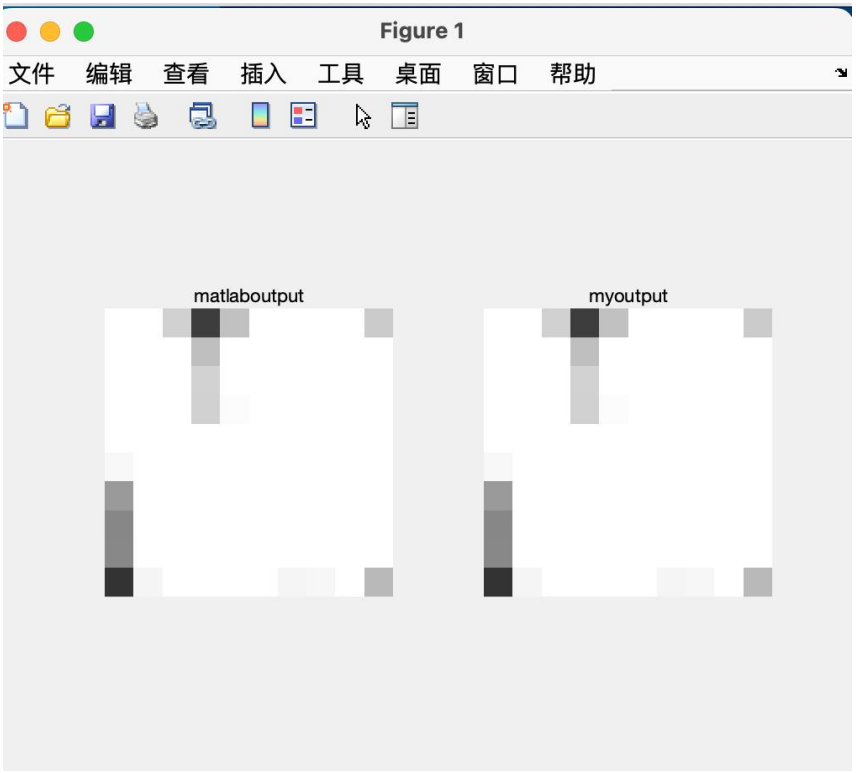
IU Lab2 Report

P1

My result is equal to MATLAB's imfilter function:



P2

- **Smoothing Filters:**

1. The box Filter is a 3*3 normal smooth filter which arrange average of surrounding pixels to the center pixel with a square size. It is more suitable to process the scene image.
2. The weighted Box Filter gives the window's pixels different weights. It makes the image more smoother than box filiter.
3. The horizontal Filter considers only horizontal pixels, it is used to detect the horizontal edge.
4. The vertical Filter considers only vertical pixels, it is used to detect the vertical edge.
5. The circular-Shape Filter is a normal smooth filter which arrange average of surrounding pixels to the center pixel with a circular size. It is more suitable to process the facial image.
6. The 5 × 5 Box Filter is a 5*5 normal smooth filter which arrange average of surrounding pixels to the center pixel with a square size. It has a larger window size. It has a more strong smooth effect on the image. It is more suitable to process the scene image.
7. The 5 × 5 Circular Shape Filter is a 5*5 normal smooth filter which arrange average of surrounding pixels to the center pixel

with a Circular size. It has a larger window size. It has a more strong smooth effect on the image. It is more suitable to process the scene image.

8. For the Lenna image, the 5 × 5 Circular Shape Filter performs the best. Because it is a facial image. Circular Shape Filter makas it focus on the facial part.
9. For the Traffic image, the weighted Box Filter performs the best. Because weighted Box Filter assign different weighted.
10. The difference between the box filter, and the weighted box filter is one arrange average of surrounding pixels to the center pixel, and another one arrange different weighrs of surrounding pixels to the center pixel. The weighted box filter makes the image more smoother.
11. Following are the image result:

bf.Lenna

wbf.Lenna

hf.Lenna

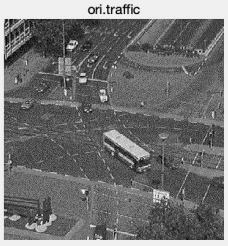
vf.Lenna

csf.Lenna

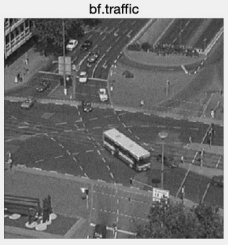
bf55.Lenna

csf55.Lenna

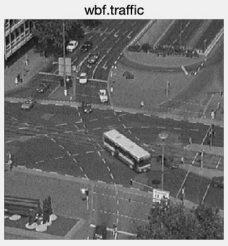
ori.traffic



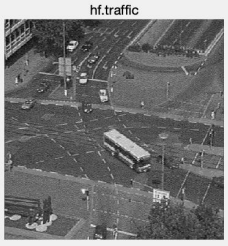
bf.traffic



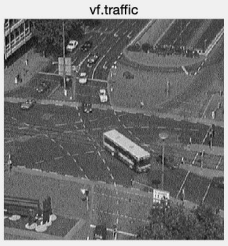
wbf.traffic



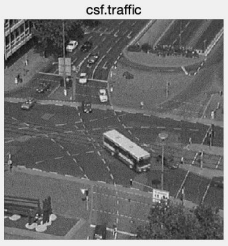
hf.traffic



vf.traffic



csf.traffic



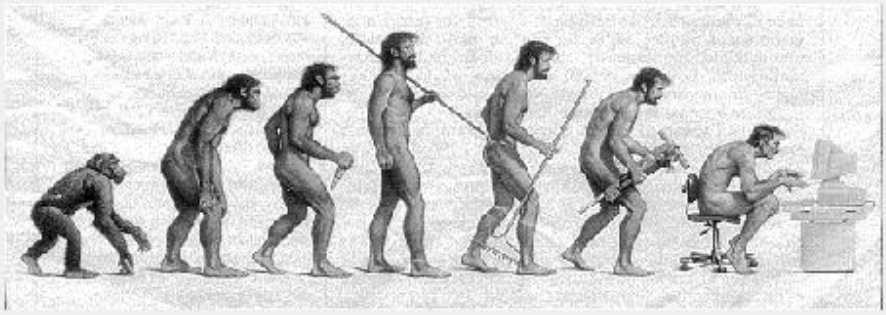
bf55.traffic

csf55.traffic

- **Enhancement Filters:**

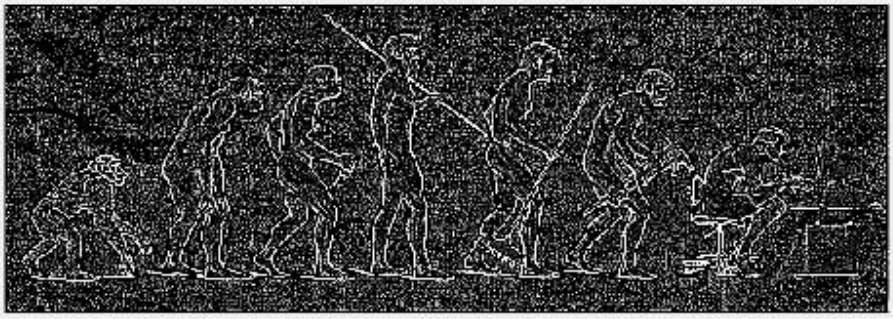
1. 4 × 4 Laplacian improve the image's high-frequency elements, such as the edges and texturing. It can sharpen the edges of the image. It always used to enhance small details in the image.
2. 8 × 8 Laplacian also improve the image's high-frequency elements, such as the edges and texturing. But it has a larger window, so it can can capture details in the image more effectively.
3. For both ImageA and ImageB, 8 × 8 Laplacian is better due to it can can capture details in the image more effectively.
4. Following are the image result:

ori.IA

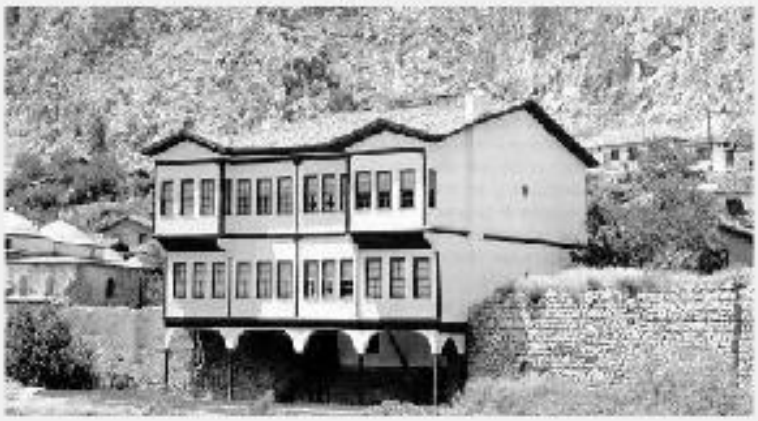


la44.IA

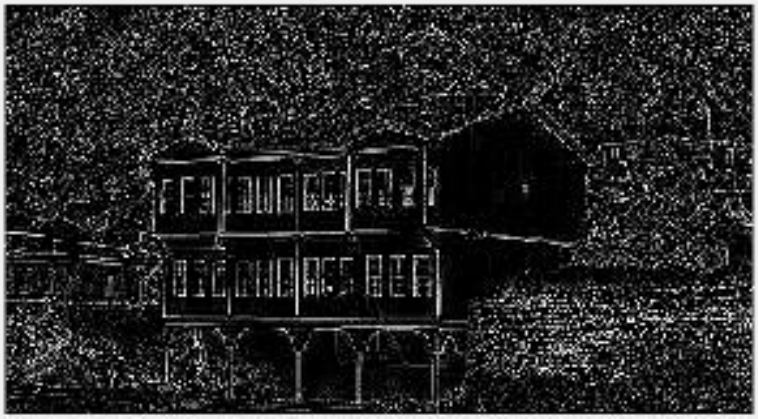
la88.IA



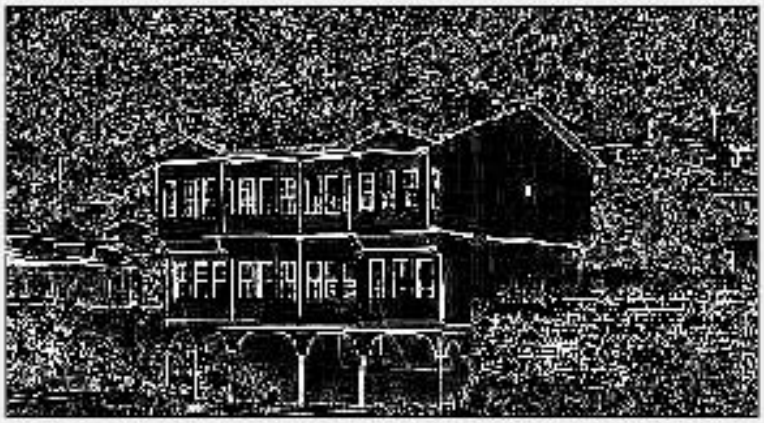
ori.IB



la44.IB



la88.IB



- **Edge Filters:**

1. For the Image A, Sobel X is the best. ImageA needs to detect vertical edges because the people stands.
2. For the Image B, Sobel Y is the best.ImageB needs to detect horizontal edges because we need to distinguish between the house and the background.
3. Weighting the center pixel makes the filter more sensitive to detect the edges, so Sobel is better than Prewitt.
4. The difference I notice between the enhancement filters and edge filters is: Enhancement filters are used to emphasize specific elements or reduce noise in order to enhance the overall quality and look of an image. Edge filters are made to recognize and draw attention to the borders or edges of objects and features in an image.
5. Here is the image result:

ori.IA

px.IA

py.IA

sx.IA

sy.IA

ori.IB

px.IB

py.IB

sx.IB

sy.IB

P3

- I use fspecial as V1 and my gauss_kernel as T1 to verity (in conidition WindowSize of 3 ,σ=0.5)

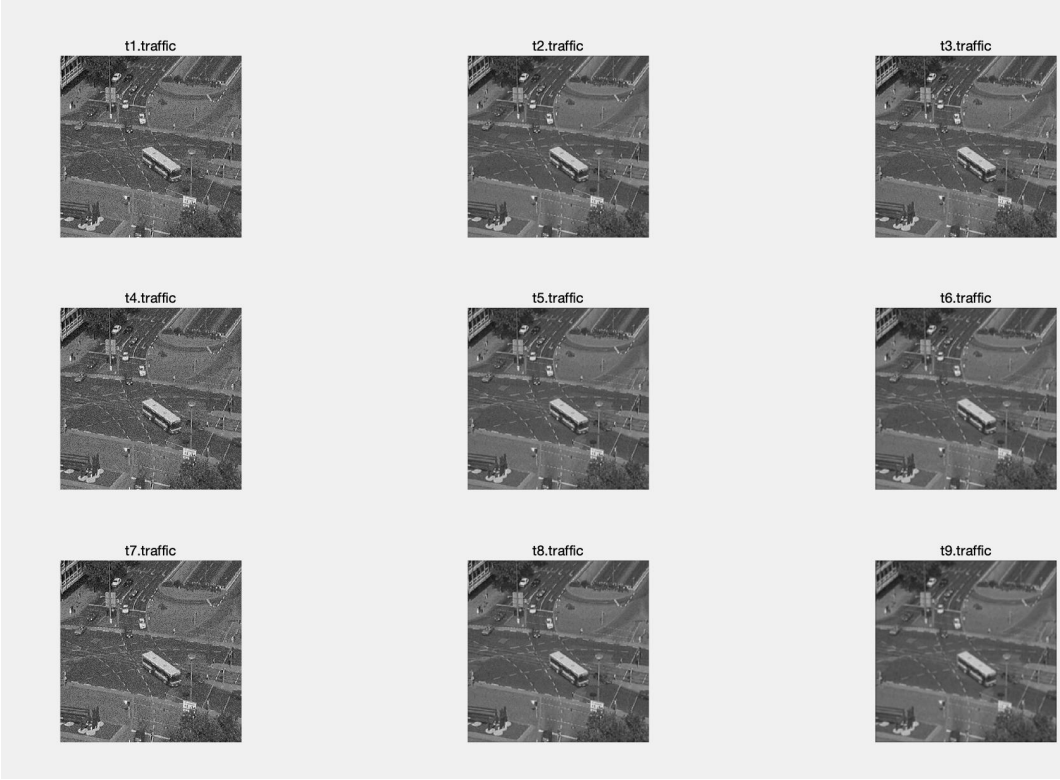
```
V1 = fspecial('gaussian',H1,s1)
T1=gauss_kernel(H1, s1)
```

```
V1 =

    0.0113    0.0838    0.0113
    0.0838    0.6193    0.0838
    0.0113    0.0838    0.0113


T1 =

    0.0113    0.0838    0.0113
    0.0838    0.6193    0.0838
    0.0113    0.0838    0.0113
```

The outcomes are the same. My function is correct!

- Report my results with these 9 combinations:

```
H1=3;
H2=5;
H3=7;
s1=0.5;
s2=1;
s3=2;

V1 = fspecial('gaussian',H1,s1);
T1=gauss_kernel(H1, s1);
T2=gauss_kernel(H1, s2);
T3=gauss_kernel(H1, s3);

T4=gauss_kernel(H2, s1);
T5=gauss_kernel(H2, s2);
T6=gauss_kernel(H2, s3);

T7=gauss_kernel(H3, s1);
T8=gauss_kernel(H3, s2);
T9=gauss_kernel(H3, s3);
```

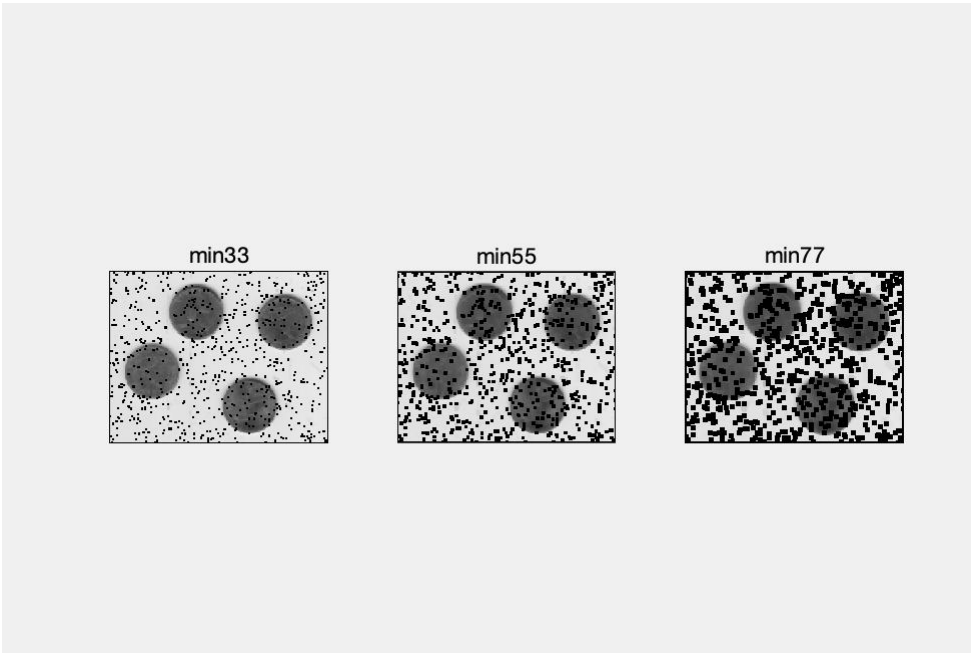


- The effect of the filter will be increasingly pronounced when the window size is increased. When determining the new value for each pixel, it will take more of the pixels around it into account.

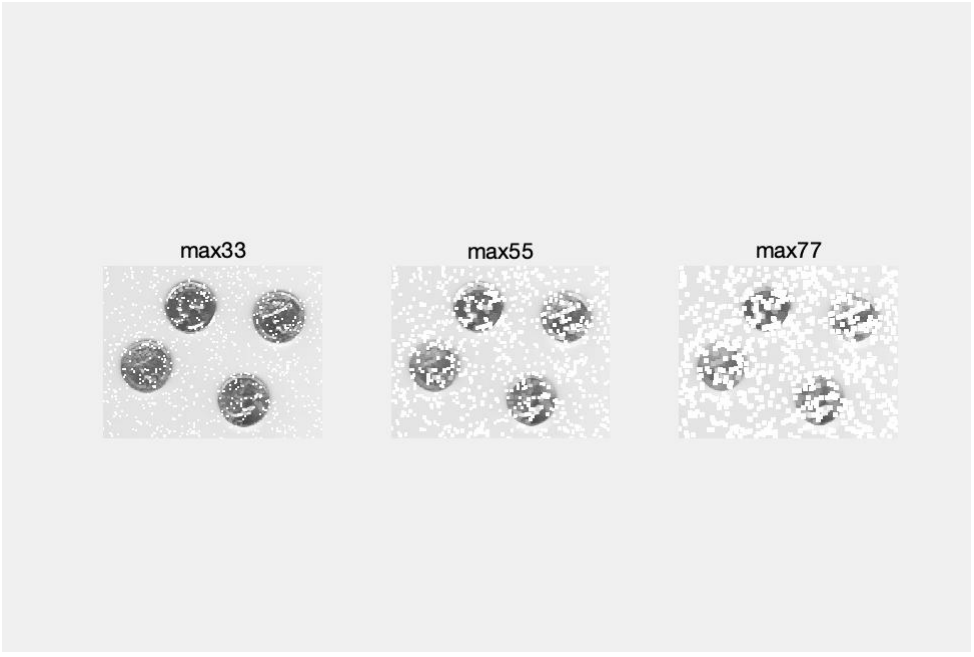
- The peak of the Gaussian distribution will get flatter as sigma is increased, giving the filter a stronger blurring effect. This implies that the filter will smooth the image more and blur the edges and features.
- Compare the Gaussian kernel with mean kernel. It makes the image more smoothed. It more effectively reduces noise and creates a soft, gradual transition between neighboring pixel values.

P4

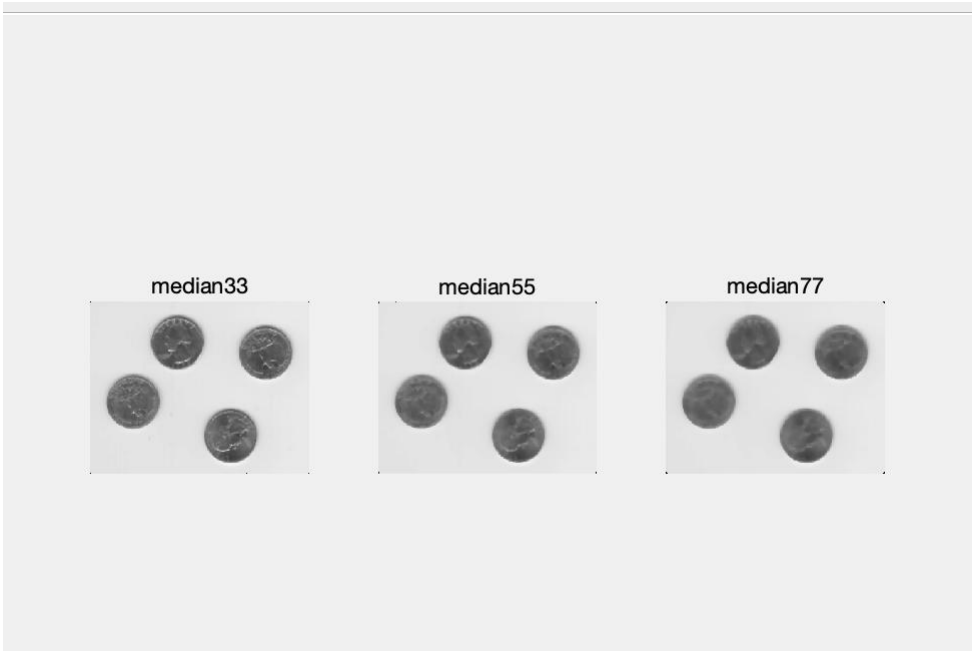
- Min filiter: 3*3 is the best, due to the noise is small.



- Max filiter: 3*3 is the best, due to the noise are small.



- Median filiter: 3*3 is the best, due to the noise is small.

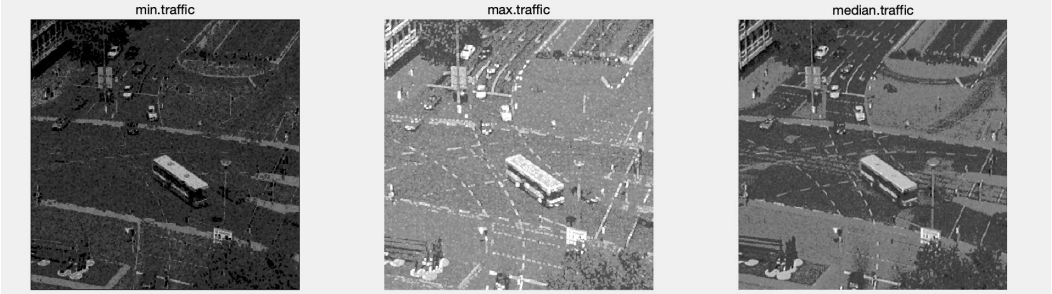


- Median Filter performs the best.

P5

- I choose 3*3 mean filiter as box filiter and Gaussian filiter with 3*3 window size and sigmoid 0.1.

- I choose 3*3 window size.

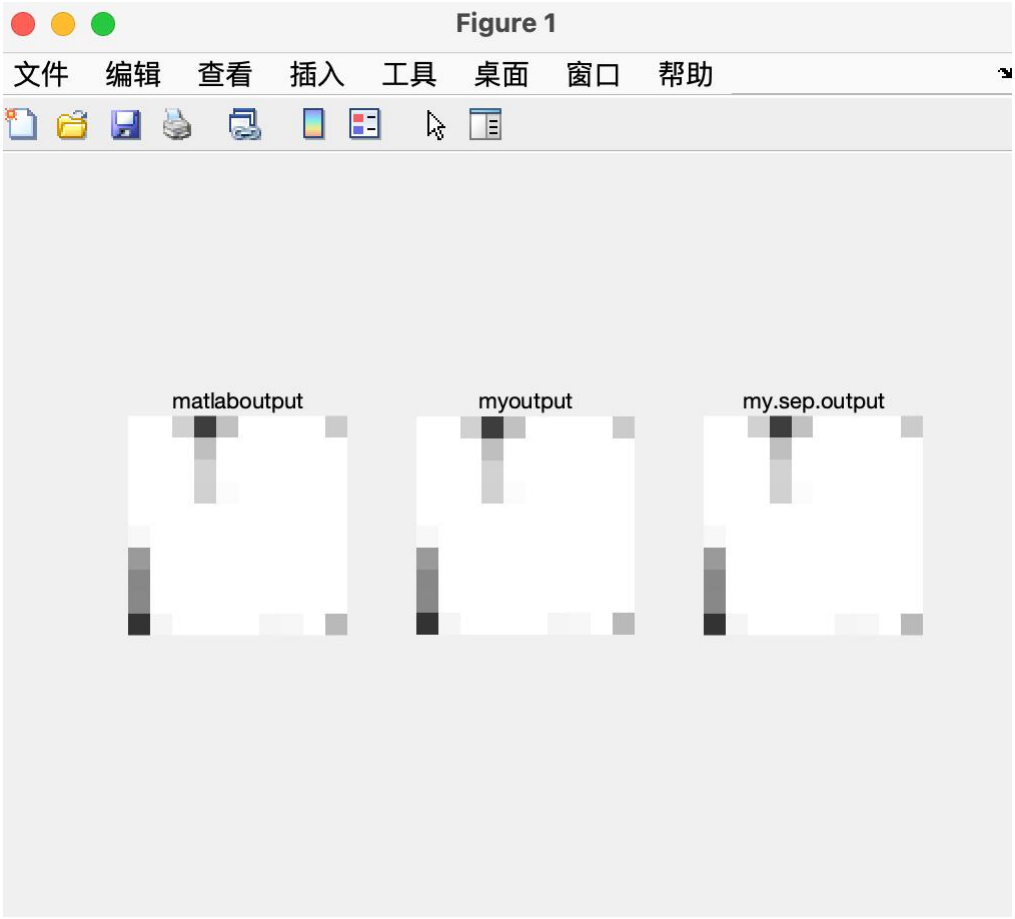


- Gaussian filter is best for white noise, because It is intended to compute a weighted average of the nearby pixels' values. In order to reduce the effects of Gaussian noise, the Gaussian filter adds a smoothing effect by assigning more weight to nearby pixels and less weight to distant ones.
- Median filter is best for salt and pepper noise, because a feature of salt and pepper noise is the sporadic appearance of extremely high or extremely low pixel values. The median filter

substitutes the median of all the pixel values in a neighborhood for each individual pixel value.

P6

- My conv separable function get the same output with matlab output and my_conv output!



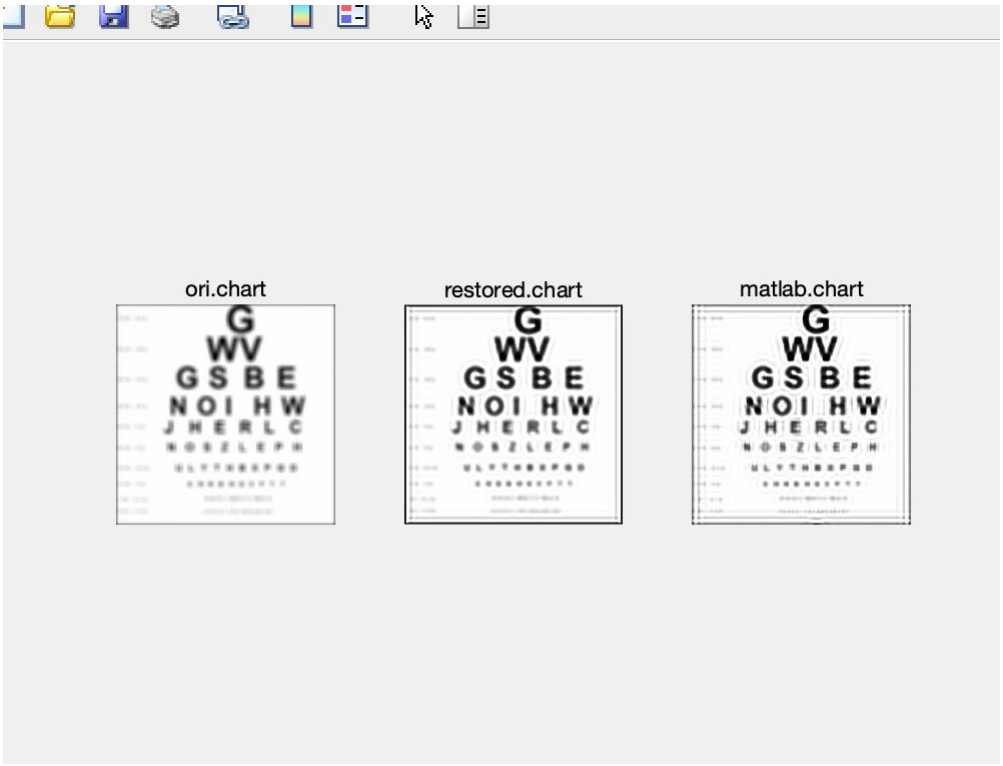
- Filters in problem 2 are separable: Box Filter; Weighted Box Filter; Horizontal Filter; Vertical Filter; 5 × 5 Box Filter; Prewitt X ; Prewitt Y; Sobel X; Sobel Y.
- A way to test whether a filter is separable is to check whether a row/column in the matrix is the multiple of the other rows/columns. If yes, it is separable.
- When the filter size is small, there isn't too much difference

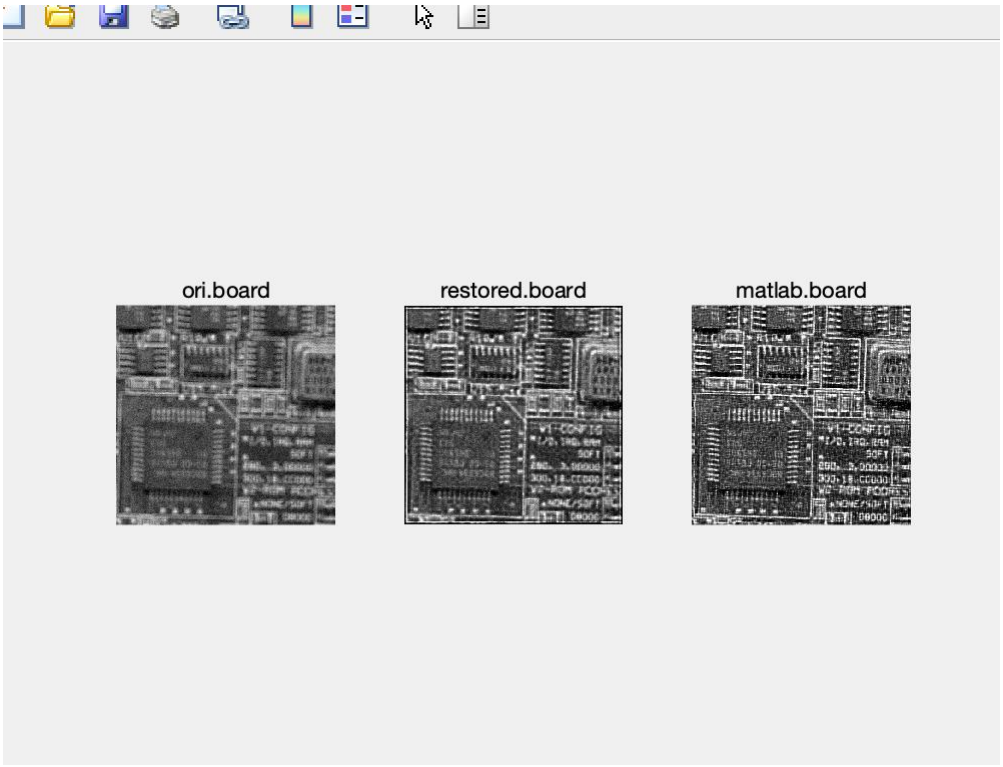
between this two function, But with the size become bigger, the time of separate convolution become less than my_conv:

```
>> P6
5 × 5 Box Filter
time for conv_separable:0.74332
time for my_conv:0.41361

5 × 5 circular shape filter
time for conv_separable:0.73836
time for my_conv:0.47144

7 × 7 Gaussian with sigma of 0.5
time for conv_separable:0.88583
time for my_conv:0.58747

17 x 17 Gaussian with sigma of 0.5
time for conv_separable:0.69751
time for my_conv:0.52455

35 × 35 Gaussian with sigma of 2
time for conv_separable:0.76873
time for my_conv:0.88981
```

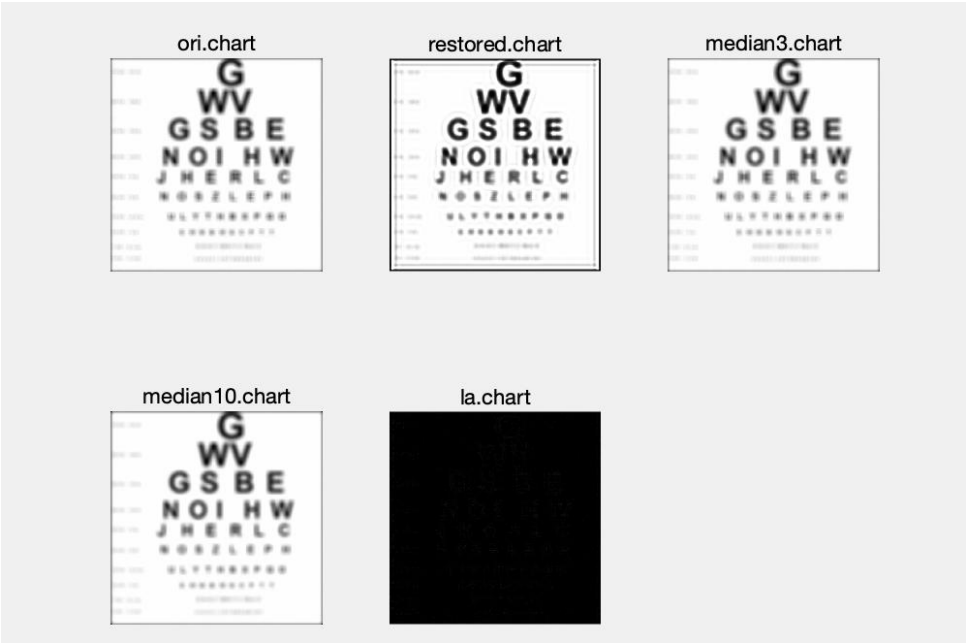
P7

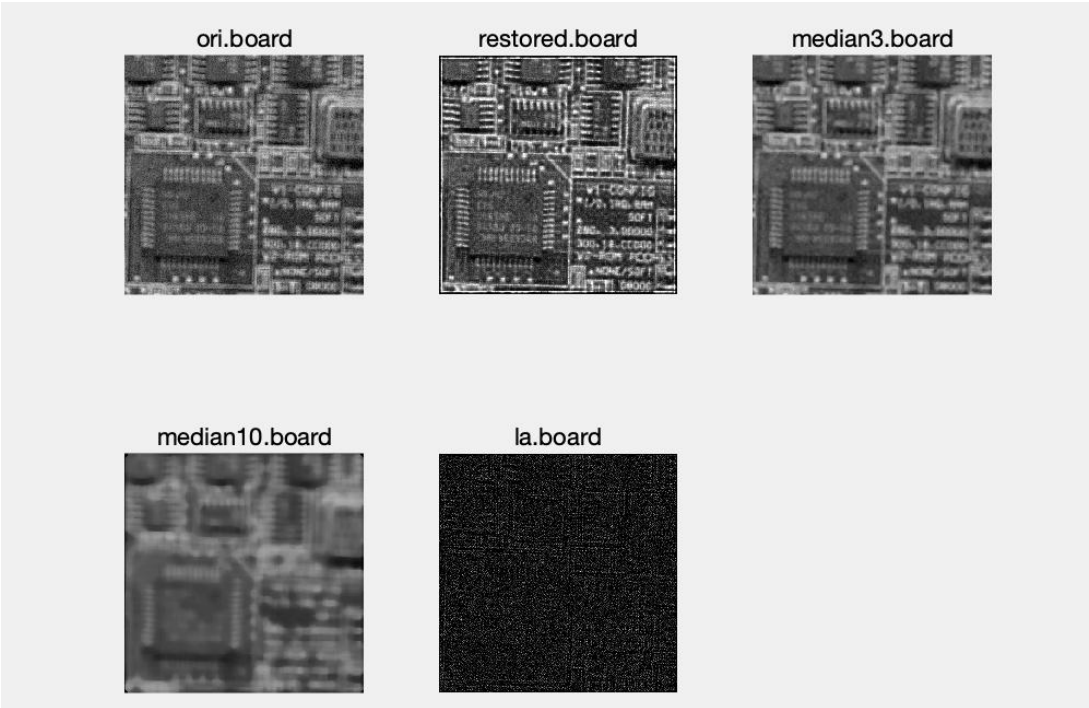
- My function's and Matlab's output matches!





- I use median filiter with window size 3 and 10. I also use 4*4 Laplacian filiter. As the result shows, Lucy-Richardson is the best. For the circuit board, smaller window size median filiter works better due to the circuit board have a lot of small details. And the Laplacian filiter is not suitable for restore these images.





- Challenge: The restored images contains some ringing due to in the process of restore, some noise in the image is amplified. I am going to use a Smoothing filiter to deal with this. I will apply a Gaussian filiter after Lucy-Richardson. As the result shows below: