LAPORAN PRAKTIKUM PENGOLAHAN CITRA DIGITAL
11. SHARPENING FILTERS IN THE SPATIAL DOMAIN

Disusun oleh :

Nama :

NPM :

Kelas : IF4..

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER DAN REKAYASA

UNIVERSITAS MULTI DATA PALEMBANG

2024

TUTORIAL : SHARPENING FILTERS IN THE SPATIAL DOMAIN

Goal

The goal of this tutorial is to learn how to implement sharpening filters in the spatial domain.

Objectives

- Learn how to implement the several variations of the Laplacian mask.
- Explore different implementations of the unsharp masking technique.
- Learn how to apply a high-boost filtering mask.

Procedure

To implement the Laplacian filter, we can either create our own mask or use the `fspecial` function to generate the mask for us. In the next step, we will use `fspecial`, but keep in mind that you can just as well create the mask on your own.

1. Load the moon image and prepare a subplot figure.

```
I = imread('moon.tif');
Id = im2double(I);
figure, subplot(2,2,1), imshow(Id), title('Original Image');
```

   We are required to convert the image to doubles because a Laplacian filtered image can result in negative values. If we were to keep the image as class `uint8`, all negative values would be truncated and, therefore, would not accurately reflect the results of having applied a Laplacian mask. By converting the image to `doubles`, all negative values will remain intact.

2. Create a Laplacian kernel and apply it to the image using the `imfilter` function.

```
f = fspecial('laplacian',0);
I_filt = imfilter(Id,f);
subplot(2,2,2), imshow(I_filt), title('Laplacian of Original');
```

Question 1 When specifying the Laplacian filter in the `fspecial` function, what is the second parameter (in the case above, 0) used for?

Question 2 What is the minimum value of the filtered image?

Question 3 Verify that a `uint8` filtered image would not reflect negative numbers. You can use the image I that was previously loaded.

You will notice that it is difficult to see details of the Laplacian filtered image. To get a better perspective of the detail the Laplacian mask produced, we can scale the image for display purposes so that its values span the dynamic range of the gray scale.

3. Display a scaled version of the Laplacian image for display purposes.

```
subplot(2,2,3), imshow(I_filt,[]), title('Scaled Laplacian');
```

The center coefficient of the Laplacian mask we created is negative. Recall from the chapter that if the mask center is negative, we subtract the filtered image from the original, and if it is positive, we add. In our case, we will subtract them.

4. Subtract the filtered image from the original image to create the sharpened image.

```
I_sharp = imsubtract(Id,I_filt);
subplot(2,2,4), imshow(I_sharp), title('Sharpened Image');
```

A composite version of the Laplacian mask performs the entire operation all at once. By using this composite mask, we do not need to add or subtract the filtered image—the resulting image is the sharpened image.

5. Use the composite Laplacian mask to perform image sharpening in one step.

```
f2 = [0 -1 0; -1 5 -1; 0 -1 0]
I_sharp2 = imfilter(Id,f2);
figure, subplot(1,2,1), imshow(Id), title('Original Image');
subplot(1,2,2), imshow(I_sharp2), title('Composite Laplacian');
```

Question 4 You may have noticed that we created the mask without using the `fspecial` function. Is the `fspecial` function capable of generating the simplified Laplacian mask?

Question 5 Both Laplacian masks used above did not take into account the four corner pixels (their coefficients are 0). Reapply the Laplacian mask, but this time use the version of the mask that accounts for the corner pixels as well. Both the standard and simplified versions of this mask are illustrated in Figure 11.1. How does accounting for corner pixels change the output?

Unsharp Masking Unsharp masking is a simple process of subtracting a blurred image from its original to generate a sharper image. Although the concept is straightforward, there are three ways it can be implemented. Figures 11.2–11.4 illustrate these processes.

1	1	1
1	−8	1
1	1	1

−1	−1	−1
−1	9	−1
−1	−1	−1

FIGURE 11.1 Laplacian masks that account for corner pixels (standard and composite).

Let us first implement the process described in Figure 11.2

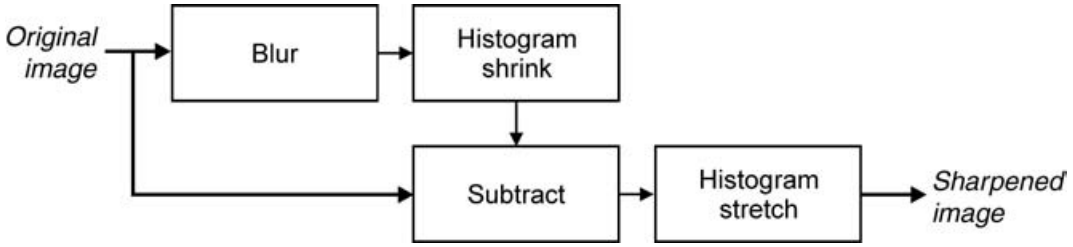

FIGURE 11.2 Unsharp masking process including histogram adjustment.

6. Close all open figures and clear all workspace variables.
7. Load the moon image and generate the blurred image.

```
I = imread('moon.tif');
f_blur = fspecial('average',5);
I_blur = imfilter(I,f_blur);
figure, subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_blur), title('Blurred Image');
```

Question 6 What does the second parameter of the `fspecial` function call mean?

We must now shrink the histogram of the blurred image. The amount by which we shrink the histogram will ultimately determine the level of enhancement in the final result. In our case, we will scale the histogram to range between 0.0 and 0.4, where the full dynamic grayscale range is [0.0 1.0].

8. Shrink the histogram of the blurred image.

```
I_blur_adj = imadjust(I_blur,stretchlim(I_blur),[0 0.4]);
```

9. Now subtract the blurred image from the original image.

```
I_sharp = imsubtract(I,I_blur_adj);
```

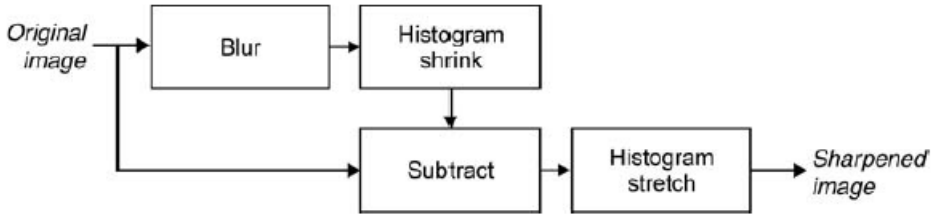

FIGURE 11.3 Unsharp masking process including histogram adjustment.

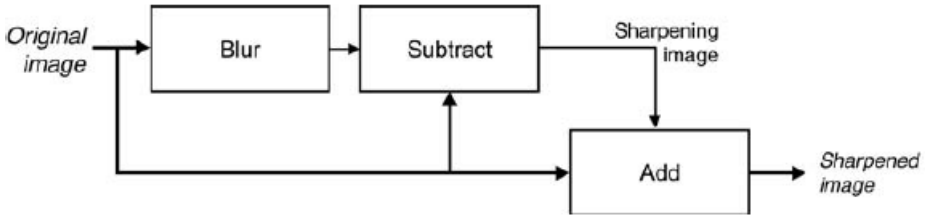

FIGURE 11.4 Unsharp masking process with sharpening image.

We must now perform a histogram stretch on the new image in order to account for previously shrinking the blurred image.

10. Stretch the sharpened image histogram to the full dynamic grayscale range and display the final result.

```
I_sharp_adj = imadjust(I_sharp);
subplot(1,3,3), imshow(I_sharp_adj), title('Sharp Image');
```

Question 7 We learned that by shrinking the blurred image's histogram, we can control the amount of sharpening in the final image by specifying the maximum range value. What other factor can alter the amount of sharpening?

11. Subtract the blurred image from the original image to generate a sharpening image.

```
I_sharpening = imsubtract(I,I_blur);
```

12. Add sharpening image to original image to produce the final result.

```
I_sharp2 = imadd(I,I_sharpening);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp2), title('Sharp Image');
```

13. Generate unsharp masking kernel using the `fspecial` function.

Question 8 How can we adjust the amount of sharpening when using this implementation?

The third implementation uses a convolution mask, which can be generated using the fspecial function. This implementation is illustrated in Figure 11.5.

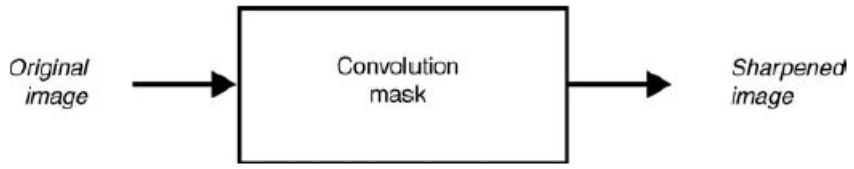

FIGURE 11.5 Unsharp masking process using convolution mask.

14. Apply the mask to the original image to create a sharper image.

```
I_sharp3 = imfilter(I,f_unsharp);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp3), title('Sharp Image');
```

Question 9 How do we control the level of sharpening with this implementation?

High-Boost Filtering High-boost filtering is a sharpening technique that involves creating a sharpening image and adding it to the original image. The

mask used to create the sharpening image is illustrated in Figure 11.6. Note that there are two versions of the mask: one that does not include the corner pixels and another that does.

15. Close any open figures.
16. Create a high-boost mask (where A = 1) and apply it to the `moon` image.

```
f_hb = [0 -1 0; -1 5 -1; 0 -1 0];
I_sharp4 = imfilter(I,f_hb);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp4), title('Sharp Image');
```

Question 10 What happens to the output image when A is less than 1? What about when A is greater than 1?

You may have noticed that when A = 1, the high-boost filter generalizes to the composite Laplacian mask discussed in step 5. As the value of A increases, the output image starts to resemble an image multiplied by a constant.

0	−1	0
−1	A+4	−1
0	−1	0

−1	−1	−1
−1	A+8	−1
−1	−1	−1

FIGURE 11.6 High-boost masks with and without regard to corner pixels.

17. Show that a high-boost mask when A = 3 looks similar to the image simply multiplied by 3.

```
f_hb2 = [0 -1 0; -1 7 -1; 0 -1 0];
I_sharp5 = imfilter(I,f_hb2);
I_mult = immultiply(I,3);
figure, subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_sharp5), title('High Boost, A = 3');
subplot(1,3,3), imshow(I_mult), title('Multiplied by 3');
```

Question 11 At what value of A does this filter stop being effective (resemble the image multiplied by a constant)?