Lies mich!

Fotos ausrichten und verkleinern

nach der kurzen Seite
mit Python

Bernd Ragutt

Im Mai 2026

Inhaltsverzeichnis

Der Zweck

Der Fotos sind meist viele an der Zahl und jedes Foto ist üppig fett an Megabytes. Werden solche Fotos in einem digitalen Reisebericht gesammelt, sprengt dieser im Nu jedes transportable Maß – der Gürtel muss also enger schnallt werden, das Übermaß an Speck der Fotos muss weg; da die Komprimierung ihre Grenzen hat, muss das Format kleiner werden.

Für die digitale Weiterverarbeitung der kleineren Formate brauchen diese allerdings weiterhin ihre Aufnahmezeit und natürlich auch den digital verwertbaren Aufnahmeort (wenn er denn vorhanden ist). Zudem sollen Fotos, die mit einem verdrehten Smartphone aufgenommen wurden und etwa auf dem Kopf stehen, korrekt ausgerichtet werden[1].

Die Randbedingungen

Die Randbedingungen sind:

- Die Fotos der digitalen Kamera müssen im jpg-Bildformat vorliegen und sie müssen mit den Standard-Exif-Daten für den Aufnahmezeitstempel versehen sein.
- Verortete Fotos wären allemal wünschenswert[2]. Gegebenenfalls sollte man sie nachverorten, wenn die Ortsdaten anderweitig zur Verfügung stehen.

Der Aufbau

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

- dem Hauptskript *manipulate_fotos_with_resizing.py*,
  dem Hilfsskript *foto_utilities.py*,
- der Text-Datei *input_data_for_resizing_fotos.csv* im csv[3]-Format, die das Hauptskript mit den notwendigen Eingabe-Daten versorgt,
- dem Python-Daten-Modul *config_data.py* im Unterverzeichnis *pyFiles*,
- der Logging-Datei *Fotos_Resizing_Logging.log*, die vom Hauptskript im Verzeichnis dieses Hauptskriptes erzeugt wird und in die es Warnungen, Fehlermeldungen, aber auch nützliche Informationen für den Nutzer schreibt, wie den Speicherbedarf der Fotos.

1 Das Foto wird so gedreht, dass der EXIF-Eintrag dann den Wert ‚Top left' hat.
2 Ich habe hier das Werkzeug QGis im Hinterkopf.
3 csv = ‚Comma-Separates Values'

Die Eingabedaten

Die Eingabedaten-Datei ‚inputdata_for_resizing_fotos.csv' stellt über 5 (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
Steuerinformation

Die *erste Spalte* der Eingabedaten-Datei enthält Steuerinformation für das Auslesen[4] der Eingabedaten 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
"Verzeichnispfad zu der verkleinernden Fotos"

Die *zweite Spalte* der Eingabedaten-Datei enthält in jeder Zeile, in Gänsefüßchen gesetzt, den Pfad des Quell-Verzeichnisses, in dem sich die Fotos befinden, die verkleinert werden sollen; ein Beispiel mitsamt den äußeren Trennstrichen | meines csv-Formates:

▷ | "C:\Users\Bernd\Pictures\2025" | ◁[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 3. Spalte
"Verzeichnispfad der verkleinerten Fotos"

Die *dritte Spalte* der Eingabedaten-Datei enthält in jeder Zeile, in hochgestellten Gänsefüßchen gesetzt, den vollständigen Pfad des Ziel-Verzeichnisses, in dem die verkleinerten Fotos abgelegt werden sollen, im Beispiel für Windows:

| "C:\Users\Bernd\Pictures\2025_sm" | [6]

Ist dieses Ziel-Verzeichnis auf dem Rechner *nicht* vorhanden, wird es vom Skript angelegt.

Die 4. Spalte
f(lache) / r(ekursive) FotoSuche

Die *vierte Spalte* der Eingabedatei enthält in jeder Zeile ohne hochgestellte Gänsefüßchen einen der zwei Kleinbuchstaben *f* oder *r*, ein Beispiel:

▷ | f | ◁

Der Buchstabe *f* steht für *f*lach, für eine flache Hierachie: Hier werden vom Python-Skript nur die Fotos verwertet, die sich unmittelbar in dem Foto-Verzeichnis befinden, welches

4 Leerzeichen werden überlesen.

5 Die weiß gefüllten Dreiecke vorne und hinten sollen nur den weißen Leerraum kenntlich machen.

6 Ohne ein weiteres \ am Ende des Pfades!

in der 2. Spalte aufgeführt ist; der Buchstabe *r* steht für *r*ekursiv, für eine verschachtelte Verzeichnis-Hierachie: Das Python-Skript läuft über alle Fotos und rekursiv über alle Unterverzeichnisse des Quell-Verzeichnisses der 2. Spalte.

Alle verkleinerten Fotos, auch die aus allen Unterzeichnissen, werden in dem einen Ziel-Verzeichnis der 4. Spalte abgelegt!

Die 5. Spalte
SollLänge kurze Seiten

Die *fünfte Spalte* der Eingabedatei enthält in jeder Zeile ohne hochgestellte Gänsefüßchen eine ganze Zahl, die die gewünschte Länge der kurzen Seiten der verkleinerten Fotos in Pixel angibt, ein Beispiel:

▷ | 960 | ◁

Diese Größenzahl muss zwischen den beiden Grenzwerten 200 und 2000 liegen; beide Werte werden in der Python-Datei ‚pyFiles/config.py' definiert und können dort bei Bedarf vom Nutzer geändert werden.

Bleibt das Spaltenfeld für die gewünschte kurze Seitenlänge in der Form

▷ || ◁ oder ▷ | | ◁

leer, so wird der Voreinstellungswert 960 benutzt; dieser Wert kann ebenfalls in der erwähnten Konfigurationsdatei angepasst werden.

Ist die kurze Seite eines Fotos kleiner als die eingestellte Solllänge, wird das Foto zwar vom Skript regulär verarbeitet und im Zielordner abgelegt, die Fotomaße bleiben aber unangetastet.

Die Vorgabe der Länge der kurzen Seite für die Verkleinerung hat nach meinem Gusto den Charme, dass der Charakter der unterschiedlichen Fotoformate, etwa 4:3, 19:9 oder 1:1, weniger verloren geht; gibt man stattdessen einfach eine maximale Länge der Breite und Höhe für die Verkleinerung vor, werden alle Bildformate einheitlich in ein Quadrat eingepasst, der Charakter des Breitformats geht so im wesentlichen verloren.

Die 6. Spalte
JPEG-Qualitätsmaß

Die *sechste Spalte* der Eingabedatei enthält in jeder Zeile ohne hochgestellte Gänsefüßchen eine ganze Zahl, die die gewünschte Bildqualität (Kompressionsstufe) des verkleinerten Fotos angibt, ein Beispiel:

▷ | 85 | ◁

Diese Qualitätszahl muss zwischen den beiden Grenzwerten 60 und 95 liegen; beide Werte werden in der Python-Datei ‚pyFiles/config.py' definiert und können dort bei Bedarf vom Nutzer geändert werden.

Bleibt das Spaltenfeld für die Bildqualität leer in der Form

▷ || ◁ oder ▷ | | ◁,

so wird der Voreinstellungswert 85 benutzt; dieser Wert kann ebenfalls in der erwähnten Konfigurationsdatei angepasst werden.

Die Konfigurationsparameter

Die Konfigurationsparameter-Datei ‚config.py' im Verzeichnis ‚pyFiles' enthält die folgenden Werte:

Dateinamen

- log_filename: „Fotos_Resizing_Logging.log"
- input_file: „inputdata_for_resizing_fotos.csv"

Voreinstellungswert und Grenzwerte für die Länge der kurze Seiten der verkleinerten Fotos in Pixel

- default_short_flank: **960***
- max_short_flank 2000
- min_short_flank 200

Voreinstellungswert der JPEG-Qualität und Grenzwerte der Kompression

- default_quality: **85***
- max_quality: 95
- min_quality: 60

Multi-Procesing

- mp True

  Die Verwendung von mehreren CPU-Kernen beschleunigt die Abarbeitung der Fotos erheblich.

Konsolenverwendung

- with_console: True

  Falls die Skripte in Entwicklungsumgebungen wie PyCharm gestartet werden, kann der Parameter auf *False* gesetzt werden.

Die beiden **fett**gedruckten und mit einem Sternchen * verzierten Zahlenwerte können in der CSV-Eingabedatendatei überschrieben werden.

Die Ausgabedaten

Fotobenennung: Die ausgerichteten und verkleinerten Fotos werden nach dem folgenden Muster benannt, es muss gegebenenfalls angepasst werden:

```
yyyymmdd_nnn__origpart_sm.jpg
```

- Der 1. Namensbestandteil *yyyymmdd* steht für den tatsächlichen Aufnahmetag des Fotos, etwa *20260317*.
- Der 2. Namensbestandteil *nnn* steht für einen dreistelligen Zähler, etwa *001*, der die an diesem einen Tag gemachten Fotos zeitlich geordnet durchnummeriert.

- Der 3. Namensbestandteil *origpart* steht für einen Namensbestandteil des Dateinamens der Original-Fotodatei, der von der digitalen Kamera oder dem Smartphone abhängt, hier einige Beispiele:

  ```
  20251005_141815.jpg
  IMG_9922.JPG
  DSC08406.JPG
  ```

  Die gelb hinterlegten Ziffernfolgen würden in den neuen Dateinamen der verkleinerten Fotos übernommen; so soll die Zuordnung zum Ausgangsfoto erleichtert werden. Enthält der Ausgangsname keine solche Ziffernfolge, wird der ganze Dateinamen als *origpart* genommen.
- Die letzten zwei Buchstaben *sm* (für *sm*all) sollen die verkleinerten Fotos leichter erkennbar machen.

EXIF-Daten: Aus der Original-Foto-Datei werden in den EXIF-Bereich des verkleinerten Fotos übernommen:

- die Aufnahmezeit des Fotos (‚DateTimeOriginal') sowie
- die vollständigen GPS-Ortsinformationen (‚GPS information') - wenn sie denn vorliegen.

Die Python-Installation

Um die Python-Skripte starten 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 benutze 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[7] als Entwicklungsumgebungen, dazu die Werkzeuge *TortoiseHg*/*Mercurial* zur Verwaltung von Versionen; alles kostet dem Hobby-Entwickler nichts.

Ich benutze in meinen Skripten die beiden Python-Pakete P*illow* und *piexif*, 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.

7 Microsoft steuert mehrere Erweiterungen für Python bei.

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 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 pillow
py -m pip install --user piexif
```

Die Installation meiner Python-Skripte beschränkt sich darauf, den gepackten Projekt-Ordner mit seinen Dateien zu entpacken.

Die Skript-Ausführung

Nach der Installation von Python kann der Nutzer die Skripte im Skriptverzeichnis durch einen Doppelklick starten - oder auch über die Windows-Konsole[8] durch die Eingabe des Skriptnamens.

Ein Doppelklick ist allerdings allzuschnell und unbedacht ausgeführt, zudem verschwinden die Textausgeben, die ein Skript in das Ausführungsfenster schreibt, nach der Skriptausführung im Nirvana. Ich empfehle daher die Verwendung der Konsole: Mit dem Kontextmenü des Skriptverzeichnisses öffnet sich flugs ein Terminal, in das man die zwei drei Anfangsbuchstaben meines Hauptskriptes eingibt; mit der Tab-Taste navigiert man zum Python-Skriptnamen und fügt diesem Namen, getrennt durch ein Leerzeichen, ein kleines oder großes Ypsilon[9] hinzu: Und schon wird das Skript ausgeführt und dem ‚y' wegen schließt sich das Skriptfenster nicht automatisch am Ende.

Das Hauptskript kann problemlos mit denselben Eingangsdaten wiederholt ausgeführt werden, die Dateien im Ziel-Verzeichnis werden einfach überschrieben.

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

Die Nebenwirkungen

Die Skripte arbeiten nicht-destruktiv, die Ausgangsdateien werden ausschließlich gelesen.

Die Danksagung

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

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

9 Verwendet man Microsofts PowerShell wird das kleine y gleich mit dem Namen ausgegeben.

- Mit der Hilfe des Python-Paketes *piexif* werden die Aufnahmezeiten und die Ortsdaten aus den EXIF-Bereichen der Ausgangsfotos ausgelesen und in die verkleinerten Fotos geschrieben.
- Mit der Hilfe des Python-Paketes *Pillow (PIL)* werden die Fotos gegebenenfalls ausgerichtet und dann auf das gewünschte Maß verkleinert.
- Und nicht zu vergessen sei der Dank an die Macher der großartigen Skriptsprache Python.

Die Gewährleistung

Die Software wird so „wie sie ist“ bereitgestellt, ohne Gewährleistung irgendeiner Art.