# PCSC - Image processing Project

Sofia Dandjee Thomas Fry

13 décembre 2019

## 1 Features of the program

### 1.1 Reader class

The Reader class loads .png and .jpeg images into either a Channel or a RGBImage. Channel is a typedef that consists in a 2-dimensional vector of integer pixels. It is used to represent either a greyscale image or a R/G/B channel of a colored image. RGBImage is another typedef that represent colored :

- loadCImg(string) : It loads a CImg<int> using the path of the image. This method is used by loadRGBImage() and GSImage().
- extractRedChannel(string) : Extracts the red Channel using the path of the image.
- extractGreenChannel(string) : Extracts the green Channel using the path of the image.
- extractBlueChannel(string) : Extracts the blue Channel using the path of the image.
- convertImage() : It converts a CImg<int> type into a RGBImage.
- convertChannel() : It converts a CImg<int> type into a Channel.
- convertColoredtoGS() : It loads a colored image into a Channel, its greyscale version (by averaging the pixel values of the RGB channels).
- loadRGBImage() : It loads a colored image directly into a RGBImage using the path of the file. It uses the loadCImg() and convertImage() methods.
- load GSImage() : It loads a greyscale image directly into a 2D vector using the path of the file. It uses the loadCImg() and convertChannel() methods. This method is useful to detect the contours of colored image.

### 1.2 ContourExtractor class

This class implements the Sobel Filter, an operator used in image processing to detect contours of an image. This operator calculates the gradient of each pixel's intensity. This gradient represents a sudden change in luminosity corresponding to contours. Its methods are :

- preprocess() : Adds a padding of 0s all around an image to prepare for convolution.
- convolution() : Computes a convolution between an input image and a filter image. The result is of the same size as the input image.
- detectVerticalEdges() : Detects the vertical contours of an image by convoluting it with the vertical gradient.
- detectHorizontalEdges() : Detects the horizontal contours of an image by convoluting it with the horizontal gradient.
- detectAllEdges() : Detect the contours of an image by computing the norm of its horizontal edges and vertical edges.

### 1.3 Writer class

The writer class creates CImg<int> type images from a Channel or an RGBImage in order to save them locally. Its methods are :

- createRGBImage() : Overloaded function - Writes a colored image either from 3 channels or from an RGB image (3D vector).
- createGSImage () : Creates a greyscale image.
- createFFTImage () : Creates a greyscale image from a 2D vector of double.

### 1.4 Histogram class

The histogram class creates histograms, vector of pixel intensity counts from either a RGBImage or a Channel, in order to plot them. There are two typedefs of histograms : ChannelHistogram and ImageHistogram. A ChannelHistogram is a 1D vector of pixel intensity values (integers), either for a greyscale image or only one color. It shows the number of pixels in an image at each different intensity value (black, red, green, OR blue). A ImageHistogram is a 3D vector of pixels intensity values for a color image. It shows the number of pixels in an image at each different intensity value (red, green AND blue). Its items are :

- computeImageHistogram(RGBImage) : Builds a ImageHistogram from a RGBImage.
- computeChannelHistogram(Channel) : Builds a ChannelHistogram from a Channel.

### 1.5 HistogramWriter functions

Two writer functions were implemented to write the two types of histograms in text file :
— WriteImageHistogram(ImageHistogram, string) : Writes the ImageHistogram in a text file.
— WriteChannelHistogram(ChannelHistogram, string) : Writes the ChannelHistogram in a text file.
Then the text files are plotted with the read_histograms Python file :
— plot_histogram(string, char, string) : Plots the intensity histogram of a Channel from its file name, the character of its color and its title.

### 1.6 FFT functions

The Fast Fourier Transform and Inverse Fast Fourier Transform are computed according the Cooley Tukey algorithm with the FFT() and IFFT() functions. For odd dimensions, the naive algorithm DiscreteFourierTransform2D() is used. The function FFTModulus() calculates the modulus of a Fourier Transform.

### 1.7 Run class

This class represents the user's interaction with the program. A Run contains a Reader, a Writer, a ContourExtractor and a Histogram. Its methods are :
— chooseImagePath() : Lets the user choose the filepath of the image.
— isColored() : Asks the user if the image is colored or not, because the image will be processed differently according to the answer.
— askUser() : Asks the user if he wants to compute an intensity histogram, extract the contours of the image or compute the Fast Fourier Transform of the image.
— extractContours() : Extracts the contours of an image and saves it in a file in the results folder as "contour.jpeg".
— computeFFTandIFFTGS() : Computes the FFT and IFFT of a greyscale image and saves its modulus and reconstruction in "FFT.png" and "reconstructed.png" in the results folder..
— computeFFTandIFFColor() : Computes the FFT modulus of each channel of a colored image and reconstructs the colored with the IFFT. Results are saved in "reconstruced_colored.png", "red_FFT.png", "green_FFT.png", "blue_FFT.png".
— computeHistogram() : Computes an intensity histogram of an image and saves it as .txt file in the results folder.

## 2 Tests of the program

### 2.1 ReaderTest

— RGBImageisLoaded : Checks that loadRGBImage on mandrill.png (a colored image) works, that is it loads a 3-channelled RGBImage with the right dimensions.
— LennaIsLoaded, MountainIsLoaded : Checks that the Channel returned by loadGSImage() on lenna.jpeg and mountain.png (two greyscale image) has the right dimensions.
— RGBChannelsAreLoaded : Checks that extractRedChannel(), extractBlueChannel(), extractGreenChannel() on mandrill.png return Channels with the right dimensions.
— convertColoredToGSWorks : Checks that convertColoredtoGS() on mandrill.png returns a Channel with the right dimensions.

### 2.2 contourExtractorTest

— zeroPadding : Checks that the process() method works on a random 4x3 Channel.
— ConvolutionVertical, ConvolutionHorizontal, convolutionAllEdges : Check that the detectVerticalEdges(), detectHorizontalEdges(), detectAllEdges() methods give a correct result on a random 4x5 Channel.
— VerticalEdgeDetectorLenna, HorizontalEdgeDetectorLenna, AllEdgeDetectorLenna : Check that the detectVerticalEdges(), detectHorizontalEdges(), detectAllEdges() methods give a correct visual result on "lenna.jpeg" and check that the original Channel and output Channel have the same dimensions. The output of this test "vertical_contours_lenna", "horizontal_contours_lenna", "contours_lenna" are saved in the results folder.
— VerticalEdgeDetectorMandrill, HorizontalEdgeDetectorMandrill, AllEdgeDetectorMandrill : Same as above with "mandrill.png"

— VerticalEdgeDetectorLich, HorizontalEdgeDetectorLich, AllEdgeDetectorLich : Same as above with "lich.png"
— AllEdgeDetectorFruits, AllEdgeDetectorGirl, AllEdgeDetectorMonarch, AllEdgeDetectorTulips, AllEdgeDetectorMountain : Check that the detectAllEdges() methods give a correct visual result on "fruits.png", "girl.png", "monarch.png", "tulips.png", "mountain.png" and check that the original Channel and output Channel have the same dimensions.

### 2.3 writerTest

— GSImageIsWritten : Checks that the "lenna.jpeg" image is well rewritten with createGSImage() (see "rewritten_lenna.jpeg" in the results folder).
— RGBImageIsWritten : Checks that the "mandrill.png" image is well rewritten with createRGBImage() (see "rewritten_mandrill.png" in the results folder).
— RGBConvertedtoGS : Checks that the "mandrill.png" image is converted to its greyscale version(see "greyscale_mandrill.png" in the results folder).

### 2.4 histogramTest

— ImageHistogramIsComputed : Checks that computeImageHistogram runs without failure and returns a ImageHistogram with the right size on the "mandrill.png".
— GreyScaleChannelIsComputed : Checks that computeChannelHistogram runs without failure and returns a ChannelHistogram with the right size on the "lenna.jpeg".
— ColorChannelHistogramsAreComputed : Checks that computeChannelHistogram returns a ChannelHistogram with the right size for each of the three color Channels of "mandrill.png".
— BlackImageHistogramIsRight : Checks that computeImageHistogram returns the right histogram for a black RGBImage.
— WhiteImageHistogramIsRight : Same as the test above, for a white RGBImage.
— ImageHistogramCountsAreRight : Checks that the sum of the histogram counts equals the number of pixels for : all color intensity, red intensity, green intensity and blue intensity, on the "mandrill.png".
— ImageHistogramAndChannelHistogramsCorrespond : Checks that every axis of the ImageHistogram correspond to the right ChannelHistogram (in order : red, green, blue), on the "mandrill.png".

### 2.5 fftTest

— ConvertInComplexWorks : Checks that convertImageInComplex converts a Channel in the right ComplexVector.
— BlackDFTIsRight : Checks that the DiscreteFourierTransform2D and FFTModulus returns null vectors of complex from a black Channel.
— DFTIsRight : Checks that the DFT of a random Channel and its size are right with the naive algorithm.
— FFTWorksOnZeros : Checks that the FFT of a null/black Channel is null.
— FFTWorksOnOnes : Checks that the FFT of a whole one intensity Channel is right.
— FFTWorksOn2x2Vector, FFTWorksOn2x2Vector2, FFTWorksOn4x4Vector, FFTWorksOn4x4Vector2, FFTModululusWorkOn4x4Vector, FFTWorksOn4x4Vector3, FFTWorksOn8x8Vector : Seven tests to check that the FFT or its modulus on random Channels are right.
— FFTLenna, IFFTLenna, FFTMandrill, IFFTMandrill, FFTFruits, FFTMountain : Checks the image product of the FFT on some pictures, and check that the Inverse Transform Fourier (IFFT) builds them back into their original pictures.
— IFFTFruitsColored : Performs the FFT and IFFT on each Channel of an image, and check that the original image is reconstructed with createRGBImage().
— DFTWorkson3x3, DFTWorkson2x4, DFTWorkson3x4 : Three tests to check that the FFT on random Channel of different dimensions is right.

## 3 Compile and execute the program

This project needs the JPEG and PNG libraries installed to compile. On Linus, the lines "define cimg_use_png 1" and "define cimg_use_png 1" have to be commented in the Reader.h file. The default settings have these lines already commented.

### 3.1 Install the required libraries

Open the terminal and install the PNG library (first command is for Linux, second is for Mac) :

```
$ sudo apt-get install libpng-dev
$ brew install libpng
```

Open the terminal and install the JPEG library (first command is for Linux, second is for Mac) :

```
$ sudo apt-get install libjpeg-dev
$ brew install libjpeg
```

### 3.2 Build the project on the terminal

1. Go to the project directory :

```
$ cd project_directory_path
```

2. Create a build folder and go inside it :

```
$ mkdir build
$ cd build
```

3. Configure the CMake files :

```
$ cmake ..
```

4. Build and compile the project :

```
$ make
```

5. Run the main program :

```
$ ./main
```

6. Run the tests :

```
$ ./name_of_the_test
```

7. Generate the documentation (in the doc folder) :

```
$ doxygen Doxyfile
```

## 4 Typical flow

The program lets you pick either an image from different test images located in the images folder or a custom image. The user has to specify if it is a greyscale or an RGB image. The user then decides if he wants to compute an intensity histogram, extract the contours of the image or calculate the Fast Fourier Transform of the image. If the user chooses to extract the contours of the image, the contours are extracted using ContourExtractor and are saved in the results folder. If the user chooses to calculate the Fast Fourier Transform, the modulus and the reconstruction with the inverse are computed and can be found in the results folder. For a RGB image, the FFT of each spectrum will be computed. Finally, if the intensity histogram is chosen, the histogram is saved as a .txt file in the histograms folder and can be plotted with the Python script provided. For a RGB image, 2D histogram of each spectrum is computed.

## 5 Limitations and prospects

The project only supports .png, .jpeg and .jpg formats for now but could be extended to .gif, .bmp and .tiff images. To process the images, filters could be added to the code (spatial/Fourier, lowpass/highpass) and affine transformations could be implemented (reflection, scaling, rotation, shear). Also the code could be generalized to sound processing.