Magneto Summer2020 Summary

Benjamin Ott

DigitizationTODO.py

The need for a program that told us what images had been digitized by previous versions became apparent, as it wouldn't be possible to re-digitize all of the images when the program was stopped. This was easiest to run in python because of its efficient computing capability. It asks the user for two .txt files *Figure: DigTODO1*; These files contain the path of one item, in our case these were the paths of the origional image and the path of the digitized image.

Original Image Path

/home/ben/magneto/Images/AGC-D-19010530-19040829/AGC-D-19031103-19031105.tif

Digitized Image Path

/home/ben/magneto/Digitized_Data/Algorithm2/AGC-D-19031103-19031105.tif.csv

---

The path was needed so that the digitize function would be able to find the image that hasn't been digitized yet (see TISForAutomation for more details about digitization function).

The path was then split into the name, the path and the file type; This program was designed to take any file type (.tiff, .tif, .tif.csv, .tif.png) as it separates by both "." and "/" to obtain

---

File Path: */home/ben/magneto/Images/AGC-D-19010530-19040829/*

File Name: *AGC-D-19031103-19031105*

File Type: *.tif*

This aloud us to only compare the file names of images and digitized images, without needing to worry about the path or the file type. This caused an decrease of the computation time as there wasn't any splitting done when the comparison was happening. This caused another problem, as we were only comparing the names, in some cases there was digitized images that don't have original images; code was created to identify when this happens and print it to screen for the user to see. We recorded these in a document **DigitizedNonImages.txt** to hopefully identify the original images one day.

As the comparisons happen in the function, They are recorded and outputed to a .csv file that gets saved locally. Example of the output *Figure: DigTODO2*. In order of the columns, it saves: **Image Path, Image Name, If the image has been digitized, Digitized Path, Digitized Name**. Along with this .csv, there is also information that is printed to the user about the data *Figure: DigTODO3*; This includes information to ensure that all of the files are accounted for in the final document, along with the number of digitized images and the number of images that still need to be digitized.

Finding Specific Days

To aid with the automation of TIS, a function was created to find all of the files we have that contain the specified **keyword**, which could be a day, a year, or any specific keyword that one would want to search for. This function is aptly named getImagesForDate which uses the fs package. Part of the fs package is the

```
C:\Users\rocke\Documents\PycharmProjects\venv\Scripts\python.exe C:/Users/rocke/Documents/PycharmProjects/Magnetograms/digitizationTODO.py
What is the image file?
Images.txt
What is the comparison file?
DigitizedBatch5.txt
What is the desired output file name? (must be .csv)
DigitizedB5.csv
```

Figure 1: DigTODO1: The user input

/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990911-18990913.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990911-18990913.tif.csv
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990913-18990915.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990913-18990915.tif.csv
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990915-18990917.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990915-18990917.tif.csv
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990917-18990919.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990917-18990919.tif.csv
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990919-18990921.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990919-18990921.tif.csv
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990921-18990923.tif	FALSE		
/home/ben/magneto/Images/AGC-D-18980111-19010529/	AGC-D-18990923-18990925.tif	TRUE	/home/ben/Magneto2020/DigitizedImages/	AGC-D-18990923-18990925.tif.csv

Figure 2: DigTODO2: .csv Output File Example

```
The amount digitized are: 3139
The amount that still needs to be done is: 23271
There are  12856 duplicate images in  Images.txt
There are  0 duplicate comparison files in  DigitizedBatch5.txt

VERY IMPORTANT
The ones that are not matched: [['AGC-D-19010729-19010730.tif.csv' 'in first array']
 ['AGC-D-19010908-19010910-BlackTraceOnly.tif.csv' 'in first array']
 ['AGC-D-19011008-19011010.tif.csv' 'in first array']
 ['AGC-D-19020402-19020404-allthecrazyredtrace.tif.csv' 'in first array']
 ['AGC-D-19020418-19020420.tif.csv' 'in first array']
 ['AGC-D-19020526-19020528-startTrace1.tif.csv' 'in first array']
 ['AGC-D-19020923-19020925.tif.csv' 'in first array']] First: comparison No duplicates, Second: matches with images
Length of notAccountedFor is 7

Check for the Total Comparison files:  3146 includes the files that aren't accounted for
Total Comparison files:  3146
Total Image files:  39266 subtract 12856 = 26410
Check for the total in end Array:  26410
Length of the final array made into csv:  26410
```

Figure 3: DigTODO3: Extra Information for User

```
[1] "/home/ben/magneto/Images/AGC-D-18640423-18800710(TOR)/AGC-D-18650228-18650229.tif"
[2] "/home/ben/Magneto2020/ImagesPNG/AGC-D-18650228-18650229.tif.png"
Warning message:
[EACCES] Failed to search directory '/home/ben/magneto/lost+found': permission denied
```

Figure 4: FindingDays1: Permission Denied

ability to use commands like grep from bash to search. It can be narrowed to a specific directory, or it can search recursively throughout all of the files from your home directory.

One problem that was created when making this function was the handling of un-foreseeable errors two, are great examples of this.

Using Grep with Symlinks

Symbolic links, or Symlinks, are pointers to another directory on another system for example. This became prevalent when using the magneto directory on the Trent server; I had access to it, but because it is a Symlink, to search that directory we had to first find all of the pointers/Symlinks then look recursively into all of those specific directory's separately.

Exeption handeling

Even with the symlinks working, there was still a problem with the chance of a directory being un-accessable by that specific user. This problem came when looking through the symlink directory for magneto *Figure: FindingDays1*. As this directory was a hidden directory that no one has access to, the handling had to be completed in such a way that the program would still complete, and just print a warning to the screen notifying the user that there was such a directory in the search radius.

This keyword searcher was implemented into the **AutomationScript**; It allows users to digitize only specific images or specific ranges of images based off of the **DigitizationTODO** csv. The user only needs to know the keyword and the function will find anything in the specified or non specified searching radius.

Finding Image Gaps

The long term goal for the magneto project is to obtain a continuous time series of all the data, in order to ensure that we had all of the images for each day a python program was written.

The first thing that needed to happen was to split the names from our file system. Using part of the algorithm created for the DigitizationTODO function, we were able to split the names into key parts; For example, AGC-D-19031103-19031105.tif, was split into these key parts:

The Observatory: *AGC*

The Axis of the plot: *D*

Start Date: *19031103*

End Date: *19031105*

Using the imported datetime package, the Start Date and End Date were successfully changed into a datetime object which we used later to produce the gaps. Any dates that weren't able to be converted were then sent to the user at the end of the script. An example of this was an image that was dated 18650228-18650229;

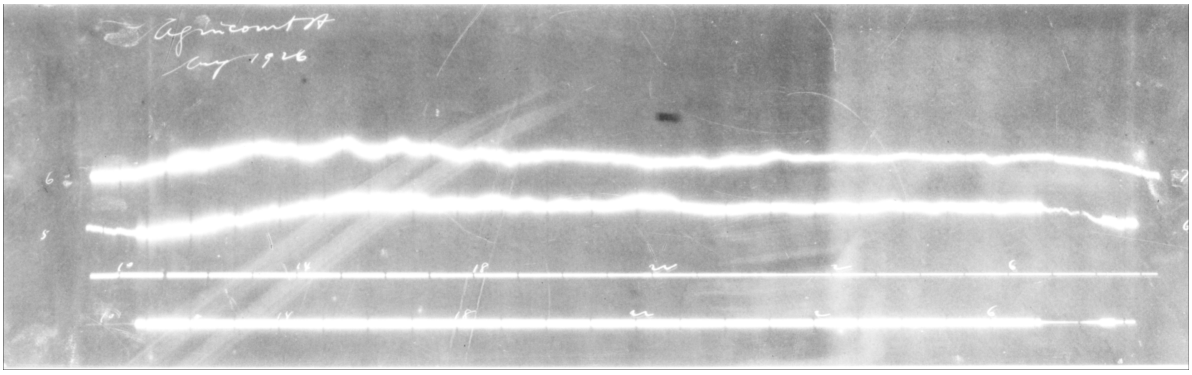

Figure 5: Bright1: Considered to be bright

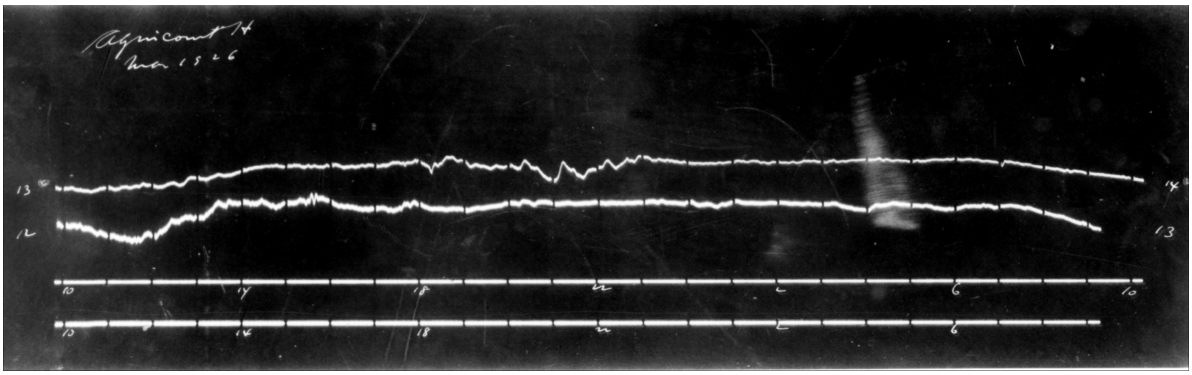

Figure 6: Bright2: Considered to be a good contrast

The program successfully returned an error, and was verified with a calendar from 1865 that there us no date February 29th in that year.

Arranging the successful conversions into an array consisting of columns for each of the cases, AGC-D, AGC-H, AGC-V, AGC-D V and H, TOR-D, TOR-H, TOR-V, and TOR-D V and H; Storing a tuple of (Start Date, End Date). It was then possible to rearrange the items in the column by their start date.

A problem arose here, as we needed to keep the start and end date of the directory the original image came from. This was not easy as some of the directories contained more then one Axis from a specific day, therefore this directory start and end date had to be added to any columns that were appended by the images in that directory.

Now starting at the first element in a row, each End Date was compared to the next Start Date. The advantage of using the datetime import is, subtracting two datetime objects gives the difference in days between them, which is the gap of the images we don't currently have. This data is being sent to NRCAN to obtain these missing images so we can start the work on making the continuous time series in the future.

Brightness Testing

A future goal put up by Mark Weygang in his masters thesis was to determine a way of automating the brightness part of TIS. Currently the user would have to manually set if an image we were digitizing was bright **Figure: Bright1** or could be considered to be okay/not bright **Figure: Bright2**

The first thing that needed to happen was to establish a baseline for what could be considered to be bright. Taking a sample of 392 images, We created a vector of 1 or 0 based off of a decision we made if an image was bright or not using a list of criteria

```
Call:
glm(formula = opinion1_392 ~ standard, family = binomial, data =
forlm)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4500  -0.7780   0.3331   0.7988   1.9759

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.7743     0.3686  -7.526 5.23e-14 ***
standard     51.9169     5.8885   8.817  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 514.59  on 390  degrees of freedom
Residual deviance: 382.14  on 389  degrees of freedom
AIC: 386.14

Number of Fisher Scoring iterations: 5
```

Figure 7: Bright3: Regression Summary

- Is the image border over exposed
- Is the traces lost because of whiteness
- Is there an over exposed part to the picture.

If two or more of these were true, we answered yes to the image being bright.
Once this vector was established, there was a need to find a way of predicting the outcome; A logistic regression model was used for this, as it gave us the probability of the image being bright. The response(our decisions about the 392 images) was modeled by:

$$\frac{\text{The pixels in the image above 0.8}}{\text{Total number of pixels}}$$

See **Figure: Bright3** for the results from the regression model. As one can see we have a significant p value for the slope of model aswell as passing the assumptions of the linear model