Practical 1:
Introduction to Python Image Processing Libraries

Unit Coordinator & Lecturer: Dr. Maryam Haghighat

This short tutorial aims at introducing some basic fundamentals of Python Image Processing Libraries. In particular, useful information and functions will be presented. You are invited to test all these functions on different images to get a good understanding of their purpose, syntax and effect on images or arrays. Throughout this document, you will also be asked to realize some additional tests or answer specific questions using the possibilities of the libraries mentioned here.

In this lab, we will refer to two common Python libraries: OpenCV (cv2) for image processing; Matplotlib (plt) for plotting; NumPy (np). You may use the images you like to test and process, however, make sure they are monochromatic (i.e. grayscale) images. If you are using a colour image, consider the conversion to grayscale via the function `cv2.cvtColor(f, cv2.COLOR_BGR2RGB)`.

1 Digital Image Representation

1.1 Coordinate Conventions

Consider a sampled image $f(x, y)$, with M rows and N columns (size $M \times N$). In many image processing references, as in the lectures notes of this course, the image origin is set at $(x, y) = (0, 0)$, which means that x ranges from 0 to $M - 1$, and y from 0 to $N - 1$, in integer increments. In OpenCV, an image matrix class can be created with the help of NumPy. Note they integrate seamlessly `f = np.zeros((rows, columns, channels), dtype=np.uint8)`.

1.2 Images as Matrices

Given the coordinate convention of OpenCV, a digital image of 1 channel can be represented as the following matrix:

$$f = \begin{bmatrix} f(0,0) & f(0,1) & ... & f(0,N-1) \\ f(1,0) & f(1,1) & ... & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & ... & f(M-1,N-1) \end{bmatrix}$$

2 Reading Images

Images are read into a Python environment using the function `cv2.imread`:

```
cv2.imread('filename')
```

`filename` being a string containing the complete file name of the image file (including extension). For example:

```
f = cv2.imread('myimage.jpg')
```

reads the JPEG image `myimage` into an array `f`. You can use the function `f.shape` to see row, columns, and channels dimensions of the image.

3 Displaying Images

Images are displayed in Python using the function `cv2.imshow(f)`, however, in interactive notebooks such as a Jupyter Notebook, OpenCV might not be able to generate a GUI. For this reason, in the lab we are going to use the Matplotlib function `plt.imshow(f)`.

For images gray scale or binary, when using Matplotlib for plotting, we need to specify `cmap='gray'` for correct color map:

```
plt.imshow(f, cmap='gray')
```

For binary images non-logical (e.g. 2 values, 0 for black, and 255 for white) we also need to specify `vmin` and `vmax`:

```
plt.imshow(f, cmap='gray', vmin=0, vmax=255)
```

4 Colour Convention

Attention when using Python image processing libraries because of colour conventions. For example, OpenCV reads a coloured image in format BGR, while Matplotlib reads in format RGB. This means that if we read an image in OpenCV with `f = cv2.imread('filename')` and then plot it with `plt.imshow(f)`, we will visualise the image in the BGR convention. To mitigate the mismatch, we need to convert the image to RGB before plotting:

```
f_rgb = cv2.cvtColor(f, cv2.COLOR BGR2RGB)
```

Test the effects in Python.

5 Writing Images

Images are written to disk using the function `cv2.imwrite`:

```
cv2.imwrite('filename', f)
```

where `filename` includes a recognized file format extension (check the accepted formats). The `cv2.imwrite` function can have other parameters, depending on the file format specified. For example, to save an image in the JPEG format, you can specify a quality parameter which will set the level of compression achieved:

```
cv2.imwrite('filename', f, [cv2.IMWRITE_JPEG_QUALITY, 100]
```

where the compression value is an integer between 0 and 100 (the lower the number, the higher the degradation due to JPEG compression, but the smaller the size of the file).
Realize some tests of quality on different kinds of images by varying the compression value.

6 Converting between Data Classes and Image Types

In Python, we obtain image type conversions by mathematical operation (see above), or by the OpenCV function `cv2.cvtColor(f, cv2.COLOR_BGR2GRAY)`. Another example is via NumPy casting `f.astype(np.float64)`.

7 Data Classes & Image Types

As OpenCV and NumPy integrate seamlessly, the data classes are shared between the libraries. Please refer to Python and NumPy tutorials for the standard data classes (e.g. `np.uint8, np.float64, np.bool, etc.`).

Similarly, the image types align with the data classes, and they are functions of standard mathematical operations. For example, what image type will we obtain following the operation:

```
f > 100 ?
```

Note that Python performs data type conversion automatically; however, be aware to always work with the correct format.

8 Array Indexing

Similarly, array indexing remains unchanged from the standard syntax.

Using only array indexing, realize the following operations:

- Flip vertically an image and display the result.
- Subsample an image (e.g., take one pixel over two).
- Crop an image (e.g., extract the central part and display it).

9 Some Important Standard Arrays

The following functions can be useful to generate simple image arrays to try out ideas, test the syntax of functions during development, or test some image processing algorithms.

- `np.zeros((M,N), dtype=np.<TYPE>)` generates an $M \times N$ matrix of 0s of given type
- `np.ones((M,N), dtype=np.<TYPE>)` generates an $M \times N$ matrix of 1s of given type
- `np.random.rand(M,N)` generates an $M \times N$ matrix whose entries are uniformly distributed random numbers in the interval $[0, 1]$
- `np.random.randn(M,N)` generates an $M \times N$ matrix whose entries are normally distributed (i.e. Gaussian) random numbers with mean 0 and variance 1.

NB: to add more than one channel, we can write `(M,N,C)`.

Use these functions to create some test arrays. Manipulate them with the previous conversion functions.