**Project Title:  Image Processing Fundamentals**

**Project Number**: Project 4

**Course Number**: CEG 7850-01

**Student's Name**: Alex Reigle

**Date Due**: October 7, 2021

**Date Submitted**:  October 7, 2021

**Declaration Statement:**

I hereby declare that this Report and the Matlab codes were written/prepared entirely by me based on my own work, and I have not used any material from another Project at another department/ university/college anywhere else, including Wright State. I also declare that I did not seek or receive assistance from any other person and I did not help any other person to prepare their reports or code.  The report mentions explicitly all sources of information in the reference list. I am aware of the fact that violation of these clauses is regarded as cheating and can result in invalidation of the paper with zero grade. Cheating or attempted cheating or assistance in cheating is reportable to the appropriate authority and may result in the expulsion of the student, in accordance with the University and College Policies.

**Abstract:**

In this project I developed a two-dimensional fast Fourier transform that accepts arguments which specify one of seven return possibilities. As demonstrated below, the fast Fourier function (fft2dFilter) is capable of image enhancement by employing both highpass and lowpass filters using are Ideal, Gaussian, or Butterworth kernels. The function is also able to return the spectrum of an input image and output image dimensions may be specified by the user. All functionalities mentioned are shown below as replication of processing techniques used in the textbook.

**Technical Discussion:**

The project code makes use of the MatLab built in 2D fast Fourier transform (fft2) as well as the corresponding built in inverse fast Fourier transform function (ifft2). However, the project required that a manual application of frequency shifting before and after all Fourier transforms be conducted through a multiplication of the image by the following frequency shifting equation:

f(x,y)(-1)x+y ⬄ F(*u*-M/2, *v*-N/2) (eqn 4-76)

where, f(x,y) is the original image of special/time dimensions x and y, and F(*u*-M/2, *v*-N/2) is the centered/shifted image in the frequency domain of u and v of M rows and N columns.

All filters used in this project rely on the same book definition of element-wise frequency,

D(u,v) = [(u – P/2)2 + (v – Q/2)2]1/2 (eqn 4-112)

where P and Q are the number of rows and columns after zero-padding, respectively. The filters in this project were applied as prescribed by the definitions of an Ideal, Gaussian, and Butterworth lowpass filter. That is to say the an Ideal Lowpass filter (ILPF) is defined as follows:

H(u,v) = { 1,0, if D(u,v) <= D0 if D(u,v) > D0 (eqn 4-111)

Where D0 is the cutoff frequency to serve as the boundary of the filter. Similarly, the definition of a Gaussian Lowpass filter (GLPF) is defined as

H(u,v) = e-(D^2)/(2D0^2) (eqn 4-116)

To the same end, the Butterworth lowpass filter (BLPF) is defined as

H(u,v) = 1/(1+(D(u,v)/D0)2n) (eqn 4-117)

where n is a variable to control the shape of the edge of the filter and is only assigned the values of 2 and 4 below.

The high pass filters in this project were computed using the following equation from the book, in an effort to reduce redundancy.

HHP(u,v) = 1 – HLP(u,v) (eqn4-118)

In the above equation, HHP and HLP are the highpass and lowpass filter, respectively, of the predefined type.

The built in regexp MatLab function was used to identify the filter type passed into my fast Fourier transform function. This allows a single argument to identify the type of initial filter and whether to then make it a highpass filter without creating extra arguments or ‘if’ statements. I have also implemented a question dialog box to display the answer to problem 2(c) in a way similar to the rest of the figures. **However** due to this implementation choice (lines 27-36), the tester of this code will be required to click on his or her choice of “Correct!” or “Incorrect.” Before the code will continue executing. This could easily be interchanged with only the use of msgbox, but I felt that this presents the data in a more uniform and visually pleasing way.

**Results:**

Problem 1 was implemented as accepting arguments to defined the size and filter offset of the output image. Additionally, a string defining the type of filter is accepting as an argument along with the cutoff frequency and the order (n) of the Butterworth filter, described above in eqn. 4-117. In all cases that the Butterworth filter is not used, any double value can be assigned to this argument.

Problem 2 tasked me with taking Figure 4.40(a) from the text (Fig. 1, below) and applying my implementation of the 2-d fast Fourier transfrom (2dfft) to the image. Text

Description automatically generated

Figure 1: Figure 4.40(a) from the textbook.

However, I was asked to return and display on the spectrum of the image prior to the completion of the the inverse 2-D fast fourier transform (2difft), resulting in the results displayed in Fig. 2 seen below.

Shape

Description automatically generated

Figure 2: Resultant images from problem 2(b)

The dimensions of Fig. 2 are double that of the original image of Fig. 1 due to the application of zero-padding. Additionally, Fig. 3 shows the average intensity of Fig. 2 which is the value 81.4012.

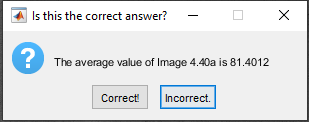


Figure 3: The output box to display the average image intensity of the images from Fig. 1 and Fig. 2. The value 81.4012 is the mean intensity asked for in problem 2(b). **Be sure to select a response to allow the code to continue executing.**

Problem 3 asked for me to implement the Gaussian lowpass filter described in eqn. 4-116, above. In order to duplicated the results of the text, the cutoff frequencies of

R = {10 30 60 160 460} (eqn 1.000)

was used to create the figures seen in Fig. 4 below.

A picture containing qr code

Description automatically generated

Figure 4: Resultant images from problem 3 with the corresponding cutoff frequency indicated above the subplot image.

It is clear to see the blurring effects induced by the varying cutoff frequencies selected for the GLPF’. These results match very closely the results of Figure 4.44 from the textbook.

Problem 4 applies a very similar process to the same original image. In this case however, the filter function is made into a highpass filter using eqn. 4-118, above. The results are seen in Fig. 5, below.

A picture containing diagram

Description automatically generated

Figure 5: Resultant image from problem 4, showing a the image form Figure 4.40(a) after having a highpass filter of type ILPF (first column), GLPF (second column), and BHPF (third column) with the specified values.

These results correspond very well with the images shown in Figure 4.53 of the textbook.

For problem 5, I approximate the images seen in Figure 4.55 in the textbook. However instead of using the Butterworth filter like the textbook, I apply a GHPF and then a binary thresholding function to both highlight the high frequency aspects of the image and then increase the contrast. The results are satisfactory, as seen in Fig.6, below.

Text

Description automatically generated

Figure 6: This image of a smudged finger print has been passed through a GHPF with a cutoff frequency of 50, and then passed through a binary thresholding function.

These results clearly resemble the images from Figure 4.55 in the textbook.

**Discussion of Results:**

Problem 1 shows the steps taken in a Fourier transformation. These steps highlight the small differences in processing outputs that occur with small changes in function call (i.e. using fftshift or ifftshift instead of multiplying by (-1)x+y ). There is also a notable similarity between the process implemented in my function fft2dFilter and the convolution process, as we have discussed in lecture. To expand upon this thought, the comparison lies heavy on the fact that both methods of image processing require a transformation into an abstracted domain (whether it be frequency or Laplacian) to preform a more computationally simple task.

Problem 2 shows the Fourier spectrum (Fig. 2) resultant of Fig. 1 after being converted into logarithmic units to match the reference image from the textbook (Fig. 4.40(b), p. 275). While I have replicated the desired spectral distribution, I have not superimposed the radii used for the following problem, as done in the text. After having acquired the spectral image (Fig. 2), I was able to calculate the simple mean intensity of the image (which, as a *‘uint8’* image, had a possible value range of 0-255). I found that 81.4012 (as shown in Fig. 3) is the mean intensity of the spectral image, Fig. 2. It is worth noting that this process is possible due to the image being returned in the frequency domain, which forgoes a majority of the implemented processing capability in the fft2dFilter function I developed.

Problem 3 requires the implementation of a GLPF (eqn. 4-116) . This lowpass filter was tested with varying radii (as defined in eqn 1.000) to reproduce the textbook results of Figure 4.44 (a-f). My results (as seen in Fig. 4, above) recreate the textbook images with one notable difference. This problem did not require mirror padding (or row/column replication, as shown in Project 1) to be applied to the zero padded image during processing. This results in my images having their edges broadened. This effect is most notable in Fig. 4(row 1, cloumn 2), where the blurring effect is greatest. This effect arises because there is a synthetically high frequency at the boundary of our image and the zeros that we padded around it.

Problem 4 require the implementation of an ILPF (eqn. 4-111) and BLPF (eqn. 4-117). I implemented the low pass filters dirrectly to reduce duplicated effort, since I had already developed the GLPF (as shown to be function in Fig. 5). After developing the lowpass filters, I made use of eqn. 4-118 to convert any lowpass filter kernel into a highpass filter kernel. After doing this, I was able to duplicate the textbook Figure 4.53 successfully. These results are shown in Fig. 5 and detail inputs used to generate each image. This effect most notably highlights edges very well, and could be used in an image sharpening process, as discussed next.

Problem 5 demonstrates an application of highpass filtering (in this case GHPF) being used in combination with simple binary thresholding , as detailed in the text, to enhance an initially blurry image. In Fig. 6 these results are shown. The notable differences that arise are a result of the difference between the textbook image (Figure 4.55) having applied a BHPF and our problem requiring a GHPF. There is also an edge broadening condition that becomes apparent in this problem, as seen in Fig. 6(column 3). This, as discussed above, could be resolved with row/column replication at the boarder of the original image, or even by simply cropping out (or removing) the outter most columns and rows after processing has been completed.

**Figures:**Text

Description automatically generated

Figure 1: Figure 4.40(a) from the textbook.

Shape

Description automatically generated

Figure 2: Resultant images from problem 2(b)

Graphical user interface, text, application

Description automatically generated

Figure 3: The output box to display the average image intensity of the images from Fig. 1 and Fig. 2. The value 81.4012 is the mean intensity asked for in problem 2(b). **Be sure to select a response to allow the code to continue executing.**

A picture containing qr code

Description automatically generated

Figure 4: Resultant images from problem 3 with the corresponding cutoff frequency indicated above the subplot image.

A picture containing diagram

Description automatically generated

Figure 5: Resultant image from problem 4, showing a the image form Figure 4.40(a) after having a highpass filter of type ILPF (first column), GLPF (second column), and BHPF (third column) with the specified values.

Text

Description automatically generated

Figure 6: This image of a smudged fingerprint has been passed through a GHPF with a cutoff frequency of 50, and then passed through a binary thresholding function.

**Appendix:**

**Written Programs:**

* main
* fft2dFilter
* scalePixelValues
* binaryThreshold

**Utilized Programs:**

* subplot
* imsho
* cast
* mat2grat
* nlfilter
* mean
* median
* sum
* max
* min
* abs
* zeros
* size
* log10
* questdlg
* num2str
* msgbox
* disp
* nextpow2
* meshgrid
* fft2
* w
* imread
* pwd
* ifft2
* sqrt
* regexp
* isempty
* real
* imag
* exp