Published in Towards Data Science

Kenneth Leung Follow
Nov 23, 2021 · 5 min read · Member-only · Listen

Read and Edit Image Metadata with Python

Using Python's exif library to extract and modify metadata of digital image files

Photo by JESHOOTS.COM on Unsplash

For every photo, there is more than meets the eye. The images taken with digital cameras and smartphones contain rich information (known as **metadata**) beyond the visible pixels.

This metadata can be helpful in many business cases. For instance, **fraud detection** systems for **insurance** claims analyze metadata of submitted photographs to check whether the claimant took them **before** the accident.

In this article, we explore how to use the *exif* library to read and edit metadata of digital images.

Contents

Feel free to check out all the codes in the accompanying **GitHub repo**.

What are Metadata and Exif?

Metadata refers to the set of data describing the data, and you can think of it as **data about the data**. The metadata of photos consists of information such as camera model and date of capture.

This metadata is stored in **Exif (Exchangeable image file format)**, a format **standard** for the various types of media (e.g. images, videos, audio) taken by devices such as digital cameras and smartphones.

The Python library used in this project is *exif*, which happens to be the namesake of the Exif format.

Read Image Metadata

We start by installing *exif* with this command:

```
pip install exif
```

For this demo, here is the photo that we will be working on:

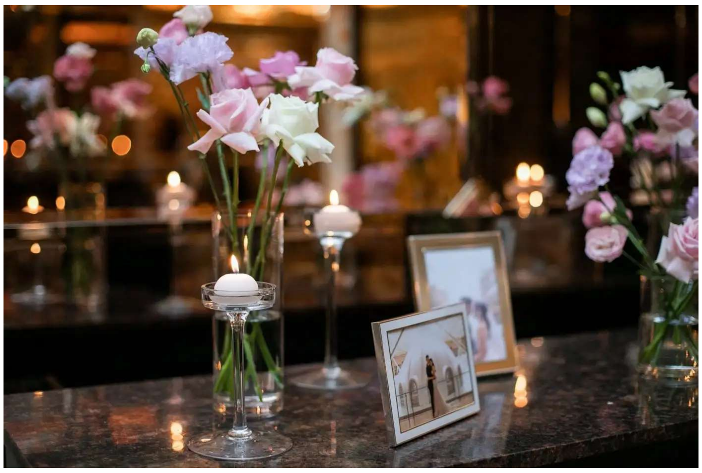


Image by author

We instantiate an *exif* `Image` Class by reading the image in a binary format before checking whether it contains any metadata. If so, the `has_exif` method will return *True*.

```python
from exif import Image

folder_path = 'sample_images'
img_filename = 'image_1.jpg'
img_path = f'{folder_path}/{img_filename}'

with open(img_path, 'rb') as img_file:
    img = Image(img_file)

print(img.has_exif)
```

metadata_exif_1.py hosted with ❤ by GitHub view raw

Different photos have varying metadata attributes (aka Exif **tags**) due to the different devices used to captur[illegible] view the existing attributes for an image with the `list_all()` method.

76 1

```python
# List all EXIF tags contained in the image
sorted(img.list_all())
```

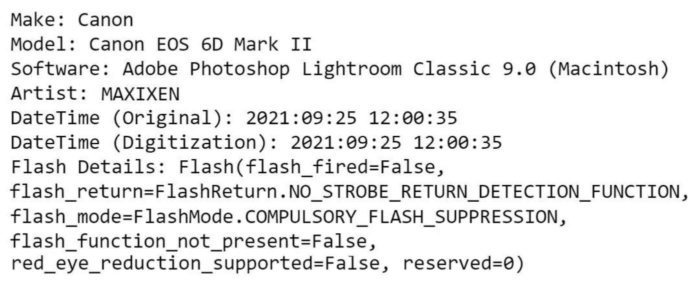
metadata_exif_2.py hosted with ❤ by GitHub view raw

```
['_exif_ifd_pointer',
 'aperture_value',
 'artist',
 'body_serial_number',
 'color_space',
 'compression',
 'copyright',
 'custom_rendered',
 'datetime',
 'datetime_digitized',
 'datetime_original',
 'exif_version',
 'exposure_mode',
 'exposure_program',
 'exposure_time',
 'f_number',
 'flash',
```

To read values of specific attributes, we can use the `get()` method. While there are other methods, I prefer `get()` as it gracefully handles cases where attributes do not exist by returning *None* (instead of throwing an error).

```python
# Make of device which captured image
print(f'Make: {img.get("make")}')

# Model of device which captured image
print(f'Model: {img.get("model")}')

# Software involved in uploading and digitizing image
print(f'Software: {img.get("software")}')

# Name of photographer who took the image
print(f'Artist: {img.get("artist")}')

# Original datetime that image was taken (photographed)
print(f'DateTime (Original): {img.get("datetime_original")}')

# Details of flash function
print(f'Flash Details: {img.get("flash")}')
```

metadata_exif_3.py hosted with ❤ by GitHub view raw



```
Make: Canon
Model: Canon EOS 6D Mark II
Software: Adobe Photoshop Lightroom Classic 9.0 (Macintosh)
Artist: MAXIXEN
DateTime (Original): 2021:09:25 12:00:35
DateTime (Digitization): 2021:09:25 12:00:35
Flash Details: Flash(flash_fired=False,
flash_return=FlashReturn.NO_STROBE_RETURN_DETECTION_FUNCTION,
flash_mode=FlashMode.COMPULSORY_FLASH_SUPPRESSION,
flash_function_not_present=False,
red_eye_reduction_supported=False, reserved=0)
```

Output of get() methods | Image by author

P.S. Check out the Image_Metadata_Extraction_EXIF.ipynb notebook for the function to extract all metadata of an image into a Pandas DataFrame.

Photo by Mylene Tremoyet on Unsplash

Modify Image Metadata

Besides reading metadata, we can perform a series of modifications such as adding, updating, and deleting attributes.

New attributes not currently present can be added to further enrich the metadata.

One important thing to note is that the attribute added **must be** a recognized EXIF tag. Otherwise, the addition will not take place. You can find the complete list of recognized image attributes **here**.

For example, we can add the recognized **copyright** attribute. After assigning a value (*Kenneth Leung 2021*) to the **copyright** attribute, the `get()` method will give us this new value instead of *None*.

```python
# Add new attribute (Copyright)
img.copyright = 'Kenneth Leung 2021'

# Check updated metadata
print(f'Copyright: {img.get("copyright")}')
```

metadata_exif_4.py hosted with ❤ by GitHub view raw

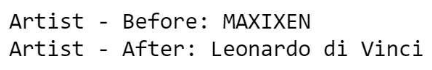
Join Medium with my referral link - Kenneth Leung
Access all my content (and all Medium articles) at the price of just one coffee!
kennethleungty.medium.com

(ii) Update metadata

We can also update the existing values of the image metadata attributes.

```python
# View existing value for artist attribute
print(f'Artist - Before: {img.get("artist")}')

# Update name of artist attribute
img.artist = 'Leonardo di Vinci'

# Check updated metadata
print(f'Artist - After: {img.get("artist")}')
```

metadata_exif_5.py hosted with ❤ by GitHub view raw



```
Artist - Before: MAXIXEN
Artist - After: Leonardo di Vinci
```

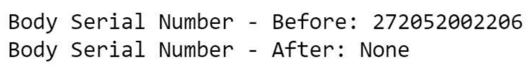
Output after metadata update | Image by author

(iii) Delete metadata

If we want to delete specific attributes instead of updating them, we can do so with `.delete()`.

```python
# View existing value for body_serial_number attribute
print(f'Body Serial Number - Before: {img.get("body_serial_number")}')

# Delete body_serial_number attribute
img.delete('body_serial_number')

# Check updated metadata
print(f'Body Serial Number - After: {img.get("body_serial_number")}')
```

metadata_exif_6.py hosted with ❤ by GitHub view raw



```
Body Serial Number - Before: 272052002206
Body Serial Number - After: None
```

Output after metadata deletion | Image by author

After all the modifications, the final step is to save the image with the modified metadata as a new file.

```python
# Write image with modified EXIF metadata to an image file
with open(f'{folder_path}/modified_{img_filename}', 'wb') as new_image_file:
        new_image_file.write(img.get_file())
```

metadata_exif_7.py hosted with ❤ by GitHub view raw

With the above, we have saved the modified image using a filename with the prefix of '*modified_*' so that the original image is not overwritten.

Wrapping It Up

- There are many other interesting attributes to explore, and you can find more details on the *exif* documentation page.
- What we have done so far is process a single image. The value of the *exif* package is realized through **batch processing**, where the extraction and modification of metadata are done on a **large** set of images. To see batch processing in action, have a look at the `batch_process_metadata.py` script in the **GitHub repo**.
- An important thing to keep in mind is to back up your photos before using this library to prevent any unexpected data loss.

Before You Go

I welcome you to **join me on a data science learning journey!** Follow this Medium page and check out my GitHub to stay in the loop of more exciting data science content. Meanwhile, have fun reading and modifying image metadata!

Top Python libraries for Image Augmentation in Computer Vision
Featuring the best augmentation libraries (along with sample codes) for your next computer vision project
towardsdatascience.com

Russian Car Plate Detection with OpenCV and TesseractOCR
Detecting, recognizing, and extracting car license plate numbers with the power of computer vision
towardsdatascience.com

Key Learning Points from MLOps Specialization — Course 2
Insights summary (with lecture notes) of the Machine Learning Engineering for Production (MLOps) Course
towardsdatascience.com