Photolab

Authored And Maintained By

Bhamidipati Venkatakrishnasameer

Abstract

Photolab is a Python package oriented towards various image processing techniques. It consists of various algorithms which are commonly used during various image processing techniques. Photolab takes its role in computer vision and machine learning at the feature extraction and feature engineering stage of model development. Photolab also takes caters to various image restoration techniques when a distorted image is given as an input. Thus Photolab helps one in various image processing activities like image smoothening, image sharpening, edge detection, etc.
Photolab is open-sourced under
Creative Commons CC BY-SA 4.0 license.

Photolab can be downloaded by pip install photolab

Setup Photolab

To Setup Photolab:

- Go to Command prompt (Windows) or Terminal (Linux)

- Type the following command
  pip install photolab

- Import Photolab in your Python script

- Run the script to ensure no errors occur

Github: https://github.com/bvks2020sameer/Photolab
PyPi: https://pypi.org/project/photolab/

Creative Common License CC BY-SA 4.0

Creative Commons Attribution-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. Adapted Material means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. Adapter's License means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. Copyright and Similar Rights means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. Effective Technological Measures means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. Exceptions and Limitations means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. Licensed Material means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. Licensed Rights means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. Licensor means the individual(s) or entity(ies) granting rights under this Public License.

i. Share means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. Sui Generis Database Rights means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. You means the individual or entity exercising the Licensed Rights under this Public License. Your has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

   a. reproduce and Share the Licensed Material, in whole or in part; and

   b. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

a. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

b. Additional offer from the Licensor – Adapted Material. Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

c. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

a. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

b. indicate if You modified the Licensed Material and retain an indication of any previous modifications, and

c. indicate the Licensed Material is licensed under this Public License, and

Common Conventions Used

This text in black color is used for description

This text in blue color is used for code snippets and keywords (functions and classes)

This text in red color is used for instructions to users

This text in purple color (bold italic) is used for arguments to the functions

Index

Contents	Page Number

Photolab

Photolab is a Python package to implement various image transformation techniques. These transformation techniques can be mainly implemented in feature engineering during machine learning, creating custom artificial neural networks, etc along with non-machine learning-based image processing approaches to a given problem. It is wholly built on numpy for faster execution.

Project Photolab consists of algorithms classified into 5 categories

- Edge Detection
- Frequency Filtering
- Spatial Image Operations
- Image Noise
- Image Restoration

These algorithms help one to perform operations like image pre-processing, image restorations, edge detection, image smoothening, noise reduction, image blurring, etc.

Getting Started with Photolab

Photolab takes input in the form of numpy matrices. The source image matrix and the dimensions of the image must be given as inputs while creating the initial Photolab object.

An example code snippet is given below:

```
import cv2
import Photolab

photo = cv2.imread("shark.jpg")
photo = cv2.resize(photo,(1200,800))
e = Photolab.edge(photo,800,1200)
```

Here the User is expected to resize the image one imports to the desired dimensions[1].

Each object of Photolab takes 3 inputs:

- The image source matrix
- Parameter M (The height of the image)
- Parameter N (The width of the image)

[1] Contrary to common dimension notation of (width, height) in images, Photolab objects take all inputs in the form of (height,width)

Edge-based Image Transforms

Edge-based image transforms are widely used in object detection, image morphology operations, etc. Here the images are to be in the black-and-white format before implementing the edge-based algorithms. The object used here is Photolab.edge(photo, M, N) . An instance of the edge class has to be made by the user and then various functions below it can be accessed.

An example code snippet is given below

```
e = Photolab.edge(photo,800,1200)
out = e.sobelx()
```

The different algorithms available in edge-based image transforms are given below

Double Thresholding

Double Thresholding is a technique where it accepts all pixels within the given ranges of pixel intensities[2]. In Photolab double thresholding can be performed by "double_thresholding" function of edge class

Syntax: double_thresholding(th1,th2) where *th1* and *th2* represent the upper and lower threshold intensity values

[2] Both the upper and lower threshold values are inclusive

Sobel Operator

The Sobel operator is a second-order edge detection operator. In Photolab 3 different variants of Sobel operators are available

1. sobelx(): performs Sobel operation in the horizontal direction
2. sobely(): performs Sobel operation in the vertical direction
3. sobel_avg(): takes an average of sobelx() and sobely() outputs

Gradientation

Gradientation is used to remove extra unwanted lines/edges in the image after the Sobel operation. This reduces noise in edge detection by highlighting only the major important lines. In Photolab Gradientation is available as gradientation() function

Prewitts Operator

Prewitts Operator is a first-order edge detection algorithm. In Photolab Prewitts Operator is available as

- prewitts_horizontal(): It identifies all horizontal edges
- prewitts_vertical(): It identifies all vertical edges

Roberts Operator

Roberts Operator is used to detect edges in diagonal directions. In Photolab Roberts Operator is available as:

1. roberts45(): It is used to detect edges in the 45° diagonal of the image

2. roberts135(): It is used to detect edges in the 135° diagonal of the image

Frequency-Based Image Transforms

Frequency-based filtering or image transforms are used in operations like de-blurring, image sharpening, etc. Here the images are to be in the black-and-white format before implementing the frequency-based algorithms. The object used here is Photolab.freq_filter(photo, M, N) . An instance of the freq_filter class has to be made by the user and then various functions below it can be accessed.

The different algorithms available in frequency-based image transforms are given below

Ideal Filters

Ideal filters are frequency-filtering image transforms to suppress either lower or higher frequencies. They are faster than other frequency filters but cause a ringing effect in the images. In Photolab the ideal filters available are

1. ideal_low_pass(cuttoff): It is used to suppress frequencies higher than the given *cuttoff* intensity.

2. ideal_high_pass(cuttoff): It is used to suppress frequencies higher than the given *cuttoff* intensity.

Butterworth Filters

Butterworth filters are frequency-filtering image transforms to suppress either lower or higher frequencies. They reduce the ringing effect over Ideal filters but, cause a ringing effect in the images at higher orders. In Photolab the ideal filters available are

1. butter_low_pass(cuttoff, order): It is used to suppress frequencies higher than the given *cuttoff* intensity. The *order* of these filters is used to increase the roll-off factor of the filter giving a steeper transition

2. butter_high_pass(cuttoff, order): It is used to suppress frequencies lower than the given *cuttoff* intensity. The *order* of these filters is used to increase the roll-off factor of the filter giving a steeper transition

Gaussian Filters

Gaussian filters are frequency-filtering image transforms to suppress either lower or higher frequencies. They reduce the ringing effect in images to a very large extent but are much slower than other filters. In Photolab the ideal filters available are

1. gauss_low_pass(cuttoff, sigma): It is used to suppress frequencies higher than the given *cutoff* intensity. The *sigma* value of these filters is used to increase the variance of the filter giving a steeper transition

2. gauss_high_pass(cuttoff, sigma): It is used to suppress frequencies lower than the given *cuttoff* intensity. The *sigma* value of these filters is used to increase the variance of the filter giving a steeper transition

Spatial Image Transforms

Spatial image transforms are the operations performed directly on the pixels themselves. It is used to filter the pixels based on the intensity levels of individual pixels. It can be used in various applications like image enhancement, etc. **Here the images are to be in the black-and-white format before implementing the spatial image transform algorithms**. The object used here is Photolab.spatial_bw(photo, M, N) . An instance of the spatial_bw class has to be made by the user and then various functions below it can be accessed.

The different algorithms available in spatial image transforms are given below

Image inversion

Image Inversion is the process of reversing the color tones and intensities.
Syntax: invert()

Image Thresholding

Image thresholding is the process of filtering all pixel values below a certain intensity. In Photolab, the different image thresholding techniques available are

1. threshold(off): Here all pixel intensities below ***offset*** are set to 0 and other pixels are set to 255

2. threshold_mask(off): Here all pixel intensities below ***offset*** are set to 0 and other pixels are retained

3. threshold_bg(off): Here all pixel intensities below ***offset*** are retained and others are set to 255

Image Slicing

Image Slicing is the process of filtering all pixel values which are not in the range of given pixel intensities. In Photolab, the image-slicing techniques available are

1. gray_slice(off1,off2): Here the image pixel values in the range of (*off1,off2*) *[both off1 & off2 are inclusive]* are set to 255 and the rest of the pixels are set to 0

2. gray_slice_mask(off1,off2): Here the image pixel values in the range of (*off1,off2*) *[both off1 & off2 are inclusive]* are retained and the rest of the pixels are set to 0

3. gray_slice_bg(off1,off2): Here the image pixel values in the range of (*off1,off2*) *[both off1 & off2 are inclusive]* are set to 255 and the rest of the pixels are retained

Image Noise

Image Noise is a process of adding defects to the image. The process of creating noisy images can be used in research on image restoration methods. Photolab offers 2 classes of noises:

- spatial_noise_col(photo)
- probabalistic_noise(photo)

Spatial noise

Spatial noise is the defects that occur at pixel levels. **Here the images are to be in the color format before implementing the spatial noise algorithms**. The object used here is Photolab.spatial_noise_col(photo). An instance of the spatial_noise class has to be made by the user and then various functions below it can be accessed.
Photolab offers the following spatial noise models:

1. salt(cycles): Adds salt noise *(white dots)* to the image. *cycles* is the measure of salt noise *(no of iterations)*

2. pepper(cycles): Adds pepper noise *(black dots)* to the image. *cycles* is the measure of pepper noise *(no of iterations)*

3. salt_pepper(cycles): Adds salt and pepper noise *(white and black dots)* to the image. *cycles* is the measure of salt and pepper noise *(no of iterations)*

Probabilistic Noise

The probabilistic noise model generates noise according to a predefined probability type.

Here the images are to be in the black-and-white format before implementing the probabilistic noise algorithms. The object used here is Photolab.probabilistic_noise_col(photo). An instance of the probabilistic_noise class has to be made by the user and then various functions below it can be accessed.
Photolab offers the following probabilistic noise models:

1. gaussian(std_dev): Generates Gaussian[3] noise in the image by taking the standard deviation ***std_dev*** as the input

2. uniform(a,b): Generates uniform[4] noise in the image taking ***a,b*** as the parameters

3. erlang(a,b): Generates erlang[5] noise in the image taking ***a,b*** as the parameters

4. exponential(a): Generates exponential[6] noise in the image taking ***a*** as the parameter

[3] Refer https://en.wikipedia.org/wiki/Gaussian_noise
[4] Refer https://aishack.in/tutorials/generating-uniform-noise/
[5] Refer https://en.wikipedia.org/wiki/Erlang_distribution
[6] Refer https://en.wikipedia.org/wiki/Exponential_distribution

Image Restoration

Image Restoration is the process of correcting defective images corrupted by noise and other factors. Here the images are to be in the black-and-white format before implementing the image restoration algorithms. The object used here is Photolab.img_restore(photo, M, N). An instance of the img_restore class has to be made by the user and then various functions below it can be accessed. Photolab offers the following image restoration algorithms:

1. median(): replaces the pixel with the median value of its neighbors

2. mean(): replaces the pixel with the mean of the pixel and its neighbors

3. geo_mean(): replaces the pixel with the geometric mean of the pixel and its neighbors

4. min(): replaces the pixel with the minimum value of the pixel and its neighbors

5. max(): replaces the pixel with the maximum value of the pixel and its neighbors

6. mid_point(): replaces the pixel with the midpoint value ([minimum + maximum]/2) of the pixel and its neighbors

7. contra_har_mean(): replaces the pixel with the contra-harmonic[7] mean value of the pixel and its neighbors

[7] Refer https://en.wikipedia.org/wiki/Contraharmonic_mean

Example Python Script

An example of implementing the Photolab library is given below. Here the sobel_avg() function of edge class is used.

Python Script

```
import cv2
import Photolab
photo = cv2.imread("car.jpg",0)
photo = cv2.resize(photo,(1200,800))
e = Photolab.edge(photo,800,1200)
out = e.sobel_avg()
cv2.imwrite("out.jpg",out)
disp = cv2.imread("out.jpg")
cv2.imshow("frame",disp)
cv2.waitKey()
```

Input Image

Output image

References

- https://en.wikipedia.org/wiki/Contraharmonic_mean
- https://en.wikipedia.org/wiki/Exponential_distribution
- https://en.wikipedia.org/wiki/Erlang_distribution
- https://aishack.in/tutorials/generating-uniform-noise/
- https://en.wikipedia.org/wiki/Gaussian_noise
- https://www.geeksforgeeks.org/image-processing/
- https://en.wikipedia.org/wiki/Canny_edge_detector
- https://www.youtube.com/watch?v=C48AI4FvOKE&t=2s
- https://www.youtube.com/@madepython

Author Connect

Mail: sameer2020bvks@gmail.com

Github Repositories: https://github.com/icebergelectronics
https://github.com/bvks2020sameer