Image Processing in Python - Introduction

This practical assumes some knowledge of the python programming language and familiarity with a python development environment. A principal aim is to allow you to develop solutions to very simple image processing problems, using available implementations of core techniques.

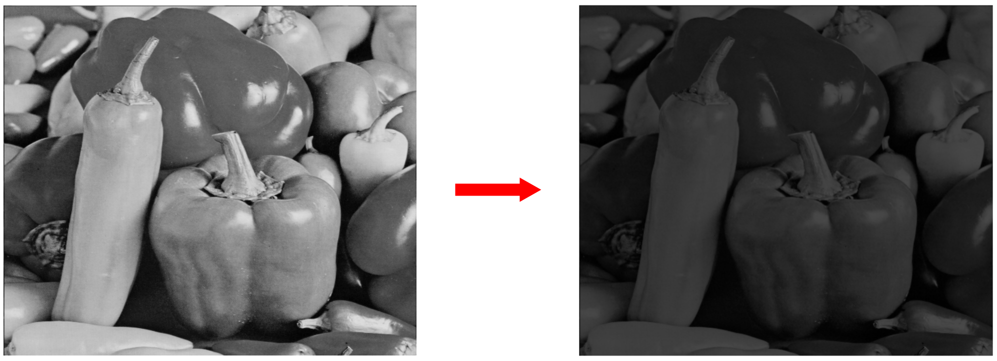
We will be using OpenCV (version 4.1.x) computer vision library with Python 3.7.x. Whilst OpenCV is primarily aimed at computer vision tasks (i.e. image understanding and analysis), it is built on a central image processing and manipulation component that is both well-documented, open-source and free. It is also the only library offering coverage of all of the course topics with interfaces in Python (used here) and Java. It has already been setup for use on the Lab PCs (Windows and Linux).

OpenCV has full documentation available at: https://docs.opencv.org/4.1.1/

Part 1: Let's Start Strong

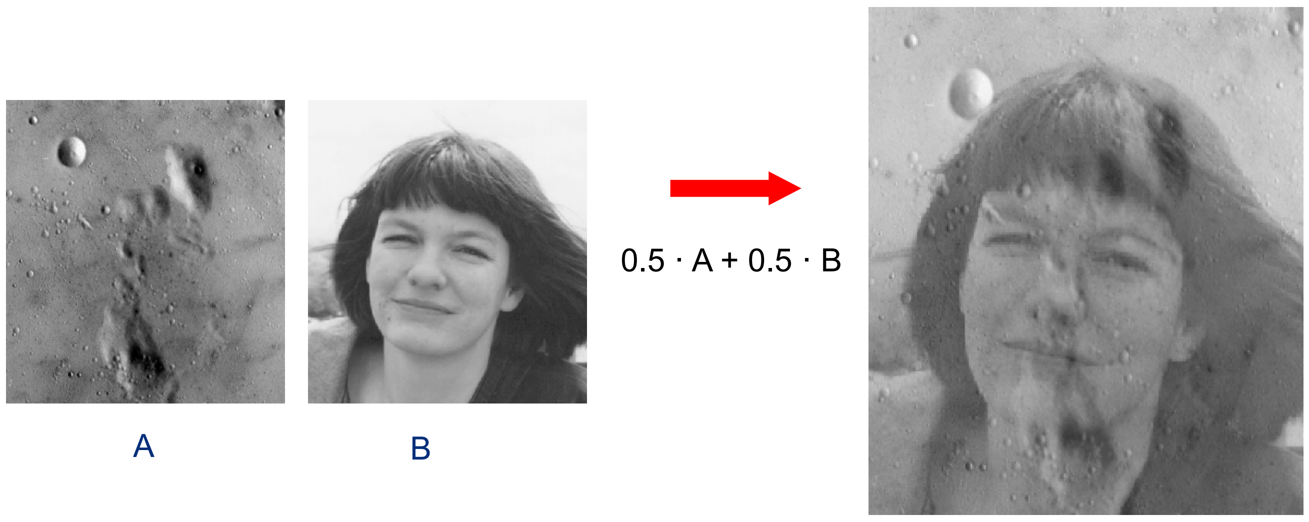
We have been learning about what image processing is and how images are represented and can be dealt with. We have also seen a demo of what image processing can do. So, it might be best to get into some image processing straight away. The exercise is very simple and will only need a few lines of code.

We are going to be working with Arithmetic Operations on images. Remember from the lecture that images are matrices of number so we should be able to perform simple mathematical operations on them. We can thus process (**transform**) an image by manipulating the corresponding matrix. For example, by multiplying an image by 0.33, we can change the brightness of the image (see the figure below). Do you know why?

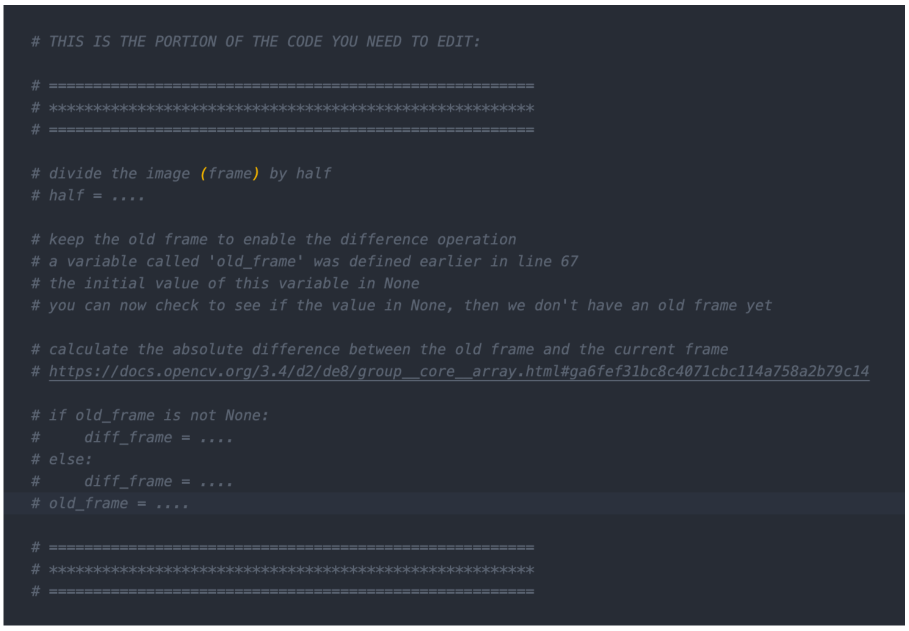


A   0.33 · A

We can also process (transform) multiple images by manipulating the corresponding matrices. For instance, by calculating the average of two images, or possibly the weighted average of two images, we can perform image blending. An example of this can be seen in the following figure.



You should have access to the file arithmetic_transforms.py. Most of the code is already written. You should even be able to run the code but it won't have the results we are looking for. Our goal is to perform two simple arithmetic operations on images received from your webcam. The only portion of the code you need to complete starts from **line 125.**



```
# THIS IS THE PORTION OF THE CODE YOU NEED TO EDIT:

# =======================================================
# *******************************************************
# =======================================================

# divide the image (frame) by half
# half = ....

# keep the old frame to enable the difference operation
# a variable called 'old_frame' was defined earlier in line 67
# the initial value of this variable in None
# you can now check to see if the value in None, then we don't have an old frame yet

# calculate the absolute difference between the old frame and the current frame
# https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga6fef31bc8c4071cbc114a758a2b79c14

# if old_frame is not None:
#     diff_frame = ....
# else:
#     diff_frame = ....
# old_frame = ....

# =======================================================
# *******************************************************
# =======================================================
```

The image coming from the camera is stored in a variable called "frame". First let's divide the image by two and put the result in a variable called "half". Note that if you change the name of the variables without keeping these consistent throughout the rest of the program, there will be errors. Also note that we are looking for integer division only. Can you think of why?

Some guidance for divisions in Python can be found here:
https://www.geeksforgeeks.org/division-operator-in-python/

As a next step, we also want to calculate the difference between every two frames coming from the webcam feed. This will emphasise any movements in the incoming video, which can be a useful analysis tool in many applications.

In order to calculate the difference between the current and previous frame, we can use the absdiff() function in OpenCV. You can read about how it works here:

https://docs.opencv.org/3.4/d2/de8/group__core__array.html#ga6fef31bc8c4071cbc114a758a2b79c14

In order to calculate the difference between the old and the current frames, we need to actually keep the old frame, which is an interesting (but easy-to-solve) challenge on its own.

Once you have completed the code, run it to see what the result is.

Part 2 (if you have time): A Simple OpenCV Example with Python

You should have access to the python example smooth_image.py and an example image of your choice from the internet (put them in the same directory). If you are struggling for inspiration on an example image use peppers.png, or another image from the SIPI database, which provides some of the most commonly-used standard test images used for image processing evaluation at: http://sipi.usc.edu/database/database.php?volume=misc

In the example code (smooth_image.py), change **line 20** of the example (if needed) to reference your image file and run the code. Spend some time understanding the program structure, objects and the flow of control. This program simply loads and displays in a window an image, specified by a string file name, after performing some mild smoothing on it.

Look up the imread() method in the manual to find out how it operates and the image file formats it supports. Like all open-source software, the documentation is not perfect, but this is the real world!

Operations on Images

Next we examine how we use OpenCV functionality to perform specified operations on images. The key concepts to take note of are:

- Automatic initialisation of a new (output) image objects prior to use.

- Common way of interfacing the provided OpenCV functions to images and parameters.

As should be apparent from the source code (smooth_image.py) and running the program, this program applies a smoothing filter (here, Gaussian blurring) to the input image and displays the result. The details of this filter are provided here:
http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

This image filtering technique is common in image processing and is outside the scope of this practical exercise. Right now, we are just using it as an example of an image processing operation.

Programming Tasks

1. Look up the entry for the GaussianBlur() function in the OpenCV manual.

- Explore the effects of varying either of the mask (i.e. kernel) parameters from 5.
- Try larger and smaller values (which must always be odd). What happens?
- How does this affect the direction of the blurring?

2. Extend the program to use multiple window objects – to display the original input image as well as smoothed outputs with different parameters!

3. Investigate the use of the following OpenCV functions by replacing the GaussianBlur() operation in the code:

(a) flip()

(b) Canny() – an edge detection method.

You will need to first convert the image to grayscale using:

cvtColor(inputImage, cv2.COLOR_BGR2GRAY)

N.B. Suitable threshold parameters for the canny operator would be 10 and 200. You can leave the other parameters at the suggested defaults from the manual (or play around and experiment!!)

(c) resize()

N.B. You can just specify the fx and fy resize scales and let the function compute the required destination image size.

Investigate the effects of changing type of interpolation used (for example between cv2.INTER_NEAREST and cv2.INTER_LINEAR).

Use the OpenCV manual for reference. Experiment with different parameters and different images.

Part 3: Further Work (optional)

A good set of open source tutorials exist for the use of OpenCV with Python here:

https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

The following tutorials re-affirm and extend the topics covered in this practical. They are not essential but are designed to reinforce your learning of the key concepts.

- Getting Started with Images:

  https://docs.opencv.org/master/db/deb/tutorial_display_image.html

- Basic Operations on Images:

  https://docs.opencv.org/master/d3/df2/tutorial_py_basic_ops.html

- Drawing Functions in OpenCV:

  https://docs.opencv.org/master/dc/da5/tutorial_py_drawing_functions.html

- Mouse as a Paint-Brush:

  https://docs.opencv.org/master/db/d5b/tutorial_py_mouse_handling.html