Historic Magnetogram Digitization

A Thesis Submitted to the Committee on Graduate Studies
in Partial Fulfillment of the Requirements for the Degree of Master of Science
in the

Faculty of Arts and Science

TRENT UNIVERSITY

Peterborough, Ontario, Canada

Applied Modelling and Quantitative Methods M.Sc. Graduate Program

September 2019

ABSTRACT

Historic Magnetogram Digitization

Mark Weygang

The conversion of historical analog images to time series data was performed by using deconvolution for pre-processing, followed by the use of custom built digitization algorithms. These algorithms have been developed to be user friendly with the objective of aiding in the creation of a data set from decades of mechanical observations collected from the Agincourt and Toronto geomagnetic observatories beginning in the 1840s. The created algorithms follow a structure which begins with pre-processing followed by tracing and pattern detection. Each digitized magnetogram was then visually inspected, and the algorithm performance verified to ensure accuracy, and to allow the data to later be connected to create a long-running time-series.

Keywords: Magnetogram, geomagnetic, digitization

Acknowledgements

Thanks to: Natural Resources Canada, Aaron Springford, David Riegert, Robyn Fiori, and my supervisors Sabine McConnell and Wesley Burr

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminaries

The Earth's magnetic field (or geomagnetic field) is an ever-changing phenomenon that influences human activity and the natural world in a myriad of ways[17]. Disturbances in this field can be caused by the interactions between the sun, and Earth. These interactions can impact our technologies every day, and are known as Space Weather[19]. They include solar flares, which are bursts of electromagnetic radiation which can result in ionospheric disturbances called geomagnetic storms such as the one that occurred in 1859, referred to as the Carrington Event.

Unlike 1859, our modern societies have become extremely dependent on complex interweaving electronic technologies[25]. Even a short term disruption of a critical part of our systems of can have negative long term effects, such as when two

Canadian telecommunication satellites were disabled in 1994. One was quickly recovered, while the second was fully restored approximately six months later. This left many northern communities without means of communication for an extended period of time, and the financial cost estimated to be as high as $70 million[24].

The modern study of these phenomena have been limited by what data can be obtained, and then easily used on computers. The Natural Resources Canada website only offers data from the year 2000 onward, which makes predictions far into the future difficult[17]. However, in storage at the Geomagnetic Laboratory in Ottawa, are thousands of days of observed data that have yet to be used on a large scale for data processing. This data is stored as magnetograms (see Figure 1.1), which are visual paper records of moment-to-moment variations in local magnetic fields[23]. They were created on photographic paper at the Toronto, and then Agincourt Geomagnetic observatories starting the 1840's, with the recordings continuing later on in Ottawa, where the current observatory resides.

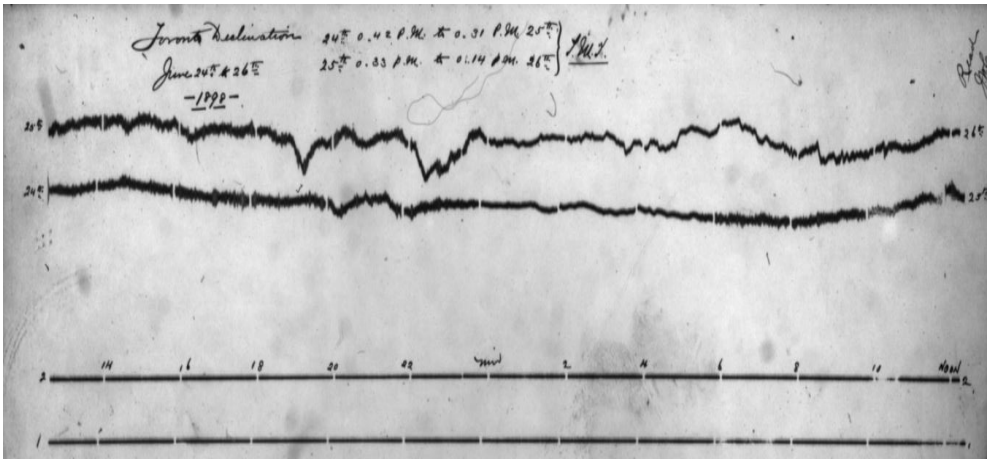

FIGURE 1.1: A Magnetogram from the Toronto Geomagnetic Observatory June 24-26, 1898. The data is spread over two separate lines with their corresponding baselines found near the bottom of the image.

Interest in the digitization of these magnetograms has increased over the past few years, especially since there are only a very small number of organizations in the world who have any data similar to what is in Ottawa. The acquisition of this data from the magnetograms has the potential to create one of the longest, if not the longest, digital time series of geomagnetic phenomena in the world.

Progress to create that end result started in 2012. A combined effort by a research group at Queen's University and Natural Resources Canada took the first step, and scanned a large percentage of the magnetogram images using a high-resolution 35mm microfilm scanner owned by the Queen's Library. With the images now on disk, the team at Queen's were able to create a proof of concept digitization algorithm. This proved that the process could be done under extremely specific conditions.

1.2 Thesis Statement

This thesis will explore if the magnetograms from the Toronto, and Agincourt geomagnetic observatories can truly be successfully digitized, in the sense of creating a validated digital time series record of the local magnetic field for more than just a few select cases. We will explore if the proof-of-concept algorithm is a viable option for digitizing the magnetograms, and specifically, if changes could be successfully implemented that would allow the proof-of-concept algorithm to produce consistent, and accurate results.

1.3 Objectives and Goals

There are three general objectives for this thesis:

1. Take the proof-of-concept, highly supervised, limited scope digitization algorithm, and develop it into a functioning automated unsupervised digitization algorithm.

2. Create an additional algorithm that can produce a similar output to that of the proof-of-concept algorithm to be used for comparison, and increase the number of magnetograms that can be digitized, i.e., add fail-safes to control edge cases.

3. Create a Shiny Web-App that can be used to aid in the identification of successfully digitized magnetograms via crowd-sourcing to produce a validated data set of magnetic field strength as a function of time.

1.4 Motivation

The digitization of the magnetograms collected from the Toronto and Agincourt geomagnetic observatories is of high interest to Natural Resources Canada. They plan on using the data for many projects, both in researching geomagnetic phenomena, and in helping create better prediction models for solar phenomena. The data might also be of interest to other organizations, and research groups. Therefore there are plans to release the data to the public when completed.

1.5 General Outline

This chapter briefly introduces geomagnetic phenomena, magnetograms, and outlines the thesis with its objectives and goals.

Chapter 2 explains geomagnetic phenomena, and why it is an important area of research. The background of relevant concepts will also be provided, as this will inform the reader on previous work done on the project. This will be accomplished by discussing in depth analysis on the past digitization algorithm, with identification of problems. This will then be followed by a brief introduction to image classification by means of crowd-sourcing through the example of the *Galaxy Zoo* project.

The work done on the project will be discussed in Chapter 3. This includes: the changes, and improvements made to the original digitization algorithm; the creation of an additional digitization algorithm; and the creation of a *Shiny Web-App* to aid in the classification of successful digitized magnetograms.

Chapter 4 is focused on the optimization of the working magnetogram digitization algorithm. It begins with identifying the areas of the algorithm that cause slow downs or "bottle necks", investigates why they occurred, followed by solutions.

Chapter 5 will cover the comparison of the main digitization algorithms, and the results obtained thus far. The use of the digitization algorithms will also be demonstrated, along with the future plans for the algorithms.

Chapter 6 will wrap up the thesis with a conclusion, and plans for future work. This includes what still needs to be accomplished, and a strategy for overcoming the more difficult magnetograms.

Chapter 2

Background

2.1 Geomagnetic Data

Space weather has a direct impact on our lives[27]. This includes electrical power transmission, radio/satellite communications, and GPS systems[19]. These systems, and service we have become so accustomed to, can fail when experiencing a geomagnetic storm. The effects of these storms range from minor interference, to the collapse of entire electrical power grids such as the Hydro Quebec blackout in 1989 which left millions without power[11]. Thus many important electronic systems have been designed to reduce the effects of these storms, along with the aid of warning systems.

Prediction of geomagnetic activity has been limited to one-to-three day forecasting via observation of the Sun[18,19]. Although these monitoring systems collect

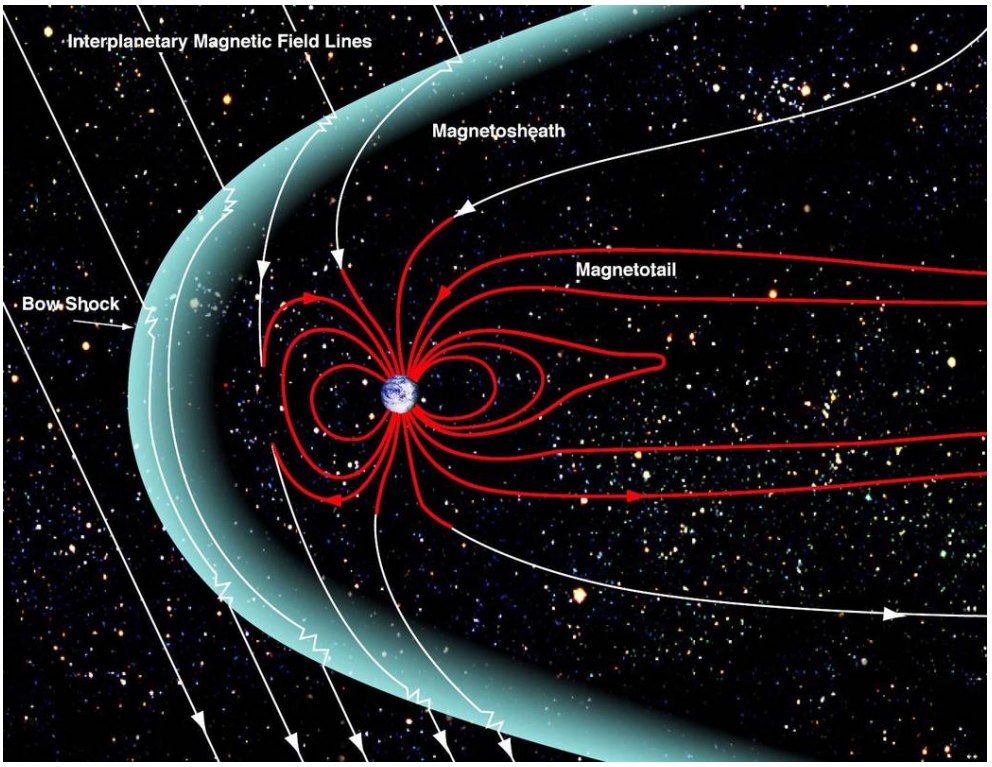

FIGURE 2.1: An illustration of Earth's magnetosphere. It is controlled by the planet's magnetic field but its shape is determined by solar winds. Credit: NASA/Goddard/Aaron Kaase

a huge amount of data, there is a serious lack in contiguous temporal data, e.g., long-running time series. This may not mean the data does not exist, but rather that it is very different to access. This is the case for the magnetograms. Since a disturbance in the Earth's magnetic field can be an indicator of space weather the magnetograms provide an opportunity to fill this void. If the digitization is successful, Canada will have access to over 150 years of local geomagnetic data, which may help improve forecasting, and ultimately help protect the systems and service we depend on. An additional benefit for the retrieval of this data may be for its use in verifying times of large earthquakes as was discovered by Krüger[8].

2.1.1 Earth's Magnetic Field Components

The Earth's magnetic field is a vector quantity, and can be represented by a three-dimensional vector. The data was collected with this in mind, and having all the components is important. This is why the available magnetograms belong to one of three groups: declination, vertical, or horizontal.

TABLE 2.1: Magnetic field components as defined by Natural Resources Canada

Component	Description
D	magnetic declination, defined as the angle between true north (geographic north) and the magnetic north (the horizontal component of the field). D is positive eastward of true North.
V	the vertical component of the magnetic field vector; by convention V is positive downward
H	the horizontal intensity of the magnetic field vector

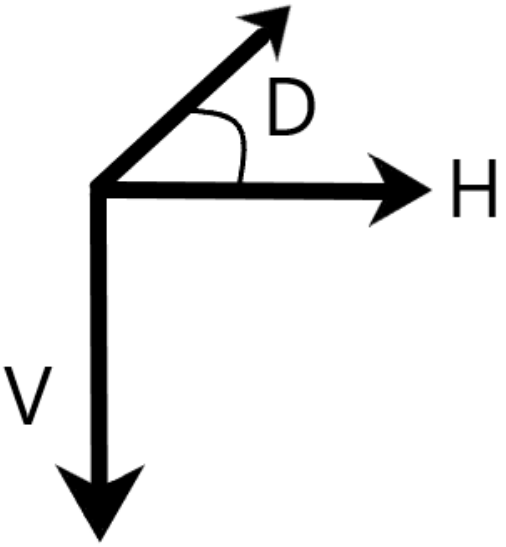

FIGURE 2.2: Visual representation of the magnetic field components

2.2 Extracting Time-Series Data from Images

The digitization of two-dimensional data from images has been vital in extracting accurate data from hand made, and mechanically created images/graphs. Prior to scanning the images to a computer, it was common for individuals to use a ruler, and extract the data by hand from a given image. Software has since been developed to meet this need. There are a number of open-source software programs currently available such as Engauge Digitizer, that provide simple-to-use single image digitization[16].

In addition to the magnetograms created in Canada, there exist magnetograms from other geomagnetic observatories such as those in the UK that have yet to be fully digitized. There has been efforts made toward this goal, and we discuss this below. In addition we will explore work done for similar challenges such as the digitization of seismograms.

2.2.1 British Geological Survey

The British Geological Survey began collecting geomagnetic data in the mid 1800s. They started with one observatory in 1840, and continued to nine total with each having a different operational life span. From these nine observatories about 300,000 magnetograms have been collected from across the UK. Most of these have been scanned, and are available as images on their website[5].

Efforts to preserve their magnetograms began in 2008 with the scanning of the original photographic papers. Then in an attempt to extract time-series data from select images, the BGS began development of a Java program to accomplish this goal in 2009[3]. The user would select 'control points' which identified start/end positions for both the traces (magnetic field strength recordings), and baselines. The algorithm would then move across the image left to right taking vertical slices. The distribution of the pixels in these slices was then used to identify the greatest peak in intensity which would then become the 'point' to represent that slice[3].

The algorithm was not completed as it was later noted that they selected an open source digitization software, Engauge Digitize to work with instead[4]. This software is similar to that of the program that was in development by BGS as it required manual input, and could only handle one image at a time as a result. The main difference is that the user must select an additional axis. This is because *Engauge Digitize* was primarily designed to extract data from graphs.

The use of this program seems to have been limited to magnetograms from dates that are well known such as the Carrington event instead of a solution to digitize the whole data set[4].

2.2.2 Data Extraction from Graphs

As with the magnetograms, there exists the need to extract time-series data from many different types of mechanically created images observed prior to the improvements that came with computers. An example is meteorological data which includes environmental attributes such as rainfall, humidity, and temperature. The Italian Meteorological Service processes such data which is stored on pluviograms (rainfall), thermograms (temperature), and udograms (humidity)[21].

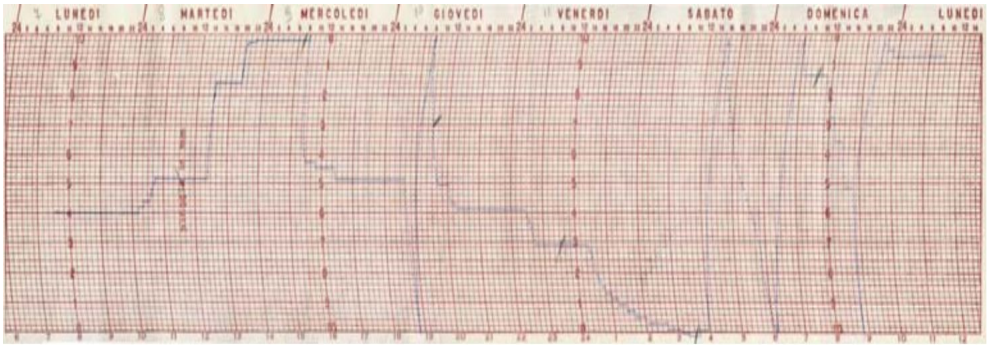

FIGURE 2.3: Orginal pluviogram from the Italian Meteorological Service, and displayed in "The automatic digitization of time series recorded on graph paper supports"[21].

To digitize these types of images, the software *DigiGraph* was created[21]. Once the image is loaded into this program the following steps are taken.

- Check Requisites - no multiple entangling traces (overlapping itself), no long gaps in the trace, and good visibility of the trace
- Detect the curve grid-lines - using an algorithm based on a 2-D Fourier transform

- Verify - was the identification of the curve grid-lines successful?
- Rectify the image - transformation from cylindrical to Cartesian coordinates, see Figure 2.4
- Optimize - remove noise, and optimize the contrast between the tracks and the background (force dark parts to black, and light to white)
- Data Extraction - an algorithm that produces a vector containing only one value for each point on the corresponding axis
- Verify - digitized data is superimposed onto the original image

The program is designed for a single trace that is of 'good' quality, and must go through a number of visual verification stages. The first stage is to determine whether or not the image will be successful. The user ensures that the image has no large gaps, the data is clear, and minimal noise. If these conditions are not met, the image is simply discarded. The image then needs to have the curvature removed, and this is done by identifying the grid lines by means of an algorithm based on a 2-D Fourier transform. This algorithm's output is then inspected.

If the transformation was successful, the image is then 'optimized' by means of removing any remaining noise, and forcing any dark points on the image to black along with any light to white. What is left are points that when placed into a vector are the digitized data points from the image, see Figure 2.5.

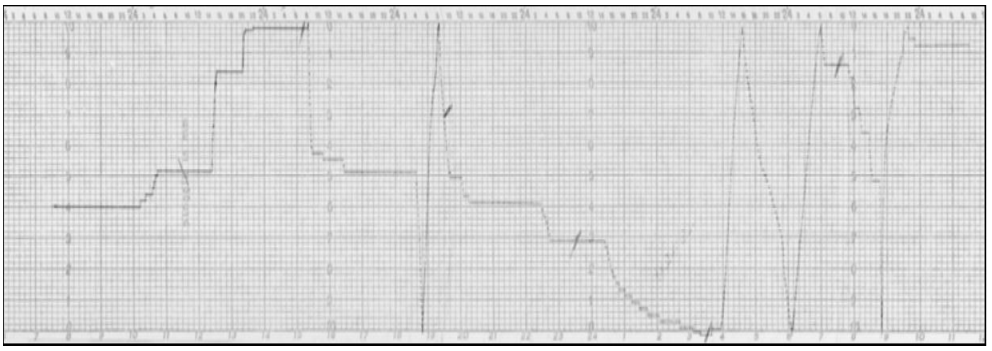

FIGURE 2.4: Converted pluviogram from cylindrical to Cartesian coordinates[21].

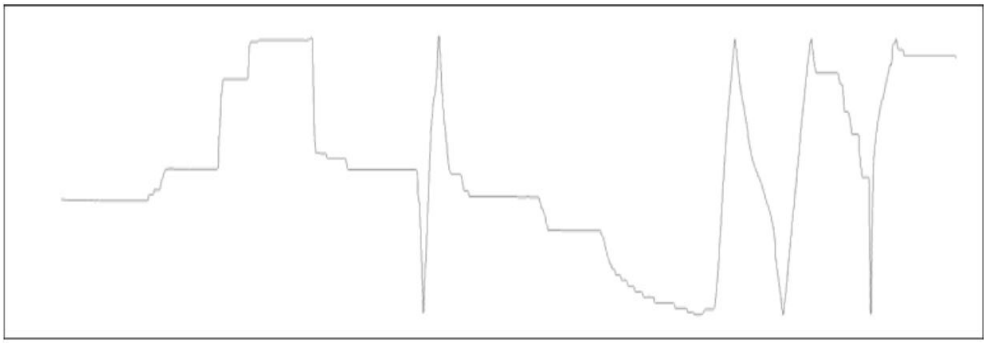

FIGURE 2.5: Extracted pluviogram Data[21].

On a more basic level, a simple approach might be preferred. One solution that was briefly mentioned earlier was *EnGauge Digitize*[16]. This software requires the user to manually select the axis points to define the work space. The user then selects a tool that creates a vector that is then clicked and dragged from one point to the next. This is continued until the user reaches that last point on the curve. This process can be tedious, as it requires the user to perform many precise movements for each image. For a magnetogram, this process could take a long time to complete as there exists constant fluctuation in the traces, meaning that

there will not be many opportunities to create a straight path that covers a large distance (with respect to the image).

A similar, but more sophisticated approach is that of *WebPlotDigitizer*[22]. Instead of using a vector tool, the user has two options: Manual Extraction, and Automatic Extraction. The manual is a simple point and click, while the automatic is much more interesting. They can use a box tool that highlights a large region, and then remove the non-trace areas. There is also a pen tool that allows quick tracing of the lines, see Figure 2.6.

Unfortunately, all these programs are "supervised", meaning the user has to provide inputs at every step. They excel at single image digitization, but are not designed for mass producing digitized data, as the labour requirements are much too high.

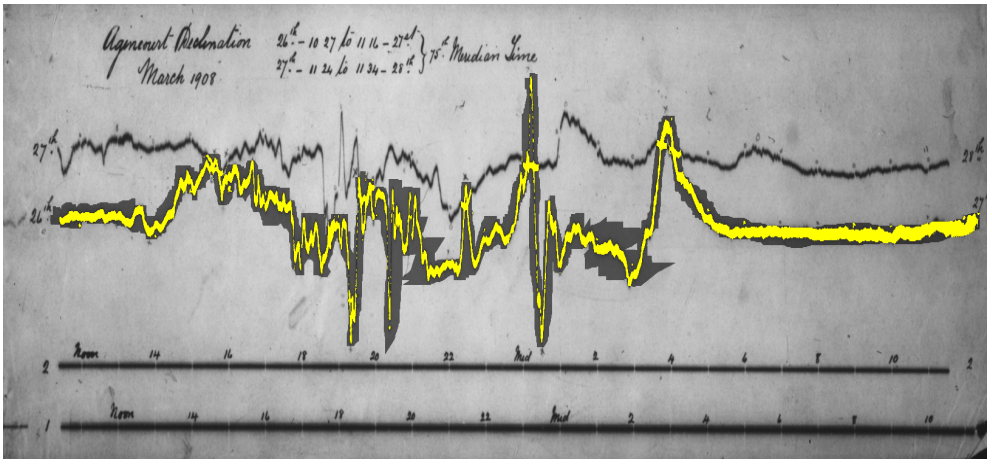

FIGURE 2.6: Example of a complex image being digitized by WebPlotDigitizer. The process begins with the manual selection of the axis points, followed by highlighting the desired region. The points of intersection can be easily seen as the highlighted region was too wide, and selected small amounts of the adjacent trace. Those regions would have to be highlighted with a reduced width.

2.2.3 Digitizing Seismograms

A similar type of challenge to our problem of magnetograms is the digitization of analog seismograms, which are records of seismic activity. Far more work has been done addressing this challenge. Fortunately, seismograms and magnetograms have many similarities. This can be seen when comparing Figure 1.1 and Figure 2.7. The particular seismogram in Figure 2.7 has been magnified so that the reader can see the similar structure to that of the magnetograms displayed. In reality the seismogram is very large, and contains twenty-four traces of which only seven are in the displayed image.

The digitization process for this seismogram, done by Church[6], started with preprocessing the image via vectorization initially through the software program RaveGrid. Their results can be seen in Figure 2.8. The author then used MATLAB for all further work. The first step was to determine the number of traces in a given image, along with their start and end points. Though these seismograms have many similarities to the magnetograms from the Toronto and Agincourt geomagnetic observatories, there are many differences. One of the main differences, is the timing marks. In the seismograms these are little dashes above the timing marks that appear to be created from the seismograph mechanism, whereas in the magnetograms they are hand written numbers.

Both are considered noise in the digitization process, but they have very different characteristics. The timing dashes are related to the length of the timing gap, and have an element of consistency. The hand written numbers do not. In fact, there

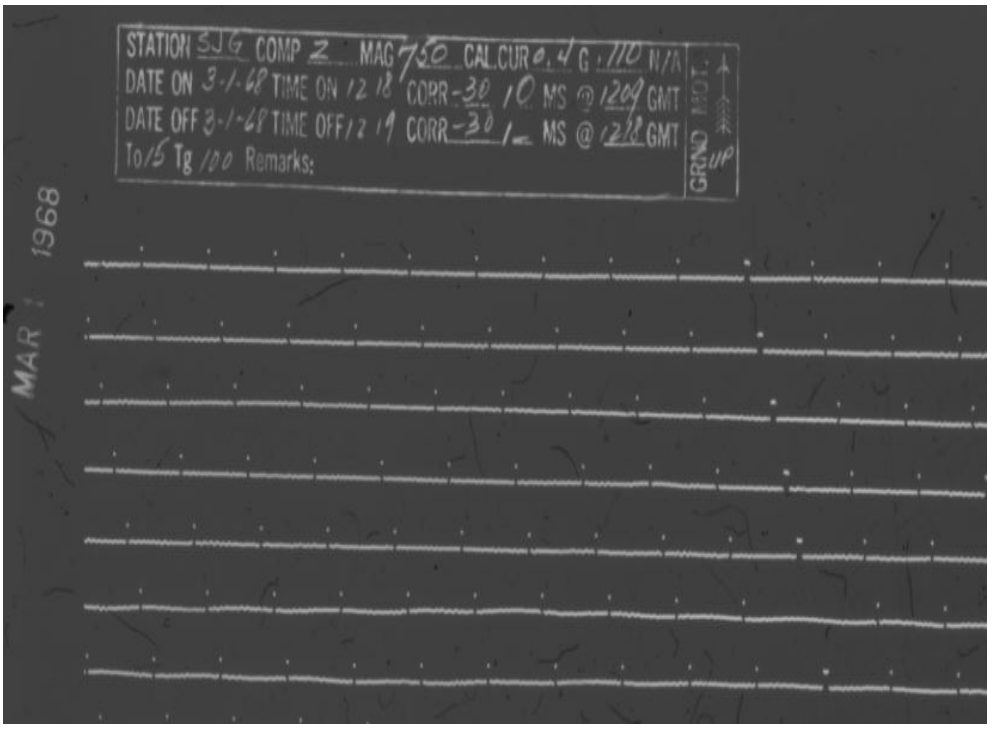

FIGURE 2.7: Seismogram from a seismographic station in Puerto Rico 1968. The method for recording the data was similar to that of the magnetograms with timing gaps occurring at a fixed interval.

are situations where the hand writing intersects the traces, and covers up timing marks.

Lastly, the seismograms are given a start and end time, where as the scanned magnetograms often just provide the dates, with no easy way to determine specific start and end times in hours. This often means that a magnetogram with two traces could actually contain data from three different days. This last difference will be a main focus in future work.

The current work done to digitize the seismograms is impressive, however most approaches share a similar characteristic: they require manual supervision. This is seen in the work done by Church[6], and even more so in the open source

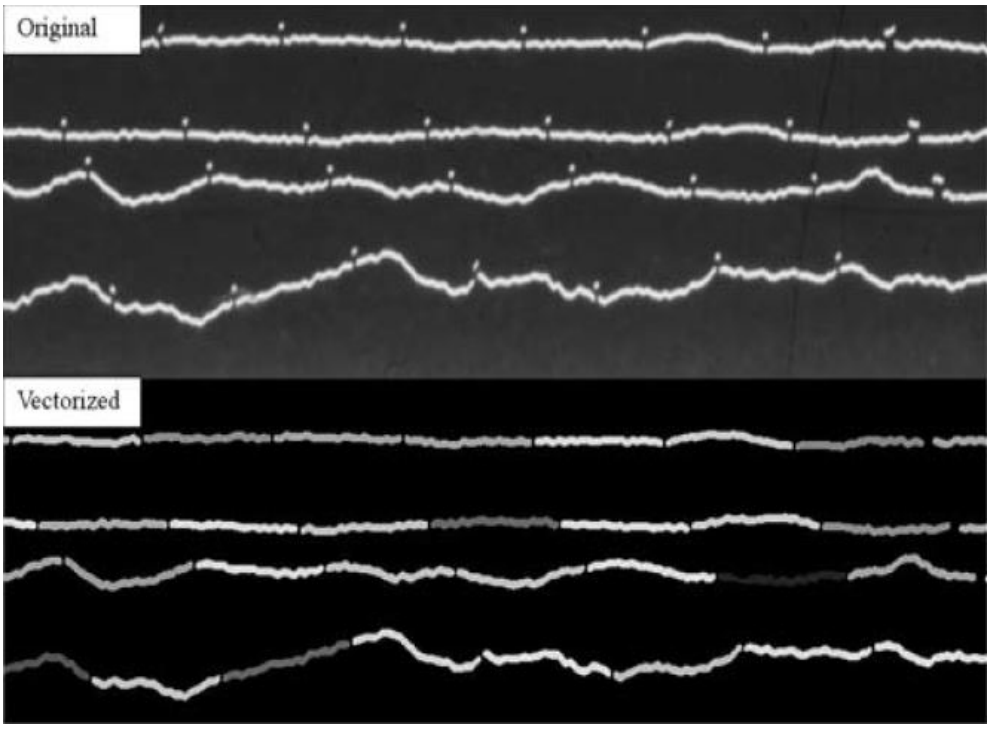

FIGURE 2.8: Seismogram from Raster-to-Vector Image Analysis by Church[6]. The vectorized version has the timing gap markers removed, as well as the background noise.

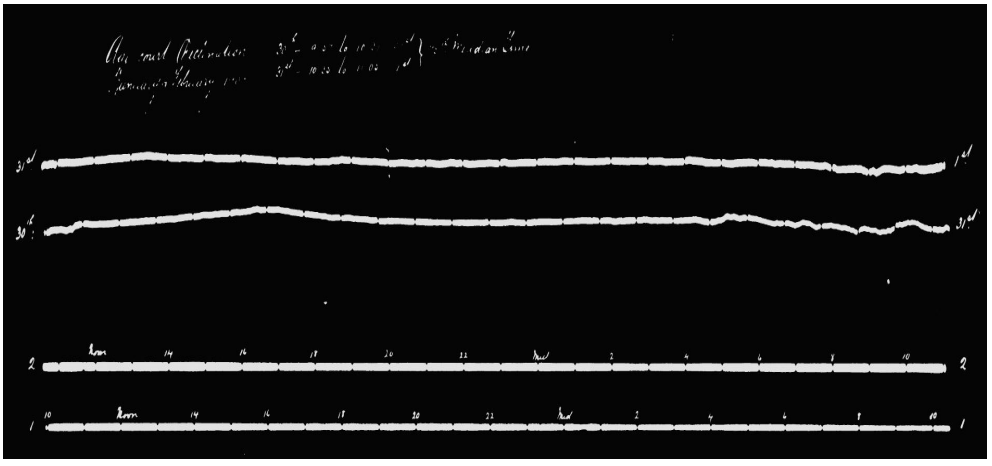

FIGURE 2.9: A negative Magnetogram used for easy comparison with the Raster-to-Vector seismogram in Figure 2.8.

software tool SKATE[1], the "Software Kit for Automatic Trace Extraction", this was also developed for extracting data from seismogram images but with more user input than in the Raster-to-Vector Image Analysis method.

This approach again takes a single image in at a time, with the user modifying selected aspects to improve results. The tools typically provide the user with editor tools that can remove spurious features, change the positions of lines which the seismic traces oscillate, and then selects the regions of interest. The user can then run tracing algorithms with these inputs to produce accurate data extraction. This comes at a cost of both computational, and operator time.

2.3 Work done by Natural Resources Canada

The magnetograms used in this project were created by researchers starting in the 1840's at the Toronto, and Agincourt geomagnetic observatories. These would have been located just South-West of the Ottawa (OTT) observatory as seen in Figure 2.10. The original magnetograms are analog images on photographic paper, recorded using on optical system which was "exposed" on the paper. Then, in the 1980s, the magnetograms were converted to 35mm film, in order to preserve the images, as the paper had begun to degrade.

The creation process for these magnetograms was not simple, and involved constant human interaction. As time moved on, the locations of the geomagnetic observatories changed, and so did the number of days recorded on a single sheet of photographic paper. These changes can be clearly seen in the images, e.g., Figures 2.11 and 2.12. In the early days, there was not much consistency. The operators experimented with different numbers of traces for a single image, similar to that of

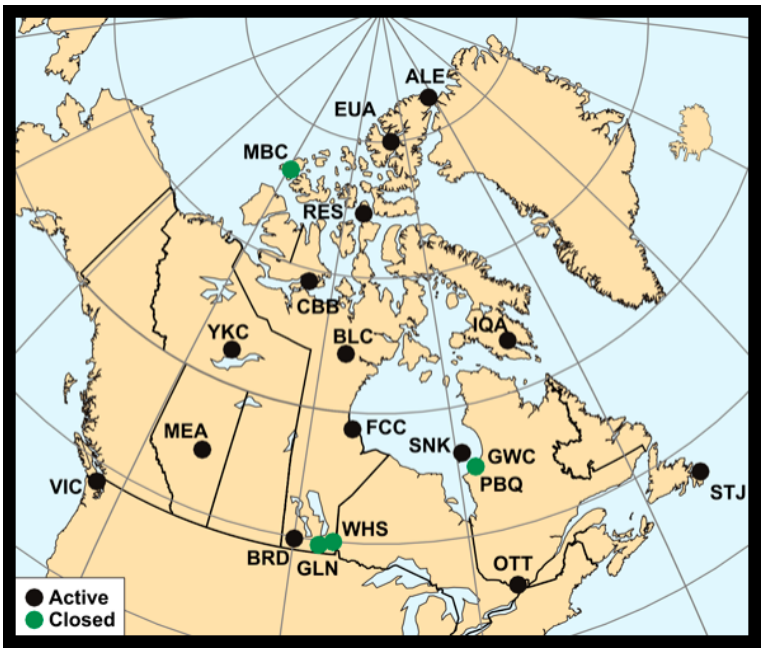

FIGURE 2.10: Current Canadian Geomagnetic Observatory locations as provided by Natural Resources Canada

the seismograms. They eventually identified the many problems that were created from jamming many days into a single image, leading to more standard formats later in time.

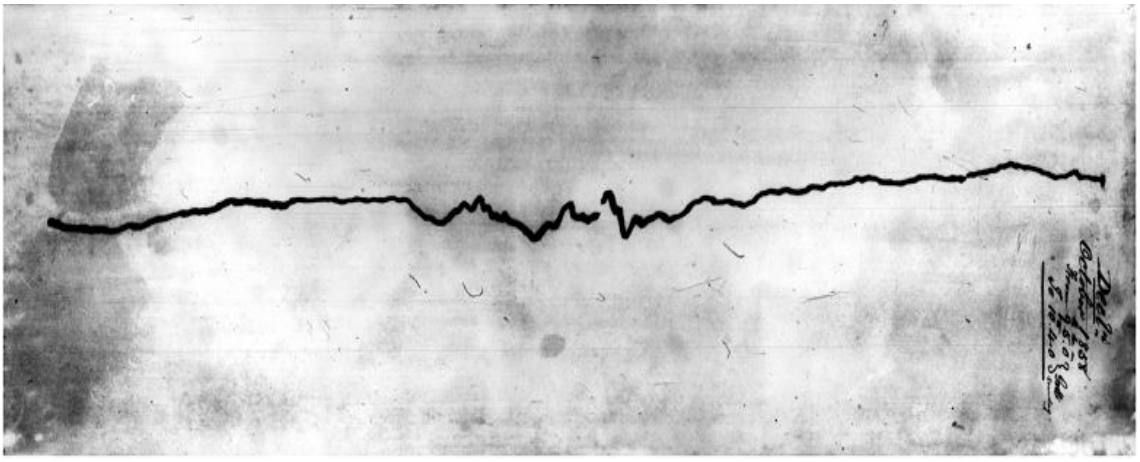

FIGURE 2.11: Magnetogram from the Toronto Observatory displaying a single trace with no baselines (1858). Image ID:TOR-D-18581009-18581010

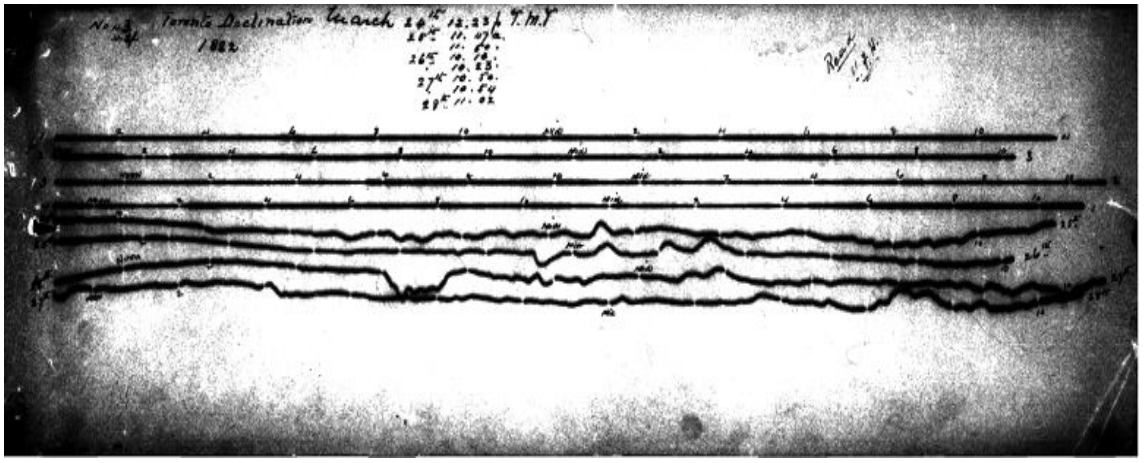

FIGURE 2.12: Magnetogram from Agincourt Observatory displaying four traces with baselines above (1882). Image ID: AGC-D-18820324-18820328

2.4 Work done by Queen's University

In 2012, a short-term project was launched by a research group at Queen's University in Kingston Ontario, led by Dr. David J. Thomson. They took the 35mm film records, and scanned a large number of the images to disk as TIFF (Tagged Image File Format) files, chosen as a lossless image format to ensure scan fidelity[23]. Now that they could be accessed on a computer, a digitization algorithm was attempted for extraction of the geomagnetic data. This algorithm followed a four stage process illustrated in Figure 2.13:

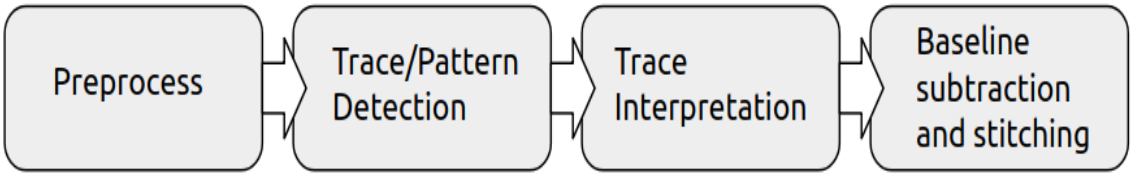

FIGURE 2.13: Original Digitization Model, developed by Queen's team

1. Preprocessing: Using deconvolution

2. Tracing/Pattern Detection: `getStarts` and `getTraces` functions

3. Trace Interpretation: `cleanTraces` function

4. Baseline Subtraction and Stitching

The preliminary work proved successful as the team was able to digitize 90 days worth of data for the year 1896 (approximately 45 images)[23]. However when attempts were later made in 2017 to recreate past results there was little success. The digitization algorithm had been claimed to work on a select few images, but additional labour had to be done in order to see those results. In addition, the digitization process was highly supervised, requiring operator interaction, and parameter selection. This was not practical for digitizing the majority of image set, which consists of over 30,000 images.

2.4.1 The Data

When the magnetograms were initially scanned they were placed into images groups. Each image group contains approximately three years worth of magnetograms for their corresponding magnetic field component. As mentioned previously the components are broken down into three groups: D (declination), V (vertical), and H (horizontal). In Table 2.2 we can see how many images are in each of the image groups for the D field component.

TABLE 2.2: Available Image Groups from 1898-1938 for the D category

Image Group by Date	Number of Images	Unusable/Errors	Missing
1898/01/11-1901/05/29	575	8	∼49
1901/05/30-1904/08/29	595	22	6
1904/08/30-1907/11/24	596	18	2
1907/11/25-1911/02/01	590	NA	NA
1911/02/02-1914/05/11	598	NA	NA
1914/05/12-1917/07/22	584	18	2
1917/07/23-1920/12/31	638	NA	NA
1921/01/01-1924/03/05	589	NA	NA
1924/03/06-1927/04/26	581	36	NA
1927/04/27-1930/06/19	585	58	NA
1930/06/19-1933/06/12	546	22	2
1933/06/01-1936/06/30	559	92	5
1936/07/01-1938/12/31	177	47	NA
Total	7213	321	66

TABLE 2.3: Distribution of Image Types for selected Image Groups. Error images are included as long as the type can be determined.

Image Group by Date	Type 1	Type 2.1	Type 2.2	Other
1898/01/11-1901/05/29	10	530	34	1
1907/11/25-1911/02/01	21	515	54	0
1927/04/27-1930/06/19	25	383	175	2

The images can be further separated by *types* or classes. These are determined by the number of traces that are present in the images. A single trace (one day worth of data) is classified as *type 1*, two traces (two days worth of data) is a *type 2*, and so on. As a result of the possible overlap of traces, there exists sub classes; these are intersection and parallel which are found in *types 2+*.

These images groups, along with their corresponding V and H component

groups are dominated by *type 2* images, with a small number of *type 1*s. In addition, these image groups contain the most 'consistent' images, meaning that the quality of the images share similar attributes such as contrast, and image classes.

Often found within each image group is a *readme* file. These contain image observations from the team that scanned the 35mm film to disk. In the first listed image group found in Table 2.2 (1898-1901), one of the *readme* file entries stated that "the device was out of operation for these three months". From this we know there exists a large gap in the data. This will be important to know when combining the images later on to create a time series. They also contain information about the most noticeable errors, and unusable images. Not all errors are noted, but this does give a quick glance at each image set. The 'Unusable/Errors' column of Table 2.2 contains their observations only, and not the true number.

The quality of the images also varies from year to year. There are image groups such as 1907-1911D that do not have a *readme* file, and upon inspection have 30 unusable/error images. Group 1933-1936D has 92 problem images identified prior to our inspection, accounting for 16% of the total images for that image group, and the real number is likely much higher. This is based on the visual inspection of image group 1907-1911D which yielded 30 errors, when the expected number was zero as no other information was supplied.

Solutions for a number of potential errors have been addressed, and can be found in Chapter 3. For those images that cannot be successfully digitized at this

time due to their appearance will have to be re-scanned if the original photographic magnetograms still exist. The reason for scanning from the photographic papers as it appears that many of the errors may have come from scanning the 35mm film. This argument is based off of the scanned magnetograms from the British Geological Survey (BGS) when compared to the magnetograms from Toronto and Agincourt. There are errors that are observed in the Canadian image set that are not seen at all from the BGS, such as Figures 2.14 and 3.1.

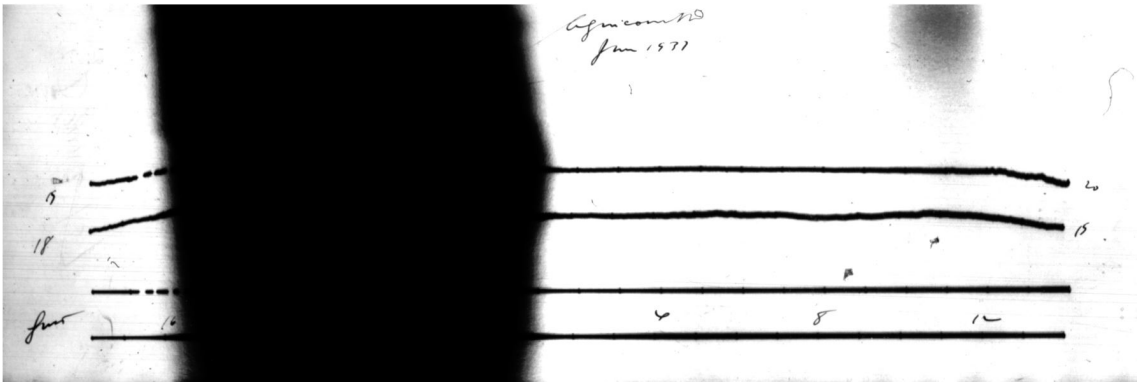

FIGURE 2.14: Error that occurred in preservation process. A finger print can also be seen in the top right which is only possible on the 35mm film version of the magnetogram.

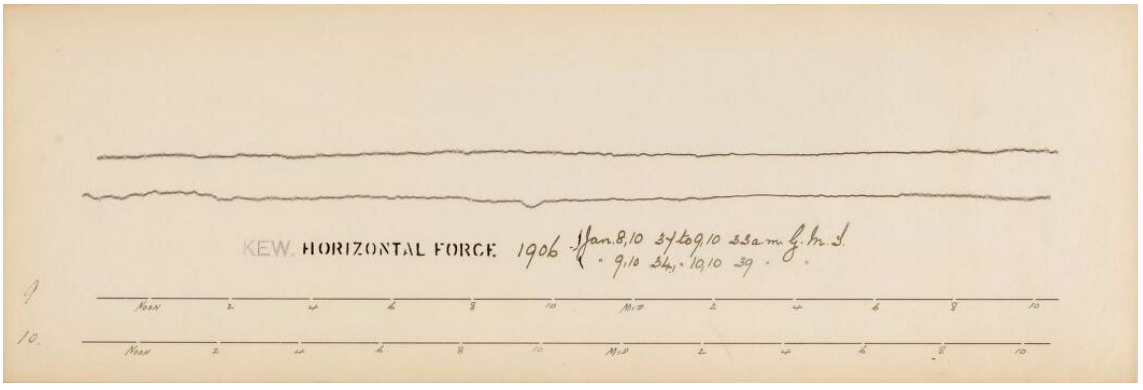

FIGURE 2.15: A 1906 British Geological Survey magnetogram from the Kew Observatory[5]

2.4.2 Preprocessing

In order to select the best preprocessing technique for the magnetograms, one needs to know how they were created. The device that records the local geomagnetic activity is called a magnetometer, and in its original, pre-electric form, projected a light source onto photographic paper while moving. A sketch of one of the early magnetometers can be seen in Figure 2.16. In the most basic case this can be considered a point light source, and after passing through the optical apparatus would be recorded as an Airy Disk[23], see Figure 2.17. This idea can be verified by examining the images.

The result on the photographic paper can be modeled as a point light source convolved with an Airy Disk then moved in time (x). If the effects of the optical system could be removed, all that would remain would be the point light source path. If $f(\cdot)$ is the path of the point light source, and $h(\cdot)$ is the point-spread (which represents the effects of the Airy Disk) where x is the row number of a given pixel, then the convolution of these two would result in $g(x)$ for a given column. As a mathematical expression, with $*$ representing convolution:

$$g(x) = (f * h)(x) = \int_1^n f(y)h(x-y)\,dy \tag{2.1}$$

The desire is then to obtain $f(\cdot)$, and deconvolution of $g(x)$ would do so. Before anything else can be done, we have to define $h(\cdot)$. It was suggested that a substitution of a Gaussian function for the Airy Disk would make an excellent

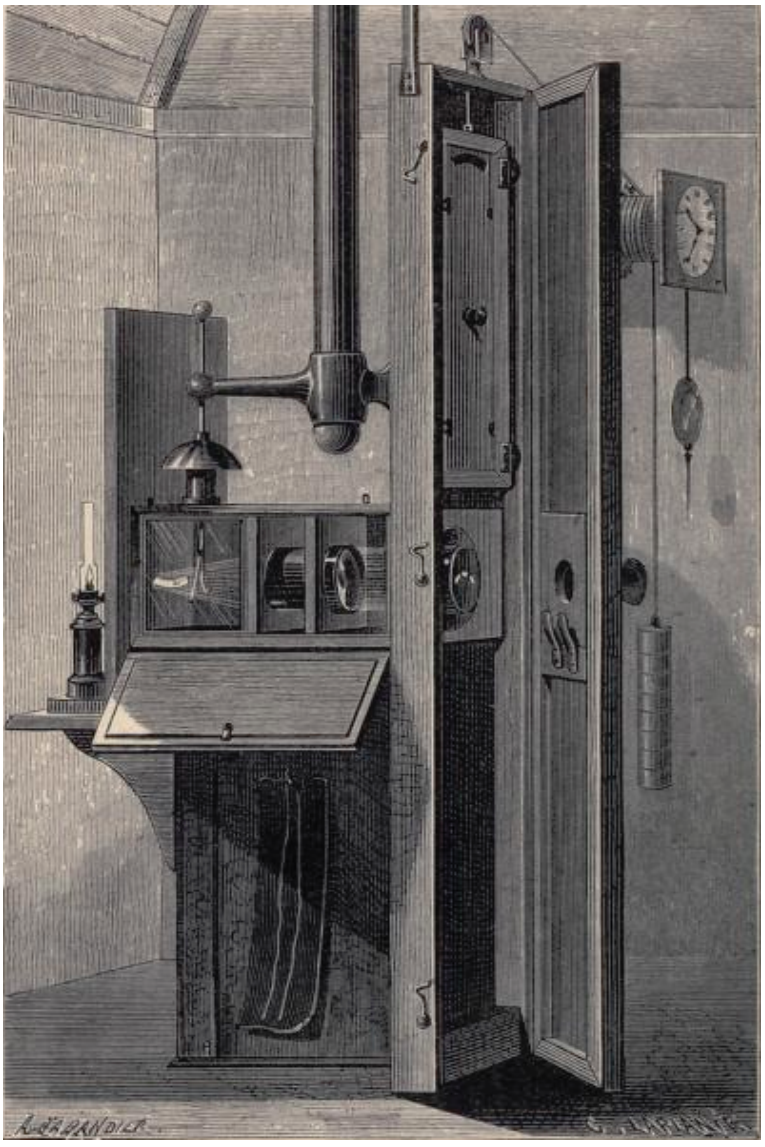

FIGURE 2.16: A sketch of one of the first magnetometers that was located at the Kew Observatory (UK) in the 1840's[9].

approximation in practice[23]. There are certain conditions where the convolution operation can be written as multiplication in the Fourier domain[23].

The procedure is then as follows:

- Compute the Fast Fourier Transform (FFT) of the pixel greyscale values, and the point spread function.

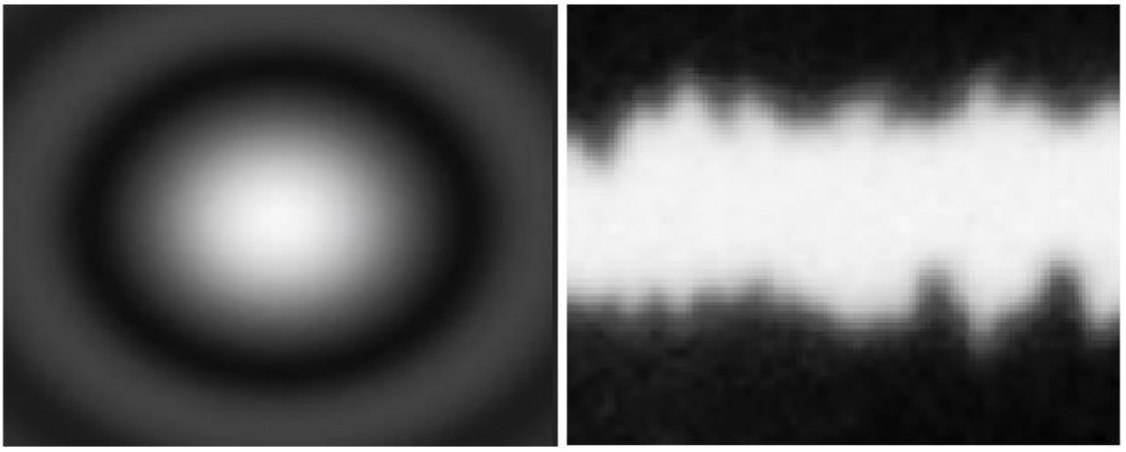

FIGURE 2.17: Example of an Airy Disk[14], with a close up negative version of a magnetogram trace. If a single column was selected from the magentogram, it would have a similar appearance to that of the center column from the Airy Disk on the left.

- Compute the ratio between the two
- Compute the inverse FFT

This is done for all columns in the selected image.

The result of the deconvolution is the removal of the effects created by the optical apparatus. Thus the path of the point light source can now be extracted with ease. The effects of the optical device used in the creation of the seismograms are still slightly visible after being vectorized, whereas in the deconvolved magnetograms it appears to have been completely removed upon close visual inspection. Note that since we do not have access to their data at that stage of the digitization, it is difficult to verify this claim.

2.4.3 Tracing/pattern Detection

After large amounts of noise are removed in the preprocessing stage, the tracing of the magnetic field strength can be done. The algorithms originally created for this task were called $getStarts$, and $getTraces$. The function $getStarts$ was the first step in the trace detection. It would attempt to identify columns that would make good starting points for the $getTraces$ algorithm. This was done by using the $findpeaks$ function from the *pracma* package[2].

Once the starting columns have been selected, the paths are then traced to the left, and then to the right of each starting column for all columns in the image. This is then repeated for each trace. The path is chosen by finding the local maximum, which is determined by computing the differences (2.2).

$$\begin{aligned} d_1 &= z_{j,i+1} - z_{j-1,i+1} \\ d_2 &= z_{j+1,i+1} - z_{j,i+1} \end{aligned} \tag{2.2}$$

Once d_1, and d_2 have been computed, they are then used as indicators to find the direction of the local maximum. There are four potential outcomes.

- $d_1 > 0$ and $d_2 > 0$, the function is increasing as we move down the column.
- $d_1 < 0$ and $d_2 < 0$, the function is increasing as we move up the column.
- $d_1 > 0$ and $d_2 < 0$, the function is decreasing in both directions.
- $d_1 < 0$ and $d_2 > 0$, the function is increasing in both directions.

If all went well, then there will be at least two traces to choose from between each starting point. We then must attempt to select the best trace between each starting column.

2.4.4 Trace Interpretation

The tracing algorithm works very well for the little-to-no activity images, but no system was implemented for selecting the "best" trace save for the highest grey-scale value. This method can work, but in more complex situations such as when the traces intersect, and/or bounce off one another, it fails. In addition the

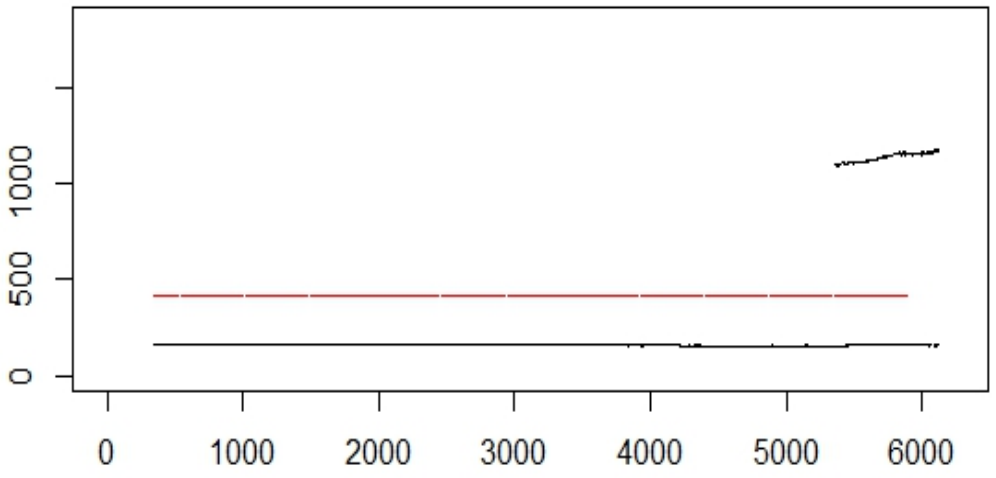

FIGURE 2.18: Cleaning function failure. The two baselines at the bottom remain but the traces have been removed, leaving only the far right of the first trace intact.

orginal *cleanTraces* function causes all sorts of problems, such as removing parts

of traces, and in some situations, entire traces. This can be seen in Figure 2.18, where the top trace has been completely removed, and the bottom only has the end remaining.

2.4.5 Baseline Subtraction and Stitching

Once the traces have been extracted, we still need some information from the baselines before we can remove them. In addition to the scanned magnetograms, there exists scanned logbook files. Within these files is recorded information related to the magnetic field strength, as the magnetograms on their own do not contain this information. Only the distance from the corresponding baseline to each data point can be obtained. This then means that before the digitized magnetograms can be used, this information will need to be pulled from these logbooks.

2.5 Code Optimization

Throughout this project, work has been completed in the *R* programming environment [20]. The proof-of-concept digitization algorithm was written in this language, and thus was the reason for its selection. Once the digitization algorithms were fully functional, the time required (around two minutes) to process a single image was not acceptable for the 30,000+ image data set. In the best case scenario, it would take over a month to digitize the magnetograms. To reduce the required

time, the code was first profiled to help identify "bottle-necks". These would then become the focus areas for the optimization. We discuss this work in Chapter 4.

2.6 Classification by Crowd-sourcing

One of the final steps in the project is to verify that the digitization process was successful for each image. We plan to accomplish this by using image classification via crowd-sourcing. Crowd-sourcing can be thought of as outsourcing some sort of work, but instead of hiring an external organization to do a particular task, it is open to anyone who is willing. This is nothing new, and has been used many times as can be seen on the "Zooniverse" website[31]. Many Kaggle machine learning competitions have used crowd-sourcing to help classify images so that competitors could use them to create prediction models, such as in "Planet: Understanding the Amazon from Space" [13]. Crowd-sourcing provides an alternative to tasks, such as the classification of images, that in many cases is faster, and cheaper than hiring a small team of experts.

However, crowd-sourcing does have its issues. In the Kaggle competition mentioned above, they acknowledged that there were incorrectly classified images in the data-set[13], since the classifications might have been done by those who do not fully understand the conditions for each classification, or simply did not take an appropriate amount of time. This is why it's important to have a way of verifying the selections of these helpers without doing it manually (which would

defeat the whole purpose). One option is to use a majority wins rule as used in the classification of the Sloan Digital Sky Survey (dubbed the *Galaxy Zoo* project[26]).

The objective of the *Galazy Zoo* project was to classify hundreds of thousands of galaxies. This was done through a website, where volunteers would be the classifiers[31]. Before proceeding to classify the images, they were asked to go through a tutorial which showed examples of galaxies with their classifications. Lastly they were tested, and those that passed were given access to the whole site[26]. They wanted as many volunteers as possible, so the bar was set relatively low.

The images were selected at random from the database, and after the user classified a given image the information was stored into a live Structured Query Language (SQL) data base, along with the timestamp, user ID, and galaxy identifier[26]. With all of this new data coming in, a method of selecting the best user classification for a given galaxy. As was mentioned before, they used a majority wins rule. However, it was clear that all classifiers should not be treated the same, as some would not take it seriously, or would not spend enough time analyzing the image. So in addition they used a weighted sampling method to identify the 'good' users[26]. The more a given classifier selects the majority classification, the higher their weight becomes. The mathematical construction used can be seen in equation 2.3.

$$w_k = \sum_i \frac{h_i(j \text{ chosen by user } (k) \text{ for galaxy } i)}{N_g(k)} \tag{2.3}$$

A problem that they identified was that the weighting favours the majority, and sometimes the majority can be wrong. So they then compared their results with another sample that contained some of the same galaxies. Their conclusion was that the general public can classify large data sets of galaxies with similar accuracy to that of professional astronomers[26].

2.7 Chapter Summary

This chapter introduced the history, and the background of the project. It includes similar digitization challenges, previous work done on the project by Natural Resources Canada, and a proof-of-concept digitization algorithm created by the research team at Queen's University. It is this algorithm that claimed that the magnetograms collected from the Toronto and Agincourt geomagnetic observatories could be digitized, and serves as one of the foundations for the remainder of the thesis. The reader was also introduced to classification through crowd-sourcing via the galaxy image classification project *GalaxyZoo*. This will be important later in the thesis.

Chapter 3

Methods

3.1 Changes

The first step was to take the original proof-of-concept script, passed on by the research group at Queen's, and make it operational. The preprocessing by deconvolution had no identifiable problems, and this remained the case throughout the project. The first problem was identified much further on, but was traced back to the beginning of the second stage, Tracing/pattern detection.

The issue was a threshold value which regulated the width of the trace path. This was done by finding a particular percentile based on the overall grey-scale values from the deconvolved image. Originally we set it to find the 95^{th} percentile by default, but this did not work for many images, especially the magnetograms that had more geomagnetic activity than normal. In this situation the path width

would be thinner as the light source would not have a significant amount of time on the given point on the photographic paper. But more commonly, the *getStarts* function would fail because it could not identify starting locations at that percentile.

A simple script was created to deal with this problem. It would start with the 95^{th} percentile, and if it failed, would lower the percentile by five, and run again. These values were determined through experimentation. There were cases where only a slight decrease (say 1) was required but still were successful with the decrease of five. This would then continue until failure which was determined by reaching zero. In those situations, there is a large problem with the image itself. An example image that causes this problem can be seen in Figure 3.1.

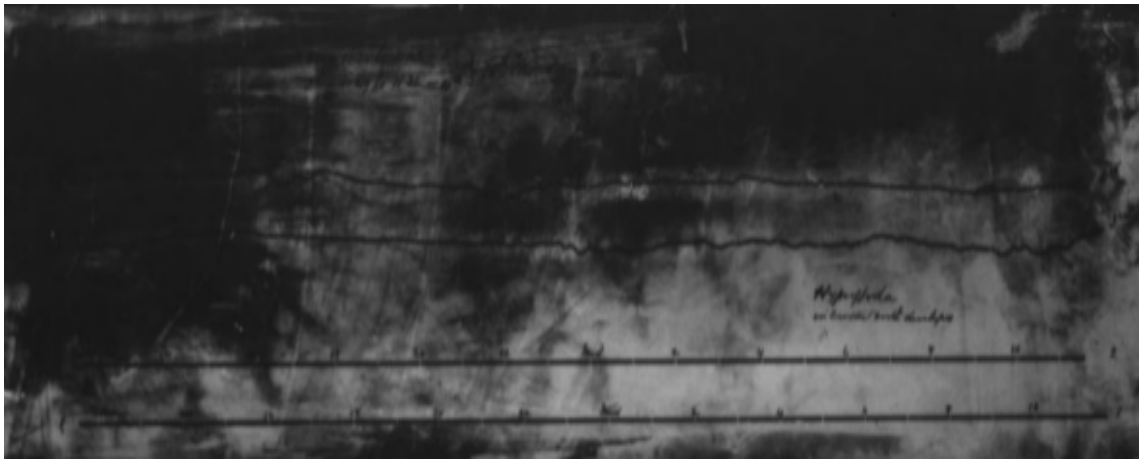

FIGURE 3.1: Human/mechanical Error: no percentile possible for digitization.

Additional preprocessing has increased the number of images able to be digitized. Most magnetograms have writing running along the top of the image,

and this can cause problems. In particular, this writing can cause the *getStarts* function to fail. Since this function identifies the starting locations for the tracing algorithm, the digitization process of that image is halted. This is not always the case, but has been encountered. An extremely simple, but efficient solution was to convert all the grey-scale values in a particular region, where the writing is often found, to the image median value. This takes place after the deconvolution, as it has a higher success rate in that position, and only if the *getStarts* function fails.

The original *getTraces* function attempted to create many traces in hopes of obtaining the correct data. It then weighed the traces based on their total greyscale value, as it would make sense that the trace with the higher value would provide the best chance or accurate data extraction. However, there are situations where the highest value trace is not the best fit. For example the traces in an image may intersect, or bounce off each other. This creates an opportunity for the tracing algorithm to follow the wrong path at the point of contact, see Figure 3.2. Luckily in the majority of those images, only one of the traces has above average activity. The magnetograms that have all of their traces with high amounts of activity are near impossible to untangle at this time, and manual intervention would be required.

The idea is to look through the other potential traces that did not have the highest grey-scale value but are a better fit, unlike the result in Figure 3.3. But first we must identify if, at some point, the given traces intersect, and overlap.

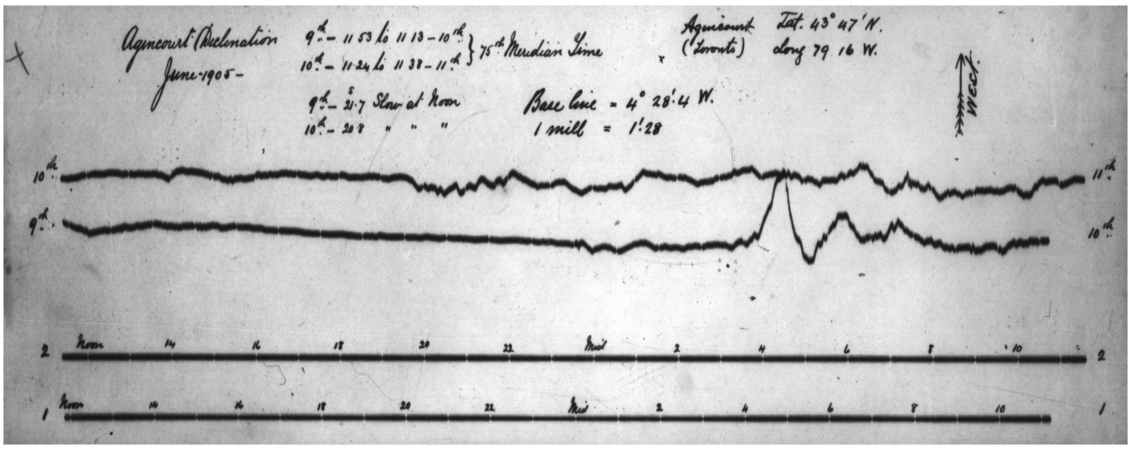

FIGURE 3.2: Magnetogram from the Agincourt Observatory (1905). Image ID: AGC-D-19050609-19050611. The first trace (bottom) has a spike in activity that intersects the second trace (top). The quick change in activity results in a thin line on the image making it difficult for the tracing algorithm to select the correct traces.

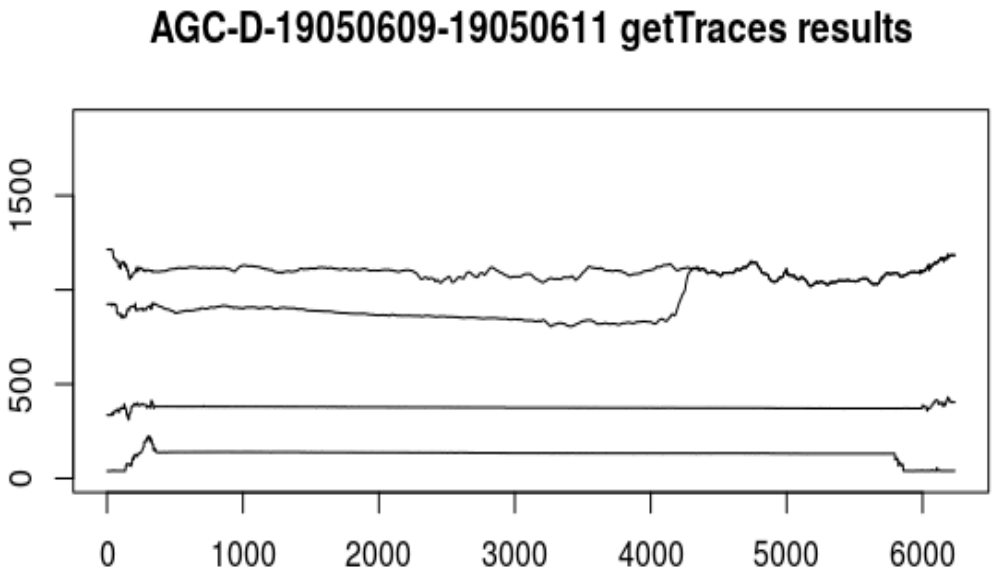

FIGURE 3.3: Orginal *getTraces* result for Image AGC-D-19050609-19050611 before modification. The first trace has merged with the second, meaning that they share the same data after the point of contact.

This can easily be done at the end of the process using a component-wise equality comparison for the highest grey-scale value traces, and storing the resulting

Boolean values in a new vector. If the sum of this new vector is over a given threshold (set by default to 8% of the image length or approximately 500 pixels), then there exists an overlap.

When a overlapping trace is identified, and an alternative trace is then selected. This process is then repeated until the number of overlapping values does not exceed the given threshold. The threshold value was determined by selecting images that created the problem described above, and then finding the breaking point. Then to be safe, the threshold value was raised to allow for images such as Figure 5.1 to pass.

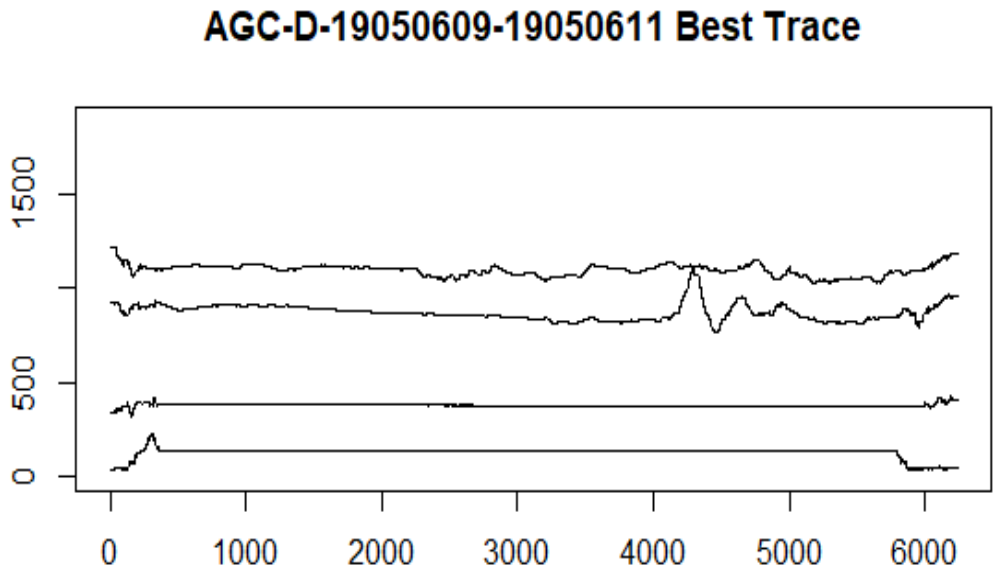

FIGURE 3.4: A modified *getTraces* best trace result after modification. The last section of the first trace is no longer confused with the second trace.

Unfortunately, there still exists more complex images, similar in nature, that do not create a "good fit" trace, such as images with large amounts of activity with very thin lines. This additional work would then not help. However, if more potential traces were created in the *getTraces* function this may increase the chance of success. Since this function takes the longest time to run, it makes little sense to increase the number of potential traces for all images. Thus in future work, if the threshold value is never broken, the code would loop back, and increase the number of traces in hopes of obtaining a better fit. For now, this addition has enabled many more images to be digitized that could not be done with the original proof-of-concept such as Figures 3.2 and 5.1. Its success also helped create a method for dealing with intersecting traces.

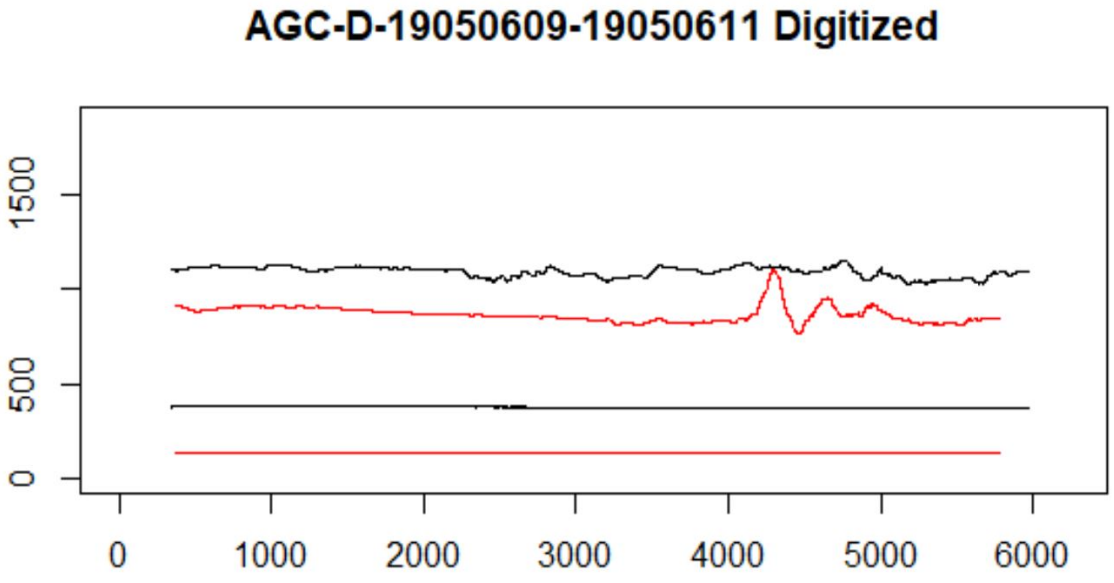

FIGURE 3.5: Final Digitized Magnetogram: AGC-D-19050609-19050611.

After the tracing is completed, there often exists handwriting residues found

on both sides of the traces, and baselines. It is very important that these residues are removed as when it comes time to stitch the traces together, any noise that remains will create poor results. The final version of the digitization process for the same case as before can be seen in Figure 3.5.

The main issue with the original cleaning function, $cleanTraces$, is that it can be over-zealous at times. There have been situations where entire traces would be classified as noise, and then removed. This was very concerning, as these particular images appeared to have no issues in previous stages. Then there was the complete reverse problem. Parts of the traces that should have been removed were kept. This occurred in both the traces themselves, and in the baselines. So an alternative cleaning algorithm was created to solve this problem.

This was accomplished by starting at the center of each trace, and moving in both directions until certain conditions were met. The algorithm looks for gaps (NA values), and then keeps a tally of how many numeric values follow. The hand writing on the images will often contain a mix of NA, and numeric values at this stage. This makes the hand writing easier to detect, and separate from the actual traces. Then as a last baseline cleaning step, we calculate the 95^{th} and the 5^{th} percentile of the baseline values, and removed any data above or below these values, respectfully.

The final stage of the digitization process is taking the traces in the image, and creating a single continuous time-series. The real difficulty is finding the correct amount of time that is not accounted for in the magnetogram due to the

operator reaction time. Some magnetograms have the time in hour/minute for the start and end of each trace hand written on the photographic paper. This can be used to fill in this knowledge gap. This step must unfortunately be saved for future work.

3.2 Dealing with Errors

Once the code had been made operational, and supervised attributes were converted to unsupervised, the need for the code to work its way through a given image set without crashing became paramount. Before changes were made, there did not exist any form of error control. A given image would either make its way through the process with no problems at all, or fail. Since many of the images are different in one way or another, the chance for error quickly mounted.

The images that caused errors in the code needed to be identified so that the cause could be made known. This was accomplished was by using commonly used error control functions on key parts of the code. These functions will try to run the algorithms that were placed inside them, and if they fail, they will not stop the script. This way if one of the main functions fails, we can then run that image again under different conditions instead of writing it off.

This then allowed the programmer to address the source of the error which would then allow the image to be digitized correctly. Every time an image was pushed through after changes were made to accommodate it, this allowed many

more images to be digitized as they shared the same problem. This was done without changing the results of other images that were already successful. However there still exists a selection of images that cause errors, and can not be digitized at this time without manual intervention.

3.2.1 Error in the Scanning Process

Some of the scanned images have areas that capture the background of the scanner, Figure 3.6 for an example. This often appears as a solid black box running along any side of an image. More often than not, they do not cause any problems as they only take up a very small percentage of the image, but there are situations where they cause the tracing function to behave poorly. A small algorithm was created to identify, and remove these sections on the images. This is a much safer approach then placing restrictions on the *getStarts* function, as there exists images where the traces get extremely close to the end of the image.

When an image is selected to be digitized, row and column sums are computed, and placed into vectors. If the max column/row sum is found at the ends of these vectors, then this particular scanning error has occurred. The region is then identified, and removed.

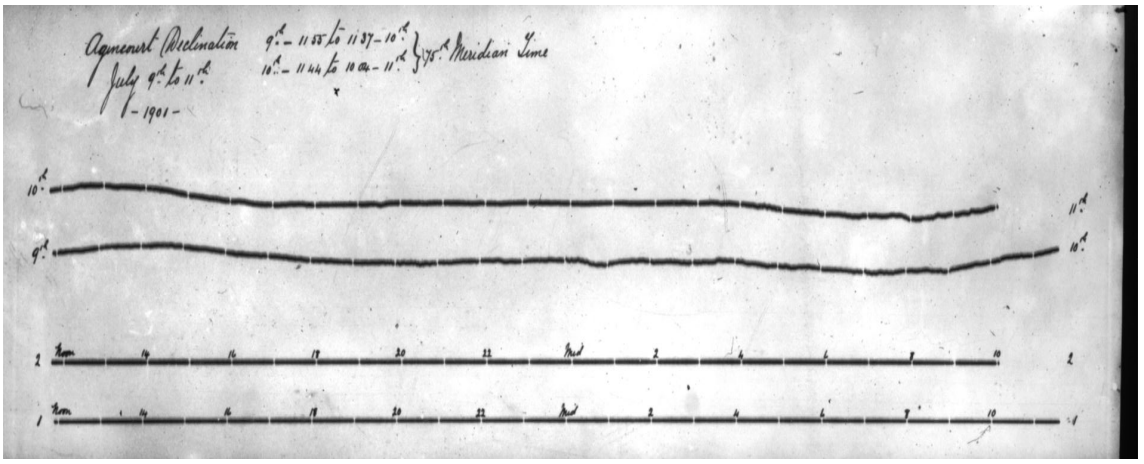

FIGURE 3.6: An example of a minor scanning error. The solid black region on the right hand side can cause tracing errors. The tracing function will attempt to start a trace from this region which can cause it to fail.

3.3 Trace Identification by Separation

In conjunction with the original code, additional algorithms were created as replacement or to work with the pre-existing algorithms. As the project moved forward, many chunks from the custom algorithms found their way into the two main digitization algorithms: *Modified Original*, and *Trace Identification by Separation*.

The driving force behind creating this algorithm was in response to the limitations of the original digitization code. A problem in extracting data of this nature is having issues with the crossing over of the traces. A theoretical solution to this problem was to use the timing marks found on both the baselines and the traces to correctly identify each trace[23]. The idea is to use the timing marks found on the baselines, and use those to match with the traces at the intersection points. However even if the trace identification by timing marks was successful,

there exists images that do not have timing marks, or have a slightly different version such as those in Figure 3.7.

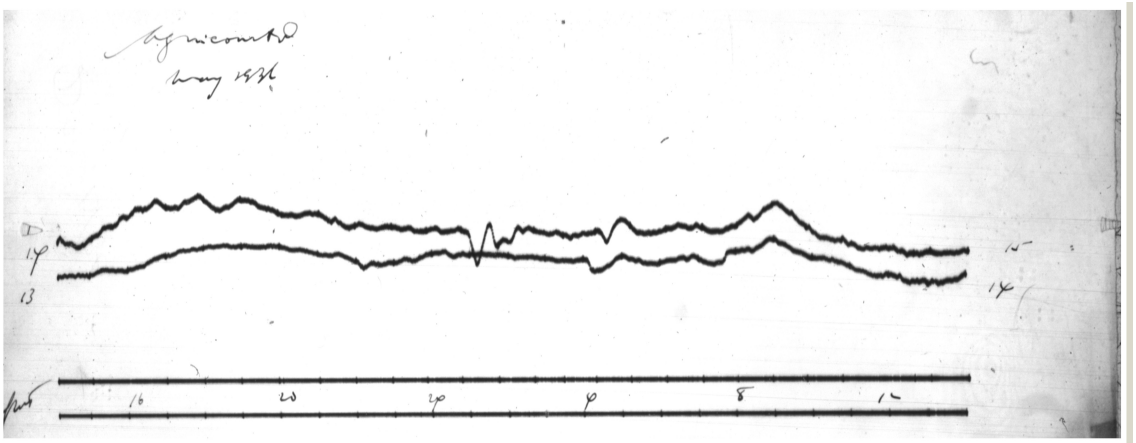

FIGURE 3.7: A magnetogram with a different style of timing marks. In place of a gap is a quick spike on the traces and baselines.

Our innovation was to encase each trace between two lines so that the given trace area can be extracted, see Figure 3.8. Then the tracing algorithm only has one trace to work with at a time. This has the potential to work extremely well when only one of the days shows high amounts of activity, which is the majority of images of the intersecting class. This algorithm shares a few ideas from the original algorithms that proved successful, such as using deconvolution for preprocessing. It can take in an image, and produce a similar output to that of the modified original code. In order for this algorithm to have any chance of success with the more complex images, it must first be able to digitize the more mundane.

The process to achieve the required results are as follows:

1. Preprocess: Decon, and noise/background removal

2. Draw Lines: Find starting points, and encase traces by creating separation lines

3. Clean: removal of any non-trace data

4. Output: Bring the data together

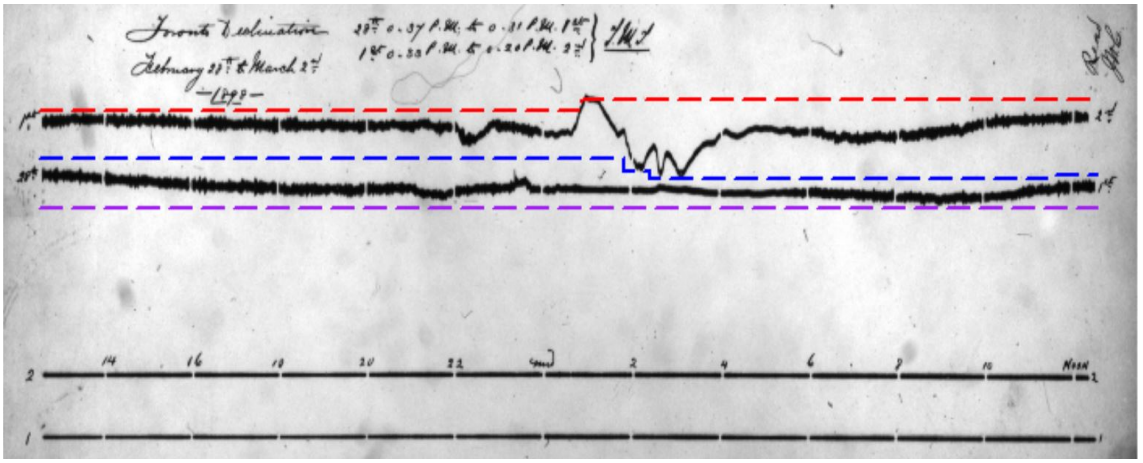

FIGURE 3.8: By isolating each trace to a given region, it becomes much easier to extract the data for each trace as most of the noise from the image is ignored.

The first step is to apply deconvolution, and set the grey-scale values that are below a given quartile to zero. This helps with peak detection, and will reduce the difficulty of extracting the traces. Once the peaks have been identified using the $findpeaks$ function, we can start creating the lines that will encase the traces. Once this has been completed, we can then find the median y value of each column within each enclosure.

The lines begin at the selected starting point, and proceed to move to the next column in the same row. If a high grey-scale (greater than or equal to the given quartile) is found in the next column, then a potential trace has been found.

The algorithm then checks above, and below the current row in the column that contained a high grey-scale value. If one is zero, and depending on what line is begin created (Upper/Middle/Lower) then that will be the next point in the separation line. It then continues until reaching the end of the image. See Figure 3.8 for an illustration of the separation lines.

There are certain situations that can arise when creating these lines. One such situation is when all points around a given test position are all high greyscale values. This is a trace intersection point. If this is the case, the algorithm then "digs" through the intersection point row wise until it hits a zero value. It then starts again but from the other side of the image. If the reverse line determines that the intersection point is the same as the exit point from the first line, then we have a "bounce". This is when no real intersection occurs, but the trace lines touch such as in Figure 3.9. Since we are interested in the middle point of the traces, the middle separation line is joined between the two intersection points.

Now that we have the three lines (for two days, four lines for three etc.) that separate the traces, we now compute the column median values for the first trace in the region between the *Lower* and *Middle* lines. Then the second between the *Middle* and *Upper*. The baselines, due to their consistent linear manner, do not require a complex method for extracting. The peak detection used in the pre-processing stage was sufficient for this task.

The data is then cleaned by removing any residues left behind from numbers and letters. These are often found on both ends of the traces, and baselines. This

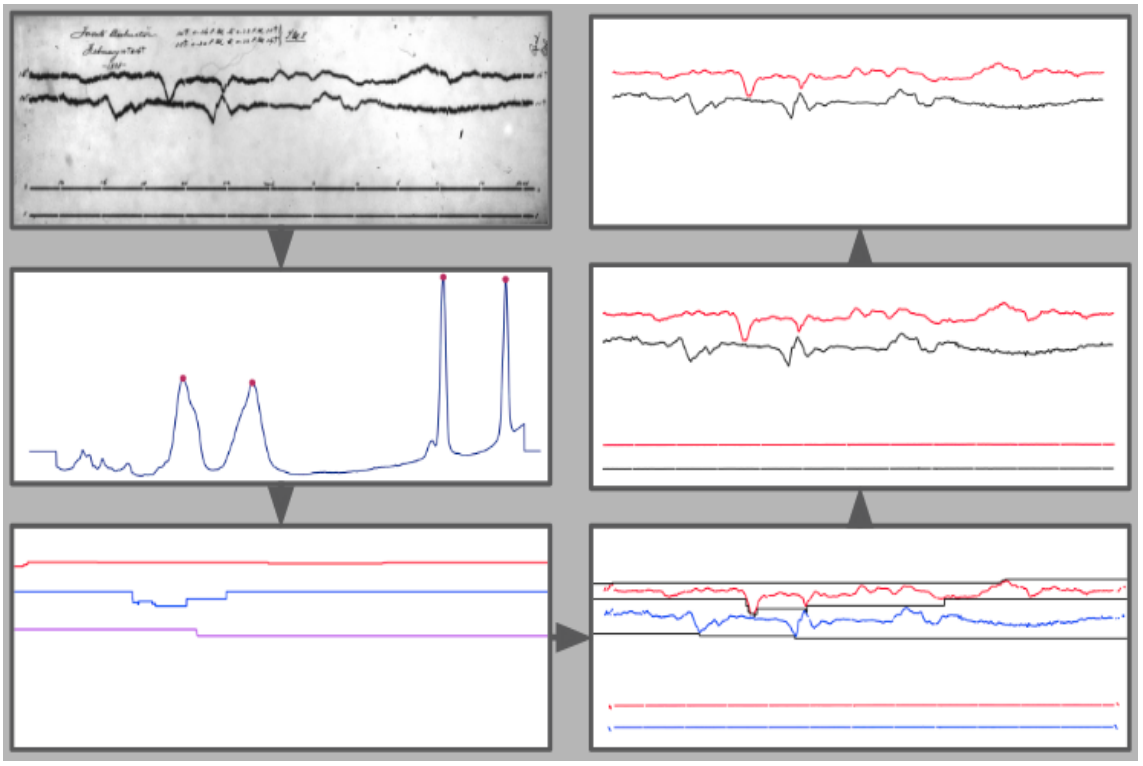

FIGURE 3.9: Overview of the Trace Identification by Separation process. The total grey-scale values for each row are calculated in order to identify peaks. These peaks then are used to identify starting positions for the separation lines. The traces are then enclosed, and with the pre-processing complete the only remaining data is the trace. The median value is collected for each column, and for each enclosure. After a final cleaning the digitization process is complete.

was accomplished by using the same approach that was successful in the other digitization algorithm.

The outcome of this digitization method has shown that it can quickly digitize the mundane image types, and some of the more complex. This method has also been seen to successfully digitize images that the original could not, and vice versa. At this stage, the combination of these two algorithms covers a large range of the images, and using both to digitize will allow a more complete set of results. This does depend on if the two methods produce comparable results. Otherwise, the quality of the digitization could be significantly different which could cause

problems for those who may use the data. This is explored in detail in Chapter 5.

3.4 Shiny Web-App

The digitization algorithm continues to improve, but the need to verify the integrity of the results remains. The time required to visually inspect each image would be substantial for one individual. They would have to plot each digitized magnetogram, and hunt down the original image in order to compare. The alternative is to have others help in this process.

A Shiny Web-App has been created to serve this very purpose. The Shiny Web-App is a mobile friendly web-application that is created through the *R* language. The app has been designed so that the user can quickly compare the digitized result with the original magnetogram image. Then they can select one of three options: *Good*, *Bad*, and *Close*. The *Close* option is for images where the digitization contains the data, but noise/holes exist. This option can also be consider as "not sure". These images can often be easily fixed by an expert as they exhibit extremely minor problems.

The current version of the Shiny Web-App, was designed for speed, and simplicity. As the final version of the app will be accessible from the web, it is possible that the user will be on their phone. So it was important that the Web-App would work on a smaller, narrower screen. Thus it was created with a fluid layout so that the images, and buttons would always appear in the same position.

Continuing to improve performance of the web-app was very important throughout the project. The magnetograms were copied, and then converted to the JPG file format from the original TIFF. The magnetograms in the their original digital format ranged from 10-13MB, and this caused problems. The app would stop loading images after a given threshold was reached. This threshold varied depending on the device used as it is a problem with the amount of memory available. With the conversion from TIFF to JPG, this issue vanished.

The Shiny Web-App does what it was designed to do, and has been vital to the identification of different errors produced by the digitization algorithm. In addition, the comparison between the two main digitization methods was made much easier with the use of this Shiny Web-App.

In addition to what this app can already do, an additional tab can be added into the web app that will allow users to aid in the identification of key points in a magnetogram that was not successfully digitized by either algorithms. As there are select magentograms that will require manual intervention in order to be digitized, this will significantly aid the operator. More about this will be discussed in the future work section.

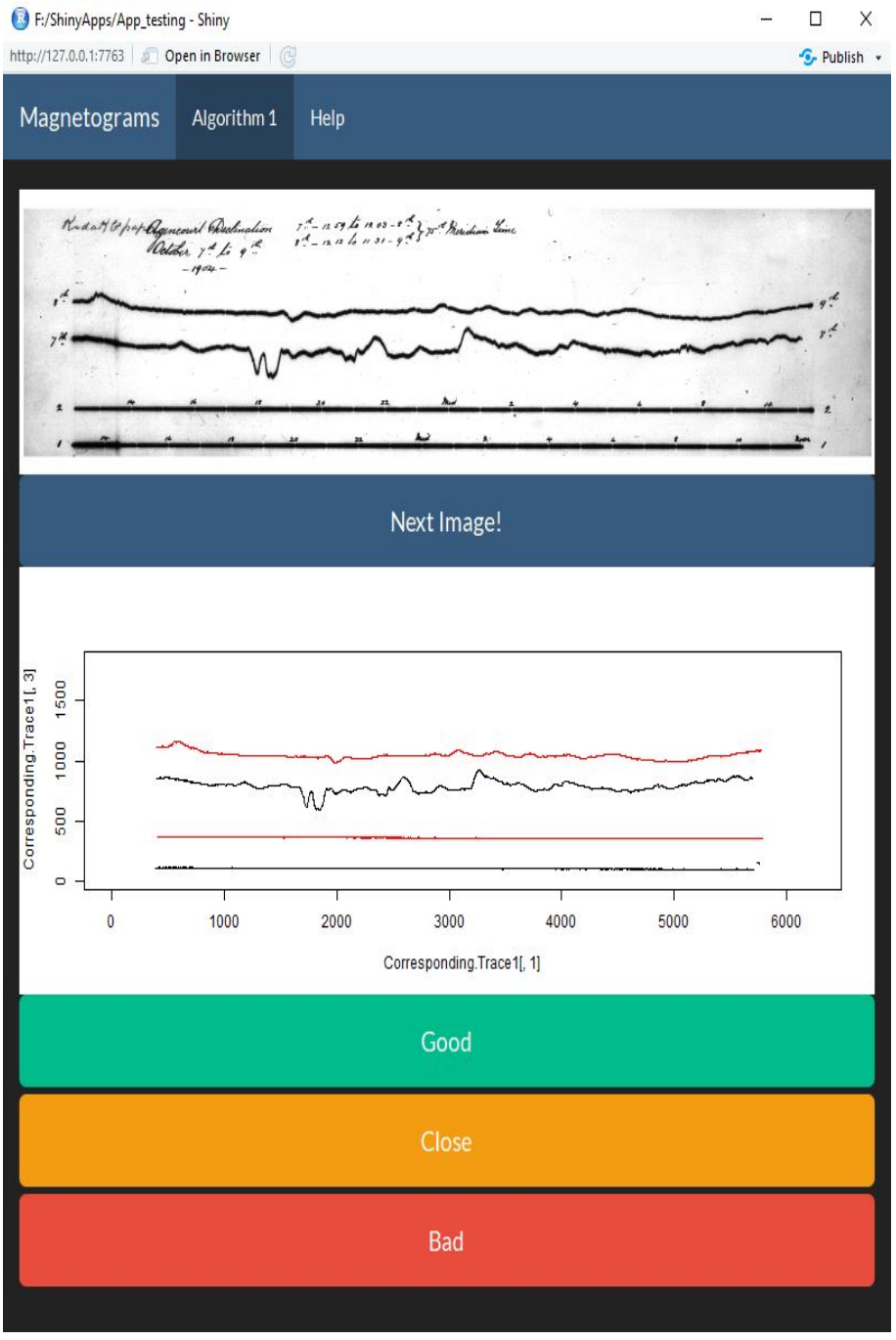

FIGURE 3.10: Working local app

3.5 Crowd Sourcing

Once this Shiny Web-App is made public, there will be the possibility that the users may make mistakes. This can either be unintentional or intentional. So we will need a way to identify reliable users. The approach will be similar to that in the *GalaxyZoo* project as it has already proven to work.

The difficulty will come when trying to determine the accuracy of the crowd-sourcing. Unlike the the *Galaxy Zoo* project, there does not exist similar geo-magnetic data to compare with. This is new territory that has not been explored before.

3.6 Chapter Summary

The proof of concept, highly supervised magnetogram digitization algorithm has been successfully transformed into a working automated algorithm with a number of additional features and more resilience. In the beginning, the algorithm was designed to process one image a time. Now entire image sets can be put through the digitization process, with no further human interaction.

Many of the problems first identified in the project, such as when traces "bounce off" one another have been addressed. Further work has also been done in the pursuit of overcoming the difficult challenge of digitizing images that contain intersections.

Additional work has been completed regarding error control throughout the digitization process. Before we began any work on the original proof of concept script, there did not exist any form of error control. This meant when a section of the code failed, or produced a result that was incorrect, there was nothing to catch it, and try something different. Now there exist many safeguards that will allow the code to skip a particular image if all other options have been exhausted, and continue with the digitization of the remaining images.

A Shiny Web-App has been created to help with the verification of the digitization process. It allows the user to compare the digitized magnetogram with the original image. The user can then select whether or not the digitization has successfully extracted the features of the magnetogram.

Chapter 4

Optimization

4.1 Strategy

The idea that "if you have the tools to complete the job, you don't need to go to the store" can be very dangerous. In the context of code optimization, we often only go looking for new tools when our current ones can't handle the job or when they perform poorly. But how can we successfully identify the tools that may cause slow downs, and find replacements? No matter what the job is, the best way to improve is to identify what worked, what didn't, what could be improved, and what can be left behind.

The strategy followed was to first measure the performance of the code via profiling. After the code was profiled, the slow sections or "bottle-necks" of the code were identified. This then allowed the programmer to make good use of their

time, as they could focus on areas of the code that actually caused slow downs. After work was done in these areas, it was very important that the programmer verified that the new output is the same as the old. This seems to be common sense, however it is extremely tempting to only do a quick visualization of the results, and this could possibly lead to problems or errors later on in your code.

4.2 Optimization in R

The default functions in the programming language *R* are convenient, and allows the many users of the language to have common functions that everyone can use without the need for packages. However, the limitations, and trade offs for these functions are not always clear or understood. The original digitization code used many unique ideas such as deconvolution for preprocessing, but often selected the default *R* functions to achieve the desired results. After additional work was done in extending the code to working order, time was invested to identify weakness in the default *R* functions used, and to find better solutions.

The digitization process, once all of the algorithms were in working order, consumed around two minutes just to digitize one image on an Intel Core i7-6700 @ 3.40GHz, running serially. An image group consists of 580-650 images. At that rate it would take approximately 20 hours to complete the digitization of just one image group, and in total there are over 30,000 images. If left unchanged, would

take around 1000 hours or approximately 41 days to digitize the magnetograms using the original algorithm alone, assuming continuous serial operation.

4.2.1 Profiling

The first step was to identify "bottle necks" in the code. This was accomplished by profiling the code, using the *R* package *profvis*[29]. The largest bottle neck identified was the *getTraces* function found in the second stage as displayed in Figure 4.1. This original function was created by the team at Queen's, with a few additional modifications added for error control.

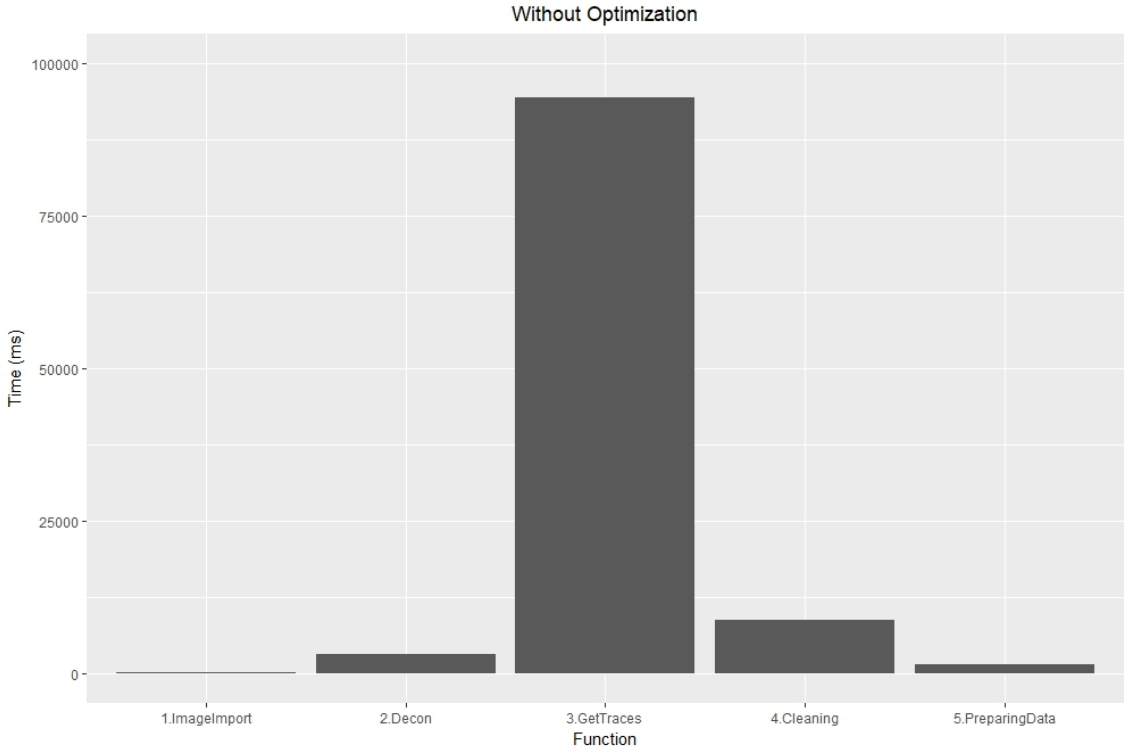

FIGURE 4.1: Initial profile results for the original algorithm. The bottle neck was identified as the getTraces function.

Not only does *profvis* provide information about each functions but it also provides a breakdown of the sub-functions. One such sub-function was called *rbind*. It is a default function within R, it can take vectors, matrices, and data frames as inputs then combines them by rows (thus the *r* in *rbind*, see Figure 4.2). For a typical type 2 image, this function alone clocked in at 1.46 minutes on average (with a standard deviation of 15 seconds) which accounted for approximately 74% of the total run time (1.97 minutes).

```
# Using vectors
          x <- c(1,2,3)

          y <- c(4,5,6)

          rbind(x,y)
```

```
##   [,1] [,2] [,3]
## x    1    2    3
## y    4    5    6
```

FIGURE 4.2: *rbind* example

This function was identified as the largest bottleneck by far, as no other single function came close. The reason *rbind* consumed so much of the run time can be traced back to its origins. It is considered a *pure function*: it always maps the same input to the same output and has no other impact on the workspace[28]. The *rbind* function first allocates space for the new object and then copies the old object to its new location using copy-on-modify semantics[28]. If this is done many

times over, the length of time required to complete is extensive. This is our exact problem.

4.2.2 Finding a solution

A solution was found in the *dplyr* package. Using the function *bind_rows* in place of the *rbind* function reduced the total run time by 45% time which, at that time, was 1 minute. A simple explanation is that the functions in the *dplyr* package are written in C++, which is known to be much faster[10]. But after exploring both functions in more detail, a much more interesting picture emerged. First, *bind_rows* is not a simple replacement for *rbind*. The *bind_rows* function has different requirements that must be met. Take for example combining vectors by rows with *bind_rows* as displayed in Figure 4.3.

The difference between Figure 4.3, and Figure 4.2 is that in order for the *bind_rows* function to work, linear names are required, ie. $\mathrm{c}(x = 1, y = 2, z = 3)$ instead of $\mathrm{c}(1, 2, 3)$. It may seem like a minor difference, as the general structure is consistent, however the output of these two functions are of different classes. *rbind* produces a matrix, while the output of *bind_rows* is a data frame. In addition, the *bind_rows* function is incredibly slow compared to *rbind* when it comes to vectors as can be seen in Figure 4.4. The reason why you don't see any results for the *rbind* is because it would be a horizontal straight line near zero. Thats how

```
# Using vectors
          x <- c(a = 1,b = 2, c = 3)

          y <- c(a = 4,b = 5,c = 6)

          rbind(x,y)
```

```
##   a b c
## x 1 2 3
## y 4 5 6
```

```
          bind_rows(x,y)
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1    1.    2.    3.
## 2    4.    5.    6.
```

FIGURE 4.3: *bind_rows* example

fast the *rbind* function can be when working with vectors as inputs. So why did *bind_rows* decrease the run time so significantly in the digitization code? It was because the inputs were data frames, not vectors.

When it comes to working with data frames, the results could not be more different. This should not be so surprising as the *dplyr* package was created to work with data frame-like objects[10]. However this exploration of the function proved to be very useful, as it was very tempting to simply replace all *rbind*'s with *bind_rows* once it decreased the run time significantly in one area. In Figure 4.5 the data frames consist of 1500 rows, and their corresponding number of columns. The *bind_rows* function significantly out-performs *rbind* when combining data frames by rows that it would appear as a horizontal line near zero. When comparing the

two R functions, the *bind_rows* is worse at combining vectors by rows than the *rbind* function, but is superior when combing data-frames by rows. This explains why there was a drop in run time when *rbind* was strategically replaced by the *dplyr* function.

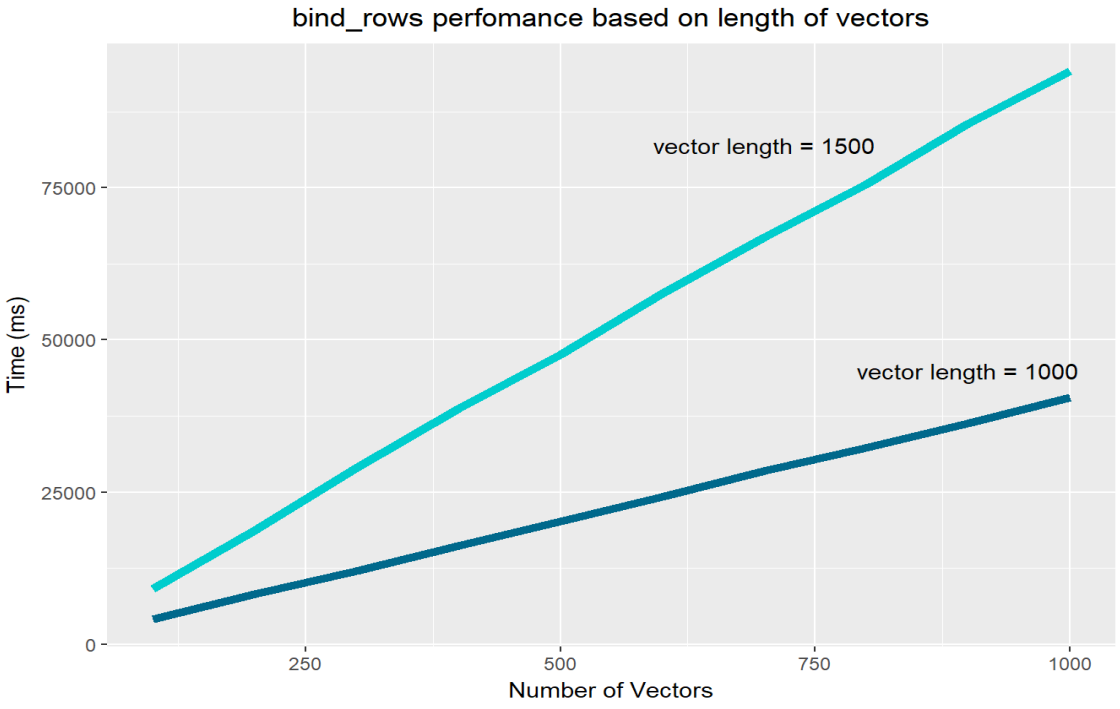

FIGURE 4.4: *bind_rows* performance using vectors

4.3 Other Optimizations

While the substitution of the *dplyr* function in place of the default R solution created the largest decrease in run time, there were other optimizations completed that reduced the average overall run time. These ranged from removing for-loops, and replacing them with vectorized operations when appropriate, to finding root

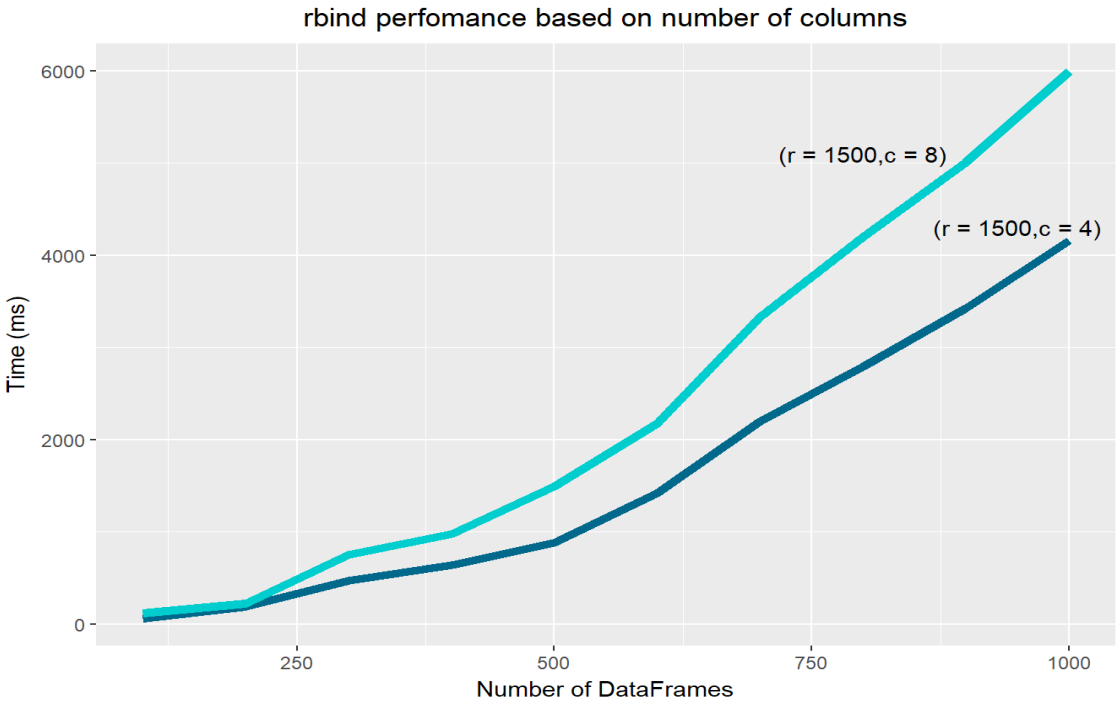

FIGURE 4.5: *rbind* performance using data frames

causes of errors that forced the code to redo large sections of the algorithm. To date, the performance-improved original algorithm can successfully digitize an image within 30-50 seconds, lowering the theoretical serial time to complete the entire image set to 13.8 days from the original 41.

The Trace Identification by Separation algorithm did not have any single identifiable bottleneck which made it very difficult to optimize. As with the original algorithm, the average completion time varied depending on the input. Throughout the project additional code was implemented to overcome errors that would prevent a successful digitization on both algorithms. This would ultimately lead to a longer digitization as the errors had to be narrowed down, and run through

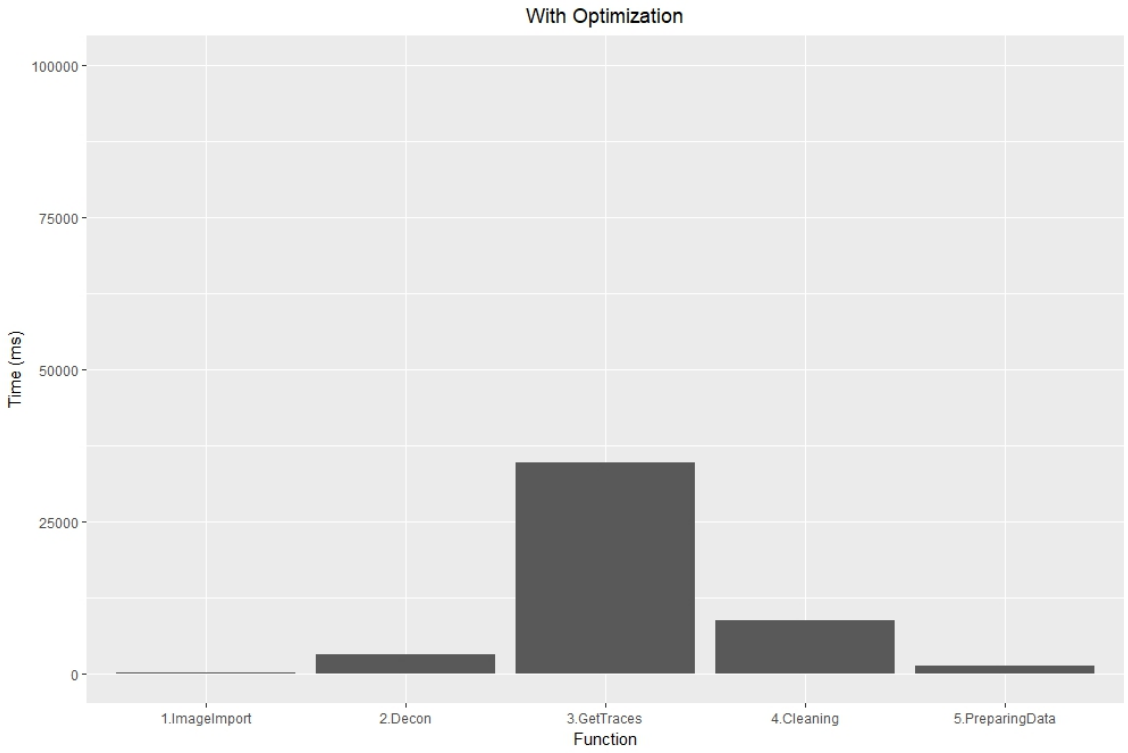

FIGURE 4.6: Profile results with bind_rows on original algorithm

various tests. If these were successful they would then start over with the the new parameters. As more updates are added to deal with these problems, the average time will increase.

Through the profiling of the modified original algorithm, we identified that combining data structures by rows can be a bottleneck. Going forward we will need to select the best function for the given situation. We know now that *bind_rows* is a better solution than *rbind* for data frames, but there are situations where we wish to combine elements of a list. Normally if we wished to combine elements of a list by rows we could use a for-loop or use `do.call(rbind,list)`. The latter being the superior choice as it is much faster. However we are still dealing with *rbind*. The alternatives are: `do.call(bind_rows,list)` if the elements of the list

are data frames, or a function from the *data.table* package named `rbindlist`.

The *data.table* package is a response to the copy-on-modify method used by R. The function we are interested in is `rbindlist`, which is a much faster replacement for `do.call(rbind,list)` as it is completely written in C for speed[7].

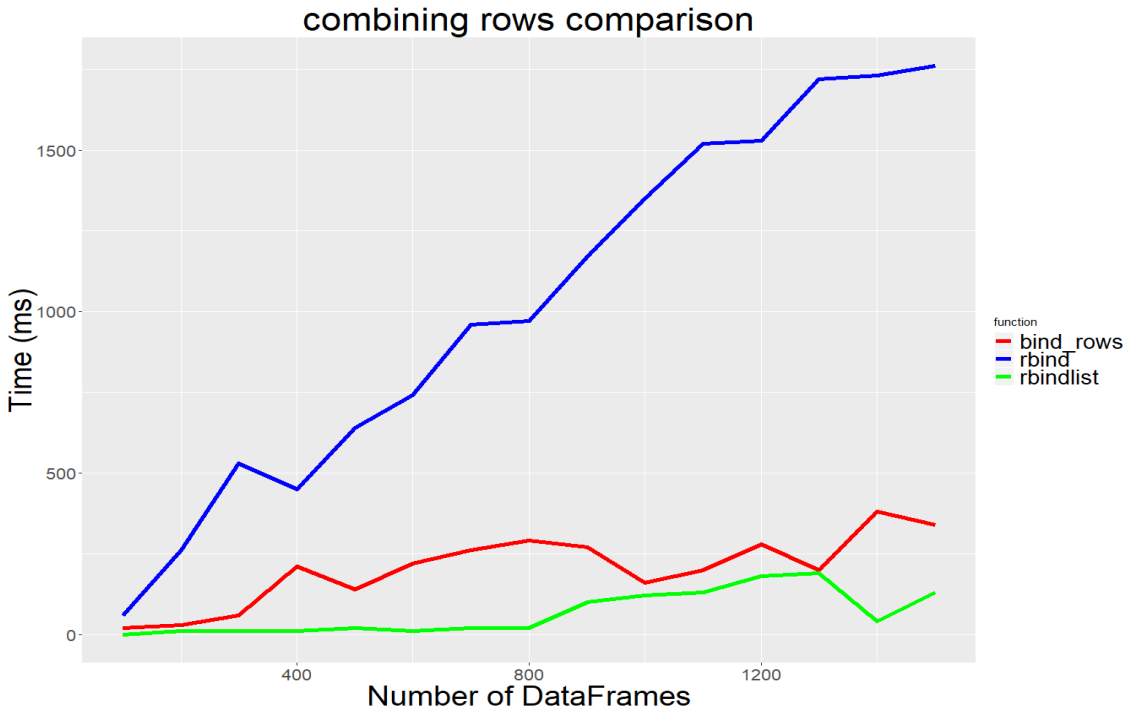

FIGURE 4.7: comparison of bind rows functions using dataframes

To test the three methods 1500 individual dataframes were created, each having 1500 rows and 20 columns. These dataframes were artificially made to test the functions with similar inputs to those found in the digitization algorithms. The performance difference can easily be seen in Figure 4.7. *rbind* performed as expected, while the other two functions produced similar results with *rbindlist* being slightly better. Based on these results a test was then performed

on the modified original digitization algorithm to compare *bind_rows*(dplyr) and *rbindlist*(data.table). This algorithm was selected as the results would have more significant impact than in the Trace Identification by Separation algorithm at this time.

The results were very similar to the initial test. The *rbindlist* function outperformed the *bind_rows* by an average of two seconds for the selected test images when the data frames were placed into a list. Two seconds may not seem that much when an image takes 30-50 seconds, but when dealing with 30,000 images it adds up quickly. This shows that if a user wishes to combine data frames by rows in R, they should first place them into a list followed by *rbindlist*.

For now the original algorithm still uses the *dplyr* function as it will take some time to convert all of the required inputs into lists to be used by *rbindlist*. The most recent discussed test showed that it is viable, and is the best way forward for future work.

4.4 Chapter Summary

Initial results showed that the original digitization algorithm required an average of two minutes to digitize an image successfully. After all optimizations to date, the performance-improved original algorithm can successfully digitize an image in the range of 30-50 seconds, and the Trace Identification by Separation takes between 10 and 30 seconds.

This was accomplished by fist profiling the digitization algorithms to identify "bottle necks" in the scripts. Once this was completed, we could then focus our optimization efforts on key areas that have been shown to cause slow downs. The main bottle neck was identified to be a default R function called *rbind*. When used with the correct input, this function can be very fast. However, in the digitization algorithm it was found to be responsible for 74% of the run time. A different function was selected for that particular purpose. Further analysis on these functions was then conducted; demonstrating the strengths, and weakness of those functions. This resulted in the proper function being selected for the given row binding operation. The result was a reduction in the total run time by 50% for the original modified algorithm.

In future work, there may continue to be additional code created to overcome special case images that have not yet been detected. Through the process of profiling and optimizing, solutions have been found for one of the most time consuming operations in R. This will enable all following work regarding this operation to have a reduced run time.

Chapter 5

Results and Discussion

5.1 Results

Thus far, the created/modified algorithms can be left to run on the image sets, and will continue working even if an error occurs. These errors are to be expected as some of images contain mechanical/human errors. These images are then identified so that they can be dealt with by the user.

As a result, many of the magnetograms have been digitized, and await visual inspection before proceeding any further. To test if the visual inspection process would be successful, a trial was performed on a three year image set. In addition, a comparison between the two digitization algorithms (modified original proof-of-concept, and newly developed) was performed. This comparison was to determine

whether or not the results produced from the two digitization algorithms are compatible.

5.2 Comparing the Algorithms

The two digitization algorithms both have their strengths, and weakness. The ideal case, would be to create a hybrid with these two algorithms which would then increase the number of successful digitizations. This does however depend on how similar the two results are.

The comparison shall first look at how similar the results of both algorithms are with respect to one another. This was done by finding the difference between each point in the shared image, along with a visual inspection via the Shiny Web-App. When doing this comparison the results between the two digitization techniques, only the magnetograms that were successful in both methods were selected. This is because it does not make any sense to compare a successful digitized image with a failure. There did arise situations where one of the algorithms captured a more accurate interpretation of the magnetogram than the other. This was also taken into account when comparing the algorithms.

Next we will compare overall performance from select samples of digitized images from both groups, along with the percentage of magnetograms that could not be digitized by either algorithm.

5.2.1 Comparison Results

The first comparison will investigate the similarity of successful images from both digitization algorithms. To clarify, a successful digitization is one in which it passes the visual inspection for a "good fit". The distance between corresponding points was computed, and stored. The grey-scale values were also extracted from the TIFF image from each path for each algorithm.

As an end result, the algorithm that contains the highest total grey-scale value likely contains more correct data. However, it was important to acknowledge that total grey-scale values change based on geomagnetic activity. For example, an image that contains two traces may have one trace "normal", while the second has more activity. The total grey-scale value would then be higher in the normal trace than in the other.

Table 5.1: Algorithm Comparison

Image Name	Mean Difference	G-S Original	G-S TIS
AGC-D-18980113-18980115	1.9 pixels	736	733
AGC-D-18980119-18980121	1.6 pixels	621	606
AGC-D-18980121-18980123	1.5 pixels	657	664

When a visual inspection is conducted on a pair of digitized images from the two different algorithms that both pass, they often look identical. This is true for more than 95% of cases. They can only really be distinguished by a numeric comparison. The images used in Table 5.1 are typical examples. What has been noted is that the modified algorithm tends to 'over step' the baseline, while the separation algorithm is more reserved (stays closer to the baseline). This however

is not a real problem, as in the baseline subtraction and stitching step, both use the baseline to remove any extra trace.

The next comparison shall look at the success rate of both algorithms, and then what could not be digitized by either. So far only a small number of images, with respect to the size of the data-set, can been compared as the visual inspection by a single individual takes time. But we do have the results for the three year period from 1898-1901. This particular image group is interesting because of the different variation in contrast, and general image quality.

TABLE 5.2: Digitization Comparison for the Image Group 1898-1901D

Image Group	Original	TIS	Undigitized
18980111-19010529	60%	77%	9.0%

The original digitization algorithm, based on the results displayed in Table 5.2, appears to under perform when compared to the results of the Trace Identification by Separation algorithm. This both is, and is not the case. The more complex the image is, the greater chance the original algorithm has on successfully digitizing over the TIS. Take for example Figure 5.1. In this magnetogram the top trace intersects the lower trace, and for a short time follows the same curve. This digitization was completed using the code that was added to digitize Figure 3.2. Collectively, the algorithms successfully digitized 91% of the images from the 1898-1901D group, with the remaining 9% requiring manual intervention.

The current state of the original modified digitization script has a greater chance at handling more complex images such as Figure 5.2, and thus preferred

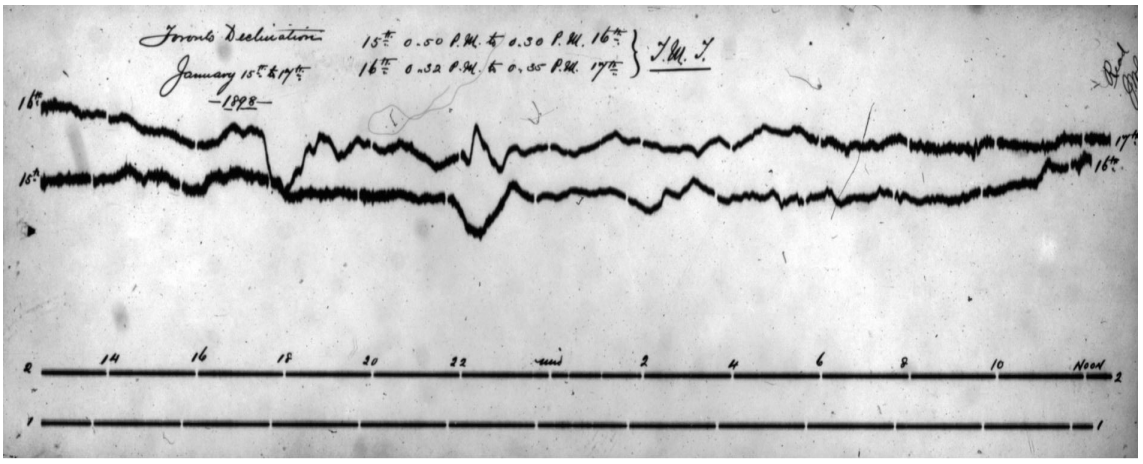

FIGURE 5.1: Complex Magnetogram

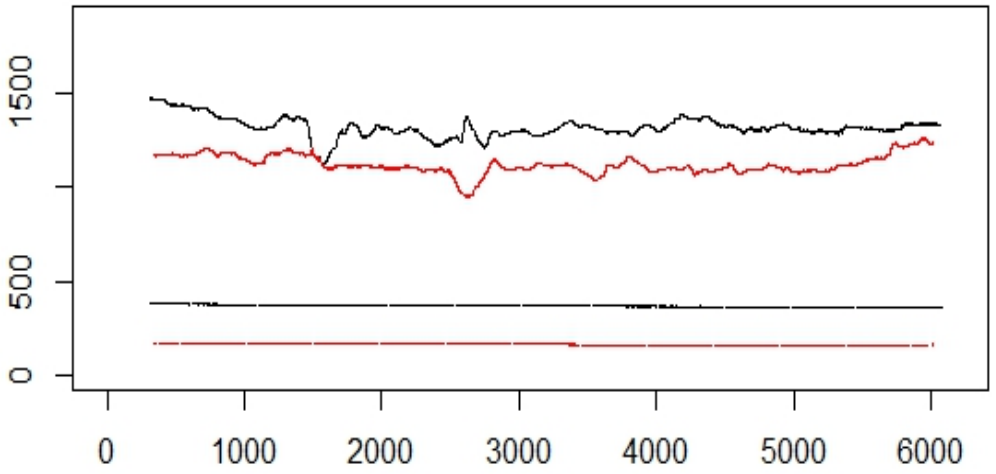

FIGURE 5.2: A successful digitization of a unique magnetogram by the original algorithm. This is also an example where the TIS algorithm failed due to the prolonged contact between the two traces.

over the TIS algorithm for the known complex images. However, there are still many cases where the simplicity of the TIS algorithm excels. This will be discussed in the Digitization Results section at the end of this chapter.

5.3 Using the Algorithms

To use the first digitization algorithm, a simple R function has been created. This function allows the easy use of the algorithm so that users do not need to know the inner workings of the algorithm in order to use it.

```
MagDigitize(file_location,image_name,withplots,
            optimization,save_results)
```

By default, the function assumes that the user wants to digitize a large group of magnetograms. So the user just has to input where the magnetograms are located by typing: `MagDigitized(file_location = folder123)` as an example. If the user wanted to digitize a single magnetogram, then the file location, and the name of the particular image would have to be put into the function. The remaining function options are for special cases.

If the user wanted to see how the function was performing as the digitization process was on going, they can type in `withplots=TRUE`. The function will then display a series of visualizations as progress was being made. These visualizations will appear after key stages such as: the tracing, cleaning, and stitching. There is also an option to not save the results, and an option to use a more optimized version of the function. These are mainly for testing purposes, and will be removed/replaced in the future.

The Trace Identification by Separation (TIS) version follows the same function structure as the original for convenience. The only difference being the name of the function, which for now is TIS. This has allowed for easy comparison, and profiling.

```
TIS(file_location,image_name,withplots,
    optimization,save_results)
```

With these immense algorithms reduced to single functions they can easily be used, and eventually will be placed in a package so that they can be transferable.

5.4 Digitization Results

Currently 18,000 of the currently available magnetograms have been digitized by the algorithms, and now require inspection. To test the process, image group 1907-1911D was selected as it appeared to have no recorded issues. This required an inspection to verify that no errors/missing images were present as it might be possible that the 'readme' file was lost. The breakdown of the image types identified by visual inspection can be found in Table 5.3.

After completing the visual inspection of the image group, it was found that 16 out of the 43 type 2.2 images are high activity, and 2 out of the 16 are high activity for a total of 18 or 3% of the image group. In addition 30 images contained attributes that are known to cause a unacceptable digitization from the algorithms

Table 5.3: Detailed break down of Digitization Results for Image Group D:1907-1911

Image Type	Frequency	%	TIS	Original
1	16	2.7%	8	13
2.1	501	85%	459	443
2.2	43	7.3%	15	12
Error	30	5%	0	3
Total	590	100%	482(82%)	474(80%)

(errors). This means that out of the 590 images for this image group, 48 (18+30) images were not expected to be successfully digitized, concluding with an 80% success rate for the original algorithm.

The TIS algorithm outperformed the original overall with an effective success rate of 88%. It did however suffer in the type 1, and the error classes. This is not strange, as the TIS algorithm was designed to make up for the weakness of the Original. Normally the TIS algorithm would be run on the images that failed the Original digitization. This method saves time, and allows both algorithms to be used where they are most effective.

5.5 Chapter Summary

It was determined through comparison that the two algorithms are compatible as they produce extremely similar results. An image group that contains a variation in image quality was selected for this comparison. This success allows the use of both digitization algorithms which will enable a larger number of magnetograms to

be successfully digitized without the fear of inconsistency between the two different digitization methods.

Currently 18,000 of the available scanned magnetograms have been digitized, and can proceed to the visual inspection stage of the process. One of the image groups from these digitized images, has successfully gone through the entire process while the results of a second has been quickly inspected. Both algorithms performed well with the Original algorithm successfully digitizing 80% of the magnetograms, and the Trace Identification by Separation with an effective 88% success rate out of the achievable images in that group.

Chapter 6

Conclusions

We have transformed a proof-of-concept highly supervised digitization algorithm into an unsupervised digitization algorithm that can identify, and pass over images that contain errors. In addition, a second digitization algorithm was created in order to evaluate the original algorithm, along with increasing the number of magnetograms that could be digitized.

The modifications to the original highly supervised digitization algorithm, along with the creation of a new digitization algorithm further prove that the data contained in the magnetogram analog images can be extracted. Then with the use of the created Shiny Web app, the data extracted from these images can easily be validated.

6.1 Future Work

In order to proceed any further in this endeavour, the remaining digitized data must first be visually inspected on a large scale. Once this has been completed, and any failed images have either been corrected through changes to the algorithms or through some manual efforts, the data from the log books must be gathered. Then, with the correct baseline values, the magnetograms can be properly stitched together as the last step in the digitization process.

It would be recommended to perform these steps one image group at a time (approximately three years worth of images). Each image group often contains similar obstacles so when additional code is added to deal with a particular image, it can then be applied to the next image set. This will reduce the number of times an image set will have to be visually checked, as there is likely to be more success.

6.1.1 Manual Interpretation

There exists images that either contain too much geomagnetic activity (multiple non-clean intersections), or human/mechanical error that cannot be digitized by any conceivable modification to the current algorithms. A solution is to add a tab to the Shiny Web app that will allow the user to manually identify points along each of the traces in a given image.

With the points periodically placed along the given traces at points of inflection, and extra at 'difficult' areas, such as intersections, a simple algorithm can be

created to connect the points together. This could use segments from the current algorithms, but instead of tracing the whole trace, it could work in sections.

No work thus far has been placed into the algorithm section of this proposed idea, but based on prior work this does not seem like a difficult task. Work has been done in testing the ability of the Shiny app to perform such a task. A slider that goes from 0 to the length of the image can easily be placed underneath the image of the magnetogram. The user can move this slider, and a vertical line will appear on the image at the position of the sliders. This could help match the baselines with their corresponding trace if done at the beginning and end of the baseline.

If given this information, the original modified algorithm might actually be able to digitize the given image as the starting, and end points may help it select the correct timing marks needed for matching. However, it is still likely that this will not solve all of the problem images. So, if a slider is then placed at the side (y-axis) of the image, the user can create an intersection with the two sliders, and create a point. The user then repeats as many times as needed.

This seems to be the most straight forward way of achieving the desired outcome with respect to coding. This is however very tedious, as a simple click would be preferred. This should be looked into for future work.

6.1.2 Future Optimization Improvements

When work began on digitizing the magnetogram data set, there existed a single incomplete non-working algorithm. Now there exists two robust digitization algorithms that each have a very different way of extracting the data.

Initial profiling made clear that the largest bottleneck was a single row-binding function called *rbind*. A replacement was quickly found in the *dplyr* package which reduced total run time in one of the digitization algorithms by a significant amount. Even with this reduction in run time, the row-binding sections of the algorithms still produced some of the longest run times as the new function could only reduce run time if the input was a data-frame.

The algorithms will continue to need improvement to overcome new obstacles that we are unaware of at this time. This will only increase the run time as the additions will likely be related to error control, which involves running a section multiple times with different parameters or using a different sub-algorithm.

If we could select the most probable option first, this would most definitely help keep the total run time down. The magnetogram could start with the predicted parameters for the most likely match, then the second, and so on. Even if this only works successfully on a small number of images, it has the potential to reduce the overall run time.

This could be accomplished by adding additional code to the algorithms to save the parameter/sub-algorithm selection data. Then using this data with a

machine learning algorithm we could create a prediction model, and use it on future images. The process would have to begin with the creation of a test, and training set in order to build the model from the parameter/sub-algorithm selection data. The first machine learning model that should be tested is a random forest model. This may seem strange, as it is not a common model to use when dealing with images. The reason for it's selection is that most of the parameter selection processes already follow a decision tree type structure or just simply goes through a list of numbers until things work. The latter being very time consuming. If the algorithms could start with a set of likely parameters it could significantly reduce the run time of the initial stages. This would be an interesting experiment if nothing else, and will definitely be done in future work.

Another idea for future work goes back to the row binding operation problem. We know now what function to use based on the data structure, but having three functions for ultimately the same purpose should be changed. A single function that selects the best row bind operation based on data input can be easily created. Having this function could quickly optimize almost any script with a simple find and replace command. This is what the function could look like:

```
r.bind(data,requested_output,compare){
    library("dplyr","data.table","profvis")
    data_class <- class(data)
    if(data_class == "class"){function1 }else{etc.}
        out}
```

The additional suggested options (requested_output,compare) are to increase the usefulness of the function. The requested_output by default should be set to be equal to that of the input class. There might be situations where the user may wish to place the results of the operation into a different data structure. The compare option could be selected with a `TRUE` or `FALSE` (by default). This would allow the user to see how the individual functions perform. Just like all the other algorithms this function will need helpful warning, and error messages added to help users to be successful. With this completed, it could then be used to help both algorithms stay optimized with respect to row binding in future additions.

6.2 Summary

Two magnetogram digitization algorithms have been created to extract time-series data from thousands of images from the Toronto and Agincourt geomagnetic observatories. Currently about 18,000 have been digitized by these algorithms, and wait to move on the the next stage: inspection. Plans for completion are through the use of a publicly available web-app where many users can help with this endeavor.

Once the digitized data has been properly inspected, the final step of remaining work will be to stitch the images together to create a long-running time-series.

An algorithm has already been created to perform this step, but in order to properly attach the images together we will need the exact start, and end times of each trace. Some images contain these times up to the minute, while others do not. The scanned magnetograms do however contain the year/day/month in their name, which makes the organization very easy. After stitching, the data will be made publicly available so that further research can be conducted on geomagnetic phenomena.

Bibliography

[1] A Bartlett, B Lichtner, M Nita, B Yashar, A Azzarone, and L Bartlett. "SKATE: A Web-Based Seismogram Digitization Tool." *Seismological Research Letters* 89: (2018) 1886–1893.

[2] Borchers, Hans W. *Practical Numerical Math Functions*, 2019. R Package.

[3] BritishGeologicalSurvey. "The Digitisation of Observatory Magnetograms.", 2009. Poster presented at the IAGA 11th Scientific Assembly.

[4] ———. "Carrington as a Benchmark: Comparisons of the September 1859 Storm Using Newly Digitised Data for London.", 2013. Poster presented at European Space Weather Week 10.

[5] ———. "Historical UK magnetic observatory magnetograms and yearbooks.", 2019. `http://www.bgs.ac.uk/data/Magnetograms/search.cfc?method=magnetogramSearchbyObservatory`.

[6] Church ED, Jourabchi MA, Bartlett AH. "Raster-to-Vector Image Analysis for Fast Digitization of Historic Seismograms." *Seismological Research Letters* 84: (2013) 489–494.

[7] Dowle, Matt. *Extension of 'data.frame'*, 2018. R Package.

[8] F Krüger, G Kulikova, A Landgraf. "Magnitudes for the historical 1885 (Belovodskoe), the 1887 (Verny) and the 1889 (Chilik) earthquakes in Central Asia determined from magnetogram recordings." *Geophysical Journal International* 215: (2018) 1824–1840.

[9] Gaston, Tissandier. *A History and Handbook of Photography.* New York: Scoville Manufacturing, 1877.

[10] Hadley Wickham, Lionel Henry Kirill Müller, Romain Francois. *A Grammar of Data Manipulation*, 2017. A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

[11] HydroQuebec. "In March 1989, Qubec experienced a blackout caused by a solar storm.", . `http://www.hydroquebec.com/learning/notions-de-base/tempete-mars-1989.html`.

[12] IRIS. "Archives by Stations: Puerto Rico.", 2017. `http://ds.iris.edu/seismo-archives/stations/puerto_rico/`. IRIS (Incorporated Research Institutions for Seismology). Seismogram from a Puerto Rico seismographic station.

[13] Kaggle. "Planet: Understanding the Amazon from Space.", 2017. `https://www.kaggle.com/c/planet-understanding-the-amazon-from-space/data`. Using satellite data to track the human footprint in the Amazon rain-forest.

[14] Linchao Pan, Fugeb Zhang, Rui Meng, Jie Xu, Chenze Zue, and Baozhen Ge. "Anomalous change of Airy disk with changing size of spherical particles." *Journal of Quantitative Spectroscopy and Radiative Transfer* 170: (2016) 83–89.

[15] M Pietrella, A Pignalberi, M Pezzopane, A Pignatelli, A Azzarone,, and R Rizzi. "A comparative study of ionospheric IRIEup and ISP assimilative models during some intense and severe geomagnetic storms." *Advances in Space Research* 61: (2018) 2569–2584.

[16] Mark Mitchell, Baurzhan Muftakhidinov, and Tobias Winchen et al. "Engauge Digitizer Software.", 2019. `http://markummitchell.github.io/engauge-digitizer`.

[17] NaturalResourcesCanada. "Geomagnetism.", 2017. `https://geomag.nrcan.gc.ca/index-en.php`.

[18] ———. "What is Space Weather.", 2018. `http://www.spaceweather.gc.ca/sbg-en.php#gen-7`.

[19] NOAA. "Space Weather Phenomena.", 2018. `https://www.swpc.noaa.gov/phenomena`. National Oceanic and Atmospheric Administration.

[20] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. `http://www.R-project.org/`.

[21] Robertomassimo Leonard, Vincenzo Malvestuto Olivia Testa, Tito Montefinale, and Maria Carmen Beltrano. "The Automatic Digitization of Time Series Recorded on Graph Paper Supports." *WMO Technical Conference on Instruments and Methods of Observation.* 1–8.

[22] Rohatgi, Ankit. "WebPlotDigitizer.", 2019. `https://automeris.io/WebPlotDigitizer`.

[23] Springford, and Riegert. "Preservation of magnetogram data by digitization: Development of software tools for digitization and storage." Technical report, Queen's University, 2016. Report delivered to Natural Resources Canada, Geomagnetic Laboratory, in reference to corpus of 32,000 magnetogram images.

[24] Lam, H.L., Boteler, D. H., Burlton, B., and Evans, J. "AnikE1 and E2 satellite failures of January 1994 revisited." *Space Weather* 10. 2012.

[25] National Research Council. *Severe Space Weather Events: Understanding Societal and Economic Impacts: A Workshop Report.* The National Academies Press, 2008.

[26] Vandenberg, Chris J Lintott, Kevin Schawinski, Anze Slosar, Kate Land, Steven Bamford, Daniel Thomas, M Jordan Raddick, Robert C Nichol, Alex Szalay, Dan Andreescu, Phil Murray,and Jan. "Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey." *Monthly Notices of the Royal Astronomical Society* 389: (2008) 1179–1189.

[27] W. Graham, et al. "Report of the Commission to Assess the Threat to the United States from Electromagnetic Pulse Attack." Executive Report, 2008.

[28] Wickham, Hadley. *Advanced R.* Taylor and Francis Group, 2015.

[29] Winston Chang, Javier Luraschi. *Interactive Visualizations for Profiling R Code*, 2018. R Package.

[30] Yang Liu, Mingyan Liu. "An Online Learning Approach to Improving the Quality of Crowd-Sourcing." *IEEE/ACM Transactions on Networking (TON)* 25: (2017) 2166–2179.

[31] Zooniverse, 2019. `https://www.zooniverse.org/projects`. A platform for citizen science research.

Appendix A

R Packages

- data.table
- dplyr
- ggplot2
- multitaper
- plyr
- pracma
- profvis
- tiff
- shiny
- shinythemes