CENG 466 Fundamentals of Image Processing Take-Home Exam 2

Mustafa Sezgin
Computer Engineering
e2380863@ceng.metu.edu.tr

I. Fourier, Hadamard, and Cosine Transformations

The tasks in the second step are implemented using `scipy.fftpack` library, which has built-in Fourier and discrete cosine transformation functions for n-dimensional data. Logarithmic scale is applied on the resulting images using (2) in order to make the data more visible by increasing the contrast. Fourier transform of the images are shifted to gather the bright parts in the center.

$$F' = 10\log(1 + Re\{F\}) \tag{1}$$

II. Low Pass Filters

In the third step, the ideal low pass filter is implemented manually using the formula of the circle. `circle_matrix()` function produces a square matrix with values 1 shaped like a circle, using the formula:

$$c_{ij} = \begin{cases} 1 & \text{if} \sqrt{(i - \frac{n}{2})^2 + (j - \frac{n}{2})^2} < n \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

An example circle with $r = 32$ produced by this function can be seen in Figure 1. The circle is then put in the middle of a filter matrix with the same sizes as the input image.

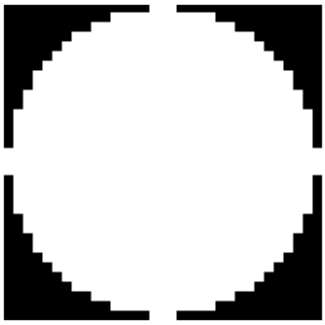

Fig. 1. Circle with $r = 32$

The Butterworth filter is implemented manually using the formula:

$$H(u, v) = \frac{1}{1 + \left(\frac{\sqrt{u^2 + v^2}}{r}\right)^n} \tag{3}$$

An example Butterworth filter can be seen in Figure 2.

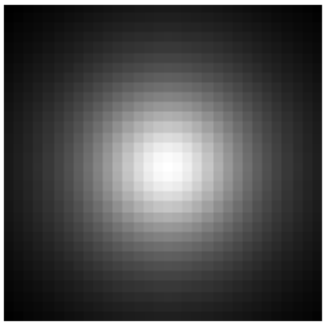

Fig. 2. Butterworth filter (32×32) with $order = 2$

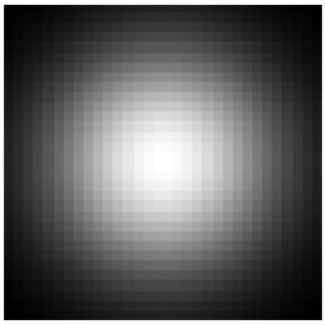

Fig. 3. Gaussian filter (32×32) with $\sigma = 8$

The Gaussian low pass filter is implemented using the `scipy.signal` library. An example Gaussian filter with $\sigma = 8$ can be seen in Figure 3.

The images are filtered using these three low pass filters with cut-off frequencies 64, 32, and 8. The filtered images are basically blurred versions of the original ones. As the cut-off frequency increases, there are more data in frequency domain that pass the filter; thus, the filtered images have more details. In other words, as the cut-off frequency decreases, the images become more blurry.

Ideal low pass filter produces outputs with some noise near the edges while Gaussian filter produces smoother outputs. Butterworth filter can be adjusted by changing the value of order. As the order increases, it becomes sharper and behaves like ideal filter.

III. High Pass Filters

In the fourth step, the implementation of high pass filters is trivial because

$$H_{HP}(u, v) = 1 - H_{LP}(u, v) \tag{4}$$

The images are filtered using three high pass filters with cut-off frequencies 128, 32, and 8. As the cut-off frequency decreases, there are more data in frequency domain that pass the filter; thus, the filtered images have more details. In other words, as the cut-off frequency increases, the images become darker, and the edges become sharper.

IV. Noise Reduction with Band Filtering

In the fifth step, we are required to remove the noises in the images 4 and 5. To do that, first, Fourier transforms of the images are inspected.

The Fourier transformation of red, green, and blue channels of Image 4 can be seen in Figure 4.

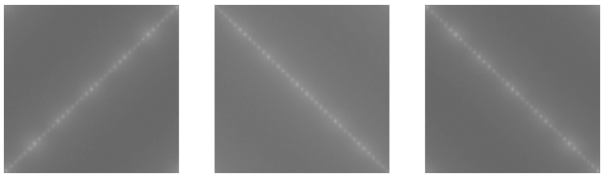

Fig. 4. Fourier transformations of red, green, and blue channels of Image 4, respectively from left to right

We can clearly see the noise in the diagonals. To clear them, I used custom diagonal band reject filters that can be seen in Figure 5.

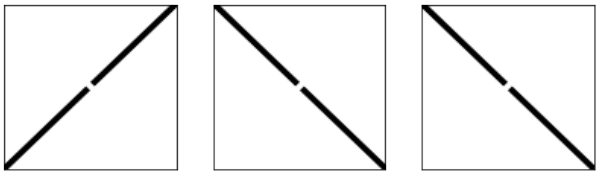

Fig. 5. Filters for red, green, and blue channels of Image 4, respectively from left to right

The filtered channels of the image in frequency domain can be seen in Figure 6.

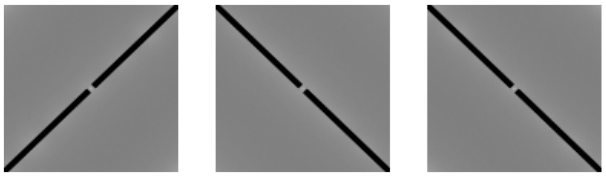

Fig. 6. Filtered red, green, and blue channels of Image 4, respectively from left to right

In addition, to obtain the noise solely, the inverse of the filters (Figure 5) are used.

I have also reduced the noise in Image 5, but I have no time to explain those.