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.

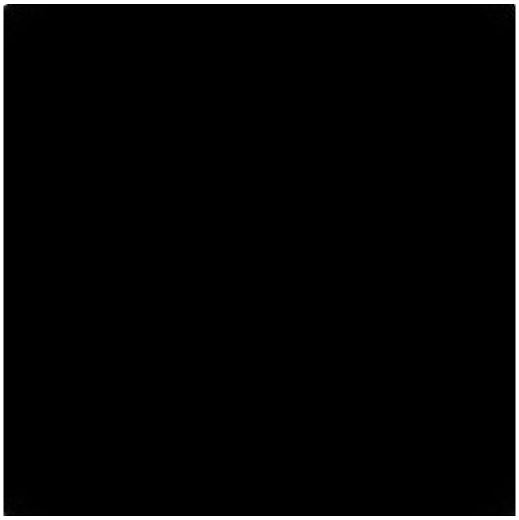

Figure 1. Spectrum of MRI not centered

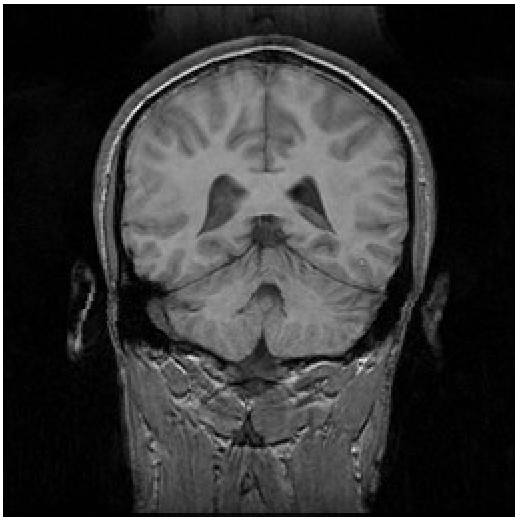

Figure 2. Reverse Image after IFFT

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 fiugres 4,5, and 6 accordingly.

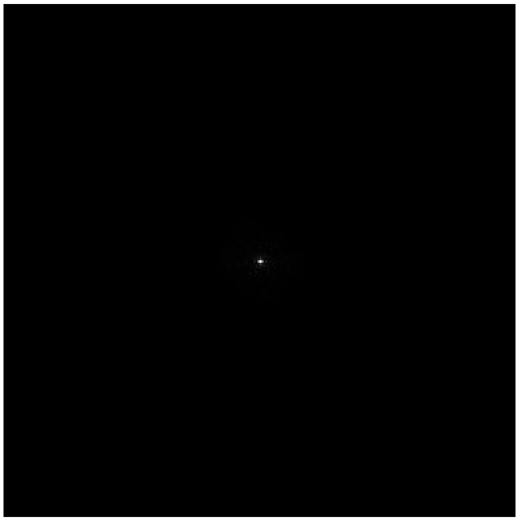

Figure 3. Centered Spectrum of MRI

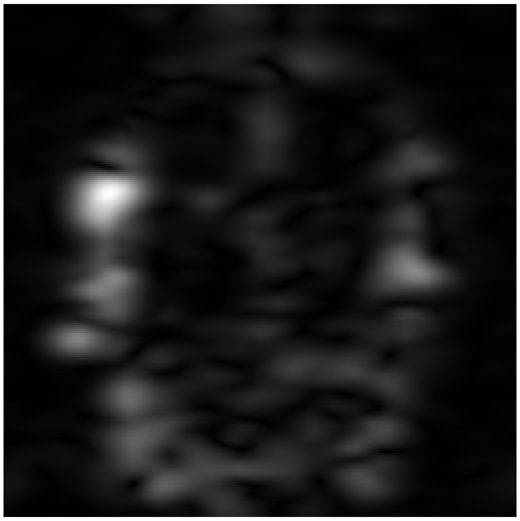

Figure 4. Filtered MRI with 10 Cutoff Freq.

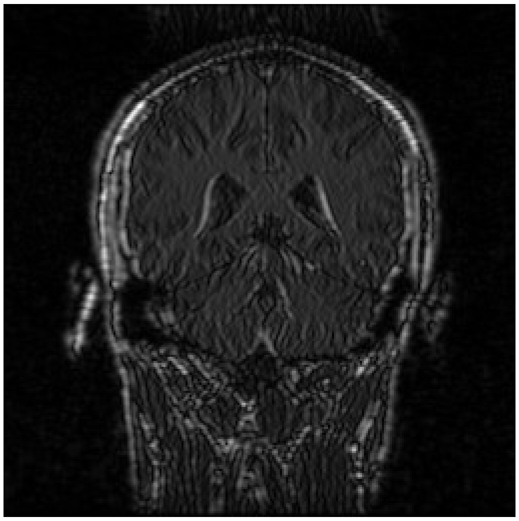

Figure 5. Filtered MRI with 60 Cutoff Freq.

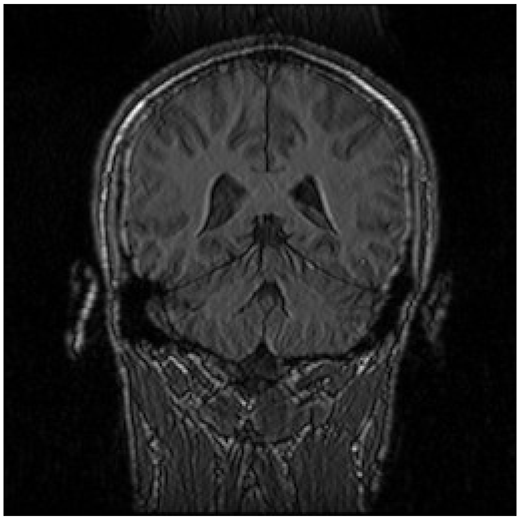

Figure 6. Filtered MRI with 80 Cutoff Freq.

Conclusion and Observations

The exercise was successful in demonstrating how to apply a Fourier transform on a two-dimensional 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

```
////////////////////// Sergio Coronado Assignment 2 README ////////////////

////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> <ySize>
  bin/HW2b <input-file> <spectrum output file> < filtered output file> < cutoff freq.> <xSize> <ySize>

//Cleanup

  $ make clean
```

Code

```
/////////////////////////// homework2a.c ////////////////////////////////////////////
/*
By:   Sergio Coronado
      16.484 Computer Vision
      Assignemnt #2
      Part 1

PURPOSE:
  Program reads in an image performs a FFT outputs the spectrum and then
  performs the inverse FFT

USAGE:

  HW2a <input-file> <ospectrum-file> <reverse-file> <xSize> <ySize>

*/

/////////////////////// Includes /////////////////////////////////////////

#include <stdlib.h>
#include <stdio.h>
#include "ImageProcessing.h"
#include "CursorCntl.h"

```

```
//////////////////////// Constants ///////////////////////////////////////////


#define NUM_ARGS 6
#define FFFT 1
#define RFFT -1

//////////////////////// Main ////////////////////////////////////////////////


int main ( int argc, char ** argv)
{
  unsigned char ** img;           // Matrix Holding Original Image Values
  float ** vectoredImg;           // Matrix holding vectorized image
  unsigned char ** result;        // Matrix holiding Image to output
  unsigned xSize;                 // NUmber of Horizontal pixels
  unsigned ySize;                 // Number of Vertical Pixels
  unsigned nRows;                 // Number of Rows Read in

  if (argc <  NUM_ARGS)
  {
    printError("Usage: HW2a <inputFile> <spectrumOut> <reverseOut> <rows> <columns>\n");
    exit(0);
  }


  xSize = atoi(argv[4]);

  ySize = atoi(argv[5]);

  printOK("Reading Image \n");

  img = ReadImage( argv[1], xSize, ySize, &nRows);

  if (img == NULL)
  {
    exit(-1);
  }

  if ( nRows != ySize)
  {
    ySize = nRows;
  }

  printOK("Vectorizing Image \n");

  vectoredImg = VectorizeImage( img, xSize, ySize);

  printOK("Performing Fourier Transform \n");

Fourier2D(vectoredImg , xSize, ySize, FFFT );

printOK("Normalizing Image \n");

result = NormalizeImage(vectoredImg, xSize, ySize);

printOK("Outputing Spectrum Image \n");

OutputImage(argv[2], result, xSize, ySize);

printOK("Destroying Spectrum Image \n");

DestroyImage (result , xSize, ySize);

printOK("Performing Reverse Fourier \n");

Fourier2D(vectoredImg, xSize, ySize, RFFT);

printOK("Normalizing Image \n");

result = NormalizeImage(vectoredImg, xSize, ySize);

printOK("Outputing Reverse Image \n");

OutputImage(argv[3], result, xSize, ySize);

 printOK ("Cleanup\n");

 DestroyImage (result , xSize, ySize);

 DestroyFloatImage (vectoredImg , xSize, ySize);

 DestroyImage (img, xSize, ySize);

 exit(0);

}
```

```
///////////////////////// homework2b.c ///////////////////////////////////////////
/*
By:   Sergio Coronado
       16.484 Computer Vision
       Assignemnt #2
       Part 1

PURPOSE:
  Program reads in an imagecenteres the spectrumand perfomrs an FFT and a pplys a
  buttersworth lowpass filer
```

```

USAGE:

  HW2b <input-file> <ospectrum-file> <filtered-file> < cutoff freq.> <xSize> <ySize>

*/

///////////////////////// Includes /////////////////////////////////////////////

#include <stdlib.h>
#include <stdio.h>
#include "ImageProcessing.h"
#include "CursorCntl.h"

///////////////////////// Constants ////////////////////////////////////////////

#define BUFFER_SIZE  50
#define NUM_ARGS 7
#define FFFT 1
#define RFFT -1

///////////////////////// Mian /////////////////////////////////////////////////


int main ( int argc, char ** argv)
{
  unsigned char ** img;           //Matrix Holding Image Values
  float ** vectoredImg;           // Matric holding vectored image values
  unsigned char ** result;        // Matric holding normalized image to output
  unsigned xSize;                 //NUmber of horizontal pixels
  unsigned ySize;
  unsigned nRows;
  int cutOff;
  char message[BUFFER_SIZE];


  if (argc <  NUM_ARGS)
  {
    printError("Usage: HW2a <inputFile> <spectrumOut> <filteredOut> <cutoff> <rows> <columns>\n");
    exit(0);
  }

  cutOff = atoi(argv[4]);

  xSize = atoi(argv[5]);

  ySize = atoi(argv[6]);

```

```
printOK("Reading Image \n");

img = ReadImage( argv[1], xSize, ySize, &nRows);

if (img == NULL)
{
  exit(-1);
}

if ( nRows != ySize)
{
  ySize = nRows;
}

printOK("Vectorizing Image \n");

vectoredImg = VectorizeImage( img, xSize, ySize);

printOK("Centering Spectrum\n");

CenterSpectrum(vectoredImg, xSize, ySize);

printOK("Performing Fourier Transform \n");

Fourier2D(vectoredImg , xSize, ySize, FFFT );

printOK("Normalizing Image \n");

result = NormalizeImage(vectoredImg, xSize, ySize);

printOK("Outputing Spectrum Image \n");

OutputImage(argv[2], result, xSize, ySize);

printOK("Destroying Spectrum Image \n");

DestroyImage (result , xSize, ySize);

sprintf(message, "Applying Low Pass Filter Cutoff = %d\n", cutOff);
printOK(message);

ApplyLowPass(vectoredImg, xSize, ySize, cutOff);

printOK("Performing Reverse Fourier \n");

Fourier2D(vectoredImg, xSize, ySize, RFFT);

printOK("Normalizing Image \n");

```

```
  result = NormalizeImage(vectoredImg, xSize, ySize);

  printOK("Outputing Reverse Image \n");

  OutputImage(argv[3], result, xSize, ySize);

  printOK ("cleanup\n");

  DestroyImage (result , xSize, ySize);

  DestroyFloatImage (vectoredImg , xSize, ySize);

  DestroyImage (img, xSize, ySize);

  exit(0);

}
```

```
//////////////////////// ImageProcessing.h /////////////////////////////////////////
/*
By:   Sergio Coronado
        16.484 Computer Vision
        Assignemnt #2
        Part 1

PURPOSE:
  Function Protoypes for image processing

*/
#ifndef IMAGE_PROC
#define IMAGE_PROC



unsigned char ** ReadImage(
  char * filename,
  unsigned xSize,
  unsigned ySize,
  unsigned * numRowsi);

void Fourier2D( float ** img, unsigned xSize, unsigned ySize, int iSign);

float ** VectorizeImage( unsigned char ** img, unsigned xSize, unsigned ySize);

unsigned char ** NormalizeImage( float ** img, unsigned xsize, unsigned ySize);

void OutputImage ( char * filename, unsigned char ** img, unsigned xSize, unsigned ySize);

void DestroyFloatImage (float ** img, unsigned xSize, unsigned ySize);

void DestroyImage ( unsigned char ** img, unsigned xSize, unsigned ySize);

void CenterSpectrum ( float ** img, unsigned xSize, unsigned ySize);

void ApplyLowPass ( float ** img, unsigned xSize, unsigned ySize, int cutOff);

#endif
```

```
//////////////////////// homework2b.c ////////////////////////////////////////
/*
By:   Sergio Coronado
        16.484 Computer Vision
        Assignemnt #2
        Part 1

PURPOSE:
  Program reads in an imagecenteres the spectrumand perfomrs an FFT and a pplys a
  buttersworth lowpass filer

USAGE:

  HW2b <input-file> <ospectrum-file> <filtered-file> < cutoff freq.> <xSize> <ySize>

*/

//////////////////////// Includes ////////////////////////////////////////////

#include <stdlib.h>
#include <stdio.h>
#include "ImageProcessing.h"
#include "CursorCntl.h"

//////////////////////// Constants ///////////////////////////////////////////

#define BUFFER_SIZE  50
#define NUM_ARGS 7
#define FFFT 1
#define RFFT -1

//////////////////////// Mian ////////////////////////////////////////////////


int main ( int argc, char ** argv)
{
  unsigned char ** img;           //Matrix Holding Image Values
  float ** vectoredImg;           // Matric holding vectored image values
  unsigned char ** result;        // Matric holding normalized image to output
  unsigned xSize;                 //NUmber of horizontal pixels
  unsigned ySize;
  unsigned nRows;
  int cutOff;
  char message[BUFFER_SIZE];


  if (argc <  NUM_ARGS)
  {
    printError("Usage: HW2a <inputFile> <spectrumOut> <filteredOut> <cutoff> <rows> <columns>\n");
    exit(0);
  }

  cutOff = atoi(argv[4]);

  xSize = atoi(argv[5]);

  ySize = atoi(argv[6]);

  printOK("Reading Image \n");

  img = ReadImage( argv[1], xSize, ySize, &nRows);

  if (img == NULL)
  {
    exit(-1);
  }

  if ( nRows != ySize)
  {
    ySize = nRows;
  }

  printOK("Vectorizing Image \n");

  vectoredImg = VectorizeImage( img, xSize, ySize);

  printOK("Centering Spectrum\n");

  CenterSpectrum(vectoredImg, xSize, ySize);

  printOK("Performing Fourier Transform \n");

  Fourier2D(vectoredImg , xSize, ySize, FFFT );

  printOK("Normalizing Image \n");

  result = NormalizeImage(vectoredImg, xSize, ySize);

```

```
 printOK("Outputing Spectrum Image \n");

 OutputImage(argv[2], result, xSize, ySize);

 printOK("Destroying Spectrum Image \n");

 DestroyImage (result , xSize, ySize);

 sprintf(message, "Applying Low Pass Filter Cutoff = %d\n", cutOff);
 printOK(message);

 ApplyLowPass(vectoredImg, xSize, ySize, cutOff);

 printOK("Performing Reverse Fourier \n");

 Fourier2D(vectoredImg, xSize, ySize, RFFT);

 printOK("Normalizing Image \n");

 result = NormalizeImage(vectoredImg, xSize, ySize);

 printOK("Outputing Reverse Image \n");

 OutputImage(argv[3], result, xSize, ySize);

 printOK ("cleanup\n");

 DestroyImage (result , xSize, ySize);

 DestroyFloatImage (vectoredImg , xSize, ySize);

 DestroyImage (img, xSize, ySize);

 exit(0);

}
```