
COMP 6771 Image Processing: Assignment 2

Student name: YUNQI XU

Student id: 40130514

October 18, 2022

1. Question 1

- (a) The minimum size of the blur mask is a $7 * 7$ filter. Since the averaging filter will calculate the average of the pixels contained in the neighborhood of the filter mask. No matter the coefficient value of the average filter is. If all pixels in the filter mask are zero, the output is zero. Based on the question, the gaps ranging from 1 to 5 pixels. So the minimum kernel size should be greater than 5 to avoid the zero output. Also, the kernel size usually is an odd number. So, $7 * 7$ is the minimum size.
- (b) Based on the question, we want those output which the filter center located on the string or hole (and be filled) equal or greater than the threshold and keep those value. Suppose the center of the filter located at a center of a gap. In this case, the filter will have a minimum output that we want to convert its value. Also, the maximum size of gap is 5 pixels. If we want to use thresholding method to convert it back at the meanwhile keep string without small hole or noise, we need to guarantee that the minimum output is greater than threshold. So, suppose we get S as the minimum useful output. We will set threshold closely less than S , but not less too much. To do this, we can keep string without small hole and avoid to bring more noise during the threshold process.

2. Question 2

The 5-by-5 Laplacian like filter has been shown in Eq. 1 below:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (1)$$

The basic rule is the sum of all coefficients is 0.

We will use $g(x, y) = f(x, y) - \nabla^2 f(x, y)$ to sharpen an image. Compared with filters in question, this one will output an image sharper result. Because, the center of the 5-by-5 filters have more weight than -4 or -8 . So, the edge of the image will be further enhanced.

3. Question 2 Fourier Transform Question

- (a) Based on the question, the Eq. 2 defines the variable t .

$$f(t) = \begin{cases} A & 0 \leq t \leq W \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$\begin{aligned}
F(\mu) &= \int_{-\infty}^{+\infty} f(t)e^{-j2\pi\mu t} dt \\
&= \int_0^W Ae^{-j2\pi\mu t} dt \\
&= \frac{-A}{j2\pi\mu} [e^{-j2\pi\mu t}]_0^W \\
&= \frac{-A}{j2\pi\mu} [e^{-j2\pi\mu W} - 1] \\
&= \frac{A}{j2\pi\mu} [1 - e^{-j2\pi\mu W}] \\
&= \frac{A}{j2\pi\mu} [e^{j\pi\mu W} - e^{-j\pi\mu W}] e^{-j\pi\mu W} \\
&= AW \left(\frac{\sin(\pi\mu W)}{\pi\mu W} \right) e^{-j\pi\mu W}
\end{aligned} \tag{3}$$

Since $A = W = 1$, In the Example, the result is:

$$F(\mu) = AW \frac{\sin(\pi\mu W)}{\pi\mu W} = \frac{\sin(\pi\mu)}{\pi\mu} \tag{4}$$

And the result of Eq. 3 is:

$$F(\mu) = \frac{\sin(\pi\mu)}{\pi\mu} e^{-j\pi\mu} \tag{5}$$

Compared with these two equation Eq. 4 and Eq. 5, the only different part is $e^{-j\pi\mu}$. So the case in this problem is a shifted part compared the result of Example 4.1.

(b) Based on the property of Convolution:

$$f(t) \star h(t) \iff H(\mu)F(\mu)$$

$$f(t) = \begin{cases} A & -\frac{W}{2} \leq t \leq \frac{W}{2} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Based on the information provided by the right image in question, the Eq. 6 is the equation.

$$\begin{aligned}
F(\mu) &= \int_{-\infty}^{+\infty} f(t)e^{-j2\pi\mu t} dt \\
&= \int_{-\frac{W}{2}}^{\frac{W}{2}} Ae^{-j2\pi\mu t} dt \\
&= \frac{-A}{j2\pi\mu} [e^{-j2\pi\mu t}]_{-\frac{W}{2}}^{\frac{W}{2}} \\
&= AW \left(\frac{\sin(\pi\mu W)}{\pi\mu W} \right)
\end{aligned} \tag{7}$$

Based on the property of convolution and Eq. 7, the Fourier Transform of the tent function is:

$$\begin{aligned} F(\mu)F(\mu) &= A^2 W^2 \frac{\sin^2(\pi\mu W)}{\pi^2 \mu^2 W^2} \\ &= A^2 \frac{\sin^2(\pi\mu W)}{\pi^2 \mu^2} \end{aligned} \quad (8)$$

4. Question 3.

(a) The code has been shown below

```

1 import numpy as np
2 import cv2
3
4
5 # read image
6 def readimg(path):
7     return cv2.imread(path, 0)
8
9 # Generate a Gaussian filter
10 def GaussinFilter(sigma, size):
11     m = (size - 1.) / 2.
12     n = (size - 1.) / 2.
13     y, x = np.ogrid[-m:m + 1, -n:n + 1]
14     h = np.exp(-(x * x + y * y) / (2. * sigma * sigma))
15     h[h < np.finfo(h.dtype).eps * h.max()] = 0
16     all_sum = h.sum()
17     if all_sum != 0:
18         h /= all_sum
19     return h
20
21
22
23 # Using gaussian to get the weighted aaverage.
24 def weightedSum(input_subimage, input_filter, filter_size):
25     new_subimage = np.zeros((filter_size, filter_size))
26     for y in range(filter_size):
27         for x in range(filter_size):
28             new_subimage[y, x] = input_subimage[y, x] * input_filter[y, x]
29     return sum(sum(new_subimage))
30
31
32 def averagingFilter(input_image, input_filter, filter_size):
33     image_height, image_width = input_image.shape[0], input_image.shape[1]
34     output_image = np.zeros((image_height, image_width))
35     padding_size = int((filter_size - 1) / 2)
36     padding_short = np.zeros((padding_size, image_width))
37     padding_long = np.zeros((image_height + (filter_size - 1), padding_size))
38     padded_image = np.r_[padding_short, input_image]
39     padded_image = np.r_[padded_image, padding_short]
40     padded_image = np.c_[padding_long, padded_image]
41     padded_image = np.c_[padded_image, padding_long]
42
43     for y in range(padding_size, image_height + padding_size):

```

```

44         for x in range(padding_size, image_width + padding_size):
45             sub_image = padded_image[y - padding_size: y + (padding_size + 1),
46                                     x - padding_size: x + (padding_size + 1)]
47             output_image[y - padding_size, x - padding_size] = weightedSum(
48                 input_subimage=sub_image,
49                 input_filter=input_filter,
50                 filter_size=filter_size)
51
52     return output_image
53
54 def calculateThreshold(input_image, C):
55     image_height, image_width = input_image.shape
56     output_image = np.zeros((image_height, image_width))
57     for y in range(image_height):
58         for x in range(image_width):
59             output_image[y, x] = input_image[y, x] - C
60     return output_image
61
62 # threshold the image
63 def threshold(input_image, max_value, threshold):
64     output_image = np.zeros((input_image.shape[0], input_image.shape[1]))
65     height, width = input_image.shape
66     for y in range(height):
67         for x in range(width):
68             if input_image[y, x] >= threshold[y, x]:
69                 output_image[y, x] = max_value
70             else:
71                 output_image[y, x] = 0
72     return output_image
73
74 # The overall Adaptive thresholding
75 def adaptiveThresholding(input_image,
76                         max_value, adaptive_method,
77                         threshold_type, filter_size,
78                         C, Gaussian_sigma=1.4):
79     if adaptive_method == "Gaussian":
80         filter = GaussianFilter(sigma=Gaussian_sigma,
81                                size=filter_size)
82
83     wa_image = averagingFilter(input_image=input_image, input_filter=filter,
84                               filter_size=filter_size)
85
86     if threshold_type == "THRESH_BINARY":
87         t_image = calculateThreshold(input_image=wa_image, C=C)
88
89     threshold_image = threshold(input_image=input_image, max_value=max_value,
90                                threshold=t_image)
91
92     output_image = np.array(threshold_image, dtype='uint8')
93
94     return output_image
95
96

```

```

97
98
99 image = readimg("Doc.tiff")
100 output_image = adaptiveThresholding(
101     input_image=image,
102     max_value=255,
103     adaptive_method="Gaussian",
104     threshold_type="THRESH_BINARY",
105     filter_size=11,
106     C = 4.5,
107     Gaussian_sigma=3)
108 cv2.imwrite('output.png', output_image)

```

- (b) In this question, we choose the Gaussian filter as the coefficient when calculating the weighted average. The reason is that, Mean filter is good at reducing random noise, but in our image, we want to split the foreground and background. And Gaussian filter can achieve a better result compared with Mean filter when dealing with the problem of splitting two different frequencies.

Also, The filter_size we choose is 11, since the letter in image are around 11 pixels for each of them. and the Gaussian_sigma = 3. Usually, a higher gaussian sigma will get a gaussian filter that can bring more neighborhood information to the center.

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Callinor

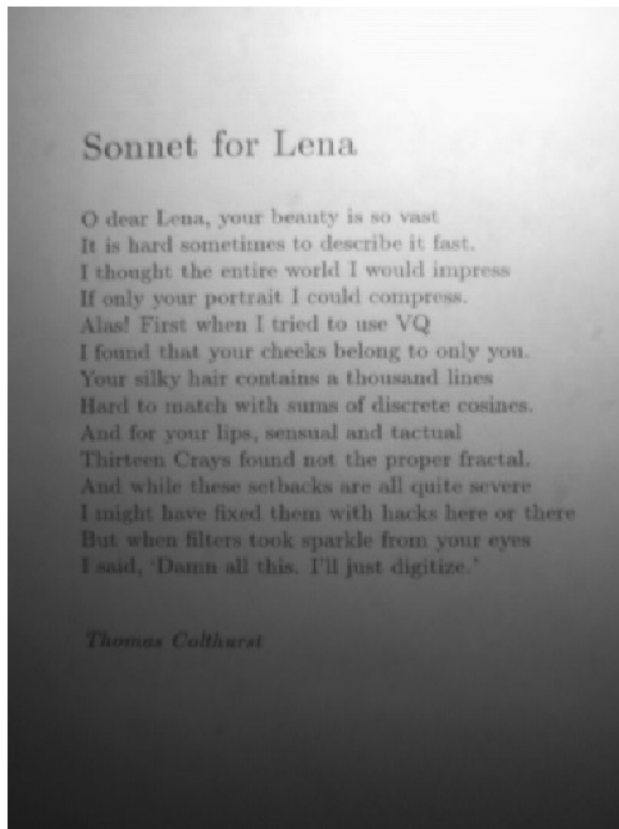
Figure 1: The output image of Python implement

Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Culthorpe

Figure 2: The output image from `open cv2.adaptiveThreshold`



Sonnet for Lena

O dear Lena, your beauty is so vast
 It is hard sometimes to describe it fast.
 I thought the entire world I would impress
 If only your portrait I could compress.
 Alas! First when I tried to use VQ
 I found that your cheeks belong to only you.
 Your silky hair contains a thousand lines
 Hard to match with sums of discrete cosines.
 And for your lips, sensual and tactual
 Thirteen Crays found not the proper fractal.
 And while these setbacks are all quite severe
 I might have fixed them with hacks here or there
 But when filters took sparkle from your eyes
 I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

Figure 3: The output image from Matlab adaptthresh

The Fig. 1 is the output by using the code above. And the Fig. 2 is the output from the method which is provided by OpenCV `cv2.adaptiveThreshold()` and we use the same parameters as we used in the implemented code. The Fig. 3 is the output image from the Matlab `adaptthresh()` with the same parameters. All of these three function can separate the input image from the dark background. The implemented code and the function from openCV can achieve a slightly better result may because there are more parameters that could be modified such as the sigma of gaussian.