note.nkmk.me

Top > Python > OpenCV

Reading and saving image files with Python, OpenCV (imread, imwrite)

Modified: 2022-10-15 | Tags: Python, OpenCV, Image Processing

In Python and OpenCV, you can read (load) and write (save) image files with `cv2.imread()` and `cv2.imwrite()`. Images are read as NumPy array `ndarray`.

This article describes the following contents.

You can also read image files as `ndarray` with Pillow instead of OpenCV.

- Image processing with Python, NumPy

See the following article for information on reading videos instead of still images.

- Capture video from camera/file with OpenCV in Python

The following image is used as an example.

Read and write images in color (BGR)

Read an image file with `cv2.imread()`

Color image file is read as a 3D ndarray of `row (height) x column (width) x color (3)` .

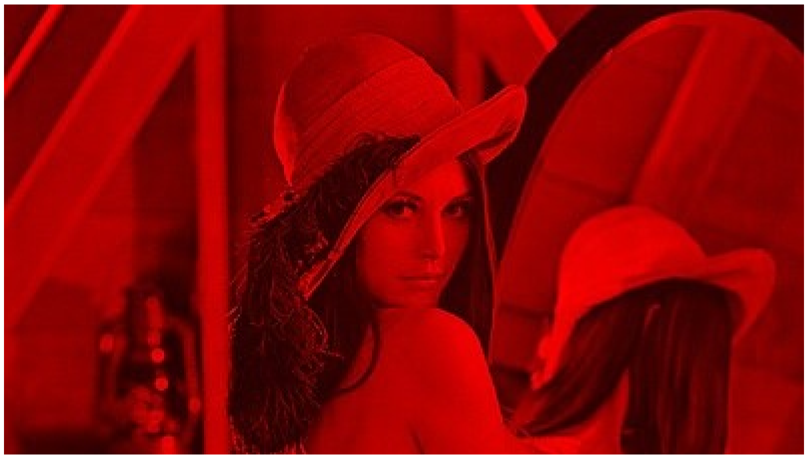
```
import cv2

im = cv2.imread('data/src/lena.jpg')

print(type(im))
# <class 'numpy.ndarray'>

print(im.shape)
# (225, 400, 3)

print(im.dtype)
# uint8
```

source: opencv_read_write.py

Note that the color order is BGR instead of RGB. As an example, set 0th (B: blue) and 1st (G: green) to 0 (black).

```
im[:, :, (0, 1)] = 0
```

source: opencv_read_write.py



If you want to convert BGR and RGB, please refer to the following article.

- Convert BGR and RGB with Python, OpenCV (cvtColor)

Write ndarray as an image file with `cv2.imwrite()`

To save `ndarray` as an image file, set the file path and `ndarray` object to `cv2.imwrite()` .

The image file format is automatically determined from the file path extension. If it is `.jpg` , it is saved as JPEG, and if it is `.png` , it is saved as PNG.

```
cv2.imwrite('data/dst/lena_opencv_red.jpg', im)
```

source: opencv_read_write.py

You can specify format-specific parameters for the third parameter. Specify with a list like `[paramId_1, paramValue_1, paramId_2, paramValue_2, ...]` .

For parameter ID (flag), refer to the official document below.

- OpenCV: Image file reading and writing - ImwriteFlags

For example, the quality of JPEG is specified by `cv2.IMWRITE_JPEG_QUALITY` . `0` is the lowest and `100` is the highest, the default is `95` .

If saved as `50` :

```
cv2.imwrite('data/dst/lena_opencv_red_low.jpg', im, [cv2.IMWRITE_JPEG_QUALITY, 50])
```

source: opencv_read_write.py

If saved as `100` :

```
cv2.imwrite('data/dst/lena_opencv_red_high.jpg', im, [cv2.IMWRITE_JPEG_QUALITY, 100])
```

source: opencv_read_write.py

Note that JPEG is lossy compression, so even if it is the highest quality `100` , when the saved image is reloaded, a difference occurs with the original pixel value. If you want to save the original image as it is, save it as PNG or BMP.

Read and write images in grayscale

Read an image file with `cv2.imread()`

By passing `cv2.IMREAD_GRAYSCALE` as the second argument of `cv2.imread()`, you can read a color image file in grayscale (black and white). Since `cv2.IMREAD_GRAYSCALE` is equivalent to `0`, you may specify `0`.

It is useful for detecting edges and other situations where color information is not required.

In this case, the image is read as 2D `ndarray` of `row (height) x column (width)`.

```
im_gray = cv2.imread('data/src/lena.jpg', cv2.IMREAD_GRAYSCALE)
# im_gray = cv2.imread('data/src/lena.jpg', 0)

print(type(im_gray))
# <class 'numpy.ndarray'>

print(im_gray.shape)
# (225, 400)

print(im_gray.dtype)
# uint8
```

source: opencv_read_write.py

You can also read the image file as color and convert it to grayscale with `cv2.cvtColor()` and `cv2.COLOR_BGR2GRAY`.

Because `cv2.IMREAD_GRAYSCALE` with `cv2.imread()` perform codec-dependent conversions instead of OpenCV-implemented conversions, you may get different results on different platforms. `cv2.cvtColor()` with `cv2.COLOR_BGR2GRAY` is safer to use if you want to handle pixel values strictly.

Write ndarray as an image file with `cv2.imwrite()`

If 2D `ndarray` of `row(height) x column(width)` is specified as the argument of `cv2.imwrite()`, it is saved as a grayscale image file.

```
cv2.imwrite('data/dst/lena_opencv_gray.jpg', im_gray)
```

source: opencv_read_write.py

If you want to save a color image (3D `ndarray`) as a grayscale image file, convert it to grayscale with `cv2.cvtColor()` and `cv2.COLOR_BGR2GRAY`.

If you save 2D `ndarray` to a file and read it again with `cv2.imread()`, it will be read as 3D `ndarray` in which each color is the same value. It does not automatically read as a two-dimensional array.

- NumPy: Compare ndarray element by element

```
im_gray_read = cv2.imread('data/dst/lena_opencv_gray.jpg')

print(im_gray_read.shape)
# (225, 400, 3)

import numpy as np

print(np.array_equal(im_gray_read[:, :, 0], im_gray_read[:, :, 1]))
# True

print(np.array_equal(im_gray_read[:, :, 1], im_gray_read[:, :, 2]))
# True
```

source: opencv_read_write.py

Notes on `cv2.imread()`

`cv2.imread()` does not raise an exception

Even if a nonexistent path is specified, `cv2.imread()` does not raise an exception and return `None`. As of OpenCV `4.6.0`, a warning is issued.

An error will be raised when you do some operation assuming it is read as `ndarray`.

```
im_not_exist = cv2.imread('xxxxxxx')
# [ WARN:0@0.069] global /tmp/opencv-20221012-82716-1q8maeo/opencv-4.6.0/modules/imgcodecs/src/

print(im_not_exist)
# None

# print(im_not_exist.shape)
# AttributeError: 'NoneType' object has no attribute 'shape'
```

source: opencv_read_write.py

Even if the file exists, `None` is returned if OpenCV does not support it.

```
im_not_supported = cv2.imread('data/src/sample.csv')

print(im_not_supported)
# None
```

source: opencv_read_write.py

You can check if the image was read correctly as follows:

```
if im_not_exist is not None:
    print('Image is read.')
else:
    print('Image is not read.')
# Image is not read.
```

source: opencv_read_write.py

```
if im_not_exist is None:
    print('Image is not read.')
else:
    print('Image is read.')
# Image is not read.
```

source: opencv_read_write.py

If the image file is read correctly:

```
im = cv2.imread('data/src/lena.jpg')

if im is not None:
    print('Image is read.')
else:
    print('Image is not read.')
# Image is read.
```

source: opencv_read_write.py

```
if im is None:
    print('Image is not read.')
else:
    print('Image is read.')
# Image is read.
```

source: opencv_read_write.py

JPEG library

As you can see in the GitHub issue below, the library used to process JPEGs depends on the OpenCV version, platform, etc. Therefore, even if the same file is read, there may be differences in the values if the environment is different.

> Reading of JPEG images is not bit-exact operation. It depends on used library (libjpeg/libjpeg-turbo) and/or versions, platforms (x86/ARM), compiler options.
>
> *imread and imwrite cause differences in image pixels · Issue #10887 · opencv/opencv*

If images cannot be read with `cv2.imread()`

Check the current directory

Like the built-in function `open()`, with `cv2.imread()` and `cv2.imwrite()`, you can specify the path of a file with one of the following methods:

- Relative path from the current directory
- Absolute path

If the file should be there but cannot be read, it is often because of a simple mistake that the current directory is different from what is expected.

You can check the current directory with `os.getcwd()`.

- Get and change the current working directory in Python

Supported formats for `cv2.imread()`

Of course, you can not read image files in formats not supported by OpenCV.

The following formats are supported by `cv2.imread()` of OpenCV `4.2.0`. See also the notes at the link.

> - Windows bitmaps - *.bmp,* .dib (always supported)
> - JPEG files - *.jpeg,* .jpg, *.jpe (see the Note section)
> - JPEG 2000 files - *.jp2 (see the Note section)
> - Portable Network Graphics - *.png (see the Note section)
> - WebP - *.webp (see the Note section)
> - Portable image format - *.pbm,* .pgm, *.ppm* .pxm, *.pnm (always supported)
> - PFM files - *.pfm (see the Note section)
> - Sun rasters - *.sr,* .ras (always supported)
> - TIFF files - *.tiff,* .tif (see the Note section)
> - OpenEXR Image files - *.exr (see the Note section)
> - Radiance HDR - *.hdr,* .pic (always supported)
> - Raster and Vector geospatial data supported by GDAL (see the Note section)
>
> *OpenCV: Image file reading and writing*

See below for other versions.

- OpenCV 3.0.0: Image file reading and writing
- Reading and Writing Images and Video — OpenCV 2.4.13.0 documentation
- OpenCV documentation index

You can check information about libraries and so on in the `Media I/O` section of `cv2.getBuildInformation()`.

- Check OpenCV Build Information: getBuildInformation()

`cv2.imread()` checks the format of a file by its content, not by its extension. If the file cannot be read, check whether the file can be read by another application (check whether the file is not corrupted).

> The function determines the type of an image by the content, not by the file extension.
>
> *OpenCV: Image file reading and writing*

Related Categories

- Python
- OpenCV
- Image Processing

Related Articles

- Concatenate images with Python, OpenCV (hconcat, vconcat, np.tile)
- Convert BGR and RGB with Python, OpenCV (cvtColor)
- OpenCV, NumPy: Rotate and flip image
- Get image size (width, height) with Python, OpenCV, Pillow (PIL)
- Detect and read barcodes with OpenCV in Python
- Binarize image with Python, NumPy, OpenCV
- Alpha blending and masking of images with Python, OpenCV, NumPy

- Detect and read barcodes and QR codes with ZBar in Python
- Detect and read QR codes with OpenCV in Python
- Generate gradient image with Python, NumPy
- Concatenate images with Python, Pillow
- Generate square or circular thumbnail images with Python, Pillow
- Composite two images according to a mask image with Python, Pillow