Medical Image Processing

Team Name: Big Pharma
Boom, Oliver
Khandahari, Hameed

February 10, 2019

Abstract

A program was written in the C++ programming language to enable a user to filtering images to aid with medical diagnostics. The `CImg` library was used for loading the image and the data structure of the pixels. A variety of filters were employed including some which make use of an image-kernel convolution algorithm.

1 Design of the Code

Fourteen filters were implemented for this project. These range from simple filters where colour images are turned into grayscale to more advanced ones where convolution algorithms are used to achieve sharpening and edge detection. The full list of filters implemented is presented in Table 1.

Simple	**Convolution-based**
Grayscale	Sharpen (High and low setting)
Brighten	Blur (High and low setting)
Low pass filter	Gaussian Blur
High pass filter	Edge detection
Band pass filter	Psychedelic
Colour swap	

Table 1: Summary of the available filters

The images were loaded using the `CImg` library [1]. This library also provides a framework for accessing the `RGB` values at every `x` and `y` coordinate. Although the images included in this repository are all `.bmp` files, this programme will work for other file formats if external dependencies (`Magick++`) are installed.

The simple filters loop through the `RGB` values for every `x` and `y` pixel and only uses this information to determine the `RGB` values for every corresponding `x` and `y` pixel in the filtered image

The convolution-based filter, however, also considers the `RGB` values of the surrounding pixels, this allows for effects and hence allows for more advanced filtering. Each convolution-based filter uses a specific kernel which is convolved with the array of pixels to produce a desired effect. This convolution works through taking a sum of products of corresponding pixel and kernel array values. Mathematically,

$$f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} k(s,t)\, p(x-s, y-t), \qquad (1)$$

where f is the filtered image, k is the kernel and p is the input image. [2]

2 Structure of the code

The software was structured with 5 C++ source files; *main, user, filter, convolution* and *display*, with each file (with the exception of *main*) having a corresponding header file. In addition to these the CImg header file was also used. An object orientated approach was chosen, and 3 classes were created *Filter, Kernel, Display* and *User* using encapsulation, abstraction, inheritance and polymorphism to optimise the work flow of the program.

The *UI* files contained functions relating to the user interface and was responsible for walking the user through the filtering process. It also contained the wrapper function where the filters are applied. The *filter* files, contained functions which applied the different effects to the image loaded in. The *convolution* files, which contained the *Kernel* class (a child of the *Filter* class), allowed for convolution based filters to be applied. And the *display* class is responsible for outputting the images for the user to view.

3 Execution

A user-friendly command-line interface was developed. When the program is executed, the user is greeted with a selection of available files. These files are taken from the listings of the input images directory (`\medical_imager\Images`). The medical images in this directory are taken from *MedPix*[3] After the file is selected, the user will be asked to select one of the filters from the choice presented in Table 1. If the greyscale filter is selected then this filter is applied to the image and then prompts the user to apply another filter if desired. The program outputs two windows, showing the image before and after the filtering. The title of the window clearly identifies which is which.

References

[1] David Tschumperle. Cimg library - c template image processing toolkit, url=http://cimg.eu/.

[2] Linda G. Shapiro and George C. Stockman. *Computer vision*. Prentice-Hall, 2001.

[3] Medpix. url = https://medpix.nlm.nih.gov/home.