Lies mich!

Verorten von Fotos mittels Trackdaten

Bernd Ragutt

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 *add_locdata_to_fotos_for_n_tracks.py* ,
- der Text-Datei *inputdata_for_location_adding_to_fotos.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, Fehlermeldung, 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.

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.

1 csv = ‚Komma separates valides‘

Die 1. Spalte

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

Ein Gartenzaun **#** 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'-Datei im gpx-Format, in der der Wanderer[3] Ortsdaten per GPS-Gerät oder dem Smartpone aufgezeichnet hat, ein Beispiel mitsamt den Trennstrichen:

```
| "2017-10-11_12_13_-_Fremont_Older_Open_Space_Preserve.gpx" |
```

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" |
```
[4]

Solch ein Pfad lässt sich mit Windows-Bordmitteln leicht mit dem Befehl *Als Pfad kopieren* aus dem Kontextmenü des Verzeichnisses ermitteln und in die Eingabedatei kopieren.

Die 4. Spalte

Die *vierte Spalte* der Eingabedaten-Datei 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" |
```
[5]

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* innerhalb des Zeitraumes, in dem ‚getrackt' wurde, wird das Foto verworfen, anders herum werden Fotos genau nur dann verortet, wenn diese Aufnahmezeit innerhalb des Trackzeitraumes liegt.

---

2 Leerzeichen werden überlesen

3 stellvertretend für all die, die ihre Unternehmungen per GPS mitloggen

4 ohne ein weiteres \ am Ende!

5 ohne ein weiteres \ am Ende!

6 konvertiert in die UTC-Zeit. UTC ist die Zeit der GPS-Daten

Die 5. Spalte

Die *fünfte Spalte* der Eingabedaten-Datei enthält in jeder Zeile ein selbst vergebenes Kürzel aus eigener Hand 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 Ort 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 Python-Datei:

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

Kürzel und Kennung stehen nur in dieser Python-Datei in Hochkomma, damit Python diese beiden Daten als Zeichenketten interpretiert.

Die 6. Spalte

Die *sechste* und *letze* 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 zu den Aufnahmezeiten der Fotos hinzuaddiert; einige Beispiele für den Zeitversatz:

Voreinstellung, kein Zeitversatz:	| +(0:0:0)
30 Sekunden:	| +(0:0:30)
15 Minuten und 30 Sekunden:	| +(0:15:30)
Und noch 1 weitere Stunde dazu:	| +(1:15:30)

Dieser Zeitversatz wird *allen*[8] Fotos des Foto-Verzeichnisses hinzugefügt und wird somit bei der Verortung über die gpx-Daten berücksichtigt.

---

7 wie sie von der IANA (Internet Assignaten Novembers Autoritär) definiert wurden

8 Also auch denen, die eventuell zeitlich und örtlich nichts mit den Trackdaten zu tun haben, die dann durch einen größeren Zeitversatz in den Trackingzeitraum geraten könnten und so auch verortet würden!

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

Die Python-Installation

Um das Hauptskript starken zu können, benötigt der potentielle Nutzer eine aktuelle Python-Installation, 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[9], beide kosten dem Hobby-Entwickler nichts.

Ich benutze in meinen Skripten zwei Python-Pakete *exif* 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[10], so muss man sich eines Windows-Terminal bedienen, dass man etwa aus dem Kontext-Menü mit dem Befehl `‚In Terminal öffnen'` erzeugt.

Mit dem ersten Kommado ein paar Zeilen weiter überprüft man nur, ob das Terminal die Python-Installation auch kennt, das zweite Kommando überprüft, ob der Paket-Installer pip auch verfügbar ist; das dritte und das vierte Kommando installieren endlich die gewünschten Pakete.

```
py –version
py -m pip --version

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

Die Skript-Ausführung

Die Installation der Python-Skripte beschränkt sich darauf, den gepackten Projekt-Ordner mit den Skripten und dem einem ‚lib'-Verzeichnis zu entpacken.

Nach der Installation von Python kann der Nutzer das Hauptskript durch einen Doppelklick starten - oder auch über die Windows-Konsole durch die Eingabe des Skriptnamens.

Wenn man das Hauptskript in der Konsole mit dem Parameter y oder Y startet, wartet das Skript nach seiner Ausführung auf die Eingabe eines beliebigen Zeichens durch den

9 Microsoft steuert auch mehrere Erweiterungen für Python bei.

10 Dann kann man auch den mit einer Warnung verbundenen Hinweis bei der Paket-Installation ignorieren, dass eine Umgebungsvariable als ‚Path' hinzugefügt werden sollte. Weiteres und mehr unter *https://packaging.python.org/en/latest/tutorials/installing-packages/*

Nutzer über die Tastatur. Ansonsten schließt sich das Konsolenfenster nach dem Ablauf des Skriptes sofort und der Nutzer bekommt etwaige Fehlerausgaben, die nur auf die Konsole geschrieben wurden, nicht zu Gesicht.

Das Skript kann problemlos mit denselben Eingangsdaten wiederholt ausgeführt werden; bereits vorhandene ‚_gxo'-Dateien werden dabei gelöscht und einfach wieder neu erzeugt. Damit die Logging-Datei nicht endlos lang und unübersichtlich wird, empfiehlt es sich, diese ab und an zu löschen.

Keine Nebenwirkungen

Das Skript arbeitet gänzlich nicht-destruktiv, es *liest* ausschließlich - und wenn es *schreibt*, dann schreibt es die ermittelten Ortsdaten ausschließlich in Kopien der Fotos[11], die unter einem *neuen* Namen durch Hinzufügung des Kürzels „_gxo" am Ende des Namens abgespeichert werden, ein Beispiel:

```
DSC00945.JPG        =>      DSC00945_gxo.JPG
```

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 Exif-Zeitdaten aus den Fotos und schreibe Exif-Ortsdaten in die Fotos.
- Mit der Hilfe des Python-Paketes ‚nvector' ermittle ich die Aufnahmeorte der Fotos durch Interpolation der Koordinaten auf dem Erdellipsoid zwischen zwei Trackpunkten.

Ein Dank auch an die Tochter, die mich mit Testdaten aus nah und fern versorgt hat.

11 Am Rande: Die Originale der Fotos hat der Fotograf sicherlich an einem sicheren Ort deponiert.