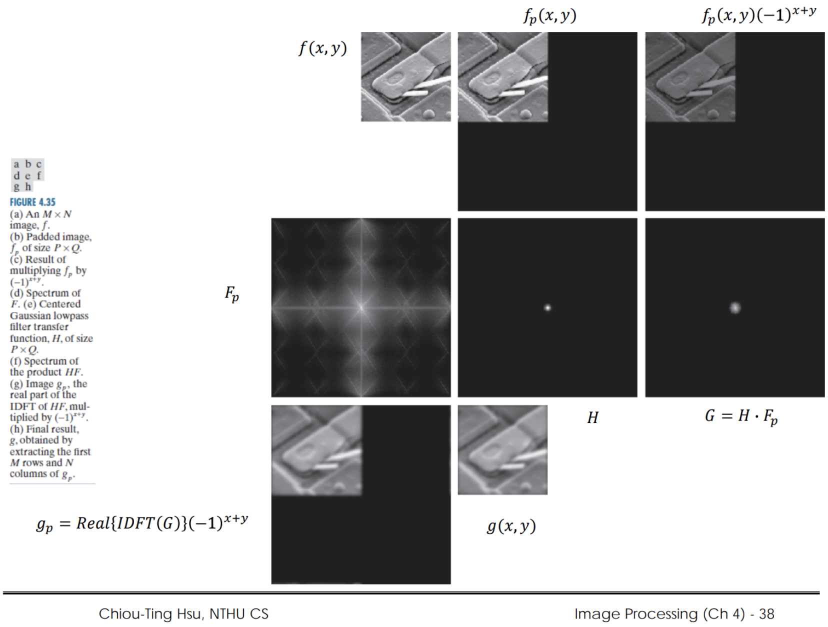
DIP Lab 3

105042015 沈冠妤 外語20

1. Proj.04-01 Two-Dimensional Fast Fourier Transform (40%)

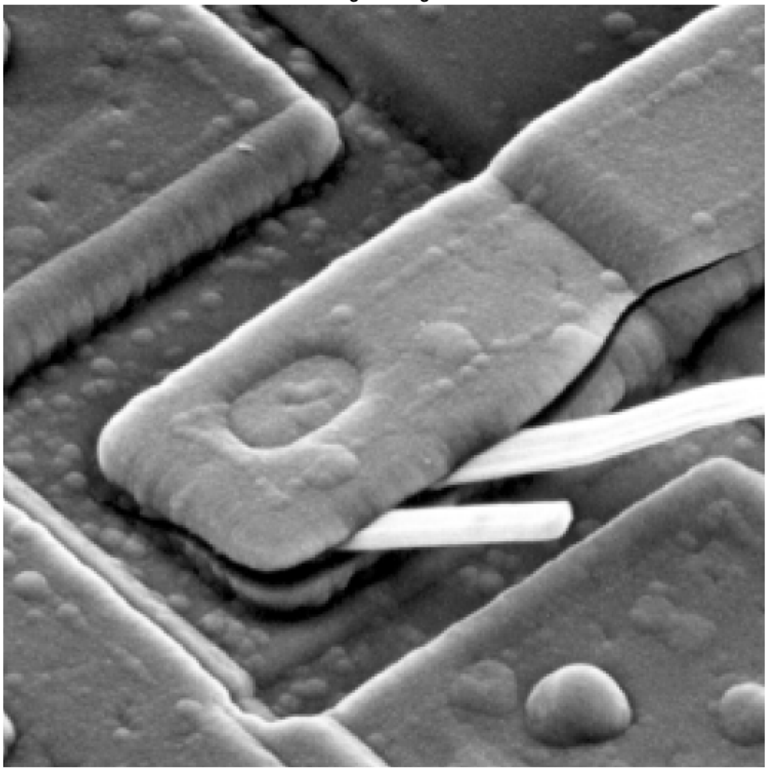
- **Explanation:**

  Do 2D-FFT manually, and repeat the process in p.38 of handout.

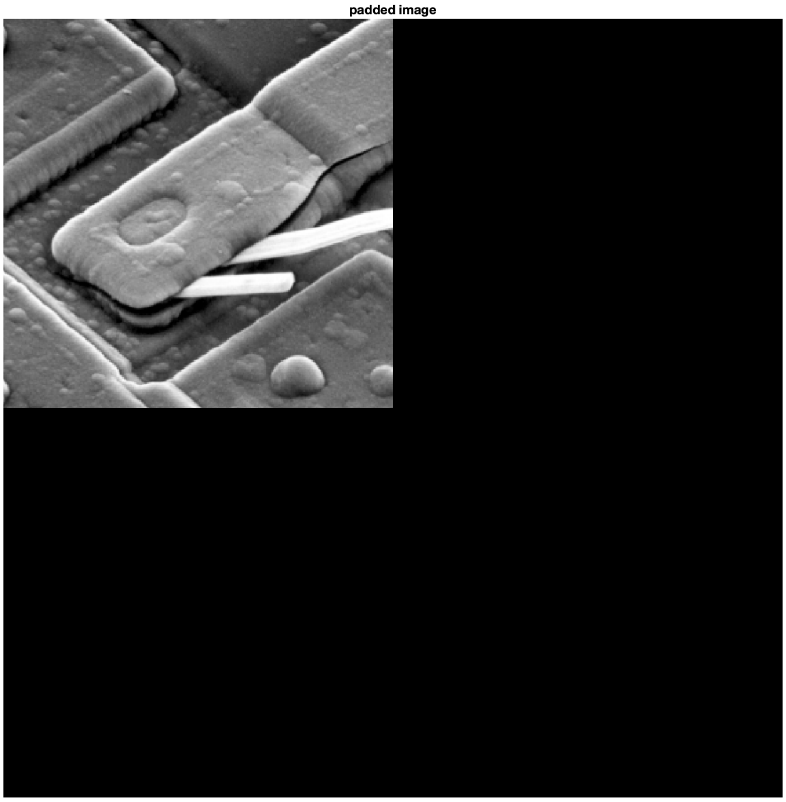


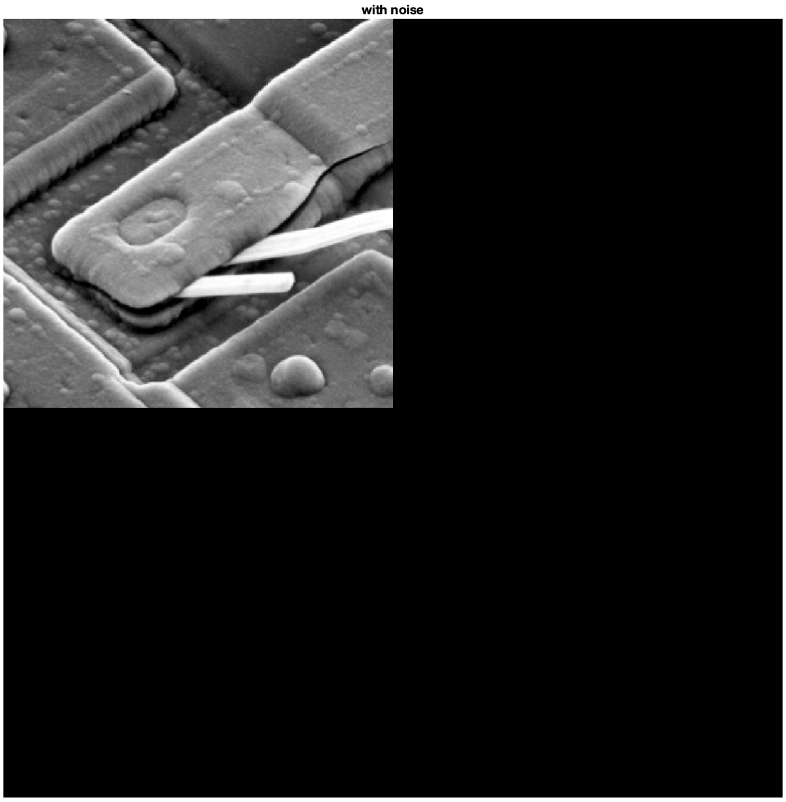
- **Result:**

original image



padded image




with noise

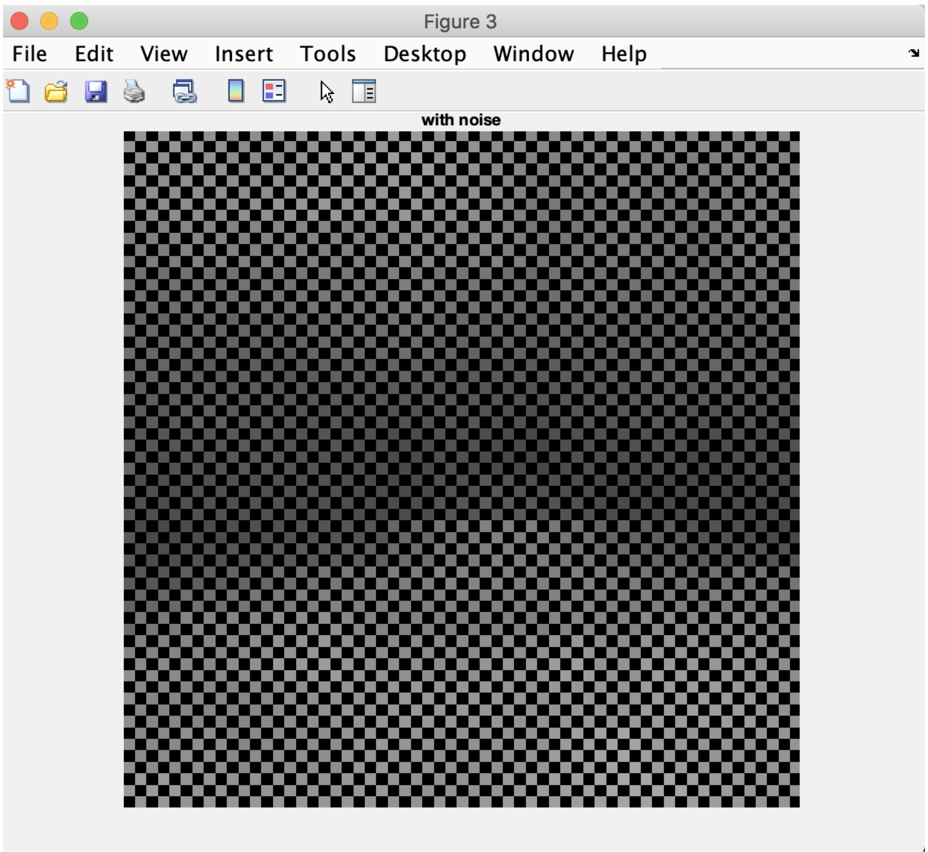
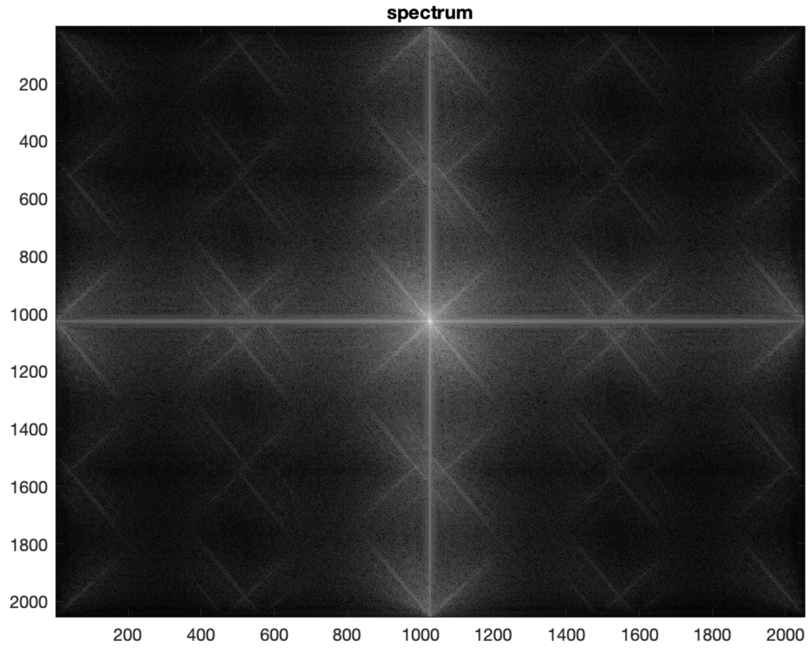
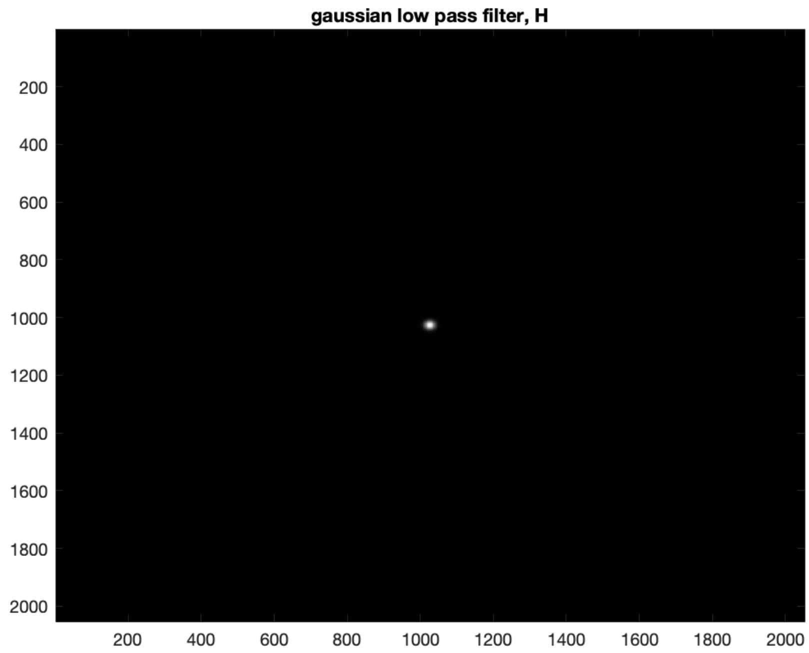
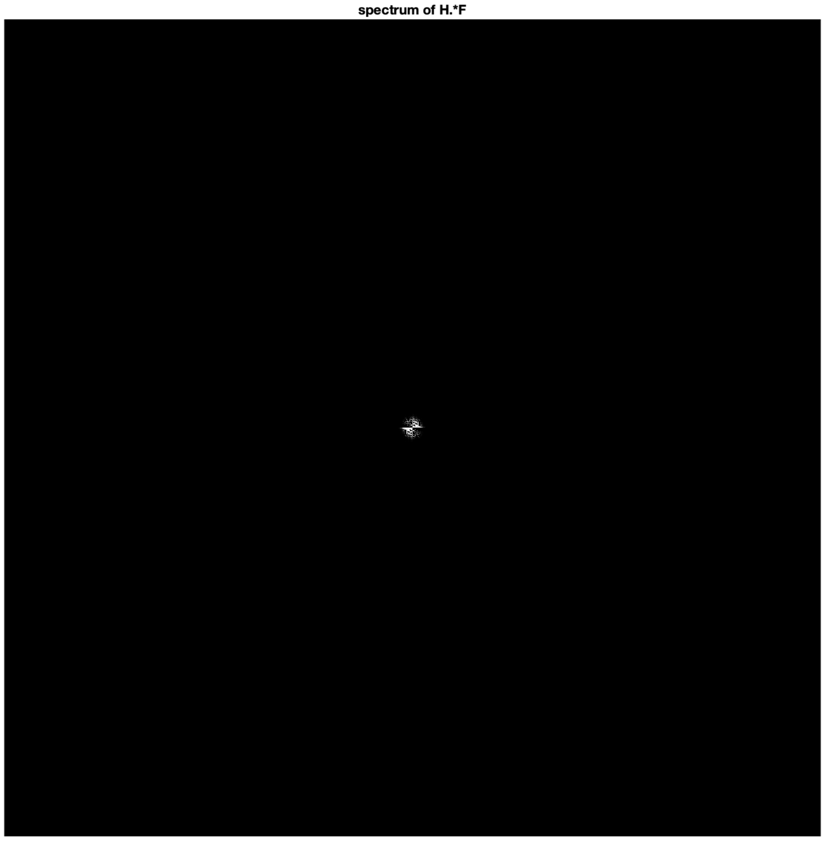


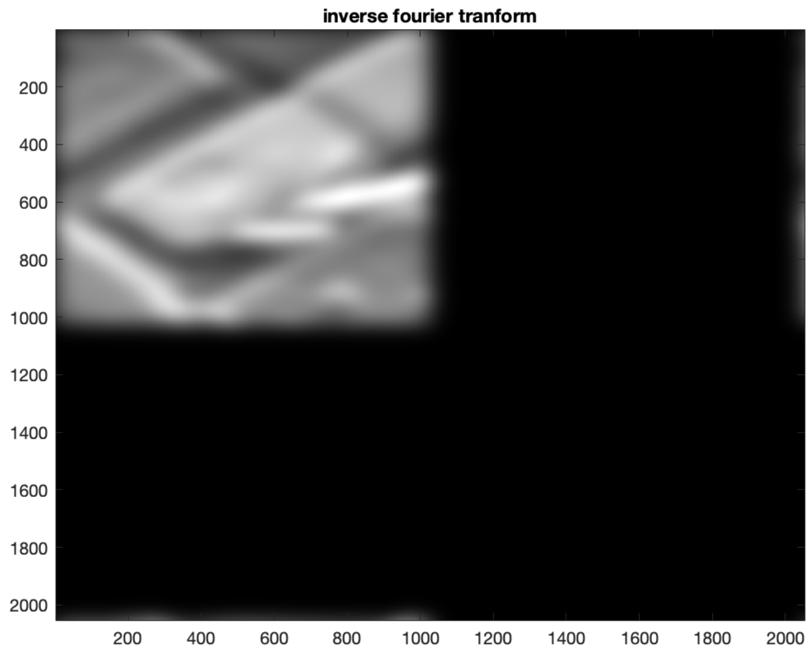
Figure 3
File Edit View Insert Tools Desktop Window Help
with noise

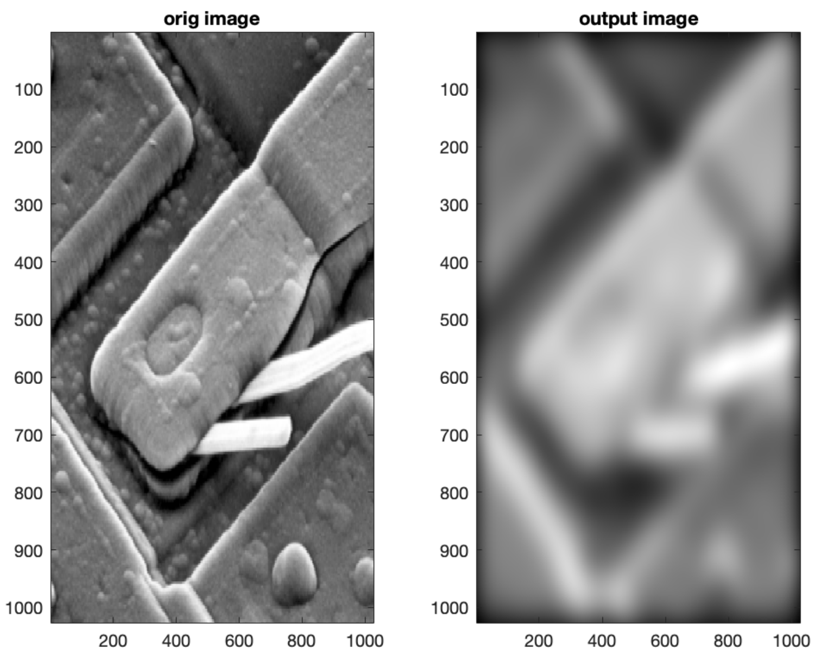
(Above: zoom in of noised image)






spectrum of H.*F


inverse fourier tranform
200
400
600
800
1000
1200
1400
1600
1800
2000
200
400
600
800
1000
1200
1400
1600
1800
2000



- **Discussion:**

  First, implement 1D-FFT:

  - Split the input vector into two, one contains odd elements of the input vector, and the other contains even elements of the input vector.
  - Do DFT according to given equation to the two vectors:
    - 1-D DFT pair
      - $F(u) = \sum_{x=0}^{M-1} f(x) e^{-\frac{j2\pi ux}{M}},\ u = 0, 1, \ldots, M-1$
  - Bind the two vectors. First half of the result(i): even_vector(i) + odd_vector(i) * exp(-(1j2pi)/N); Second half of the result: even_vector(i-N/2) + odd_vector(i-N/2) * exp(-(1j2pi)/N)

  Then, implement 2D-FFT:

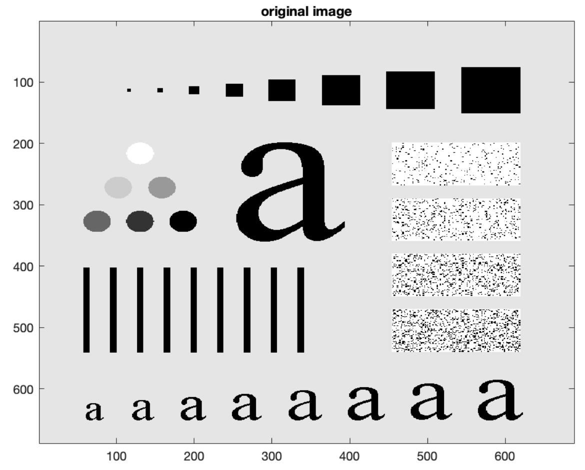
  - Split the 2D input image into two 1D-FFT.

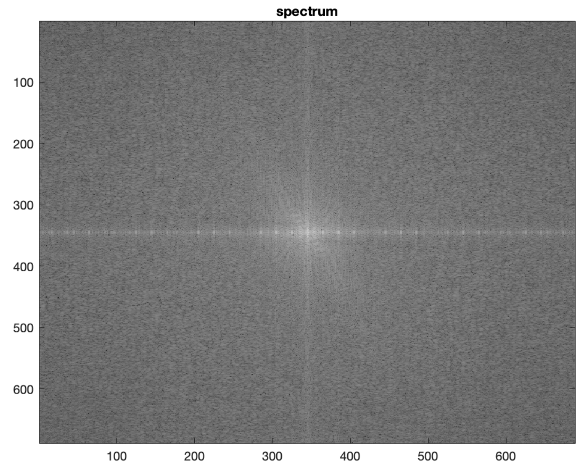
2. Proj.04-02 - Fourier Spectrum and Average Value (20%)

- **Explanation:**

  Compute a given image's centered fourier spectrum, and calculate its average value. Compare it with the average value of the original image in spatial domain.

- **Result:**





- **Comparison:**

  Center frequency component is 207.3147, and the mean of spatial domain is also 207.3147.

  One of the properties of 2D-DFT includes:

  – DC component

  - $F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) = \frac{1}{MN} \bar{f}(x,y)$

  $|F(0,0)|$ is proportional to the average of $f(x,y)$

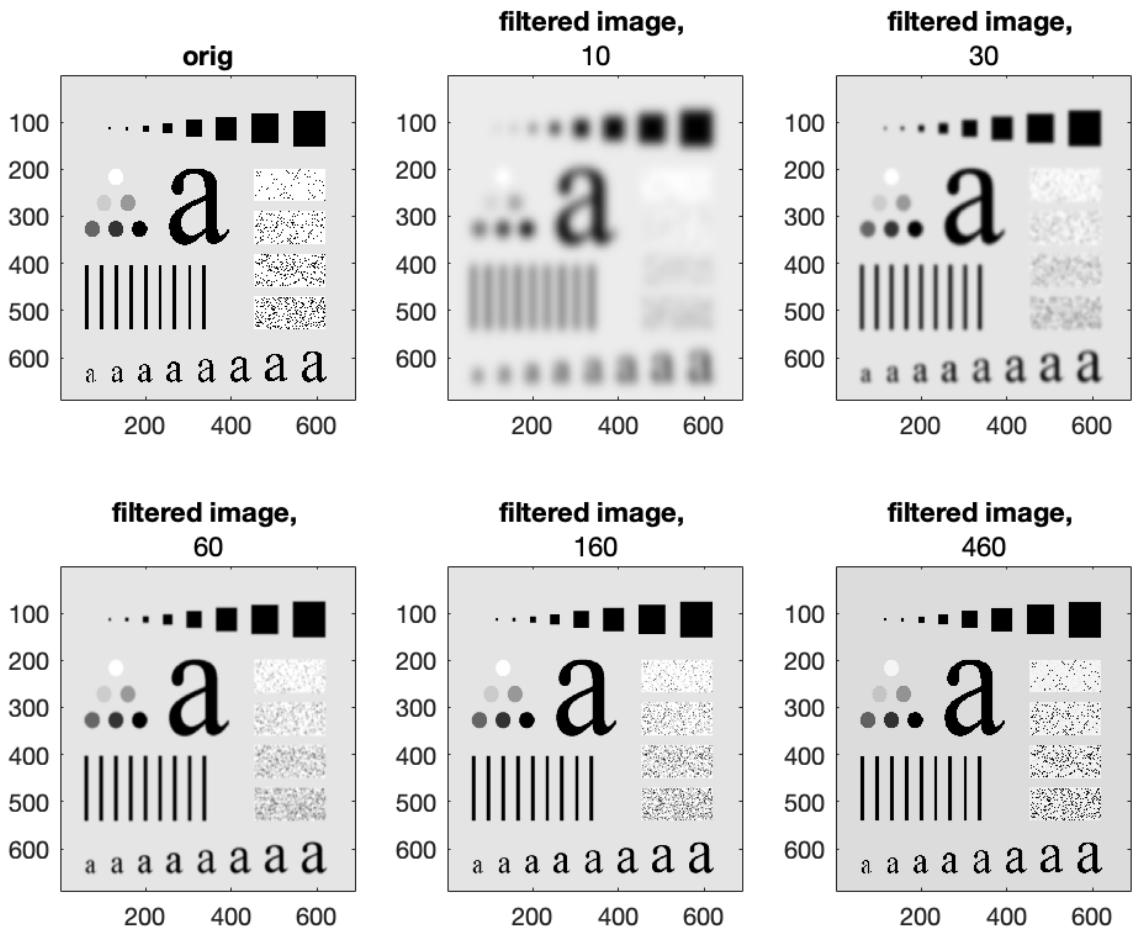
  Since the center point of spectrum indicates the lowest frequency, which will be the average of the pixels in spatial domain. Higher frequency means above or below average.

3. Proj.04-03 Lowpass Filtering (20%)

- **Explanation:**

  Implement the Gaussian lowpass filter, I/O as H = LowPassFilter(M, N, D0). M and N indicates the size of the filter, D0 as cut off frequency.

- **Result:**



- **Comparison:**

  Image becomes more blurred as the cut off frequency becomes lower, because the highter frequencies are filtered out.

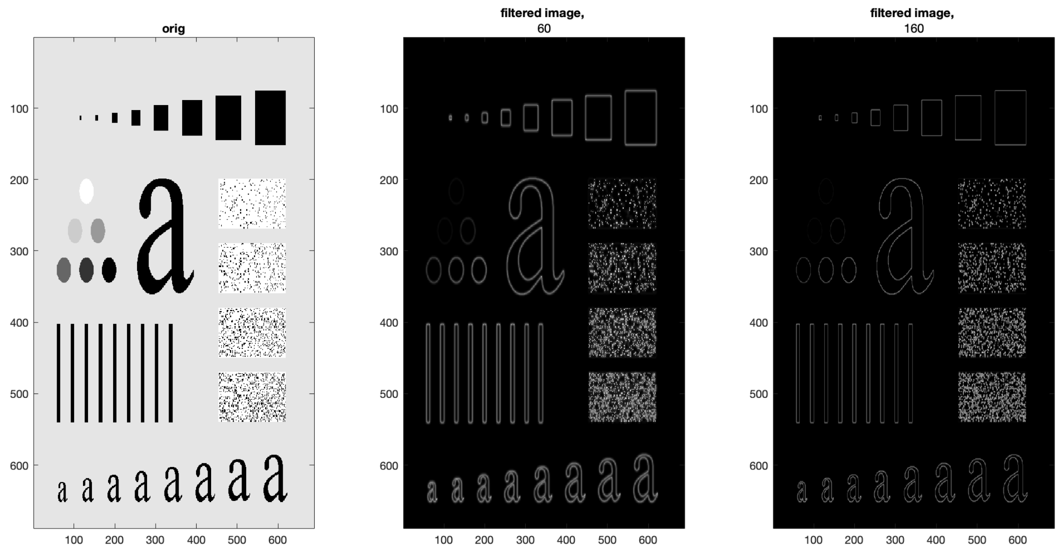
4. Proj.04-04 Highpass Filtering (20%)

- **Explanation:**

  Implement the Gaussian highpass filter. Basically it is the same program as Gaussian lowpass filter mentioned above.

  - Highpass filter
    - $H_{HP}(u,v) = 1 - H_{LP}(u,v)$

- **Result:**


orig
filtered image,
60
filtered image,
160
100
200
300
400
500
600