Practical 1

EERI 322

By:

J.P. Beukes 260281071

D.D. Krynauw 260***

Submitted in pursuit of the degree

BACHELOR OF ENGINEERING
In
COMPUTER AND ELECTRONIC ENGINEERING

North-West University Potchefstroom Campus

July 28, 2017

Supervisor: Mr. C. van der Merwe
Potchefstroom
2017

Innovation through diversity®

NORTH-WEST UNIVERSITY ®
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT
POTCHEFSTROOM CAMPUS

Contents

List of Figures

List of Abbreviations

DFT	Discrete Fourier Transform
IDFT	Inverse Discrete Fourier Transform
DSP	Digital Signal Processing
FFT	Fast Fourier Transform

1 Introduction

Digital systems are becoming increasingly popular in the modern world of signal theory. It is therefore essential to develop techniques for manipulating digital signals. A prominent division of digital signals are images.

This practical investigates methods for processing digital images in both the time and frequency domain. The platform chosen for this practical is Qt Creator, with all programming done in C++.

2 Overview and Background

2.1 Image as a discrete signal

An image can be modelled as a two-dimensional discrete signal with each pixel having three channels, namely red, green and blue [?].

2.2 Edge detection

To implement edge detection, the image is first separated into its composing channels, thereafter both vertical and horizontal edge detection are applied. The channels are then summed to produce the edges of the original image.

2.3 Grayschale for DFT

To apply a Discrete Fourier Transform (DFT), the image must first be converted to a grayschale image. Qt's standard method for grayschale conversion is used. This formula for the gray value of a pixel is implemented in Qt as follows:

$$gray = (red \times 11 + green \times 16 + blue \times 5) \div 32 \tag{2.1}$$

2.4 DFT and IDFT

After calculating the DFT, the magnitude and phase components are displayed on separate images. The Inverse DFT (IDFT) is then drawn from the DFT to produce the original grayschale image.

3 Separating RGB channels

To extract every channel, the original image is iterated pixel, by pixel, copying every channel to corresponding pixels of separate images. Standard qRed, qGreen(), qBlue() and setPixel() functions is used to implement channel separation in Qt. The original image and results form separating the channels is presented in Figure 3.1 to **??** below.

Figure 3.1: Image used in this report.

4 Edge Detection

As stated above, edge detection is done over several steps on each of the rgb channels separately. The first step is to apply a horizontal and vertical Kernel Convolution to every channel.

This practical's kernel (i.e. Sobel operator) is a 3×3 as shown in Figure **??**. For each pixel of every channel, the pixel is multiplied by the kernel. The elements of the resulting matrix are then summed to produce a pixel value for a new image.

Kernel Convolution using the horizontal Sobel operator is first applied to every channel (Figure **??** to **??**)

The process is then repeated using the vertical Sobel operator (Figure **??** to **??**)

These images carrying the resulting horizontal and vertical edges are then summed together to produce images containing all of the edges detected on each channel (Figure **??** to **??**)

Finally, the edges are then summed together to produce a composite image containing all edges detected through the use of Sobel edge detection. (Figure **??**)

5 DFT and IDFT

The Discrete Fourier Transform (DFT) is a process used to obtain the frequency spectrum of a sequence. In the case of this practical, the sequence is an image and therefore the DFT will produce a two-dimensional complex valued matrix which can be displayed as a magnitude and phase image.

The following equation was implemented to calculate the DFT of an image:

$$F[k,l] = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-j2\pi(\frac{k}{M}m + \frac{l}{N}n)}, \tag{5.1}$$

where $j = \sqrt{-1}$ [?].

After transformation, the resulting phase and magnitude of the DFT is displayed on separate images. (Figure **??** and **??**)

The reverse process yields the Inverse DFT (IDFT), which is implemented using

$$f[k,l] = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} F[m,n] e^{j2\pi(\frac{k}{M}m + \frac{l}{N}n)}, \tag{5.2}$$

where $j = \sqrt{-1}$ [?].

6 Conclusion

As presented in this report, this practical demonstrates common techniques used in digital image processing. However, the program written in this practical is not yet extremely robust. One of its limitations is that the DFT can only handle square images. It should also be noted that the DFT is only designed to handle images with an even number of pixels in its width and height.

The DFT and IDFT algorithm used in this practical is simple, but inefficient. A common alternative is to use a FFT (Fast Fourier Transform), which will reduce the computing time from minutes to milliseconds. The current DFT has a complexity order of $\mathrm{O}(n^4)$, consequently a small increase in image width and height yields a large increase in computation time. It is found that a 100x100 image takes approximately eight seconds to process.