Lies mich!

Verorten von Fotos mittels Trackdaten

Bernd Ragutt

Im Dezember 2025

Inhaltsverzeichnis

Der Zweck

Klassische Kameras haben ihre Qualitäten, sie haben allerdings (zumeist) keinen eingebauten GPS-Chip, der die gemachten Fotos gleich verortet. Heutige Smartphones haben einen eingebauten GPS-Chip und es gibt solche Apps, die wunderbar aufzeichnen, wo man seines Wegs gegangen ist. Oft reicht die Qualität der Fotos aus, die man mit den winzigen Objektiven der Smartphones schießen kann – aber nicht immer.

Zweck dieser Unternehmung ist, die Fotos einer Kamera mit den GPS-Daten eines Smartphones zu verheiraten.

Die Randbedingungen

Die Randbedingungen sind:

- Die Fotos der Kamera müssen im jpg-Bildformat vorliegen und sie müssen mit Standard-Exif-Daten für den Aufnahmezeitstempel versehen sein.
- Die Ortsdaten des Smartphones müssen mit einer App aufgezeichnet worden sein und müssen als Trackdaten im Standard-gpx-Format vorliegen.

Der Aufbau

Diese Python-Anwendung setzt sich aus den folgenden Teilen zusammen:

- dem Hauptskript a*dd_location_data_to_fotos_using_tracks.py*,
  den Hilfsskripten *analyse_adding_location_data_to_fotos_using_tracks.py*,
  recover_and_purge_from_backuped_origfotos.py,
  remove_backuped_origfotos.py.
- der Text-Datei *input_data_for_location_adding_to_fotos_using_tracks.csv* im csv[1]-Format, die das Hauptskript mit den notwendigen Eingabe-Daten versorgt,
- weiteren Python-Module im Unterverzeichnis *pyFiles*, von denen nur die Datei *time_zone_data.py* für den Nutzer wichtig ist,
- die Logging-Datei *Foto_GPX_Logging.log*, die vom Hauptskript im Verzeichnis dieses Hauptskriptes erzeugt wird und in die es fortlaufend Warnungen, Fehlermeldungen, aber auch nützliche Informationen für den Nutzer schreibt und
- dem Unterverzeichnis *libs*, das Zeitzonen-Daten *tzdata* für Python lokal bereitstellt; diese Daten müssen so nicht extra aus dem Python-Universum installiert werden.

1 csv = ‚Comma-Separates Values‘

Die Eingabedaten

Die Eingabedaten-Datei stellt über 6 (logische) Spalten die Nutzer-Daten bereit. Die Spalten haben keine feste Breite, das Steuerzeichen | trennt die Datenblöcke in den Zeilen voneinander. Die für das Skript notwendige Kopfzeile benennt die Spalten.

Die 1. Spalte

Die *erste Spalte* der Eingabedaten-Datei enthält Steuerinformation für das Auslesen[2] der Eingabe-Daten bereit.

Ein Rautenzeichen **#** markiert eine reine Kommentarzeile oder auch eine Zeile mit Nutzer-Daten, die aktuell *nicht* ausgelesen werden sollen; ein Plus-Zeichen **+** markiert eine Nutzer-Zeile, die vom Skript aktuell ausgewertet werden soll, auch mehrere Zeilen können mit einem **+** versehen werden.

Die 2. Spalte

Die *zweite Spalte* der Eingabedaten-Datei enthält in jeder Zeile, in Gänsefüßchen gesetzt, den Dateinamen der ‚Track'- oder Wegspur-Datei im gpx-Format, in der der Wanderer[3] Ortsdaten per GPS-Gerät oder Smartphone aufgezeichnet hat, ein Beispiel mitsamt den Trennstrichen meines csv-Formates:

▷ | "2017-10-11_12_13_-_Fremont_Older_Open_Space_Preserve.gpx" | ◁[4]

Die 3. Spalte

Die *dritte Spalte* der Eingabedaten-Datei enthält in jeder Zeile, in hochgestellten Gänsefüßchen gesetzt, den vollständigen Pfad des Verzeichnisses, in dem die Track-Datei aus der zweiten Spalte liegt, im Beispiel für Windows:

| "C:\Users\Bernd\PyCharmProjects\photos_to_track_4\Datensatz-Track-Fotos" | [5]

Solch ein Pfad lässt sich mit Windows-Bordmitteln leicht mit dem Befehl *Als Pfad kopieren* aus dem Kontextmenü des Verzeichnisses – oder auch einer Datei – ermitteln und in die Datei der Eingabedaten kopieren.

Die 4. Spalte

Die *vierte Spalte* der Eingabedatei enthält in jeder Zeile, in hochgestellten Gänsefüßchen gesetzt, den vollständigen Pfad des Verzeichnisses, in dem die Foto-Dateien im jpg-Format, die der Track-Datei aus der zweiten Spalte zugeordnet sind, liegen, ein Beispiel:

▷ | "C:\Users\Bernd\PyCharmProjects\photos_to_track_4\Datensatz-Track-Fotos" | ◁

Gehören zu dem einen Track weitere Fotos in weiteren Verzeichnissen, so muss für jedes weitere Foto-Verzeichnis eine neue vollständige Zeile erzeugt werden. Dabei werden Fotos, die das Skript *nicht* der aktuellen gpx-Datei zuordnen kann, beiseite gelassen; die Zuordnung erfolgt über die Aufnahmezeit[6] des Fotos; liegt diese Aufnahmezeit *nicht*

2 Leerzeichen werden überlesen.
3 Stellvertretend für all die, die ihre Unternehmungen per GPS mitloggen.
4 Die weiß gefüllten Dreiecke vorne und hinten sollen nur den weißen Leerraum kenntlich machen.
5 Ohne ein weiteres \ am Ende des Pfades!
6 Konvertiert in die UTC-Zeit. UTC ist die Zeit der GPS-Daten.

innerhalb des Zeitraumes, in dem die Wegspur aufgezeichnet wurde, wird das Foto außer Acht gelassen, anders herum werden Fotos genau nur dann verortet, wenn diese Aufnahmezeit innerhalb des Trackzeitraumes liegt.

Die 5. Spalte

Die *fünfte Spalte* der csv-Datei enthält in jeder Zeile ein Kürzel der Zeitzonenkennung nach IANA[7]. Das Kürzel besteht aus 4 Buchstaben, das *nicht* in Gänsefüßchen oder Hochkomma gesetzt wird, hier ein Beispiel für die Zeitzonenkennung Europe/Berlin:

▷ | EuBl | ◁

Wird dieses Spaltenfeld in der Form ▷ | | ◁ leer gelassen, so wird im Weiteren mit der Kennung Europe/Berlin gerechnet.

Augenblicklich sind von mir die folgenden Kürzel für Zeitzonen definiert:

EuRo	*für*	Europe/Rome
AtMr	*für*	Atlantic/Madeira
EuLb	*für*	Europe/Lisbon
AmLA	*für*	America/Los_Angeles
EuBl	*für*	Europe/Berlin
EuMd	*für*	Europe/Madrid
EuMt	*für*	Europe/Malta
EuLo	*für*	Europe/London
EuZa	*für*	Europe/Zagreb

Benötigt der Nutzer für einen Ort eine weitere Zeitzone, so erhält er über den Aufnahmeort der Fotos mithilfe der Internetseiten von https://timezone360.com/de/ die zugeordnete Zeitzonenkennung nach IANA. Ein selbstdefiniertes Kürzel aus vier Buchstaben und die Zeitzonenkennung müssen dann in der Python-Datei *time_zone_data.py* nach dem dort zu findenden Muster aufgelistet werden, ein Beispiel für die Insel Malta aus der dieser Python-Datei:

```
time_zones['EuMt']='Europe/Malta'          # Valletta, Malta
```

Das von mir gewählte Kürzel EuMt und die standardisierte IANA-Kennung Europe/Malta stehen *nur* in dieser Python-Datei in Hochkomma, damit Python diese beiden Daten als Zeichenketten interpretiert. Das Rautenzeichen # leitet einen Kommentar in Python ein.

Die 6. Spalte

Die *sechste* und *letzte* Spalte der Eingabedaten-Datei kann in jeder Zeile einen Zeitversatz der Form.

▷ | +/−(hh:mm:ss) ◁

aus Stunden, Minuten und Sekunden enthalten; dieser Zeitversatz wird bei der Verortung der Fotos über die gpx-Daten berücksichtigt und wird im Erfolgsfalle auch zu den abgespeicherten Aufnahmezeiten[8] der Fotos hinzuaddiert; einige Beispiele für den Gebrauch des Zeitversatzes:

7 Wie sie von der IANA (Internet Assigned Numbers Authority) definiert wurden.

8 Wie sie im EXIF-Datenbereich der jpg-Fotodatei festgehalten sind.

Voreinstellung, kein Zeitversatz:	▷ | +(0:0:0) ◁
30 Sekunden:	▷ | +(0:0:30) ◁
5 Minuten und 30 Sekunden:	▷ | +(0:5:30) ◁
Und noch 1 weitere Stunde dazu:	▷ | +(01:05:30) ◁

Dieser Zeitversatz wird bei all[9] denen Fotos des Foto-Verzeichnisses berücksichtigt, die selbst mit *keinen* eigenen GPS-Daten daherkommem.

Wird dieses Spaltenfeld in der Form ▷ | ◁ [10] leer gelassen, so wird im Weiteren mit dem Zeitversatz +(0:0:0), also mit keinem Zeitversatz gerechnet.

Die Python-Installation

Um die Python-Skripte starten zu können, benötigt der potentielle Nutzer eine aktuelle Python-Installation[11], herunterzuladen von *python.org*. Ich verwendete bei der Erstellung der Skripte die Version 3.14. Die Installation selbst ist unproblematisch; ich verwende aus Erfahrung immer eine lokale Installation ‚nur für den Nutzer', dann gibt es keinerlei Rechteprobleme.

Wer sich weiter mit Python beschäftigen möchte, dem empfehle ich *PyCharm* von JetBrains oder auch *Visual Studio Code* von Microsoft[12] als Entwicklungsumgebungen, dazu die Werkzeuge *TortoiseHg/Mercurial* zur Verwaltung von Versionen; alles kostet dem Hobby-Entwickler nichts.

Ich benutze in meinen Skripten drei Python-Pakete *exif*, *piexif* und *nvector*, die zusätzlich aus dem Python Package Index (PyPI) installiert werden müssen.

Benutzt man *PyCharm*, so meckert das Werkzeug, wenn ein importiertes Python-Paket nicht installiert ist – und nach einem Mausklick führt es die Installation selbstständig durch.

Visual Studio Code macht es dem Nutzer nicht so bequem, hier muss man aus dem Werkzeug-Menü mit `‚Terminal' => ‚Neues Terminal'` ein internes Terminal (Konsole) öffnen und anschließend die gleich unten aufgeführten Kommandos eingeben.

Will man einfach nur die Python-Skripte ohne weitere Python-Ambitionen ausführen, so sollte man sich zur Ausführung der Skripte eines Windows-Terminals bedienen, welches man etwa aus dem Kontext-Menü heraus mit dem Befehl `‚In Terminal öffnen'` erzeugt.

Nun zur Installation der Python-Pakete: Mit dem ersten Kommando gleich ein paar Zeilen weiter überprüft man nur, ob das Terminal die Python-Installation auch kennt, das

9 Also auch denen, die eventuell zeitlich und örtlich nichts mit den Wegspurdaten zu tun haben, die dann durch einen zu groß gewählten Zeitversatz in den Trackingzeitraum der gpx-Datei geraten könnten und so auch verortet würden!

10 Der *eine* senkrechte Strich trennt das letzte Spaltenfeld rechts vom senkrechten Strich vom vorletzten links davon ab.

11 Ich habe versucht, mit dem Python-Paket *pyInstaller* ausführbare, schlüsselfertige Programme aus den Skripten zu erzeugen. Das Python-Paket *nvector* erwies sich allerdings als unüberwindbar widerspenstig, die Erzeugung gelang nicht, selbst nicht mit der Unterstützung durch das fast allwissende KI-Werkzeug ChatGPT über einen ganzen Nachmittag.

12 Microsoft steuert mehrere Erweiterungen für Python bei.

zweite Kommando überprüft, ob der Paket-Installer pip auch verfügbar ist; die letzten 3 Kommandos installieren endlich die gewünschten Pakete.

```
py –version
py -m pip --version

py -m pip install --user exif
py -m pip install --user piexif
py -m pip install --user nvector
```

Die Skript-Ausführung

Die Installation der Python-Skripte beschränkt sich darauf, den gepackten Projekt-Ordner mit seinen Dateien und dem einen *lib*-Verzeichnis zu entpacken.

Nach der Installation von Python kann der Nutzer die Skripte im Skriptordner des Explorers durch einen Doppelklick starten - oder auch über die Windows-Konsole[13] durch die Eingabe des Skriptnamens. Die Skripte fragen vor der eigentlichen Ausführung nach, ob die Ausführung gewünscht wird; zudem warten sie nach der Ausführung auf die Eingabe eines Zeilenvorschubs über die Tastatur durch das Drücken der ‚Return'-Taste, womit verhindert wird, dass das Konsolenfenster sich nach der Skriptausführung gleich wieder schließt und somit Textausgaben auf die Konsole, insbesondere Fehlermeldungen, ansonsten nicht mehr lesbar wären.

Das Hauptskript kann problemlos mit denselben Eingangsdaten wiederholt ausgeführt werden; die Originaldateien werden dabei aus den bereits vorhandenen *‚.jpg.orig'*-Dateien wieder hergestellt. Auch die Hilfsskripte richten bei einer wiederholten Ausführung keinen Schaden an, sie tun gegebenenfalls einfach nichts.

Damit die Logging-Datei nicht endlos lang und unübersichtlich wird, empfiehlt es sich, diese ab und an zu löschen, sie wird neu erzeugt.

Ein Einsatzszenario

1. Sichern sie die Originale ihrer Fotos grundsätzlich auf einem externen Speichermedium. Wählen sie eine Trackdatei und einen zugehörigen Ordner mit Fotos im jpg-Speicherformat aus, unter den Fotos sollen auch solche Fotos sein, die nicht von der Aufnahmekamera verortet wurden; wenn sie sich dessen unsicher sind, öffnen sie bitte das Foto im Bildbetrachter *IrfanView*, klicken hier auf das Symbol *Bild-Information*, dann auf den Menüknopf *EXIF Daten** und schließlich auf *Show in OpenStreet*; sehen sie den letztgenannten Menüknopf nicht, so ist das Foto von der Kamera nicht georeferenziert worden, ansonsten wird bei einem Klick auf den Menüknopf der Aufnahmeort des Fotos auf den Landkarten von OpenStreetMap dargestellt.

2. Tragen sie ihre Daten in die csv-Eingabedatei ein. Beginnen sie mit einer Trackdatei und einem Fotoordner.

3. Starten sie das *‚analyse_...'*-Skript und prüfen sie die Textausgaben auf der Konsole oder in der Logging-Datei: Welche Fotos sind nicht verortet? Welche nicht-verorteten Fotos liegen außerhalb des Trackingzeitraumes? Müssten die letztgenannten Fotos auch innerhalb des Trackingzeitrau-

13 Im Explorer über das Kontextmenü mit ‚In Terminal öffnen' erreichbar.

mes aufgenommen worden sein? Wenn ja, müssten sie für die Verortung dieser Fotos die Aufnahmezeit der Fotos korrigieren[14], dies kann im nächsten Schritt geschehen.

4. Starten sie das ‚*add_loc...*'-Skript und prüfen sie die Textausgaben auf der Konsole oder in der Logging-Datei. Um die berechneten Ortsdaten zu überprüfen, gehen sie bitte wie folgt vor:

   a) Für Bayern öffnen sie in ihrem Internetbrowser den BayernAtlas[15] , für die weite Welt außerhalb Bayerns die OpenTopoMap[16].

   b) Klicken sie auf die gewünschte Trackdatei im Explorer und ziehen sie diese mit gedrückter Maustaste auf die Karte des Internetbrowsers.

   c) Die Logging-Datei gibt die berechnete geografische Breite und Länge eines Fotos als Zahlenpaar, etwa `48.144676, 11.416867` aus; wählen sie dieses Zahlenpaar als Ganzes aus, kopieren sie es in die Zwischenablage und fügen sie die kopierte Breite und Länge in das Suchfeld der bereits geöffneten Kartenanwendung ein.

5. Gegebenenfalls können sie den Zeitversatz in der csv-Eingabedatei ändern und das ‚*add_loc...*'-Skript erneut ausführen – bis ihre Fotos korrekt verortet sind.

6. Gehören zu ihrer Trackdatei weitere Ordner mit ortslosen Fotos, so können sie nun weitere Zeilen in der csv-Eingabedatei hinzufügen; für jeden weiteren Foto-Ordner eine eigene, vollständige Zeile, die mit einem ‚+ |' beginnt. Es spart nicht nur Logging-Ausgaben, wenn sie Zeilen mit bereits verorteten Foto-Ordner mit einem ‚# |' am Zeilenanfang auskommentieren.

7. Wollen sie (fast bis) zum Ausgangszustand zurückkehren, so können sie dieses mit dem Skript ‚*recover_...*' erreichen, es stellt die originalen Fotodateien wieder her, löscht allerdings noch nicht die Sicherungsdateien *‚...jpg.orig'*.

8. Sind sie zufrieden mit den Ergebnissen, so können sie auch die Sicherungsdateien *‚...jpg.orig'* mit dem Skript *‚remove_backuped...'* eingeschränkt endgültig löschen, eine Sicherungsdatei *abc.jpg.orig* wird nur dann gelöscht, wenn im gleichen Verzeichnis die Datei namens *abc.jpg* zu finden ist.

Die Nebenwirkungen - beherrschbar

Die Skripte arbeiten beherrschbar-destruktiv, was heißen soll, dass, wenn etwa das Hauptskript die Orts- und Zeitdaten in die Originaldatei eines Fotos schreiben will, dass es dann vorab die originale jpg-Datei in eine *Kopie* des Fotos[17] sichert; diese Kopie ist leicht erkennbar durch die Hinzufügung des Suffixes *.orig* am Ende des Dateinamens, ein Beispiel:

DSC00945.JPG => *wird gesichert in* => DSC00945.JPG.orig

Dennoch sollte der Nutzer dieser Skripte natürlich wissen, was er tut und mit Bedacht vorgehen – der allerste Schritt dazu ist die unabdingbare Datensicherung.

14 Der eingebaute Zeitmesser der Kamera ging vor oder nach oder die Zeitzone vor Ort wurde nicht berücksichtigt .

15 https://atlas.bayern.de/

16 https://opentopomap.org/ - OpenStreetMap kann leider keine Trackdateien importieren.

17 Am Rande: Die digitalen Originale der Fotos hat der vorsorgliche Fotograf – ganz unabhängig von diesen Python-Spielereien – sicherlich an einem sicheren Ort deponiert!

Die Danksagung

Ich verwende in meinen Skripten für die Verortung von Fotos im jpg-Format die folgenden externen Python-Pakete aus dem Python-Package-Index (PyPI):

- Mit der Hilfe des Python-Paketes *exif* lese ich unter anderem Zeit-Daten aus den Fotos; mit Hilfe von *piexif* schreibe ich Orts- und Zeitdaten *verlustfrei* in die EXIF-Daten des jpg-Fotos.
- Mit der Hilfe des Python-Paketes *nvector* ermittle ich die Aufnahmeorte der Fotos durch Interpolation der Koordinaten auf dem Erdellipsoid zwischen zwei Trackpunkten.
- Und nicht zu vergessen sei der Dank an die Macher der großartigen Skriptsprache Python.
- Ein Dank auch an die leidenschaftlich loggende Tochter, die mich unterstützt und nicht nur mit Testdaten aus nah und fern versorgt hat.

Die Gewährleistung