Laboratory work 5

Image processing with the OpenCV library and working with the bunch of digital image file

Goal: get knowledge and skills in digital image processing with OpenCV library and processing of a bunch digital images.

Task 1. Work with a bunch of files

1) Create a function that takes the path to a folder on disk as an argument and returns a dataframe with the following information about the images in the folder:

- file name;
- color model;
- format;
- the number of channels in the image;
- file size in megabytes, rounded to two decimal places;
- image width;
- image height;
- full path to the file;
- image thumbnail

Notes:

а) If the file is not an image, do not add information about it to the dataframe.

б) To process a bunch of files you can use built-in modules: `glob` (https://docs.python.org/3/library/glob.html), or `zipfile` https://docs.python.org/3/library/zipfile.html

в) To display a picture in a dataframe, you can use the method described in the article

Example:

```
In [176]: dff = to_dataframe(r'D:\2022\images2')
          HTML(dff.to_html(formatters={'image': image_formatter}, escape=False))
```

Out[176]:

	name	color model	format	channels	size	width	height	full path	image
0	1.png	RGBA	PNG	4	715.38	967	601	D:\2022\images2\1.png	
1	2.jpeg	RGB	JPEG	3	53.60	718	750	D:\2022\images2\2.jpeg	
2	2.PNG	RGBA	PNG	4	483.61	716	450	D:\2022\images2\2.PNG	
3	3.jpg	RGB	JPEG	3	66.55	730	411	D:\2022\images2\3.jpg	

Task 2. Create a poster from the images in the folder **using the OpenCV library**.

1) Create a function that takes following arguments:

- the path to the folder;
- name of output image file;
- the number of images in a row;
- the number of images in a column

and returns the poster image.

We assume that the poster is rectangular and the number of pictures in the poster is $a \times b$, where a is the number of columns and b is the number of rows.

2) Create a function that converts the picture into a square with the maximum possible side size (for example, for the picture 800×533, the output size should be 533×533). We crop image on both sides, that is, we calculate the size from the image center.

3) All pictures from which the poster is created must be of the same size, that is, after cropping, the size must be changed, for example, by 500×500.

4) To add images to poster, pick them randomly. Pictures may be repeated.

5) "Filter" the image with red, green, blue, yellow, magenta, or cyan colors, which are also randomly selected and may repeat.

6) Provide 3-4 versions of posters saved in files on disk.

Content of the report:

As a report, present a Jupyter notebook with a description of tasks, code and output images.

To perform the work, you can use the provided images, or any others you like.

Examples:

1)

```
In [232]: img = to_poster(r'D:\2022\images2', 'post3x2.png', 3, 2)
          img
```

Out[232]:

2)

```
In [240]: img = to_poster(r'D:\2022\images2', 'post2x2.png', 2, 2)
          img
```

Out[240]:

3)

```
In [233]: img = to_poster(r'D:\2022\images2', 'post3x3_1.png', 3, 3)
          img
```

Out[233]:

4)

```
In [243]: img = to_poster(r'D:\2022\images2', 'post2x3_1.png', 2, 3)
          img
Out[243]:
```

Hints

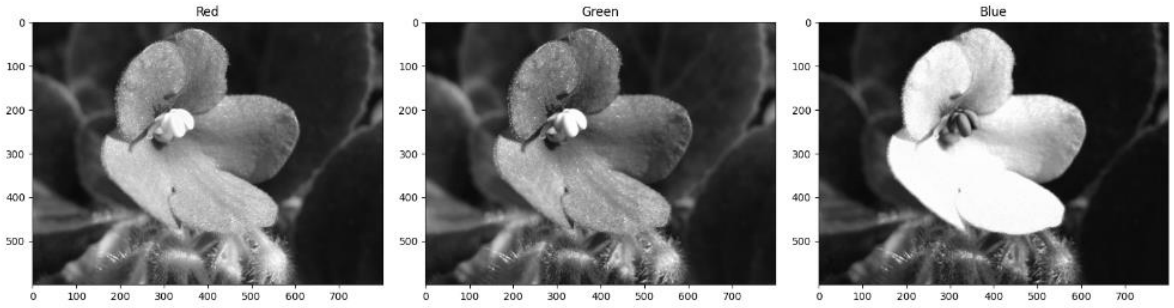
1. **How to split color channels in open CV**

Way1. Using `cv.split()`

```
b, g, r = cv.split(img)
fig, ax = plt.subplots(1, 3, figsize=(16, 8))
fig.tight_layout()

ax[0].imshow(cv.cvtColor(r, cv.COLOR_BGR2RGB))
ax[0].set_title("Red")

ax[1].imshow(cv.cvtColor(g, cv.COLOR_BGR2RGB))
ax[1].set_title("Green")

ax[2].imshow(cv.cvtColor(b, cv.COLOR_BGR2RGB))
ax[2].set_title("Blue")
```

Result:

```
Out[11]: Text(0.5, 1.0, 'Blue')
```



Way 2. Using Numpy features

```
b = img[:,:,0]
g = img[:,:,1]
r = img[:,:,2]
```

2. How to create a “zero channel”

```
b_zero = b.copy()
b_zero[:] = 0
print(b_zero)
```

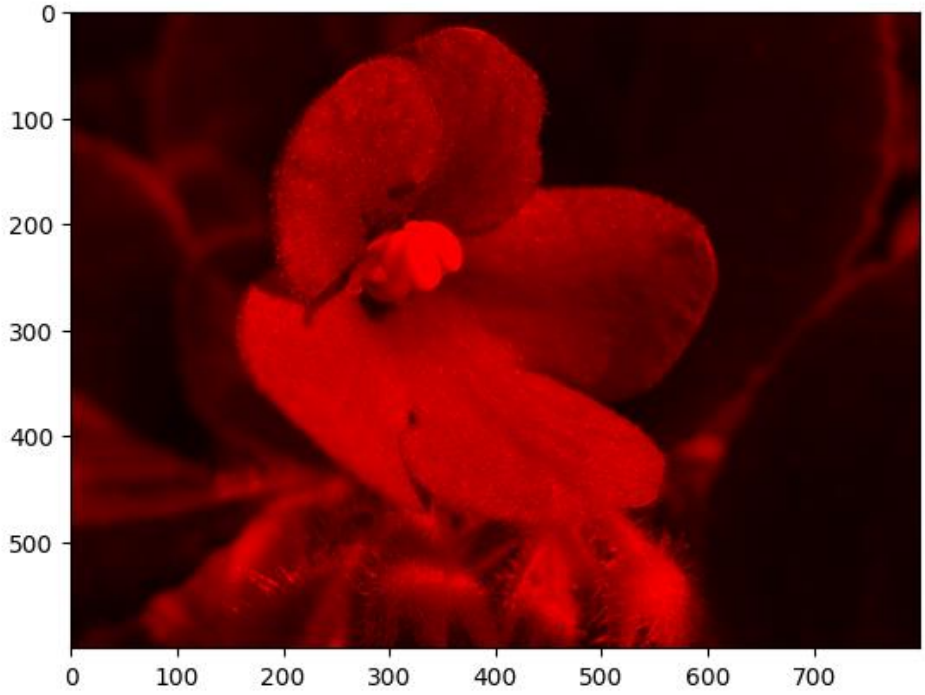
Result:

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

3. How to merge color channels together to have a “one-simple-color” image

```
new_image = np.dstack((b_zero, g_zero, r))
plt.imshow(cv.cvtColor(new_image, cv.COLOR_BGR2RGB))
```

Result:



4. **How to merge color channels together to have a “two-simple-colors” image**

```
new_image_2colors = np.dstack((r, g_zero, b))
plt.imshow(cv.cvtColor(new_image1, cv.COLOR_BGR2RGB));
```

Result: