Published in Spatial Data Science

Abdishakur Follow
Jul 29, 2021 · 4 min read · · Listen

Save

How to extract GPS coordinates from Images in Python

Geolocating Photos and Extracting Coordinates in Python.

The pictures you take with your smartphone or digital camera have rich metadata of additional information about the photo. They store this metadata in a format called EXIF, short for Exchangeable Image File Format, which has different standard versions.

> ***Exchangeable image file format*** *(officially* ***Exif****, according to JEIDA/JEITA/CIPA specifications) is a standard that specifies the formats for images, sound, and ancillary tags used by digital cameras (including smartphones), scanners and other systems handling image and sound files recorded by digital cameras.*

The EXIF metadata might have information about the model of the device, dimensions, the date of the picture, and its location.

This article will teach you how to extract the EXIF metadata from images using Python, including where that photo was taken. Extracting location information from Photos can help you as a Geospatial data scientist have additional information for your analysis or model.

EXIF in Python

We start with getting the EXIF metadata using the EXIF Python library. First, we open the image and pass it to exif.Image method.

```
from exif import Image
img_path = 'images/image_exif.jpg'
with open(img_path, 'rb') as src:
    img = Image(src)
    print (src.name, img)
```

And now, we have successfully opened the image using the EXIF library. The output returns the object, which is not meaningful as per se as we will see.

```
images/image_exif.jpg <exif._image.Image object at 0x00000223EBC62430>
```

Before we move on, let us first open the image that we will use for this tutorial using the PIL library in Python.

```
import PIL
image = PIL.Image.open(img_path)
image
```

Now, let us move on to getting the EXIF information of the image. Some images might not have the EXIF metadata, so it is good to check out if there is any exif metadata available for the picture.

```
if img.has_exif:
 info = f” has the EXIF {img.exif_version}”
else:
 info = “does not contain any EXIF information”
print(f”Image {src.name}: {info}”)
```

In this case, our image contains the 0220 EXIF standard information.

```
Image images/image_exif.jpg:  has the EXIF 0220
```

If you pass an image without EXIF, you will get no information as I show you in the following example using a photo without EXIF information.

```
# Image without EX

with open('images/foto_no_exif.jpg', "rb") as src:
 img = Image(src)
 if img.has_exif:
    info = f" has the EXIF {img.exif_version}"
 else:
    info = "does not contain any EXIF information"
 print(f"Image {src.name}: {info}")
```

And therefore, the output prints out that the image does not have any EXIF information.

```
Image images/foto_no_exif.jpg: does not contain any EXIF information
```

What kind of EXIF metadata Tags do we have?

Images can have different metadata. In our example, we can list out all tags available.

```
# Read again photo with exif info
with open(img_path, “rb”) as src:
 img = Image(src)

img.list_all()
```

And we have a rich list of EXIF metadata available at our fingertips.

```
['image_description',
 'make',
 'model',
 'orientation',
 'x_resolution',
 'y_resolution',
 'resolution_unit',
 'software',
 'datetime',
 'y_and_c_positioning',
 '_exif_ifd_pointer',
 '_gps_ifd_pointer',
 'compression',
 'jpeg_interchange_format',
 'jpeg_interchange_format_length',
 'exposure_time',
 'f_number',
 'exposure_program',
 'photographic_sensitivity',
 'exif_version',
 'datetime_original',
 'datetime_digitized',
 'components_configuration',
 'exposure_bias_value',
 'max_aperture_value',
 'metering_mode',
 'light_source',
 'flash',
 'focal_length',
 'maker_note',
 'user_comment',
 'flashpix_version',
 'color_space',
 'pixel_x_dimension',
 'pixel_y_dimension',
 '_interoperability_ifd_Pointer',
 'file_source',
 'scene_type',
 'custom_rendered',
 'exposure_mode',
 'white_balance',
 'digital_zoom_ratio',
 'focal_length_in_35mm_film',
 'scene_capture_type',
 'gain_control',
 'contrast',
 'saturation',
 'sharpness',
 'subject_distance_range',
 'gps_latitude_ref',
 'gps_latitude',
 'gps_longitude_ref',
 'gps_longitude',
 'gps_altitude_ref',
 'gps_timestamp',
 'gps_satellites',
 'gps_img_direction_ref',
 'gps_map_datum',
 'gps_datestamp']
```

Let us try out some of these tags. Forexample to get the longitude coordinates of the image, we can call:

```
img.gps_longitude
```

And you will get the coordinates in Degrees, minutes, and seconds (DMS). We will soon convert this format to Decimal degrees (DD) format.

```
(11.0, 53.0, 7.42199999)
```

It also has the coordinate references, which you can call through:

```
img.gps_longitude_ref
```

This is crucial information to get your coordinates right when converting to Decimal degrees (DD) format. In this particular image, we have:

```
'E'
```

Extracting Coordinates from Image

Let us wrap up this tutorial. Here I will create two functions. The first function converts the coordinates into Decimal degrees (DD) format.

```
def decimal_coords(coords, ref):
 decimal_degrees = coords[0] + coords[1] / 60 + coords[2] / 3600
 if ref == “S” or ref == “W”:
     decimal_degrees = -decimal_degrees
 return decimal_degrees
```

And the second function opens the image with exif python and prints out the coordinates in Decimal degrees (DD) format with some additional metadata of the image.

```
def image_coordinates(image_path):
    with open(img_path, 'rb') as src:
        img = Image(src)

    if img.has_exif:
        try:
            img.gps_longitude
            coords = (decimal_coords(img.gps_latitude,
                      img.gps_latitude_ref),
                      decimal_coords(img.gps_longitude,
                      img.gps_longitude_ref))
        except AttributeError:
            print 'No Coordinates'
    else:
        print 'The Image has no EXIF information'

    print(f"Image {src.name}, OS Version:{img.get('software', 'Not
```

Open in app ↗ Get unlimited access

Search Medium

Now, we can call this function on any image and get back the coordinates in a nice format, plus the camera model and the date of the image.

```
image_coordinates(img_path)
```

And the output is is the following:

```
Image images/DSCN0012.jpg, OS Version:Nikon Transfer 1.1 W ------
Was taken: 2008:10:22 16:29:49, and has coordinates:
(43.46715666666389, 11.885394999997223)
```

Conclusion

Extracting location features from images and additional metadata can help you carry out geospatial data analysis and modeling. This tutorial shows you how to extract coordinates from images using the EXIF Python library.

To install exif Python Library, run the following command:

```
pip install exif
```

Visit the GitLab repository of the library to get more information.

Tyler Thieding / exif

Read and modify image EXIF metadata using Python.

gitlab.com

131

Geospatial GIS Python Data Science Spatial Analysis