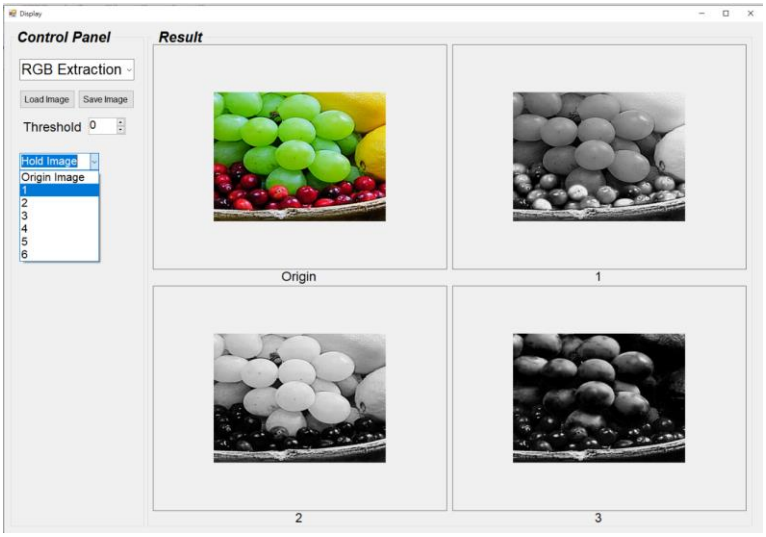


Fig.6 Hold image

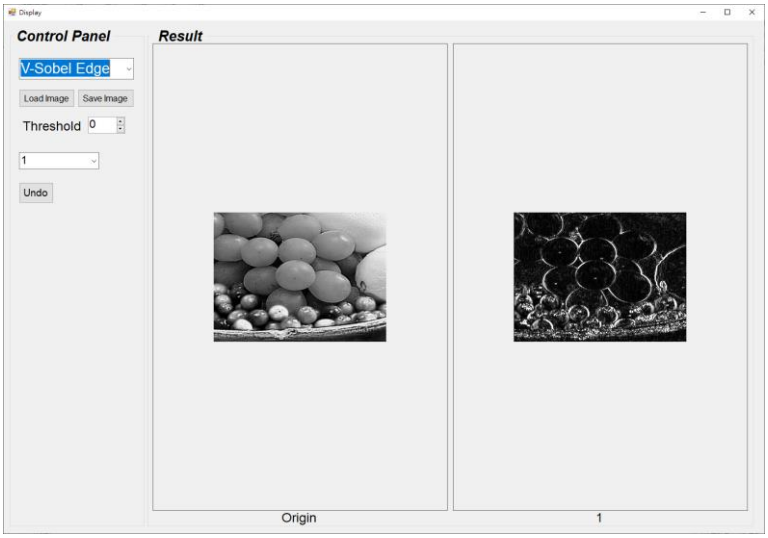


Fig.7 Change operation

2. RGB Extraction & Transformation

2.1 Problem

Extract R, G, B channel from color image and transform it to gray scale.

2.2 Methods

For the image being loaded into the program, it's is divided into 3 values, R, G, and B. In order to do R, G, and B extractions, we simply paste R value to G and B to create R extraction image, which is in gray level. We do the same thing to G and B for their extractions.

If we want to transform RGB image to gray scale, we need to calculate a value considering R, G, and B values. I use the lightness method in [1], which defined gray value as:

$$gray\ value = \frac{\max(R, G, B) + \min(R, G, B)}{2},$$

and the result of lightness method it better than average of R, G, and B.

2.3 Result

Original

R Extraction

G Extraction

B Extraction

Gray level image

3. Smooth Filter (mean and median)

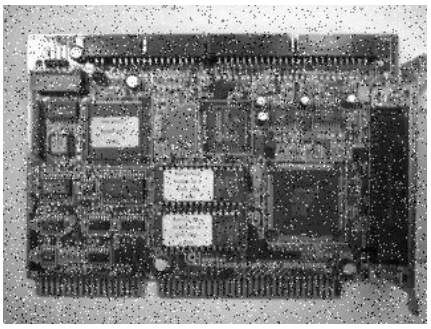
3.1 Problem

Add mean and median filter

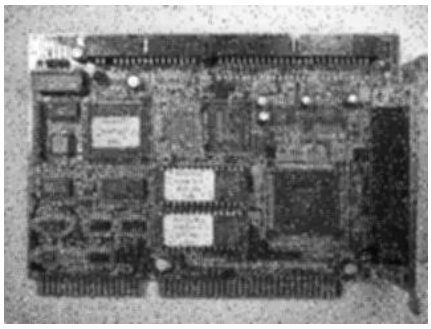
3.2 Methods

For both mean and median filter, I set a window of 3 by 3, and decide the average or median of the values inside the window respectfully. For those at the edges, which has values missing in the window, I fill in the value of the closest pixel. For example, if we want to filter pixel(0, 0), but missing pixel(-1, -1), (-1, 0), (-1, 1), (0, -1), (1, -1), then we copy pixel (0, 0) to (-1, -1), (-1, 0), (0, -1), (0, 1) to (-1, 1), (1, 0) to (1, -1). After the operations, the value we calculated we be the new value of the pixel.

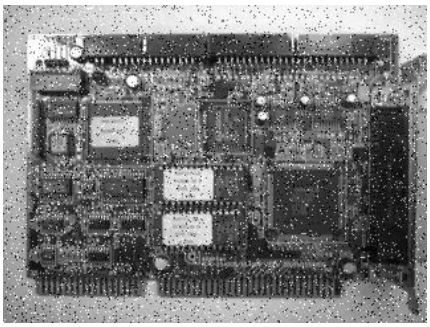
3.3 Result



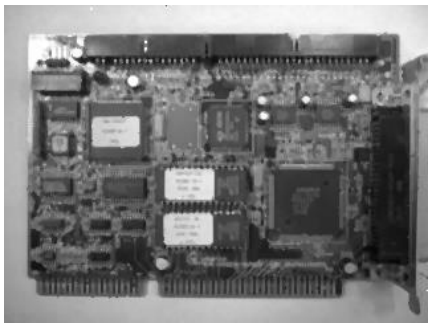
Original



Mean Filter



Original



Median Filter

3.4 Discussion

Mean filter has poor performance on salt-and-pepper noise. My deduction is that due to the average, the value can be easily interfered by the salt and pepper, indicate 0 and 255. For the median filter, 0 and 255 will be the bounded value, and won't interfered with the median, therefore, it can have better result than mean filter.

4. Histogram Equalization

4.1 Problems

Implement histogram equalization

4.2 Methods

First, I count the number of the gray levels in the picture, in order to draw the histogram of the original image. Then, I accumulate the values in each value, respect to the formula of eq. (1).

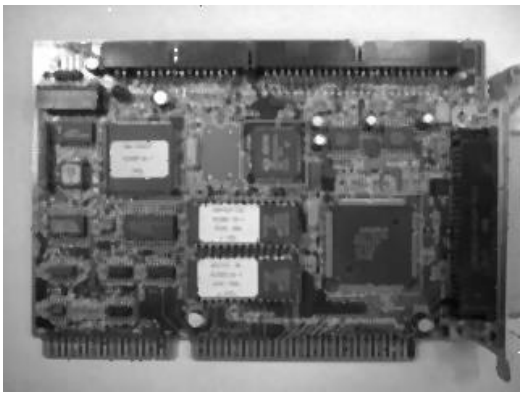
$$h_{acc}(i) = \sum_{j=0}^{i} h(j) \qquad (1)$$

with the new histogram statistics, we are able to equalize the distribution of image histogram and create the image of histogram equalization using eq. (2).

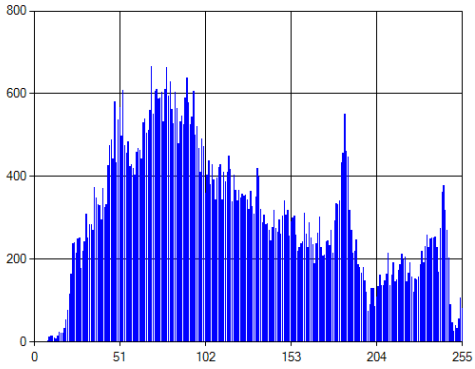
$$value_{gray}(i) = \frac{h_{acc}(i) - value_{min}}{value_{max} - value_{min}} \times 255 \qquad (2)$$

for $value_{min}$ equals to the minimum gray value whose $h_{acc} \neq 0$, and $value_{min}$ equals to the maximum gray value whose $h_{acc} \neq 0$.

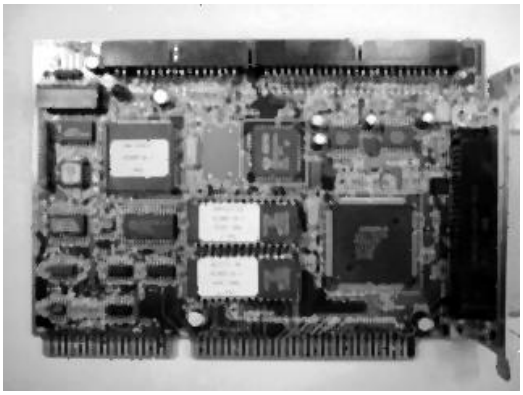
4.3 Results



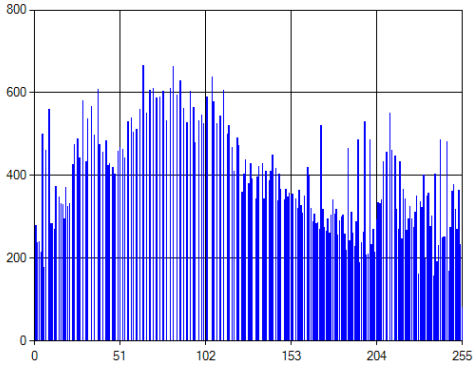
Original



Histogram



Histogram Equalization



Histogram

4.4 Discussion

After histogram equalization, we are able to make the distribution of the image more equal, therefore, the image is brighter and clear in complex areas.

5. A User-defined Thresholding

5.1 Problems

Set a threshold t, if $t \geq 255$, pixel will be set to 0, otherwise set to 1

5.2 Methods

It's obvious to calculate the threshold image, I use for loop to judge whether each pixel is greater or smaller than threshold, shown as eq. (3).

$$value(x) = \begin{cases} 255, & x \geq t \\ 0, & x < t \end{cases} \qquad (3)$$

5.3 Results

Original

Threshold=127

Threshold=30

Threshold=210

6. Sobel Edge Detection

6.1 Problems

Apply vertical, horizontal, and combined sobel edge filter to original image.

6.2 Methods

For the sobel filter [1], we need to pre-define the sobel filter, whether vertical or horizontal. When applying to the image, we need to do convolution to the image window. The vertical sobel filter is define as eq. (4), horizontal filter is defined as eq.(5).

$$sobel_v = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad (4)$$

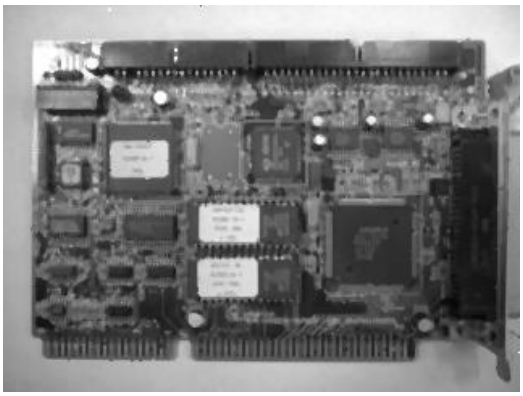
$$sobel_h = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (5)$$

After convolution, we'll get the value which is used to update the center of the window, but there's positive and negative values, both represent the edges of the picture. Therefore, we need to take absolute value of the value in order to get the negative edges. Last, if the edge is greater than 255, we have to set it to 255. Some use normalization to deal with the greater edge, which the method can show the difference of the value between 255 and maximum value, but this require extra memory, so it's not taken in this program.

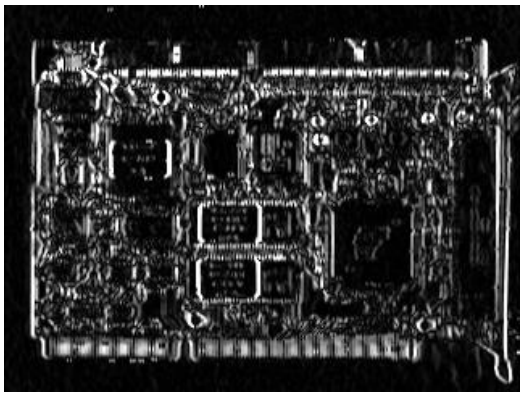
To combine vertical and horizontal sobel filter, we have to use eq. (6) to get the combine sobel filter value, due to sobel filters is operated in frequency domain, but we are in time domain.

$$value_c = \sqrt{\frac{value_v^2 + value_h^2}{2}} \quad (6)$$

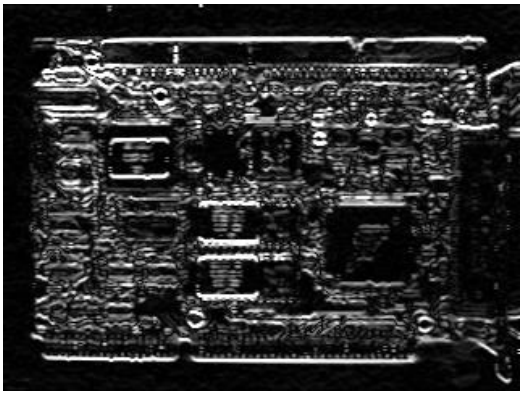
6.3 Results



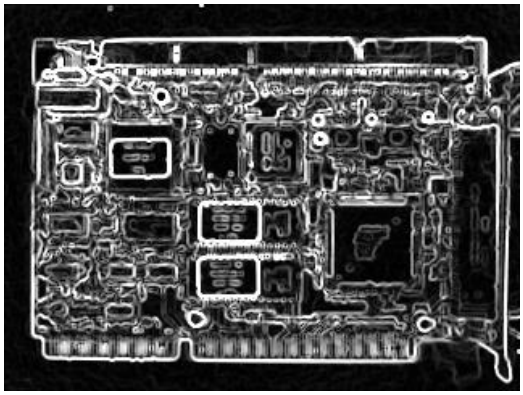
Original



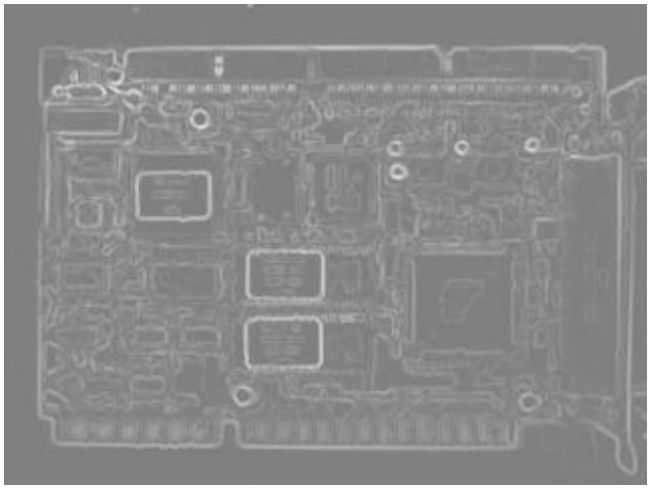
V Sobel



H Sobel



Combine Sobel



Combine Sobel using normalize

6.4 Discussion

For vertical and horizontal sobel filter, it's easy to see that the vertical edge and horizontal edge is easy to be notified. For combine sobel filter, I tried normalizing the values, we can see that there's strong and weak edges, which is not able to show in the ones not using normalize. Though it didn't show the intensity of the edges, the ones not using normalization is easier for human to see, there's benefits for both methods.

7. Edge Overlapping

7.1 Problems

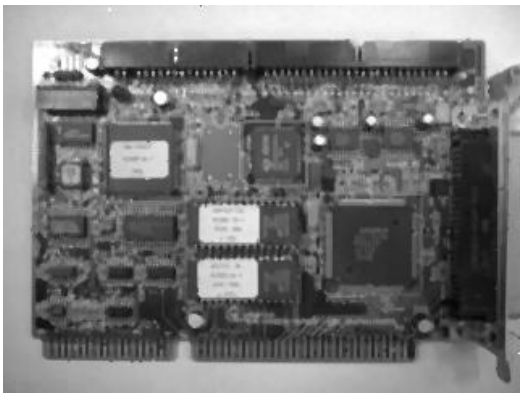
Use threshold to filter the combine sobel filter image and take it as a mask for the original picture. Then merge them together and get the result.

7.2 Methods

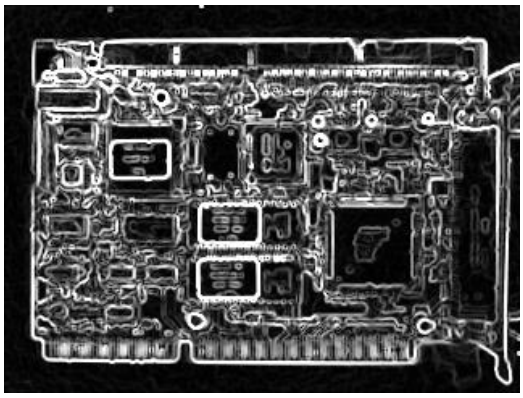
My method is quite simple, after getting the image of combine sobel filter, I do the same operation of Q5, and get the mask of edges. If the mask is 1 at

pixel (i, j), I set the pixel (i, j) in original picture to (0, 255, 0)respect to R, G, and B. After the operation, I'll get the image masked with edge detection.

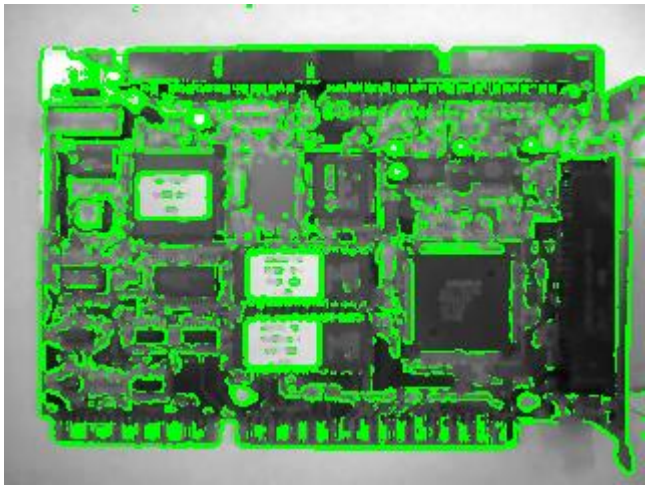
7.3 Results



Original



Combine Sobel with threshold=127



Overlapping

7.4 Discussion

By using different threshold, the edges are being filter out in different level, and we are able to extract more significant edges using higher threshold, vice versa.

8. Image Registration

8.1 Problems

To align the original image with the image already being spun, and calculate the angle, magnification, translation, and shearing.

8.2 Methods

For finding the alignment, we need to set anchor points, in this program, we

set the number of anchor points to 4, and the two anchor points, located separately on original/target image has to be corresponded. Therefore, we'll have 4 relations between the anchor points, which is then used to calculate the transformation matrix between two images.

$$ATr = B \qquad (7)$$

$$Tr = A^{-1}B \qquad (8)$$

The relation between the image is can be present as eq.(7), and our target is to find the transformation matrix, Tr, which can be found simply using A's inverse matrix, shown in eq.(8). However, A is a 4x3 matrix, which is not a square matrix, so we need to find it's pseudo matrix, defined as eq.(9). After a series of calculation, we can get the transformation matrix.

$$A^{\dagger} = (A^T A)^{-1} A^T \qquad (9)$$

In addition, we need to calculate the parameter of transformation matrix, the rotation angle theta, the magnified scale, the shifting of image, and the shearing of the image. After calculation, the matrix can be indicated as eq.(10)

$$Tr = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix} = \begin{bmatrix} C_x(S_v sin\theta + \cos\theta) & C_x(S_v cos\theta - \sin\theta) & t_x \\ C_y(S_h cos\theta + sin\theta) & C_y(cos\theta - S_h sin\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (10)$$

for θ is the rotate angle, C_x, C_y is the magnified scale, t_x, t_y is the translation, respective to x, y axis, S_v is the shearing on vertical direction, and S_h is the shearing on horizontal direction.

Since we have 7 variables need to solve, but we only have 6 equations, we aren't able to find the specific answer. Alternatively, I decided to calculate both theta if I set both $S_v = 0, S_h = 0$ separately. Then, I take the average of both thetas and set it as the angle I calculate, as shown in eq.(11). We'll get the value of rest of the variables, using the eq.(12), (13), (14), (15), (16), (17). To calculate the intensity of the image, I use the formula in ppt to calculate.

$$\theta = \left(\tan^{-1} - \frac{T_{12}}{T_{11}} + \tan^{-1}\frac{T_{21}}{T_{22}}\right)/2 \qquad (11)$$

$$S_v = (T_{11} sin\theta + T_{12} cos\theta)/(T_{11} cos\theta - T_{12} sin\theta) \qquad (12)$$

$$S_h = (T_{21} cos\theta - T_{22} sin\theta)/(T_{22} cos\theta - T_{21} sin\theta) \qquad (13)$$

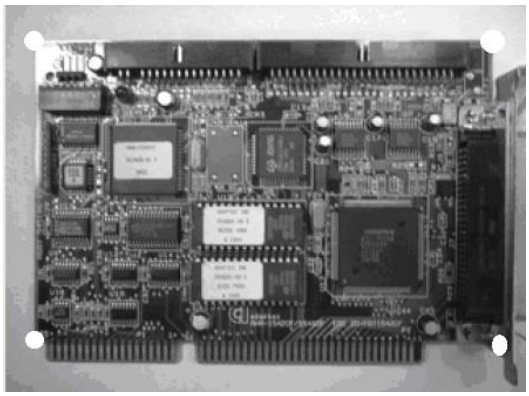
$$C_x = T_{11}/(cos\theta + S_v sin\theta) \qquad (14)$$

$$C_y = T_{21}/(sin\theta + S_h cos\theta) \qquad (15)$$

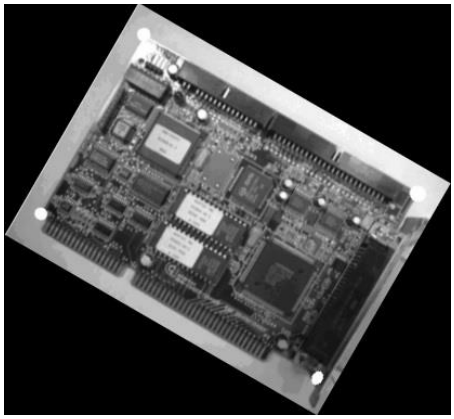
$$t_x = T_{13} \qquad (16)$$

$$t_y = T_{23} \qquad (17)$$

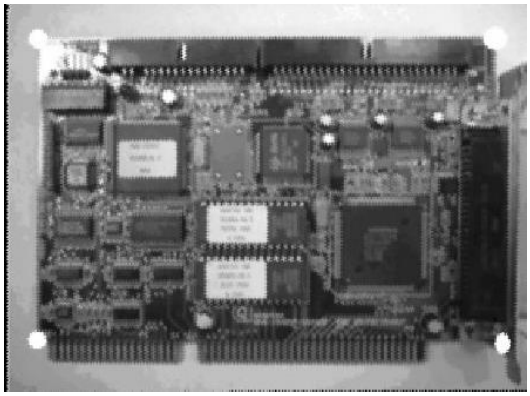
8.3 Results



Target Image



Need to transform image



After transformation

| 1.154191, -0.672378, -0.000000|
| 0.666224, 1.158580, -0.000000|
| -141.111572, 83.435410, 1.000001|

Rotation: -30.061539

Scale x: 1.332670 Scale y: 1.339551

Trans x: -141.111572 Trans y: 83.435410

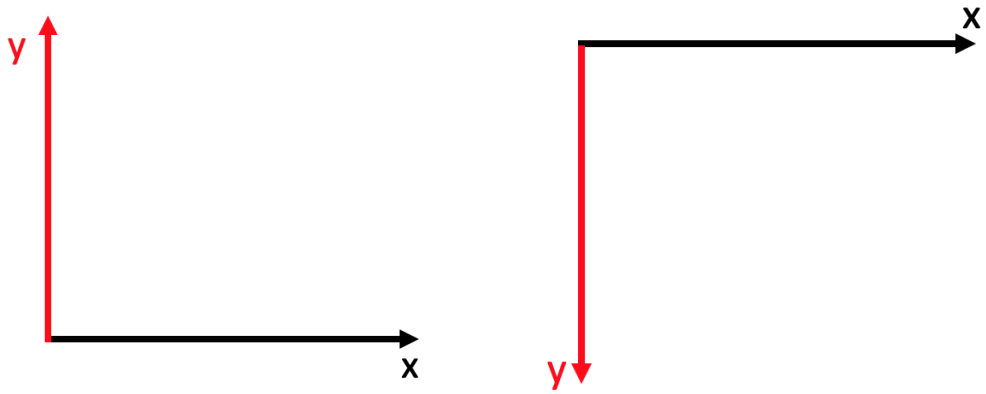
Shear v: -0.001170 Shear h: -0.001170

Intensity: 16.659559

Result printed

8.4 Discussion

For the result, we can see that the image is more transparent than original, that is because the image is being magnified, and there's some space between them, causing the image showing the color of background. Also, the rotation angle is a little bit strange, because in order to rotate the image back to the target image, it need to rotate 30 degree, not -30 degree. After analysis, I figure out that in original coordinate system, direction of y axis is pointing upward on x axis, but in my image coordinate system, I set the y axis direction the opposite, causing the direction of rotation, y transparent, and vertical/horizontal direction miss a minus sign, illustrate as follow.



Original coordinate Image coordinate

9. Undo

In order to undo the image process, I create the class named doc, which record last image and operation. If the user clicks undo button, program will get the previous image operating the previous operation.

10. Conclusion

This program allows me to get use to window form, and get better on traditional image processing technics, which I think it's very unique and useful.

11. Reference

[1] https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/