CPSC 42800 Project 3: P-GPSExtractor

Table of Contents

1. Overview

In this project you will implement and test code to scan a directory of image files and attempt to extract GPS information from any EXIF tags in those images.

This code could support a forensic investigation in which it is important to determine the time and location at which one or more photographs were taken.

Completing this project will provide valuable experience in working with Python dictionaries and classes. This project also has an element of *offensive security*; you will write code that someone who was attempting to obstruct a forensic investigation, or "cover their tracks", might write. This will give you experience in the valuable skill of thinking from the adversary's point of view.

2. Detailed Requirements

The project consists of 3 major components, which need to be completed in order.

2.1. Getting the book code to run (50 points)

Your first job is to turn the code printed on pages 151-158 of *Python Forensics* into a working **Python 3** program.

There may be bugs or incompatibilities in the book's code; it is your responsibility to track these down and fix them. structure. This will require you to carefully **read and understand the code** to determine its intended operation.

Note: The provided code for this project exhibits a greater degree of *modular decomposition* than the previous projects—that is, the code and classes are broken out into more separate files. This is a good thing. However, the book's code listing does not show where one code file ends and the next one begins! Thus, it is your job to figure this out by reading the code, and to determine what to name the code files based on the 'import' statements used in the code.

Note 2: Since we are using the newer "Pillow" fork of the Python Imaging Library, the capitalization of some class, object, and function names will be different from what is shown in the book. To figure this out, you can consult the code written in class as well as the online documenatation for the Pillow library.

The book's code has some significant bugs, a couple of which I point out here to make our lives a little easier.

Bugfix 1: The _CSVWriter class uses a non-existent log object. To solve this, have your top-level script pass its `oLog` object to the _CSVWriter constructor. Now you will be able to use that log object in the class methods.

Bugfix 2: The call to `writeCSVRow` in the top-level script does not match the arguments of the function defined in the class `_CSVWriter`. To fix this, you should, at least initially, remove the two 'altitude' entries from the code in the _CSVWriter class.

Bug-you-fix 3: The logic in the function `ExtractGPSDictionary()` is buggy—the function may return a "wrong-shaped" value. Specifically, if there is an image which *does* have EXIF data but *does not* have the GPS tags, the logic will "fall through" and the function will return `None` by default, causing an error with the caller, which expects `ExtractGPSDictionary` to return a pair of two return values. Find a good way to fix this. Testing this will require image files with no GPS data, some of which I provide you in the test archive.

Update 4: Newer versions of the Pillow library return the latitude and longitude degree, minute, and second values as a floating-point number, where older versions returned a *rational pair* (numerator, denominator). For this reason, the code for the `ConvertToDegrees` function can be simplified to what we showed in class.

You must *test your program sufficiently* to ensure that it works with all options and is as bug-free as possible. This includes inspecting the output files and log files.

Your score for this part will be based on:

- The code running correctly as submitted, including all combinations of command line options, with correct output.

2.2. Code Improvements (20 points)

You are to design and implement the following extensions to the code given in the book:

- **Add altitude (10 pts).** Add back the two fields that you removed from the CSV writer code, 'Alt Ref' and 'Altitude', and modify the rest of the code to extract the corresponding fields from the GPSInfo dictionary *if they exist* and pass them to the CSVWriter class. The CSV Writer should write the altitude value (if it exists) to the .csv file, correctly formatted as meters.

  Note that adding altitude will require handling a new case of possible missing data, in addition to the bugfix mentioned above. If an image contains latitude and longitude GPS information but not altitude, a row should still be added to the CSV file, just with blank entries for the altitude. (This can be done by passing an empty string, "", to writeCSVRow.)

- **Add clickable link to map the location (10 pts).** Add an additional column to your CSV output, named "Map Link". The field should contain a Google Maps URL constructed from the image's latitude and longitude, so that it can be copied or clicked on to open Google Maps at the image location.

Your score for this part will be based on correctly implementing these improvements, especially the robustness of your code to missing data conditions.

2.3. Malicious modification script (30 points)

For the final portion of the project, you will play the "bad guy", developing a program using the **piexif** library (*not Pillow*) that will maliciously modify the latitude, longitude, and timestamp values of an existing photograph. This will demonstrate how the data obtained from EXIF tags is not necessarily trustworthy.

Your program can be a single script file that takes two command-line arguments: an image file as input, and an output filename for writing the tampered image. It should output an image with falsified latitude, latitudeRef, longitude, longitudeRef, and timestamp values.

Your code should work as follows for either of two cases:

- If the image has EXIF data but no GPS info tag, you should create the GPS dictionary "from scratch" with the false information.
- If the image already has a GPS info tag, you should replace its latitude, longitude and time values with the falsified ones, leaving the other entries as-is.

The false data should be *randomly generated* latitude, longitude, and time values. They should be legitimate time values and coordinates (though you don't need to check if your coordinates are in the middle of the ocean.)

This part of the project will require some additional research on how to use the piexif library to rewrite image data.

You can use the code written for the first part of the project to test that your script is modifying image files as you expect. You will need to test you script on several different image files to verify that all the cases work.

Your code will be graded on the correct handling of the three cases mentioned above, and generation of a valid image file with random latitude, longitude and time values.

3. What to Submit

You are to submit on Blackboard a single .zip archive containing the following files:

- All of the Python source code files for the pGPSExtractor program and your malicious script. I should be able to immediately run the program after extracting these. DO NOT include any bytecode (.pyc) files.
- An output CSV file showing the results of running the program on the provided image set.
- A “Readme” file describing how to run your program and any known bugs.

4. Concluding remarks

I highly recommend re-reading this document several times in the course of working on the project, to make sure you understand the requirements in detail. If there is anything less than clear about the requirements, please ask me. As always, I am available to help with debugging your code.

Have fun!

Created: 2021-10-13 Wed 08:38
Validate