COMP2005 Laboratory Sheet 1A: Image Processing in Python

OpenCV is an open-source library of functions related to computer vision, image processing and machine learning. Some other notable libraries would be Pillow (PIL) and sci-kit image. In this module, we will be using OpenCV as the main image processing library. You may read the official documentation to better understand the functions available https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.

1. Using Libraries

To use the libraries in Python, they must be imported using *import*. **It is encouraged and good practice to import libraries at the beginning of the file.** *import* is used to access the code in a Python module, which is a Python file containing a set of functions and classes (essentially what a library is). We can also create our own modules to organise code by grouping it, improving code readability and reusability.

- Using *import.* The entire module is imported, and the module must be referenced when used.
- Using *from… import.* The specified functions are imported, and the module does not need to be referenced.
- Using *from… import *.* The entire module is imported, and the module does not need to be referenced. **Note: this is not encouraged and is bad practice as it makes it unclear which module a function is from and pollutes the namespace.**
- Aliasing using *as.* This modifies the name of the module during referencing. It is done when the module's name is too long or has already been used in the program.

In this lab and all future labs, we will be using OpenCV. Please ensure you have installed the OpenCV library, which was discussed in the previous lab sheet.

- Import the OpenCV library using *import cv2.*

2. Whole Image Operations

Reading and writing images is achieved via the *imread* and *imwrite* functions, while *imshow* displays images in a window. Using the image cameraman.tif from the Moodle page, perform the following:

- Read it in using *imread.*
- Display it using *imshow* (Hint: Remember to use *waitKey* to keep the window displayed).
- Write it to a new file using *imwrite.*

shape can provide basic information about the image.

- Use *shape* on the image and print the results. How many values do you get, and what does each value represent? Set the *IMREAD_UNCHANGED* flag to the *imread* function and print the results of *shape* again. Is there a difference, and why?

Image manipulation functions can affect the whole image.

- Use *rotate* to rotate the image by 90 degrees. Display the result.
- Try to rotate the image by 30 and 60 degrees. Can *rotate* be used? If not, why and how can we go about it? Display the results.
- *imshow* displays each image in a separate window, but it can get very messy having so many windows popping up. Try to display the results of the three rotations in a single window (Hint: Use *subplots* and *imshow* from the Matplotlib library, and set the *cmap* parameter of *imshow*).

NOTE: Images read in Python are arrays of type uint8, which range from 0 to 255. Some operations can generate large or negative numbers that cause overflows. You may find it useful to convert to int or float before applying some processes. When you use the *imshow* and *imwrite* functions, please pay attention to the data type as it must be of type uint8. This can be achieved by using *astype* to convert the data type.

Operations can also be applied to every element of an array.

- Make a copy of the cameraman image that is darker or lighter than the original by adding or subtracting a constant to each pixel. Display the results in a single window and use *set_title* to label each image.

3. Arrays in Python

Images are a matrix of numbers whereby each number is a pixel value. In OpenCV, images are stored as NumPy arrays. NumPy is a library which provides support for multi-dimensional arrays and matrices. Although Python on its own has lists, NumPy arrays are much faster, take up less space and provide many convenient operations. We will explore more on the functionalities of NumPy in future labs.

Without using the NumPy library,

- Use array indexing with conditionals to change pixels with intensity values < 128 to 0 and pixels with intensity values >= 128 to 255. Display the results.
- Use array indexing to draw a black cross through the centre of the cameraman image. Display the results. (Hint: Ensure the value used during the indexing is of type *int*).

4. Expected Results

Figure 1: Rotate 90 degrees

Figure 2: Rotate 30 degrees

Figure 3: Rotate 60 degrees

Figure 4: Lighter image

Figure 5: Darker image

Figure 6: Changing pixel intensities

Figure 7: Adding a black cross