Sergio Coronado 16.484 Assignment # 2 Created :3/6/16 Due: 3/8/16

### **Objective:**

The purpose of this assignment is to perform Fourier transforms on an image and apply a butter-worth low pass filter in the frequency domain.

# **Background:**

This assignment is aimed at introducing image processing in the frequency domain. For the first part of the assignment the MRI image was transformed to the frequency domain using a 2d Fast Fourier transform algorithm. In the second part a second-order butters-worth low pass filter is applied in the frequency spectrum.

## **Algorithm Used:**

The image was read in and a vectorized version of the image was created which had n\* 2 columns. In the second part of the program the spectrum was then shifted by multiplying each pixel by (-1) ^ (x +y). After the image has been vectorized it is then transformed using a 2D Fast Fourier transform. This was achieved by doing a 1D Fourier transform for each row of the vectorized image and then doing another 1D Fourier for each column in the in the image. Once the Fourier transform has completed the spectrum magnitude was calculated for both parts and normalized and sent to a output file. In the second part a low pass filter was applied to the the transformed image. Finally a the image is reverse using an inverse 2D Fast Fourier Transform.

#### Results:

The Program successfully transformed the image in into the frequency domain and back as required by the first part of the assignment as shown in figure 1. and figure 2.



Figure 1. Spectrum of MRI not centered

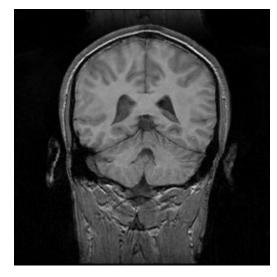


Figure 2. Reverse Image after IFFT

In the Second Part the spectrum was centered as shown in figure 3 and the the filter was successfully applied using cutoff frequencies of 10, 60 and 80 as shown in fingres 4,5, and 6

accordingly.

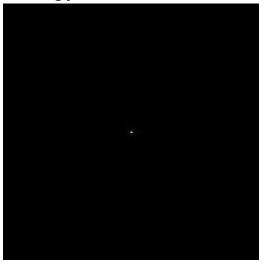


Figure 3. Centered Spectrum of MRI

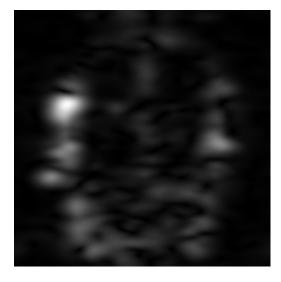


Figure 4. Filtered MRI with 10 Cutoff Freq.

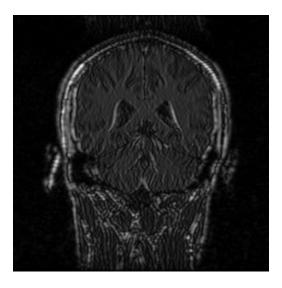


Figure 5. Filtered MRI with 60 Cutoff Freq.

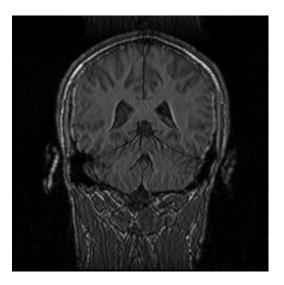


Figure 6. Filtered MRI with 80 Cutoff Freq.

### **Conclusion and Observations**

The exercise was successful in demonstrating how to apply a Fourier transform on a twodimensional image and use the frequency domain to apply low pass filters. It was interesting to observe how changing the cutoff frequency will affect the resulting image, If the the cutoff frequency is too low the image is unreadable and if it is too high there is very little change to be perceived from the application of the filter.

### Readme

24

```
////BUILD
$ make
///RUN
 NOTE: Program already add the bitmap header to the image there is no need to add a header to view
   the image only need to run
        $ XV <image-file>
bin/HW2a <input-file> <spectrum output file> <reverse -fft output file> <xSize> <vSize>
 bin/HW2b <input-file> <spectrum output file> < filtered output file> < cutoff freq.> <xSize> <ySize>
//Cleanup
 $ make clean
Code
2 /*
 3 By: Sergio Coronado
      16.484 Computer Vision
 5
      Assignemnt #2
 6
      Part 1
8 PURPOSE:
9 Program reads in an image performs a FFT outputs the spectrum and then
10 performs the inverse FFT
11
12 USAGE:
13
14 HW2a <input-file> <ospectrum-file> <reverse-file> <xSize> <ySize>
15
16 */
19
20 #include <stdlib.h>
21 #include <stdio.h>
22 #include "ImageProcessing.h"
23 #include "CursorCntl.h"
```

```
26
27
28 #define NUM_ARGS 6
29 #define FFFT 1
30 #define RFFT -1
32 /////// Main //////////
33
34
35 int main (int argc, char ** argv)
36 {
37 unsigned char ** img;
                            // Matrix Holding Original Image Values
                           // Matrix holding vectorized image
38 float ** vectoredImg;
39 unsigned char ** result;
                            // Matrix holiding Image to output
40 unsigned xSize;
                         // NUmber of Horizontal pixels
41 unsigned ySize;
                         // Number of Vertical Pixels
42 unsigned nRows;
                          // Number of Rows Read in
43
44 if (argc < NUM ARGS)
45 {
    printError("Usage: HW2a <inputFile> <spectrumOut> <reverseOut> <rows> <columns>\n");
46
47
    exit(0);
48 }
49
50
51 xSize = atoi(argv[4]);
52
53 ySize = atoi(argv[5]);
54
55 printOK("Reading Image \n");
56
57 img = ReadImage( argv[1], xSize, ySize, &nRows);
58
59 if (img == NULL)
60 {
61
    exit(-1);
62 }
63
64 if ( nRows != ySize)
65 {
66
    ySize = nRows;
67 }
68
   printOK("Vectorizing Image \n");
69
70
71 vectoredImg = VectorizeImage( img, xSize, ySize);
72
73 printOK("Performing Fourier Transform \n");
```

```
74
75 Fourier2D(vectoredImg, xSize, vSize, FFFT);
76
77
    printOK("Normalizing Image \n");
78
79
    result = NormalizeImage(vectoredImg, xSize, ySize);
80
81
    printOK("Outputing Spectrum Image \n");
82
83
    OutputImage(argv[2], result, xSize, ySize);
84
85
    printOK("Destroying Spectrum Image \n");
86
    DestroyImage (result, xSize, ySize);
87
88
    printOK("Performing Reverse Fourier \n");
89
90
91 Fourier2D(vectoredImg, xSize, ySize, RFFT);
92
93 printOK("Normalizing Image \n");
94
95
    result = NormalizeImage(vectoredImg, xSize, ySize);
96
97
    printOK("Outputing Reverse Image \n");
98
99
    OutputImage(argv[3], result, xSize, ySize);
100
101 printOK ("Cleanup\n");
102
103 DestroyImage (result, xSize, ySize);
104
105 DestroyFloatImage (vectoredImg, xSize, ySize);
106
    DestroyImage (img, xSize, ySize);
107
108
109 \operatorname{exit}(0);
110
111 }
 2 /*
 3 By: Sergio Coronado
       16.484 Computer Vision
 5
       Assignemnt #2
 6
       Part 1
8 PURPOSE:
9 Program reads in an imagecenteres the spectrumand performs an FFT and a pplys a
10 buttersworth lowpass filer
```

```
11
12 USAGE:
13
14 HW2b <input-file> <ospectrum-file> <filtered-file> < cutoff freq.> <xSize> <ySize>
15
16 */
17
19
20 #include <stdlib.h>
21 #include <stdio.h>
22 #include "ImageProcessing.h"
23 #include "CursorCntl.h"
24
27 #define BUFFER_SIZE 50
28 #define NUM_ARGS 7
29 #define FFFT 1
30 #define RFFT -1
31
32 /////// Mian /////////
33
34
35 int main (int argc, char ** argv)
36 {
37 unsigned char ** img;
                            //Matrix Holding Image Values
38 float ** vectoredImg;
                            // Matric holding vectored image values
39 unsigned char ** result;
                            // Matric holding normalized image to output
                          //NUmber of horizontal pixels
40 unsigned xSize;
41 unsigned ySize;
42 unsigned nRows;
43 int cutOff;
44 char message[BUFFER_SIZE];
45
46
47 if (argc < NUM_ARGS)
48 {
    printError("Usage: HW2a <inputFile> <spectrumOut> <filteredOut> <cutoff> <rows>
<columns>\n");
50
    exit(0);
51 }
52
53 \operatorname{cutOff} = \operatorname{atoi}(\operatorname{argv}[4]);
54
55 xSize = atoi(argv[5]);
56
57 ySize = atoi(argv[6]);
58
```

```
59 printOK("Reading Image \n");
60
61 img = ReadImage( argv[1], xSize, ySize, &nRows);
62
63 if (img == NULL)
64 {
65
    exit(-1);
66 }
67
68 if (nRows!=ySize)
69 {
70
    vSize = nRows;
71
72
73 printOK("Vectorizing Image \n");
74
75
    vectoredImg = VectorizeImage( img, xSize, ySize);
76
77
    printOK("Centering Spectrum\n");
78
79
    CenterSpectrum(vectoredImg, xSize, ySize);
80
81
    printOK("Performing Fourier Transform \n");
82
83 Fourier2D(vectoredImg, xSize, ySize, FFFT);
84
85 printOK("Normalizing Image \n");
86
87
    result = NormalizeImage(vectoredImg, xSize, ySize);
88
89
    printOK("Outputing Spectrum Image \n");
90
91
    OutputImage(argv[2], result, xSize, ySize);
92
93 printOK("Destroying Spectrum Image \n");
94
95 DestroyImage (result, xSize, ySize);
96
    sprintf(message, "Applying Low Pass Filter Cutoff = %d\n", cutOff);
97
98
    printOK(message);
99
100 ApplyLowPass(vectoredImg, xSize, ySize, cutOff);
101
102
    printOK("Performing Reverse Fourier \n");
103
104
    Fourier2D(vectoredImg, xSize, ySize, RFFT);
105
106
    printOK("Normalizing Image \n");
107
```

```
108 result = NormalizeImage(vectoredImg, xSize, ySize);
109
110 printOK("Outputing Reverse Image \n");
111
112
    OutputImage(argv[3], result, xSize, ySize);
113
114 printOK ("cleanup\n");
115
116 DestroyImage (result, xSize, ySize);
117
118 DestroyFloatImage (vectoredImg, xSize, ySize);
119
120 DestroyImage (img, xSize, ySize);
121
122 exit(0);
123
124 }
2 /*
3 By: Sergio Coronado
      16.484 Computer Vision
4
5
      Assignemnt #2
6
      Part 1
8 PURPOSE:
9 Function Protoypes for image processing
10
11 */
12 #ifndef IMAGE PROC
13 #define IMAGE_PROC
14
15
16
17 unsigned char ** ReadImage(
18 char * filename,
19 unsigned xSize,
20 unsigned vSize,
21 unsigned * numRowsi);
22
23 void Fourier2D( float ** img, unsigned xSize, unsigned ySize, int iSign);
24
25 float ** VectorizeImage( unsigned char ** img, unsigned xSize, unsigned ySize);
26
27 unsigned char ** NormalizeImage( float ** img, unsigned xsize, unsigned ySize);
29 void OutputImage (char * filename, unsigned char ** img, unsigned xSize, unsigned ySize);
30
31 void DestroyFloatImage (float ** img, unsigned xSize, unsigned ySize);
```

```
32
33 void DestroyImage (unsigned char ** img, unsigned xSize, unsigned ySize);
35 void CenterSpectrum (float ** img, unsigned xSize, unsigned ySize);
36
37 void ApplyLowPass (float ** img, unsigned xSize, unsigned ySize, int cutOff);
38
39 #endif
3 By: Sergio Coronado
4
      16.484 Computer Vision
5
      Assignemnt #2
6
      Part 1
8 PURPOSE:
9 Program reads in an imagecenteres the spectrumand performs an FFT and a pplys a
10 buttersworth lowpass filer
11
12 USAGE:
13
14 HW2b <input-file> <ospectrum-file> <filtered-file> < cutoff freq.> <xSize> <ySize>
15
16 */
17
19
20 #include <stdlib.h>
21 #include <stdio.h>
22 #include "ImageProcessing.h"
23 #include "CursorCntl.h"
24
26
27 #define BUFFER_SIZE 50
28 #define NUM_ARGS 7
29 #define FFFT 1
30 #define RFFT -1
31
32 /////// Mian //////////
33
34
35 int main (int argc, char ** argv)
36 {
37 unsigned char ** img;
                          //Matrix Holding Image Values
38 float ** vectoredImg;
                          // Matric holding vectored image values
39 unsigned char ** result;
                          // Matric holding normalized image to output
                        //NUmber of horizontal pixels
40 unsigned xSize;
```

```
41 unsigned ySize;
42 unsigned nRows;
43 int cutOff;
44 char message[BUFFER_SIZE];
45
46
47 if (argc < NUM_ARGS)
48 {
     printError("Usage: HW2a <inputFile> <spectrumOut> <filteredOut> <cutoff> <rows>
<columns>\n");
50
     exit(0);
51 }
52
53 cutOff = atoi(argv[4]);
54
55 xSize = atoi(argv[5]);
56
57 ySize = atoi(argv[6]);
58
59 printOK("Reading Image \n");
60
61 img = ReadImage( argv[1], xSize, ySize, &nRows);
62
63 if (img == NULL)
64 {
65
     exit(-1);
66 }
67
68 if ( nRows != ySize)
69 {
70 ySize = nRows;
71 }
72
73 printOK("Vectorizing Image \n");
74
75 vectoredImg = VectorizeImage( img, xSize, ySize);
76
77 printOK("Centering Spectrum\n");
78
79
    CenterSpectrum(vectoredImg, xSize, ySize);
80
    printOK("Performing Fourier Transform \n");
81
82
83 Fourier2D(vectoredImg, xSize, ySize, FFFT);
84
    printOK("Normalizing Image \n");
85
86
87
    result = NormalizeImage(vectoredImg, xSize, ySize);
88
```

```
89 printOK("Outputing Spectrum Image \n");
90
91
    OutputImage(argv[2], result, xSize, ySize);
92
93 printOK("Destroying Spectrum Image \n");
94
95 DestroyImage (result, xSize, ySize);
96
    sprintf(message, "Applying Low Pass Filter Cutoff = %d\n", cutOff);
97
98
    printOK(message);
99
100 ApplyLowPass(vectoredImg, xSize, vSize, cutOff);
101
102
    printOK("Performing Reverse Fourier \n");
103
104 Fourier2D(vectoredImg, xSize, ySize, RFFT);
105
106 printOK("Normalizing Image \n");
107
108 result = NormalizeImage(vectoredImg, xSize, ySize);
109
110
    printOK("Outputing Reverse Image \n");
111
112
    OutputImage(argv[3], result, xSize, ySize);
113
114
    printOK ("cleanup\n");
115
116 DestroyImage (result, xSize, ySize);
117
    DestroyFloatImage (vectoredImg, xSize, ySize);
118
119
120 DestroyImage (img, xSize, ySize);
121
122 exit(0);
123
124 }
```