UNIVERSITY OF LINCOLN

SCHOOL OF MATHEMATICS AND PHYSICS

Classification of Biological Organisms from Images Using Advanced Mathematical Techniques

Timothy Smith

25796944

Supervised by Dr Danilo Roccatano

February 26, 2024

Scientific Report in the 3rd Year Module MTH3009 Mathematics Project

Abstract

Contents

1 Introduction

2 The Elliptic Fourier Transform

The Fourier transform is a way of finding constituent frequencies within a function on the real domain. The Fourier transform extends the concept of the Fourier series (which openerates on a bounded interval) to the real domain.

2.1 The Fourier Series

The Fourier series is defined using the equations described below. It takes in a single repeating unit of a periodic function and allows us to generate an infinite sum which converges to the same function. This is useful because the resultant sum ends up depending on trigonometric functions.

Firstly we have a set of numbers, A_0, A_n and B_n:

$$\begin{aligned} A_0 &= \frac{1}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} s(x)\,dx \\ A_n &= \frac{1}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} s(x)\cos\left(\frac{2\pi nx}{P}\right)dx \\ A_b &= \frac{1}{P}\int_{-\frac{P}{2}}^{\frac{P}{2}} s(x)\sin\left(\frac{2\pi nx}{P}\right)dx \end{aligned} \tag{1}$$

Here A_0 is a constant, and A_n and B_n are functions of n. In fact, since both the denominator of the fraction and the range of the integral for A_0 are the same, P, A_0 is simply the average around which the function oscillates.

These coefficients allow us to define the Fourier series:

$$s(x) \sim A_0 + \sum_{n=1}^{\infty}\left(A_n\cos\left(\frac{2\pi nx}{P}\right) + B_n\sin\left(\frac{2\pi nx}{P}\right)\right) \tag{2}$$

Here we use a $\sim$ because this series doesn't always converge to the desired function, although in most cases it does.

2.2 The Fourier Transform

The transform function ($S(t)$) for frequency f is given by (3).

$$S(t) = \int_{-\infty}^{\infty} s(t)\cdot \mathrm{e}^{-i2\pi ft}dt \tag{3}$$

2.3 Fourier Analysis

2.4 Elliptical Fourier Analysis

3 Processing the Images

In order to find shapes in our images for the purposes of analysis, we will need to detect edges. For this we will use the Canny Edge detection method, developed by John F. Canny.

3.1 Convolution Kernels

The most basic aspect of edge detection is the convolution kernel. This is an $M \times M : \{M = 2n+1, n \in \mathbb{N}\}$ matrix (an odd sided square matrix). Odd side lengths allow the kernel to be centered at each pixel.

If we let the function $f(x,y)$ represent the original image, $g(x,y)$ represent the convolved image and ω represent the kernel, then:

$$g(x,y) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} \omega(i,j) f(x-i, y-j) \tag{4}$$

Notice that the coordinates of the kernel are not 1 to M as with a traditional matrix, but rather $-n$ to n.

The effect of this on an image will be to make each pixel of a convolved image a function of the surrounding pixels. The simplest convolution matrix is the identity convolution matrix, (5).

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

More examples include edge detection kernels like (6) and (7).

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{7}$$

3.2 Canny Edge Detection

4 Conclusion

Note: these are just example citations. [1, p. 150] [2] [3] [4] [5] .

5 Acknowledgements

6 Appendix

Note: These are unlikely to be in the final report, they are simply there as an example appendix for now.

main.cpp

```cpp
// Credit:
// https://gnome.pages.gitlab.gnome.org/gtkmm-documentation/sec-helloworld.html
#include "elliptic_fourier.h"
#include <gtkmm/application.h>

int main(int argc, char *argv[])
{
        auto app = Gtk::Application::create("Test Application");

        return app->make_window_and_run<EllipticFourier>(argc, argv);
}
```

elliptic_fourier.h

```cpp
#ifndef ELLIPTIC_FOURIER_H
#define ELLIPTIC_FOURIER_H

#include <gtkmm/window.h>
#include "drawing_area.h"

class EllipticFourier: public Gtk::Window
{
public:
        EllipticFourier();
        ~EllipticFourier() override;
protected:
        GraphArea ga;
};

#endif
```

elliptic_fourier.cpp

```cpp
// Credit:
// https://
// gnome.pages.gitlab.gnome.org/gtkmm-documentation/chapter-drawingarea.html
#include "elliptic_fourier.h"
#include <iostream>

EllipticFourier::EllipticFourier()
{
        set_child(ga);
}

EllipticFourier::~EllipticFourier()
{
}
```

drawing_area.h

```cpp
#ifndef DRAWING_AREA_H
#define DRAWING_AREA_H

#include <gtkmm/drawingarea.h>

class GraphArea: public Gtk::DrawingArea
{
public:
        GraphArea();
        virtual ~GraphArea();

protected:
        void draw(
                const Cairo::RefPtr<Cairo::Context>& cr,
                const int width,
                const int height
        );
        double waveform(double x);
};
```

```
#endif
```

drawing_area.cpp

```
#include <cairomm/context.h>
#include <cmath>
#include "drawing_area.h"

using std::sin;

GraphArea::GraphArea()
{
	set_draw_func(sigc::mem_fun(*this, &GraphArea::draw));
}

GraphArea::~GraphArea()
{
}

void GraphArea::draw(
	const Cairo::RefPtr<Cairo::Context>& cr,
	const int width,
	const int height
) {
	cr->set_source_rgb(0, 0, 0.5); // blue

	int i = 0;
	cr->move_to(i, height/2 + waveform(i/20.0) * height/4);
	for(i = 1; i <= width; i++)
	{
		cr->line_to(i, height/2 + waveform(i/20.0) * height/4);
	}
	cr->stroke();
}

double GraphArea::waveform(double x)
{
	double y = 0;
	double freqs[] = {1, 1.1, 0.8};
```

```
        double mags[] = {1, 1.1, 0.8};

        for(int i = 0; i < sizeof(freqs)/sizeof(freqs[0]); i++)
        {
                y += sin(x * freqs[i]) * mags[i];
        }

        return y;
}
```

7 Bibliography

References

[1] M. S. Nixon and A. S. Aguado, *Feature Extraction and Image Processing for Computer Vision*. Academic press, 2019.

[2] D. W. Thompson, *On Growth and Form*. Cambridge: Cambridge University Press, 1961.

[3] P. E. Lestrel, *Fourier Descriptors and Their Applications in Biology*. Cambridge: Cambridge University Press, 1997.

[4] M. Cartwright, *Fourier Methods for Mathematicians, Scientists and Engineers*. Chichester: Ellis Horwood, 1990.

[5] I. L. Dryden and K. V. Mardia, *Statistical Shape Analysis, with Applications in R*. Hoboken, New Jersey, United States: John Wiley & Sons, 2016.