Digital Image Processing Laboratory

Experiment Report

Experiment Title	Smoothing filter and sharpening filter
Student's Name	
Student's ID	
Class	
Date handed in	

International school, Jinan University

A. Objectives

(1) To know the concept of image restoration.
(2) To know basic function of image restoration in MATLAB.
(3) To be able to evaluate different restoration methods and select the best one.

B. Technique

In this lab, the image **lib.jpg** will be used.
(1) Add noise on the image.
(2) Apply average filter, Gaussian filter and custom filter to smooth the image.
(3) Apply Laplacian mask to get the Laplacian image, then perform image addition.
(4) Use Robert operator and Sobel operator to get the gradient image.
(5) Design one sharpening mask to sharpen the image.

C. Experiment Content

(1) **Smoothing image**

1. Read the images **lib.jpg**.

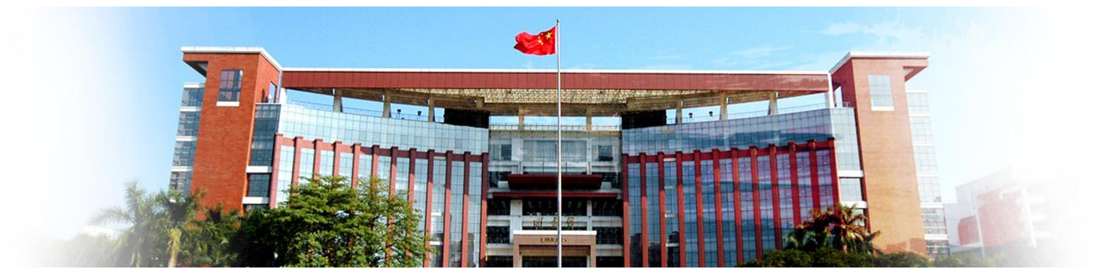


Figure 1: The original **lib.jpg**

2. Convert it into grey image.

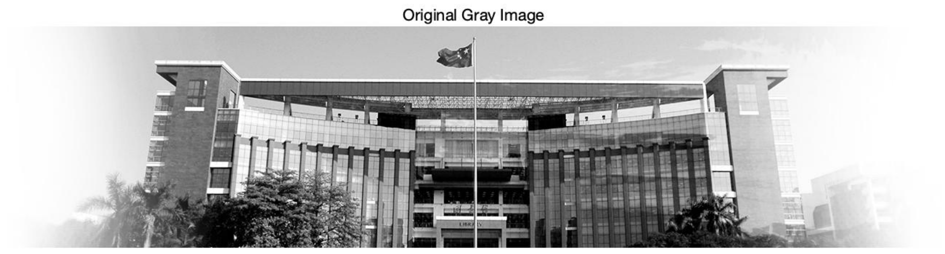


Figure 2: The original grey image **GIMG**

3. Add some noise on the grey image (name it as **GIMGNoise**). (Using **imnoise** function)

Gaussian Noise Image

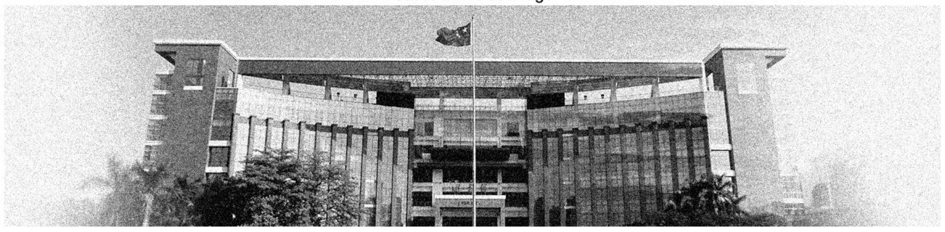
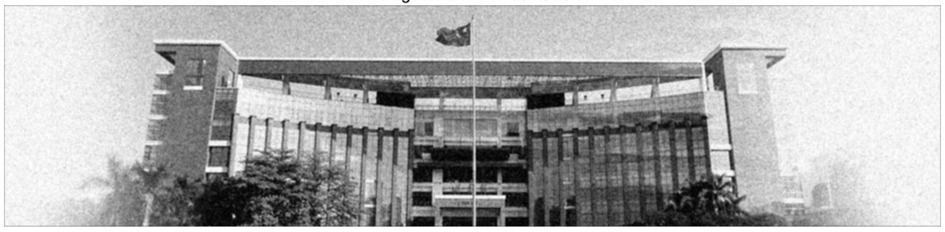


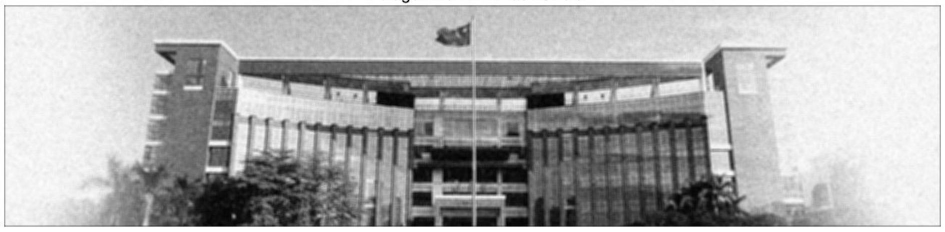
Figure 3: The **GIMG** with noise(**GIMGNoise**)

4. Using averaging filter to smooth the image **GIMGNoise**. Use different size of averaging mask to do it and compare the results.

Average Filter with Mask Size 3



Average Filter with Mask Size 5



Average Filter with Mask Size 7

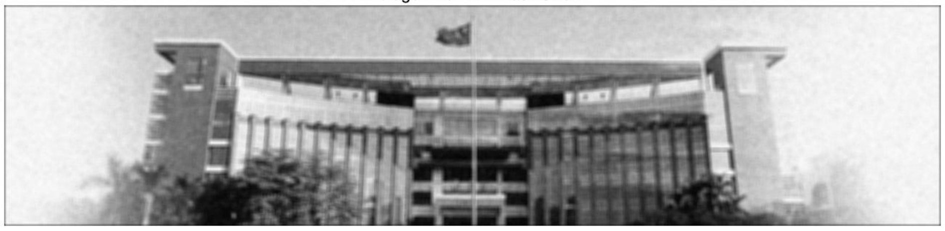


Figure 4: Using averaging filter with different size to smooth the image **GIMGNoise**

The above figures show the effect of smoothing an image using averaging filters of different sizes. From top to bottom, as the mask size increases (3x3, 5x5, to 7x7), it can be observed that the noise of the image decreases and the details become blurrier. This suggests that a larger mask size will smooth the image more effectively, but at the expense of image sharpness and detail. Therefore, in practical applications, choosing the appropriate mask size to achieve the best balance between denoising and preserving details is a key consideration in image processing.

5. Using Gaussian filter to smooth the image **GIMGNoise**. Use different size of mask to do it. After that, compare the results from Gaussian filter.

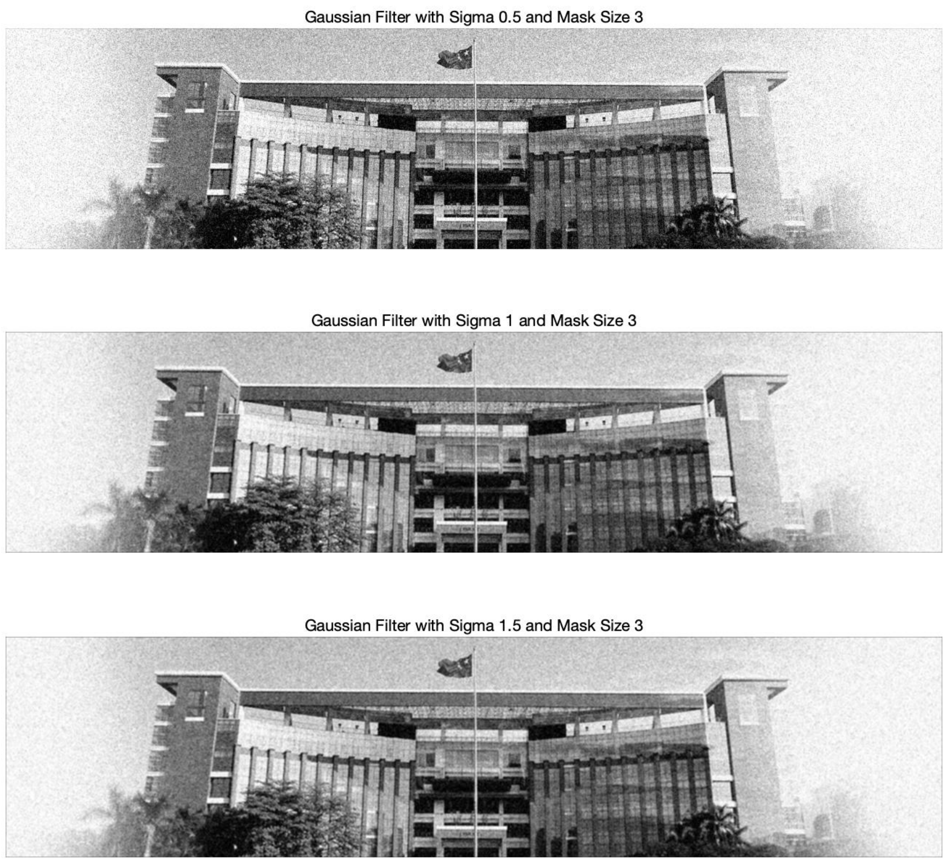
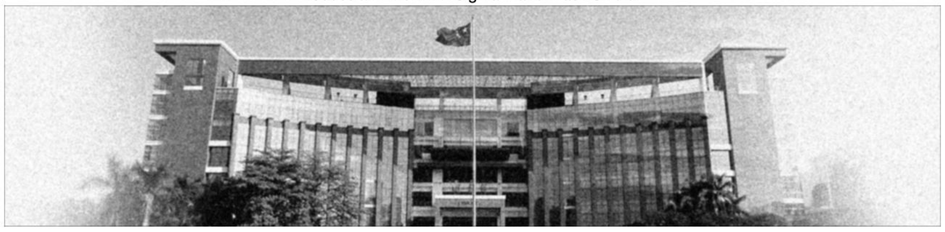

Gaussian Filter with Sigma 0.5 and Mask Size 3
Gaussian Filter with Sigma 1 and Mask Size 3
Gaussian Filter with Sigma 1.5 and Mask Size 3

Figure 5: Using Gaussian filter with different Sigma and same mask size(3) to smooth the image **GIMGNoise**

This above figures show the results of smoothing an image using a Gaussian filter, using different σ (standard deviation) values but keeping the same mask size (3x3). As the σ value increases from 0.5 to 1.5, the smoothing effect of the image gradually increases and the noise decreases. However, as the σ value increases, the details of the image also decrease accordingly. This shows that the σ value of the Gaussian filter has a significant impact on image smoothness and detail retention.

Gaussian Filter with Sigma 0.5 and Mask Size 5

Gaussian Filter with Sigma 1 and Mask Size 5



Gaussian Filter with Sigma 1.5 and Mask Size 5

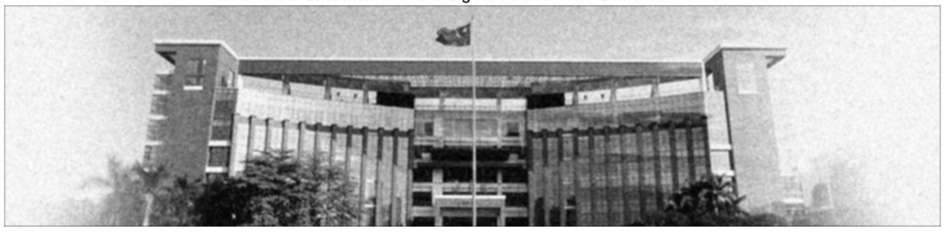
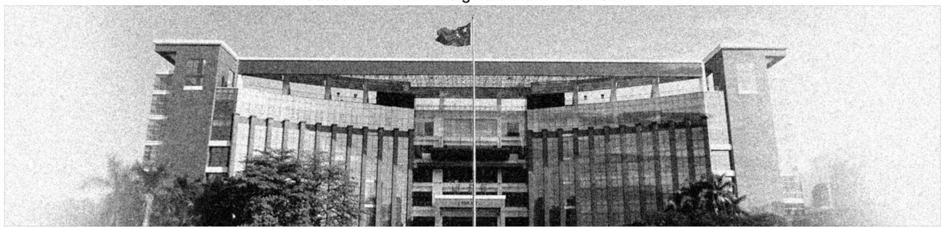


Figure 6: Using Gaussian filter with different Sigma and same mask size(5) to smooth the image **GIMGNoise**

Gaussian Filter with Sigma 0.5 and Mask Size 7



Gaussian Filter with Sigma 1 and Mask Size 7

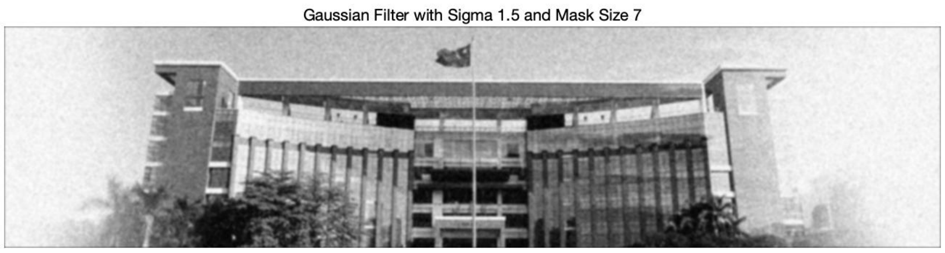


Figure 7: Using Gaussian filter with different Sigma and same mask size(7) to smooth the image **GIMGNoise**

This section shows the effect of using a Gaussian filter and increasing the mask size (from 5x5 to 7x7) to smooth the image. As the mask size increases, the noise in the image is further reduced, but it also results in more loss of image details. Specifically, when the mask size increases from 5x5 to 7x7, the smoothness of the image increases and the noise is removed more thoroughly, but the sharpness and texture details of the image decrease.

6. Design one smoothing mask to smooth the image **GIMGNoise** and show its results.

   Note: the mask depends on your idea, no any restriction.

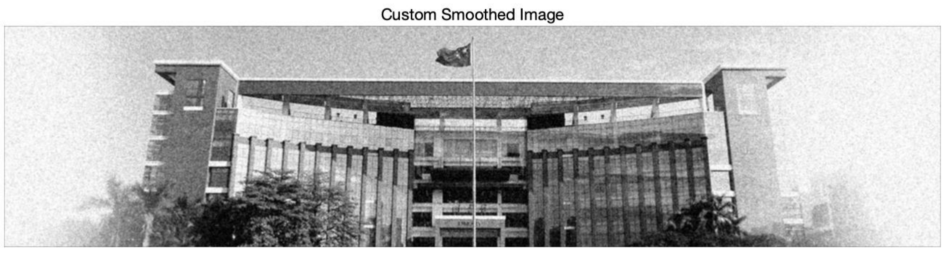


Figure 8: Using custom smoothing mask to smooth the image **GIMGNoise**

The smoothing mask I designed uses a weighted average method, with the center pixel having the greatest weight and decreasing weight as the pixels get further away from the center. This design aims to take more into account the influence of the central area of the image, while also slightly considering surrounding pixels, thereby smoothing out noise while maintaining some sharpness in the image.

According to the results shown, the mask is able to effectively reduce noise and the image looks smoother, but still retains some details. The advantage of this mask is that it provides a balanced method between a simple average filter and a Gaussian filter, which has good effects on detail preservation and noise removal in the image.

The MATLAB code to complete this experiment is as follows:

```
clc;

clear;


% Read the image

img = imread('lib.jpg');


% Convert to grayscale

grayImg = rgb2gray(img);


% Display the original grayscale image

figure, imshow(grayImg), title('Original Gray Image');


% Add some noise on the grey image

GIMGNoise = imnoise(grayImg, 'gaussian');


% Display the image with Gaussian noise

figure, imshow(GIMGNoise), title('Gaussian Noise Image');


% Using averaging filter with different size of averaging mask to smooth

maskSizes = [3, 5, 7];

for i = 1:length(maskSizes)

    h = fspecial('average', maskSizes(i));

    smoothedImg = imfilter(GIMGNoise, h);

    figure, imshow(smoothedImg),

    title(['Average Filter with Mask Size ', num2str(maskSizes(i))]);
```

```
end


% Using averaging filter with different size of mask and sigma to smooth
sigmaValues = [0.5, 1, 1.5];
for i = 1:length(maskSizes)
    for j = 1:length(sigmaValues)
        h = fspecial('gaussian', maskSizes(i), sigmaValues(j));
        smoothedImg = imfilter(GIMGNoise, h);
        figure, imshow(smoothedImg),
        title(['Gaussian Filter with Sigma ', num2str(sigmaValues(j)), ...
        ' and Mask Size ', num2str(maskSizes(i))]);
    end
end


% Design and apply a custom smoothing mask
customSmoothMask = [1 2 1; 2 4 2; 1 2 1] / 16;
customSmoothedImg = imfilter(GIMGNoise, customSmoothMask);
figure, imshow(customSmoothedImg), title('Custom Smoothed Image');
```

(1) Sharpening Images

1. Read the images **lib.jpg** given in the folder.

Figure 9: The original **lib.jpg**

2. Convert it into grey image **GIMG**.

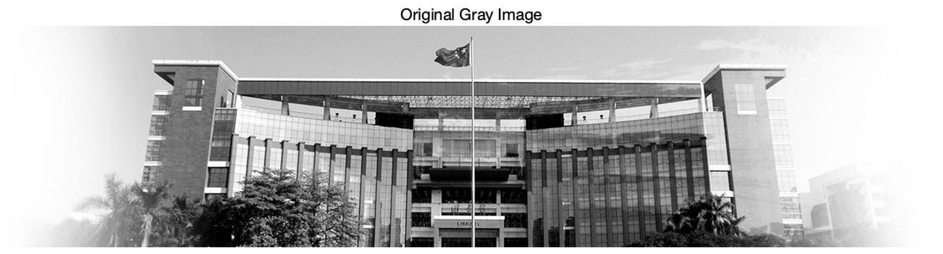


Figure 10: The original grey image **GIMG**

3. Use Laplacian mask to get the Laplacian image of **GIMG**. Laplacian mask has four types. Try to use each one to do it.

The following are the contents of the four Laplacian masks:

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

Laplacian mask 1

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplacian mask 2

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

Laplacian mask 3

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Laplacian mask 4

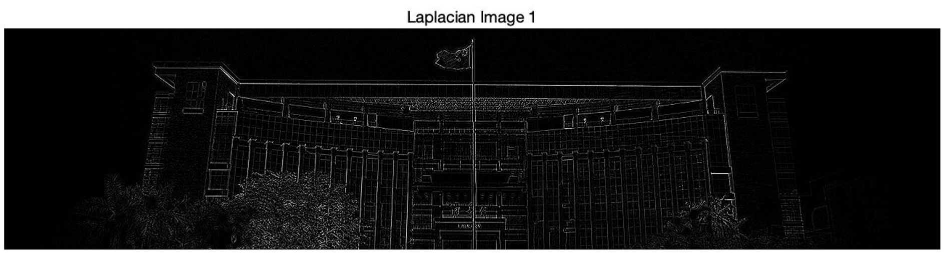


Figure 11: Use Laplacian mask 1 to get the Laplacian image of **GIMG**

The central pixel of Laplacian mask 1 is given a negative weight, and its four immediate neighbors around it are given a positive weight. This approach causes the pixel values around the edges to be reduced, creating dark lines at the edges, primarily enhancing the horizontal and vertical edges in the image. Due to its mask design, it is insensitive to diagonally oriented edges. This means that this mask is more effective if the main features of the image are horizontal or vertical. In the resulting image, vertical and horizontal edges are clearly highlighted, while diagonal details may be less obvious.

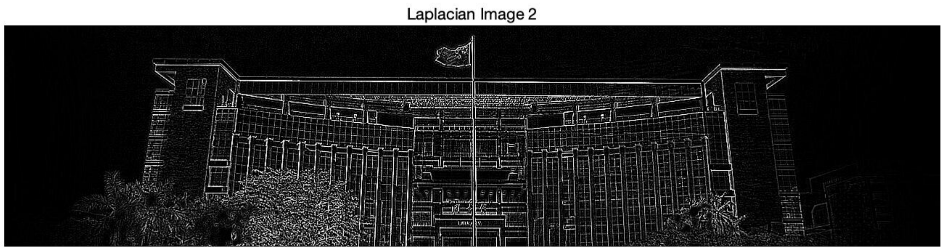


Figure 12: Use Laplacian mask 2 to get the Laplacian image of **GIMG**

Laplacian Mask 2 is a more comprehensive mask that emphasizes edges equally in all directions, with the center pixel being given a negative weight and its four immediate neighbors around it being given a positive weight, resulting in an Dark lines form everywhere. This all-around enhancement may cause edges to appear sharper and may cause more noise to be interpreted as edges, especially in more textured areas of the image.

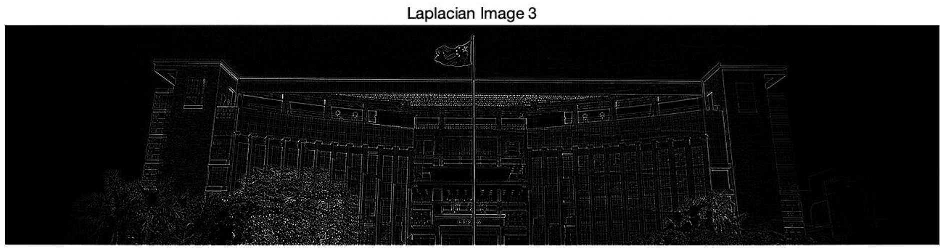


Figure 13: Use Laplacian mask 3 to get the Laplacian image of **GIMG**

Laplacian Mask 3 is very similar to Mask 1, with the central pixel having a positive weight and its four neighbors having negative weights. This causes the pixel values around the edges to increase, creating bright lines at the edges. This weight distribution may result in edge enhancement that is not as strong as Mask 1, but at the same time, it may better preserve the true edge information while reducing noise.

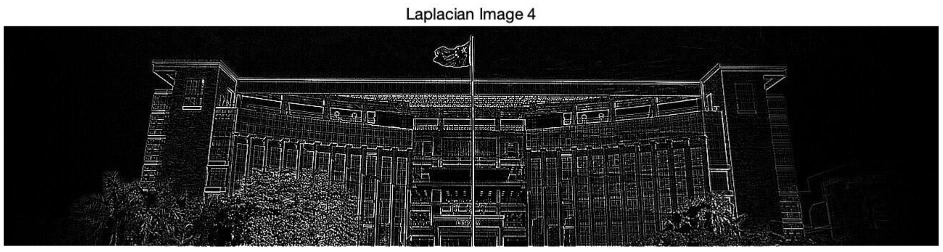


Figure 14: Use Laplacian mask 4 to get the Laplacian image of **GIMG**

Laplacian Mask 4 provides the same negative weight in all directions except the center point, resulting in bright lines at the edges. This configuration makes it very powerful at edge enhancement of images, whether horizontal, vertical or diagonal. But this also means it may be more sensitive to noise.

4. Add the four Laplacian images on the image **GIMG**. Compare the four resulting images.

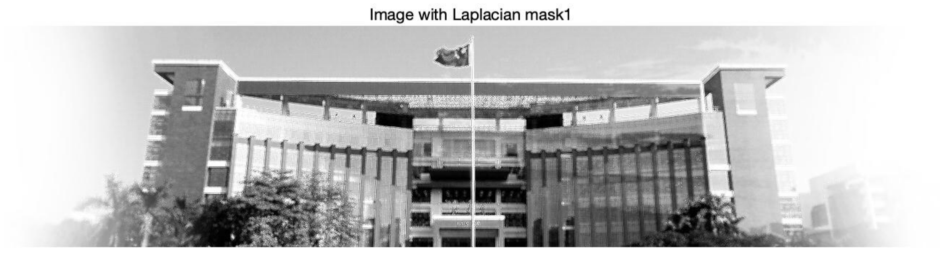


Figure 15: Add the first Laplacian image on the image **GIMG**

Laplacian Mask 1 decreases the value of the center pixel while increasing the pixel values of its immediate neighbors. This is achieved by giving the center pixel a negative weight and surrounding pixels positive weights. When such a Laplacian filter is applied to an image, there is a relative increase in brightness around the central pixel, which can lead to bright line effects around the edges.

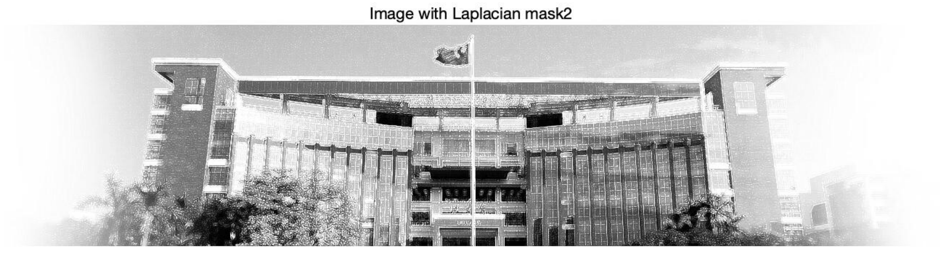


Figure 16: Add the second Laplacian image on the image **GIMG**

The resulting image from Mask 2 may show a more pronounced sharpening effect, as it enhances edges in all directions, including diagonals. This may result in more bright line effects near the edges, since Mask 2 also increases pixel values at the edges.

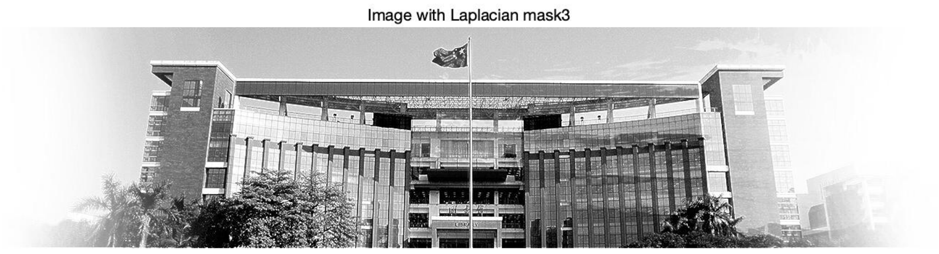


Figure 17: Add the third Laplacian image on the image **GIMG**

Since Mask 3 is designed to provide positive weights at the center pixel and negative weights at its four directly adjacent pixels, such a design tends to enhance the brightness of the center pixel relative to its neighbors. When overlaid with the original image, it usually makes the edges in the image more visible, but does not create a noticeable bright line or dark ring effect. The result is that edges look sharper and clearer, but the brightness of the overall image changes more gently without strong brightness jumps.

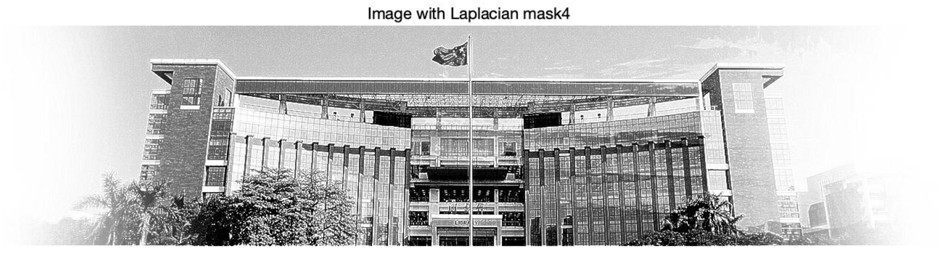


Figure 18: Add the fourth Laplacian image on the image **GIMG**

Mask 4 gives the center pixel negative weight in all directions, meaning it emphasizes the contrast between the center pixel and all surrounding neighbors. This mask causes the edges in the image to be enhanced in all directions, so they appear sharper when superimposed on the original image. Since it produces an increase in contrast at the edge, it is more likely to increase the contrast on both sides of the edge than the bright line effect, resulting in a more obvious edge enhancement effect. However, such processing may also amplify noise, especially in areas where edges are less obvious.

5. Use Robert operator and Sobel operator to get the gradient image of **GIMG**.

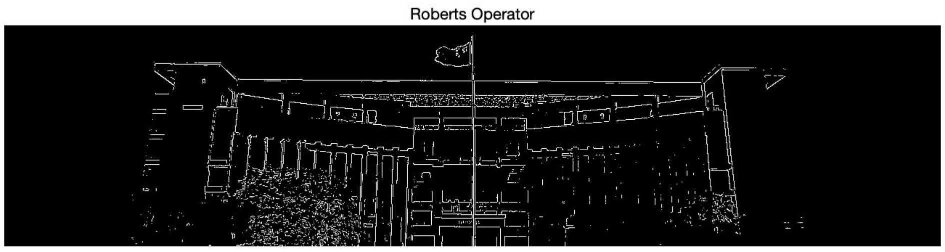


Figure 19: Use Robert operator to get the gradient image of **GIMG**

The result of Roberts operator contains small edges and noise in the image. It approximates the gradient by a simple linear method of calculating diagonal pixel differences and is therefore sensitive to noise. In the resulting image, edges may appear less continuous, especially in areas of the image where brightness changes are not large. The Roberts operator is more suitable for identifying sharp small edges, but may produce many irrelevant edges in images with noise.

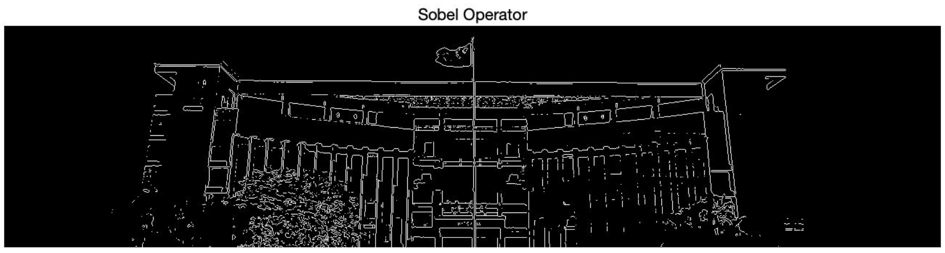


Figure 20: Use Sobel operator to get the gradient image of **GIMG**

The results of the Sobel operator usually show smoother and continuous edges. The Sobel operator calculates the gradients in the horizontal and vertical directions in the image, and combines the gradients in these two directions to obtain the overall gradient of the edge. This method is more robust as it considers a larger neighborhood of pixels, making it more resistant to noise. In the resulting image, edges are generally wider and more pronounced, and the Sobel operator is more effective for larger and smoother edge identification.

6. Design one sharpening mask to sharpen the image **GIMG** and show its results.

   Note: the mask depends on your idea, no any restriction.

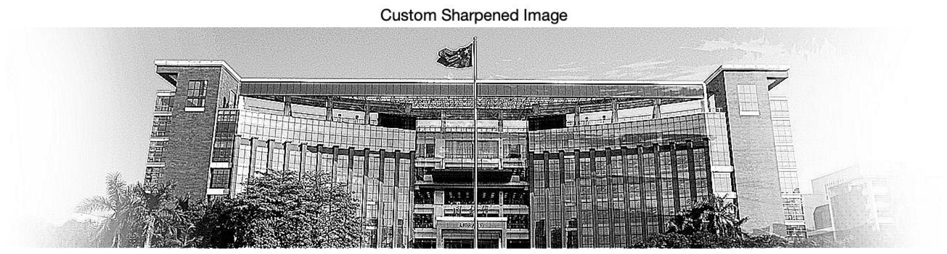


Figure 21: Using custom sharpening mask to sharpen the image **GIMG**

This mask applies an 8-neighbor negative weight and a positive center weight to each pixel. This setting increases the brightness of the central pixel while reducing the brightness of its surrounding pixels, thereby increasing the local contrast of the image and making edges and details more prominent.

When this sharpening mask is applied to an image GIMG, edges and textures will appear sharper and more defined, while the overall contrast of the image may be increased, especially where color changes are dramatic. This mask may slightly amplify noise, since sharpening also enhances high-frequency noise components in the image. The details of the image will be more obvious, but over-sharpening may cause unnatural visual effects, such as bright lines near edges or dark ring artifacts. Visually, the sharpened image should appear to have more depth and texture than the original image, but there may also be some visual artifacts, especially in areas with complex details or high noise levels.

The MATLAB code to complete this experiment is as follows:

```
clc;

clear;



% Read the image

img = imread('lib.jpg');



% Convert to grayscale

GIMG = rgb2gray(img);



% Using four different types of Laplacian masks
```

```
laplacian1 = [0 1 0; 1 -4 1; 0 1 0];

laplacian2 = [1 1 1; 1 -8 1; 1 1 1];

laplacian3 = [0 -1 0; -1 4 -1; 0 -1 0];

laplacian4 = [-1 -1 -1; -1 8 -1; -1 -1 -1];


laplacianImages = cell(1,4);

for i = 1:4

    laplacianImages{i} = imfilter(GIMG, eval(['laplacian', num2str(i)]));

    figure, imshow(laplacianImages{i}), title(['Laplacian Image ',
num2str(i)]);

end


% Add the Laplacian images to the original grayscale image and compare

addedImages = cell(1,4);

for i = 1:4

    addedImages{i} = imadd(GIMG, uint8(laplacianImages{i}));

    figure, imshow(addedImages{i}), title(['Image with Laplacian mask',
num2str(i)]);

end


% Use Roberts and Sobel operators to obtain the gradient images

roberts = edge(GIMG, 'roberts');

sobel = edge(GIMG, 'sobel');


figure, imshow(roberts), title('Roberts Operator');

figure, imshow(sobel), title('Sobel Operator');


% Design and apply a custom sharpening mask
```

```
customSharpenMask = [-1 -1 -1; -1 9 -1; -1 -1 -1];

customSharpenedImg = imfilter(GIMG, customSharpenMask);

figure, imshow(customSharpenedImg), title('Custom Sharpened Image');
```

D. Conclusions

In this experiment, I comprehensively explored the key technologies of image processing.

Starting by adding noise to the image, which simulates real-world image corruption, this step is critical to understanding the importance of image processing. I then learned how to smooth images using average filters, Gaussian filters, and custom filters, each of which offers different ways of reducing noise and preserving image detail.

By applying a Laplacian mask, I not only improve the edge contrast of the image, but also enhance the clarity of details, which is especially important for image sharpening. The use of Robert operator and Sobel operator allowed me to experience the actual operation and effect of edge detection. Finally, by designing our own sharpening mask, we put our theoretical knowledge into practice, sharpening the image and improving its visual impact.

This experiment not only improved our skills but also deepened our understanding of image processing workflows and nuances.