Abertay University

BatchExifVisualizer Development and Evaluation

Developing a tool to automate batch image location parsing and visualization.

Alex McNaughton

CMP320: Advanced Ethical Hacking

2022/23

.

.

Abstract

Exif image data is a vital tool for digital forensics researchers to obtain information about an image that would not be possible to obtain otherwise such as the camera the image was taken on, the settings for the camera, and the image location. Knowing the time and location of an image makes it valuable to investigators as it can establish an approximate location for suspects at a given time. Current metadata viewing tools allow for viewing only single images at a time, making it impossible to visualize multiple image locations at once. Therefore, development of a tool that can visualize multiple image locations at once would be beneficial to investigators.

Testing of such a tool would require creation of a set of images to test against, this was done using an android phone with location services enabled to capture around a hundred images with location data for each image. Testing the tool on this set of images revealed that many of the images did not have location data saved to them, however most of them were available to be converted into an output csv file and a map with plot points.

Development of such a tool was undertaken due to the lack of a readily available alternative, resulting in the creation of the Batch Exif Visualizer (BEV for short). The Batch Exif Visualizer takes a directory of compatible images as an input, and scans through them to gather image Exif data. Once it has this data, it either collates it into a csv file for use in external mapping software, or it visualizes the data itself using the plotly express library to plot the image points onto an OpenStreetMap.

Comparing the tool to other available multi-image Exif visualizer showed that the BEV was faster at visualizing geolocation data than alternatives, however its visualizations lack the same level of clarity as other available geolocation data visualizers.

.

Contents

1 Introduction 1

1.1 Background 1

1.2 Aims 3

2 Program and Development 4

2.1 Pre Development 4

2.1.1 Creation of image dataset 4

2.1.2 Visualisation 5

2.2 Development 5

2.2.1 Explanation of code 5

2.2.2 Testing for expected output of tool 7

3 Discussion 11

3.1 Evaluation 11

3.1.1 Comparing functionality of BEV against similar tools 11

3.2 Future Work 12

4 References 13

Appendices 14

Appendix A: Contents of BatchExifVisualizer.py 14

Appendix B: Location Data of each image in GPS_dataset 16

.

1 INTRODUCTION

1.1 BACKGROUND

Photography in forensics has played a vital role in crime investigation since the inception of the field, and in the modern day the importance of its role has expanded from a purely evidential gathering method to the use of digital photograph data in digital forensics. EXIF image data can provide an incredible amount of information about a given image, including the device the image was taken on, the settings of the device, and , if appropriate hardware is available, the GPS coordinates of where the image was taken. This data is invaluable to digital forensic investigators as it can provide evidence of the location of a suspect, and the use of a suspects device to perform illegal acts.

The use of metadata in court cases is not purely speculative, and has been used in court cases as damning evidence to decide the outcome of a case. In Kairos Shipping Ltd & Anor v Enka & Co LLC & Ors, the main evidence for inspection of a vessel was photographs confirmed to be legitimate by their metadata. On the 3rd April 2013, the Atlantik Confidence sunk in the Gulf of Aden, after slowly filling with water for 4 days after being abandoned by her crew due to a fire in the engine room. Inspection of the vessel was extremely limited, and the timeline of compartments filling with water was unclear to naval architects involved with the case. This was until various photographs of the vessel were submitted to evidence of the ship before it sank, with EXIF data across all of them confirming the timeline of events. These photographs were "the most important evidence for the naval architects to study in forming their views as to the compartments which had flooded and in what sequence" (Royal Court of Justice, 2016). Use of this evidence eventually lead to the Judge deciding that the sinking of the ship came as no accident. Such importance placed on the use of EXIF data highlights its importance in real life investigations and its legitimacy as digital evidence in court cases.

However, EXIF data is susceptible to manipulation like most forms of digital evidence. There are several tools available to edit image metadata, even some available online, but the act of editing image data is an added effort that may cyber criminals may forget to apply, in some cases "In a digital data-heavy investigation … you may be recovering tens of thousands of images, and in these situations, manipulation of data ahead of time tends to be forgotten" (CaseGuard, 2019). Lack of knowledge of Exif data combined with the time required to manually edit a large amount of images' metadata makes it reasonable to give higher importance as evidence to a large number of photos which all have similar metadata properties, as the patterns generated by multiple photos together give a more solid ground as evidence than just one photo alone.

However, most popular photo metadata viewers available do not support the use of multiple metadata sources, making it hard to visualise hundreds or thousands of photos at once. The most popular multi Exif viewer, Pic2Map, is a solely online service that offers no offline alternative, making it difficult to use as an investigative tool. This difficulty is credited to the fact that Pic2Map offers no privacy in regards to image viewing, stating in their own privacy policy that “Each image is still accessible via its own URL. Public photos/albums will be publicly viewable if featured by our editors and may be indexed by search engines.” (Pic2Map, 2023) Uploading evidence to such a service is therefore impossible as it is unlikely that it will be securely saved if the service is used. Therefore, there should be a tool that enables the visualization of multiple images Exif information that is available offline that processes the information on the hosts computer.

1.2 AIMS

- Develop a tool that converts multiple images location data into a more easily visualized medium.
- Evaluate the effectiveness of the tool as a metadata visualizer and as a tool for digital forensic specialists.

2 Program and Development

2.1 Pre Development

Before development of the tool could begin, there were several elements required for the testing and development of the tool: Firstly, a dataset of images with GPS location data was required in order to effectively test multiple image EXIF data sources being used. Secondly, the method of visualisation needed to be decided on so that development of the visualisation feature could start.

2.1.1 Creation of image dataset

Finding a suitable dataset for testing the tool proved difficult as there were no large sets of images with their metadata still intact. In order to test the tool against a large number of images, a set of images would have to be made.

The type of dataset required for testing such a tool should align with the aims of the project, mainly to evaluate the use of the tool in an investigative light. Therefore, the dataset should simulate the type of data captured from an average image capturing device. To achieve this, an android phone was used to take multiple images at various locations around Dundee, Scotland so that the phone could write GPS EXIF data to each image and produce multiple images with GPS data pointing to different locations around the city. The resulting dataset consists of 92 images with varying levels of GPS data to accurately simulate the type of EXIF data found in real investigations. The geolocation data of each image in the set can be found in appendix B, along with the images that did not capture image data.

Figure 1: An example of the type of image contained in the dataset

2.1.2 Visualisation

Visualisation of the data collected could be performed in various ways, with varying help from outside libraries. Google Earth Pro supports the importing of csv tables into its map to display multiple different points on the map at once. This would allow for a highly interactable experience and allow for further analysis of data through Google Earth Pro's internal tools, however this requires the user to have to import the table into google earth, and can take a considerable amount of time dependent on the size of the table. However, it would be beneficial for the tool to support converting its data into a csv table for use in Google Earth Pro, due to the software's prevalence and negligible cost.

Despite support through Google Earth, there should still be an option to visualize the location of each photo on a map more seamlessly, as Earth Pro's import feature requires extra work to visualise the collected data. The python library Plotly Express allows for easy visualisation of multiple data types, including location points using an integrated OpenStreetMap map. Using the features supplied in this library should allow for seamless visualisation of the data generated.

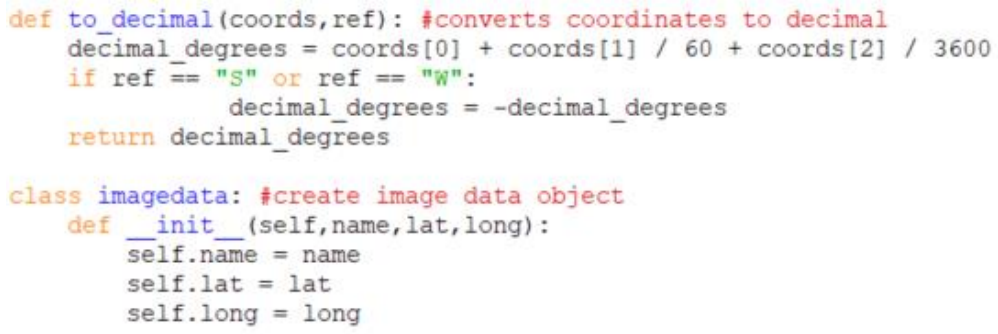
2.2 DEVELOPMENT

After developing the set of images for testing as well as deciding the method of visualization. Development of the Batch Exif Visualizer could begin. The result of development produced a single python script that handles both the gathering of image metadata and the visualization functions that are required to complete the aims set out.

2.2.1 Explanation of code

The expected functionality of the script and its code are as follows:

Firstly, the script requires name of a directory that is in the same directory as the script as an input, and either (or both) visualization flags so it can decide which visualizations to supply. Once it has this information, it defines a function and a class that will be used later on during the image data processing process.



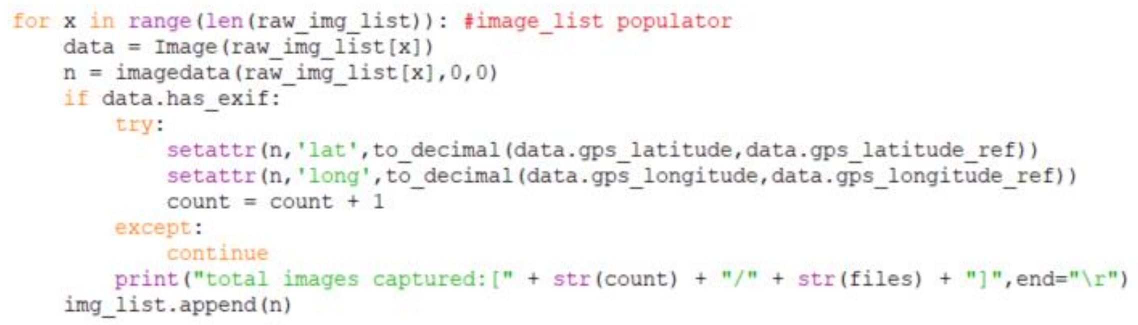
```
def to_decimal(coords,ref): #converts coordinates to decimal
    decimal_degrees = coords[0] + coords[1] / 60 + coords[2] / 3600
    if ref == "S" or ref == "W":
                decimal_degrees = -decimal_degrees
    return decimal_degrees

class imagedata: #create image data object
    def __init__(self,name,lat,long):
        self.name = name
        self.lat = lat
        self.long = long
```

Figure 2: The function and class defined after arguments have been given

The function to_decimal is used to convert degrees, minutes and seconds coordinates into decimals. This function is required as the Exif python library returns an images geolocation in degrees format, when the visualization functions of both google earth and Plotly need decimal coordinates. The class imagedata is used to store image data for a single image once it is processed into the tool.

Once these are defined, the tool traverses to the directory specified and stores the name of every image in a list called raw_image_list that is of type JPEG, this is so that the tool only collects the name of images that will potentially have geolocation data.

Once raw_image_list has been populated with every JPEG in the directory. A new list is created called img_list that will store a list of imagedata objects that contain the name, latitude and longitude of every jpeg in the directory. This list if populated using the following loop:

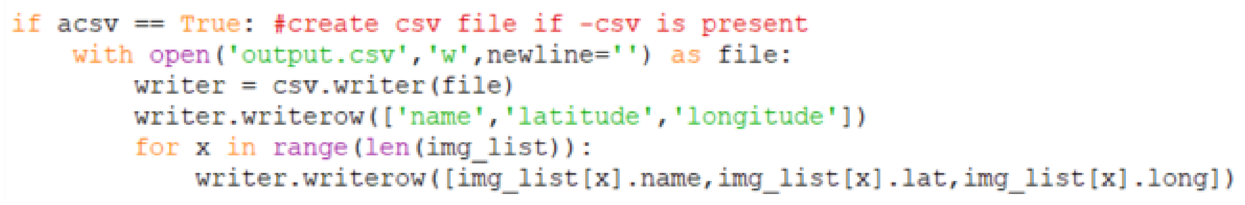


```
for x in range(len(raw_img_list)): #image_list populator
    data = Image(raw_img_list[x])
    n = imagedata(raw_img_list[x],0,0)
    if data.has_exif:
        try:
            setattr(n,'lat',to_decimal(data.gps_latitude,data.gps_latitude_ref))
            setattr(n,'long',to_decimal(data.gps_longitude,data.gps_longitude_ref))
            count = count + 1
        except:
            continue
        print("total images captured:[" + str(count) + "/" + str(files) + "]",end="\r")
    img_list.append(n)
```

Figure 3: image_list populator loop

In this loop, the script uses the raw_img_list to look up the name of an image and store its data in the data variable, it then creates an imagedata object with the image name as the name, and two placeholder latitude and longitude values at 0,0. Then, it checks if the given image has Exif data and, if successful, sets the placeholder location values to the actual values contained in the image. If data could not be found or the location data is not found, then it skips over the image and doesn't add it to the final img_list list. Finally, the loading bar is updated and the image list is appended.

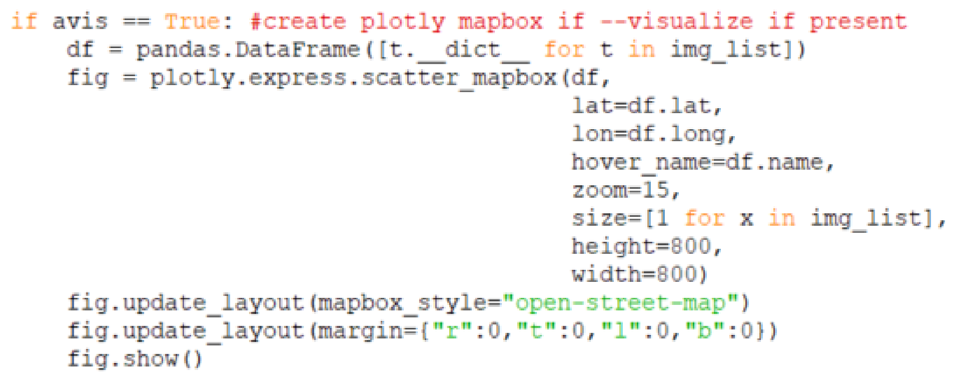
Once the image list has been populated with all available image metadata, functionality of the script is dependent on the initial arguments set at runtime. If the user used the –csv or -c flag, the script will begin creating a CSV file with the image data collected in img_list.



```
if acsv == True: #create csv file if -csv is present
    with open('output.csv','w',newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name','latitude','longitude'])
        for x in range(len(img_list)):
            writer.writerow([img_list[x].name,img_list[x].lat,img_list[x].long])
```

Figure 4: Snippet of the code used to write the scripts CSV output

If the -v or –visualize flag has been selected, the script runs the following code to facilitate visualization:



```
if avis == True: #create plotly mapbox if --visualize if present
    df = pandas.DataFrame([t.__dict__ for t in img_list])
    fig = plotly.express.scatter_mapbox(df,
                                        lat=df.lat,
                                        lon=df.long,
                                        hover_name=df.name,
                                        zoom=15,
                                        size=[1 for x in img_list],
                                        height=800,
                                        width=800)
    fig.update_layout(mapbox_style="open-street-map")
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    fig.show()
```

Figure 5: Snippet of code used to run the Plotly visualization

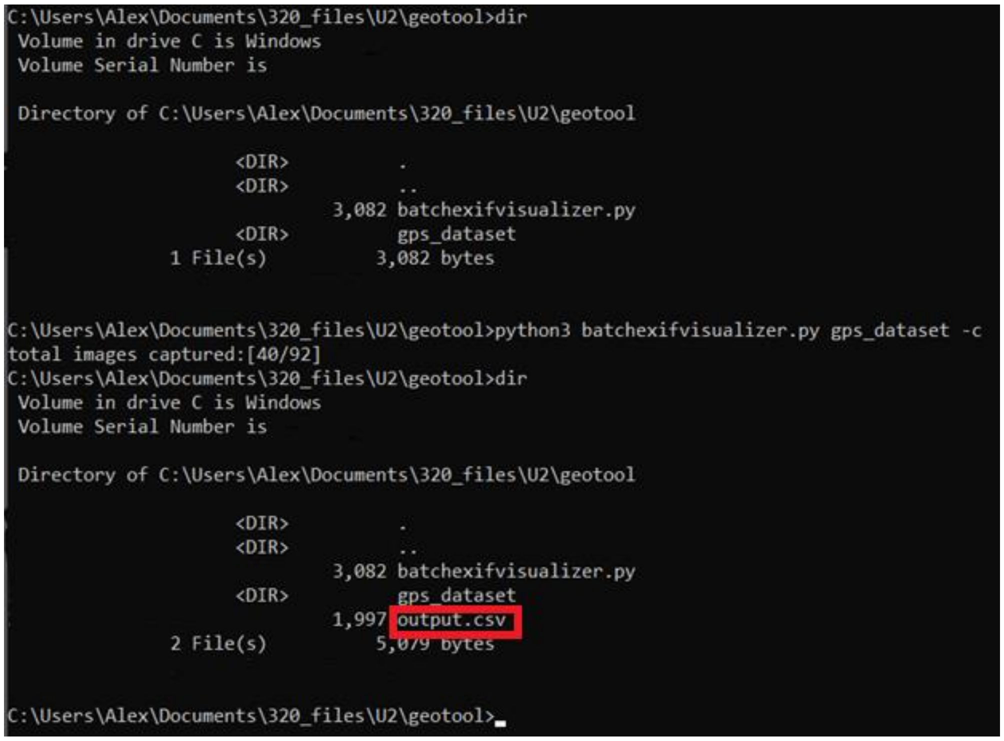
This code converts the image list into a dataframe, then generates a map box using the data stored in the dataframe. This mapbox is visualized in a web browser page that has the ability to save sections of the map as an image . For more information, see 2.2.2.2 Evaluation of visualization through Plotly. The full code can be found in Appendix A.

2.2.2 Testing for expected output of tool

In order to confirm that the developed BEV tool works, the output of each visualization method was performed using the test image data to confirm that the tool could effectively visualize all available GPS Data.

2.2.2.1 *Evaluation of CSV output*

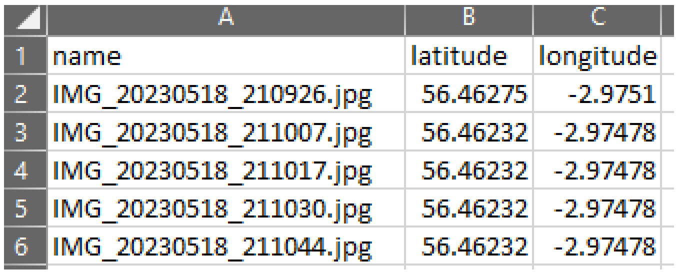
The use of the csv output of BEV is designed to function with Google Earth Pro's CSV import feature, to evaluate if this functions appropriately, the test set of images were passed into the tool to generate the output CSV file



```
C:\Users\Alex\Documents\320_files\U2\geotool>dir
 Volume in drive C is Windows
 Volume Serial Number is

 Directory of C:\Users\Alex\Documents\320_files\U2\geotool

                     <DIR>          .
                     <DIR>          ..
                              3,082 batchexifvisualizer.py
                     <DIR>          gps_dataset
               1 File(s)          3,082 bytes

C:\Users\Alex\Documents\320_files\U2\geotool>python3 batchexifvisualizer.py gps_dataset -c
total images captured:[40/92]
C:\Users\Alex\Documents\320_files\U2\geotool>dir
 Volume in drive C is Windows
 Volume Serial Number is

 Directory of C:\Users\Alex\Documents\320_files\U2\geotool

                     <DIR>          .
                     <DIR>          ..
                              3,082 batchexifvisualizer.py
                     <DIR>          gps_dataset
                              1,997 output.csv
               2 File(s)          5,079 bytes

C:\Users\Alex\Documents\320_files\U2\geotool>
```

Figure 6: Result of running the tool with the -c flag on the gps_dataset directory

Reading the information in the csv file using Excel shows that the tool successfully added every image with GPS data to the table.

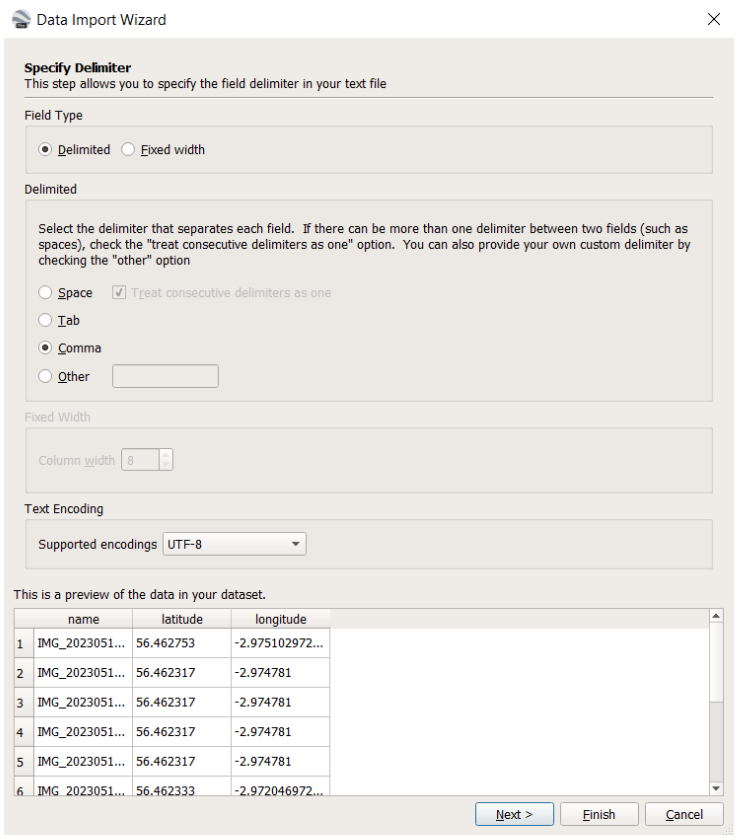


	A	B	C
1	name	latitude	longitude
2	IMG_20230518_210926.jpg	56.46275	-2.9751
3	IMG_20230518_211007.jpg	56.46232	-2.97478
4	IMG_20230518_211017.jpg	56.46232	-2.97478
5	IMG_20230518_211030.jpg	56.46232	-2.97478
6	IMG_20230518_211044.jpg	56.46232	-2.97478

Figure 7: The first few rows of output.csv

Comparing the information in this output file to the data located in each file confirmed that the location data of each image was successfully imported into the output file (each file's location data can be found in Appendix B).

After confirming the file is populated the table was imported into Google Earth Pro.



Data Import Wizard

Specify Delimiter
This step allows you to specify the field delimiter in your text file

Field Type

(●) Delimited ( ) Fixed width

Delimited

Select the delimiter that separates each field. If there can be more than one delimiter between two fields (such as spaces), check the "treat consecutive delimiters as one" option. You can also provide your own custom delimiter by checking the "other" option

( ) Space [✓] Treat consecutive delimiters as one
( ) Tab
(●) Comma
( ) Other

Fixed Width

Column width 8

Text Encoding

Supported encodings UTF-8

This is a preview of the data in your dataset.

	name	latitude	longitude
1	IMG_2023051...	56.462753	-2.975102972...
2	IMG_2023051...	56.462317	-2.974781
3	IMG_2023051...	56.462317	-2.974781
4	IMG_2023051...	56.462317	-2.974781
5	IMG_2023051...	56.462317	-2.974781
6	IMG_2023051...	56.462333	-2.972046972...

Figure 8: output.csv is imported into Earth Pro using the default settings

After all the data is imported, Earth Pro displays all the locations of each image as a point on its map:

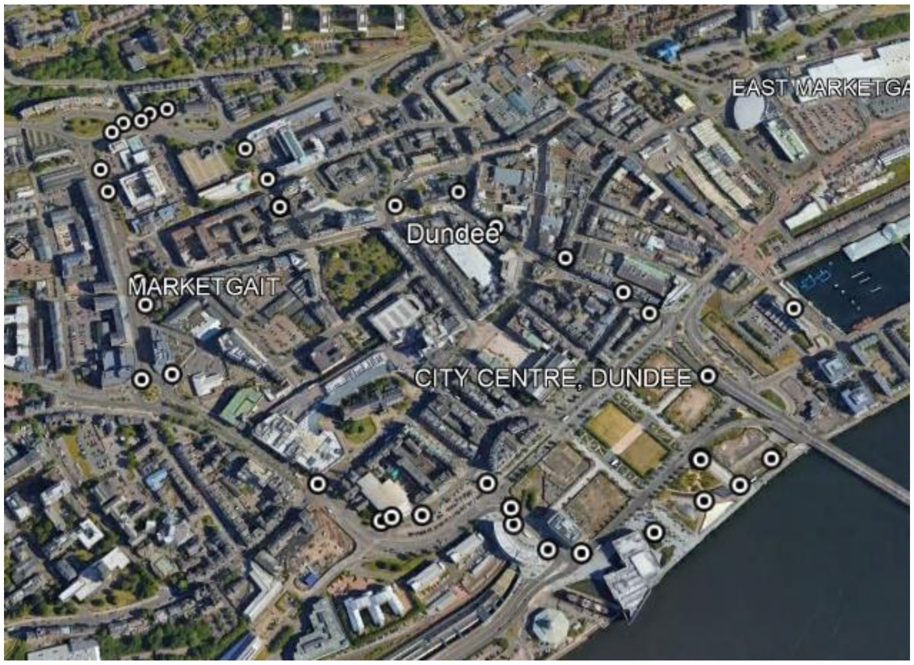
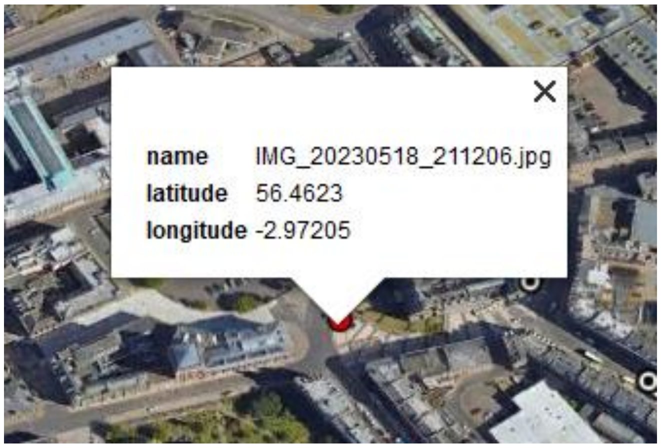


Figure 9: Resulting map after importing the data from output.csv



name IMG_20230518_211206.jpg
latitude 56.4623
longitude -2.97205

Figure 10: Clicking on a point reveals the name of the file

2.2.2.2 Evaluation of visualization through Plotly

Testing the plotly map box visualization was done by running the script on the test data with the -v flag. Doing this resulted in the script opening an instance of the default browser with a map box inside, with the points gathered plotted on the map. The interface allowed for the easy exporting of the map by simply pressing a button in the interface to save a image.

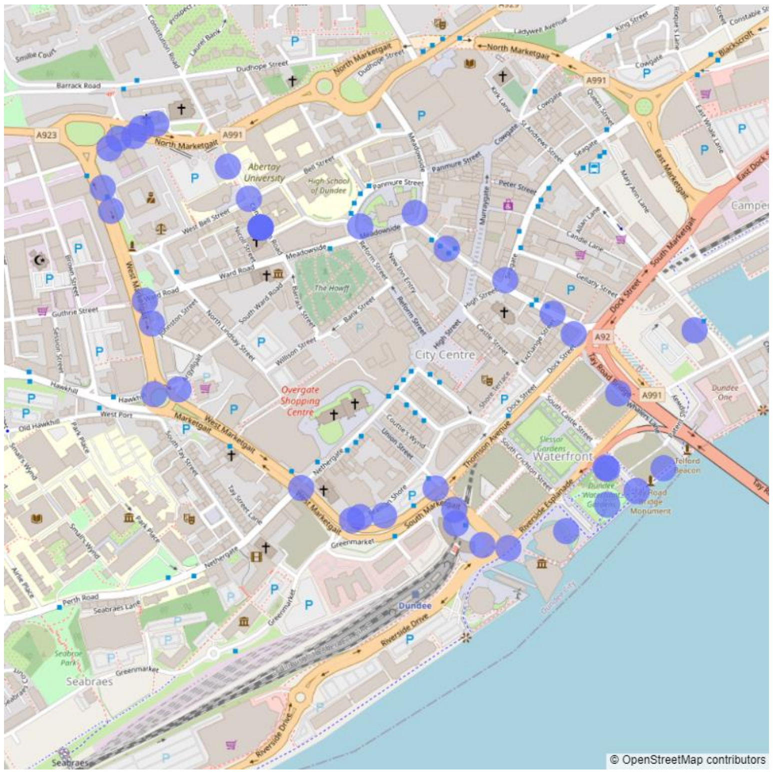


Figure 11: Output of Plotly Map Box using in-built image output function

3 DISCUSSION

3.1 EVALUATION

Through testing of the BEV, we can evaluate the functionality of the tool as a Exif data viewer. The BEV provides information about multiple different images at once such as their location and their name, making it easy for its users to identify the locations of multiple images at once. This functionality achieves the aim of visualizing multiple images in one medium, however its effectiveness as a tool for digital forensic specialists is still up for question. One of the reasons for undertaking this project was because of the tendency for large amounts of electronic evidence to have their metadata untampered in investigations, making it more likely for multiple images to confirm a date and location than just one. This tendency can be exploited through the use of the BEV, as it is able to quickly sort through many images that would take forensic investigators hours to manually visualize.

While speed of processing isn't an essential part of the project, it is still important for a tool to be fast, as slower scripts reduce the productivity of digital forensic specialists. To evaluate its effectiveness as a tool for investigators, the BEV's speed against other batch image visualizer should be tested.

3.1.1 Comparing functionality of BEV against similar tools

In terms of visualization speed, the BEV is able to outperform other available batch image visualizers:

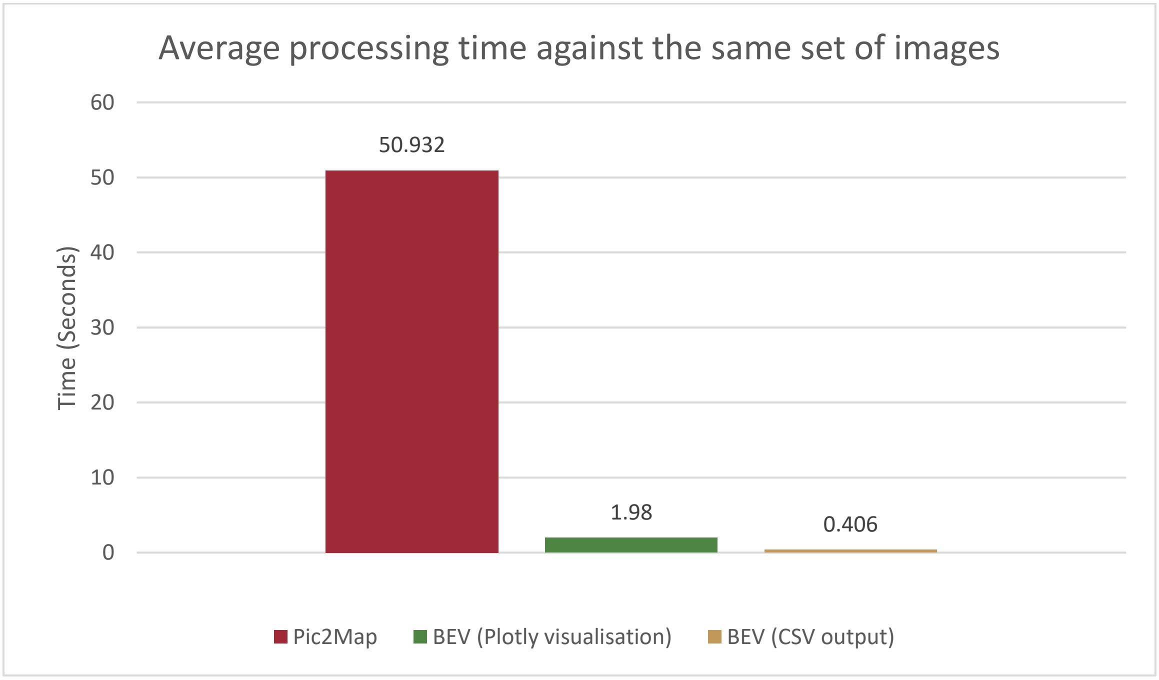


Figure 12: Graph comparing the time taken for Pic2Map and BEV to process the same amount of images

Both tools were tested multiple times to achieve an average processing time on the image set created during pre development. Results show that BEV is considerably faster at processing similar amounts of data than other available tools. While the CSV output average does not take into account the time required to put the data into the appropriate software, the plotly visualization achieves similar results to the Pic2Map in a considerably faster time. This speed is likely attributed to the need to send image data to Pic2Map through its website, compared to BEV being able to access image data much faster as it is locally available.

3.2 Future Work

While the current implemented visualization options are suitable for viewing the location of each image, there is significant work that could be undertaken on the BEV to allow it to provide more data about each image such at date of image taken and the other camera settings that are usually provided in Exif Data. In most cases for images taken on a mobile phone the name of the image usually shows the date and time that the image was taken, meaning that some date and time data would be visualized in the BEV depending on the source of the images. However, this is not always the case leading to a lack of information provided by the BEV. Implementation of these features would make it a much more useful tool for investigating Exif data and make it more useful for digital forensic purposes.

One crucial issue with the BEV is its lack of image viewing functionality within its visualizations. Currently, is a user wanted to view the contents of an image through the BEV on either visualization method, they would have to find it in the directory scanned using the name of the point as a reference. This method isn't particularly user friendly and it would be much more functional if the visualization tools allowed for previewing of image data in their visualizations.

Improved Google Earth integration would allow for increased usability of the BEV. While support for CSV files in earth is supported, there is little functionality besides displaying the name and location of each point. To improve functionality, the use of the KML file format could have been implemented to provide further ease of use in google earth such as being able to preview which image is at each point, rather than having to search for the image using the name provided.

4 References

CaseGuard, 2019. *Digital Evidence EXIF Data.* [Online]
Available at: https://caseguard.com/articles/digital-evidence-exif-data/
[Accessed 2023].

Pic2Map, 2023. *Privacy Policy.* [Online]
Available at: https://www.pic2map.com/privacy.html
[Accessed 2023].

Royal Court of Justice, 2016. *Kairos Shipping Ltd & Anor v Enka & Co LLC & Ors.* [Online]
Available at: http://www.bailii.org/ew/cases/EWHC/Admlty/2016/2412.html

APPENDICES

APPENDIX A: CONTENTS OF BATCHEXIFVISUALIZER.PY

```
import os,csv,argparse,pandas,plotly.express,PIL,time
from exif import Image

#argument handling

start_time = time.time()

parser = argparse.ArgumentParser(description="converts location data from multiple images into easy to
parse formats")
req_args = parser.add_argument_group('required arguments')
#required arguments
req_args.add_argument("path", help="Path to directory to be scanned for images")
#optional arguments
req_args.add_argument("-c","--csv", help="create csv file",action='store_true')
req_args.add_argument("-v","--visualize", help="create an interactable and savable map using
plotly",action='store_true')

args = parser.parse_args() #parse argumentsdir

path = os.path.dirname(os.path.abspath(__file__)) + "\\" + args.path
acsv = args.csv
avis = args.visualize

if acsv == False and avis == False: #exit if no visualization selected
  print("Neither visualisation option selected. Quitting...")
  quit()

def to_decimal(coords,ref): #converts coordinates to decimal
  decimal_degrees = coords[0] + coords[1] / 60 + coords[2] / 3600
  if ref == "S" or ref == "W":
        decimal_degrees = -decimal_degrees
  return decimal_degrees

class imagedata: #create image data object
  def __init__(self,name,lat,long):
    self.name = name
    self.lat = lat
    self.long = long


os.chdir(path)
```

```
raw_img_list  = os.listdir() #list of all files in directory
raw_img_list = [a for a in raw_img_list if a.endswith('jpg')] #only include jpgs

count = 0 #count num of images tagged
files = len(next(os.walk(path))[2]) #number of files in directory

img_list = [] #list of image objects

for x in range(len(raw_img_list)): #image_list populator
    data = Image(raw_img_list[x])
    n = imagedata(raw_img_list[x],0,0)
    if data.has_exif:
        try:
            setattr(n,'lat',to_decimal(data.gps_latitude,data.gps_latitude_ref))
            setattr(n,'long',to_decimal(data.gps_longitude,data.gps_longitude_ref))
            count = count + 1
        except:
            continue
        print("total images captured:[" + str(count) + "/" + str(files) + "]",end="\r")
    img_list.append(n)

os.chdir(os.path.dirname(os.path.dirname(os.path.abspath(__file__)))) #change directory back to file

if acsv == True: #create csv file if -csv is present
    with open('output.csv','w',newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['name','latitude','longitude'])
        for x in range(len(img_list)):
            writer.writerow([img_list[x].name,img_list[x].lat,img_list[x].long])

if avis == True: #create plotly mapbox if --visualize if present
    df = pandas.DataFrame([t.__dict__ for t in img_list])
    fig = plotly.express.scatter_mapbox(df,
                                    lat=df.lat,
                                    lon=df.long,
                                    hover_name=df.name,
                                    zoom=15,
                                    size=[1 for x in img_list],
                                    height=800,
                                    width=800)
    fig.update_layout(mapbox_style="open-street-map")
    fig.update_layout(margin={"r":0,"t":0,"l":0,"b":0})
    fig.show()

print("\nfinished in " + str(int((time.time() - start_time) * 1000)) + "ms")
```

APPENDIX B: LOCATION DATA OF EACH IMAGE IN GPS_DATASET

File Name	Latitude	Longitude
IMG_20230518_210849.jpg	N/A	N/A
IMG_20230518_210926.jpg	56.46275	-2.9751
IMG_20230518_211007.jpg	56.46232	-2.97478
IMG_20230518_211017.jpg	56.46232	-2.97478
IMG_20230518_211030.jpg	56.46232	-2.97478
IMG_20230518_211044.jpg	56.46232	-2.97478
IMG_20230518_211121.jpg	N/A	N/A
IMG_20230518_211206.jpg	56.46233	-2.97205
IMG_20230518_211251.jpg	N/A	N/A
IMG_20230518_211320.jpg	56.46254	-2.97055
IMG_20230518_211351.jpg	N/A	N/A
IMG_20230518_211421.jpg	56.462	-2.96967
IMG_20230518_211508.jpg	N/A	N/A
IMG_20230518_211534.jpg	N/A	N/A
IMG_20230518_211540.jpg	56.46152	-2.96807
IMG_20230518_211658.jpg	56.46101	-2.96677
IMG_20230518_211723.jpg	N/A	N/A
IMG_20230518_211741.jpg	56.4607	-2.96619
IMG_20230518_211813.jpg	N/A	N/A
IMG_20230518_211847.jpg	56.46076	-2.96287
IMG_20230518_211912.jpg	56.4598	-2.96497
IMG_20230518_211922.jpg	N/A	N/A
IMG_20230518_211952.jpg	N/A	N/A
IMG_20230518_212023.jpg	N/A	N/A
IMG_20230518_212051.jpg	N/A	N/A
IMG_20230518_212119.jpg	56.45867	-2.96372
IMG_20230518_212144.jpg	N/A	N/A
IMG_20230518_212202.jpg	56.45832	-2.96446
IMG_20230518_212221.jpg	N/A	N/A
IMG_20230518_212245.jpg	56.45866	-2.96527
IMG_20230518_212258.jpg	56.45868	-2.96531
IMG_20230518_212316.jpg	N/A	N/A
IMG_20230518_212350.jpg	N/A	N/A
IMG_20230518_212413.jpg	56.45812	-2.96527
IMG_20230518_212451.jpg	N/A	N/A
IMG_20230518_212513.jpg	56.45771	-2.96639
IMG_20230518_212534.jpg	N/A	N/A
IMG_20230518_212558.jpg	N/A	N/A
IMG_20230518_212620.jpg	56.45745	-2.96798
IMG_20230518_212640.jpg	N/A	N/A

IMG_20230518_212707.jpg	56.4575	-2.9687
IMG_20230518_212834.jpg	56.45783	-2.96942
IMG_20230518_212852.jpg	56.45804	-2.96947
IMG_20230518_212904.jpg	N/A	N/A
IMG_20230518_212917.jpg	N/A	N/A
IMG_20230518_212930.jpg	56.45836	-2.96998
IMG_20230518_212957.jpg	N/A	N/A
IMG_20230518_213029.jpg	N/A	N/A
IMG_20230518_213044.jpg	56.45795	-2.97139
IMG_20230518_213104.jpg	56.45793	-2.97205
IMG_20230518_213119.jpg	56.45788	-2.97227
IMG_20230518_213138.jpg	N/A	N/A
IMG_20230518_213213.jpg	N/A	N/A
IMG_20230518_213241.jpg	56.45837	-2.97367
IMG_20230518_213315.jpg	N/A	N/A
IMG_20230518_213340.jpg	N/A	N/A
IMG_20230518_213405.jpg	N/A	N/A
IMG_20230518_213427.jpg	N/A	N/A
IMG_20230518_213448.jpg	N/A	N/A
IMG_20230518_213505.jpg	N/A	N/A
IMG_20230518_213542.jpg	N/A	N/A
IMG_20230518_213556.jpg	N/A	N/A
IMG_20230518_213614.jpg	N/A	N/A
IMG_20230518_213636.jpg	56.45986	-2.97703
IMG_20230518_213651.jpg	56.45979	-2.97771
IMG_20230518_213715.jpg	N/A	N/A
IMG_20230518_213744.jpg	N/A	N/A
IMG_20230518_213816.jpg	56.46086	-2.97777
IMG_20230518_213837.jpg	N/A	N/A
IMG_20230518_213900.jpg	N/A	N/A
IMG_20230518_213911.jpg	56.4612	-2.97796
IMG_20230518_213933.jpg	N/A	N/A
IMG_20230518_213953.jpg	N/A	N/A
IMG_20230518_214029.jpg	N/A	N/A
IMG_20230518_214057.jpg	N/A	N/A
IMG_20230518_214107.jpg	56.46257	-2.97889
IMG_20230518_214124.jpg	N/A	N/A
IMG_20230518_214134.jpg	56.46292	-2.97912
IMG_20230518_214154.jpg	N/A	N/A
IMG_20230518_214217.jpg	N/A	N/A
IMG_20230518_214230.jpg	56.46352	-2.97894
IMG_20230518_214246.jpg	56.46366	-2.97868
IMG_20230518_214303.jpg	56.4637	-2.97825

IMG_20230518_214320.jpg	N/A	N/A
IMG_20230518_214334.jpg	56.46382	-2.97806
IMG_20230518_214356.jpg	56.46389	-2.97764
IMG_20230518_214405.jpg	N/A	N/A
IMG_20230518_214421.jpg	N/A	N/A
IMG_20230518_214438.jpg	N/A	N/A
IMG_20230518_214456.jpg	N/A	N/A
IMG_20230518_214511.jpg	N/A	N/A
IMG_20230518_214542.jpg	56.46324	-2.97568