Python Photoshop

Martin Roelfs & Peter Kroon

April 6, 2020

1 Python Photoshop

1.1 Introduction

Editing pictures is an essential skill for the Facebook generation. In this exercise you will be using Python to do some basic image editing, which is similar to the way a software suite like Photoshop would handle things at the core level. We will be using numpy to import images as numpy arrays. Numpy can open a file in several modes, including RGB mode and grey scale.

RGB stands for Red, Green and Blue, which are the primary colours. Yellow is not! This usually baffles the average girl I tell this in the club. Each pixel has three values: one for each colour. Because every colour in a bitmap is one byte long, it is a value in the domain [0, 255]. 0 means no colour whatsoever, whereas 255 means maximal colour.

If you open an image in RGB using numpy, each pixel is accessible as an RGB array. In this way, [0 0 0] would be black, [255 0 0] red, and [255 255 255] would be white. This means that an image will be a three dimensional array! The first dimension is the height of the image, the second the width, and the third the colours.

Note that if some image formats (png in particular) result in arrays with the shape (Y, X, 4). This fourth "colour" describes transparency. For the purpose of this exercise, just remove this column.

Note that some image formats result in colour values normalized between 0.0 and 1.0. Convert these to integers using the following:

```
image = (image * 255).astype('uint8')
```

2 Exercises

2.1 Exercise 1: Reading images

The first step is to read in an image as a numpy array. `matplotlib` has all the required functionality to read, write and show images. Look through the matplotlib documentation to find the appropriate functions.

1. Read an image.
2. Make sure the image you read is a numpy array.
3. Make sure the shape of your numpy array is (Y, X, 3).
4. Make sure the dtype of your numpy array is 'uint8'.

2.2 Exercise 2: conversion to grey scale

To discover how to make an image black and white, we will convert an image by hand. Do not use some ready-made function, as you won't learn the mechanics of how an image works if you do.

The first question you should answer is how grey is represented as RGB value. Hint: black is [0 0 0], and white is [255 255 255], and grey is in between black and white. What about dark grey? Light grey?

The second question you should answer is: how do you decide how light (or dark) a pixel should be?

Make sure your function returns an array of the shape (Y, X, 3), and has dtype uint8.

1. Turn the image into a grey scale image using only numpy functions.
2. Save your image to the hard drive.

2.3 Exercise 2: Thresholds

Very similar to the previous exercise, but this time we will round our pixel values, so that everything above a certain threshold will yield 255, and below that 0. This way we can make a silhouette. How will you decide which pixel should be white, and which one should be black?

1. Make an image where every pixel from the original is rounded up or down. Display the image, it should be a silhouette!
2. Make a silhouette such that exactly half the pixels are black, and half are white, irrespective of the initial image. See also: `numpy.median`.
3. Save the image.

2.4 Exercise 3: Finding edges

Now for something a bit more complicated. We're going to try and find edges in an image. Make your first test case simple, and pick an image with very clear edges, and not much else. First thing we need is some definition for what an "edge" is. To keep it manageable we'll say that an edge is any pixel where the colour changes more than a certain threshold.

1. Find the distances between all neighbouring pixels. This probably involves 2 steps, where first you find the difference per colour channel, and then combine those to get a single number (e.g. Euclidean: $d = \sqrt{\Delta r^2 + \Delta g^2 + \Delta b^2}$).
2. Find all pixels where the distance is more than a threshold. These pixels are part of an edge!
3. Give those pixels a distinctive colour, such as black.
4. Save the image.

2.5 Exercise 4: Be creative

Using your new found knowledge of image editing, make a Python script that performs any image editing you like! There are a lot of possibilities. To name a few: making a stamp function which sets the value of a pixel on the basis of those around it. This type of operation is commonly used to remove acne. Make your entire image more blue to resemble The Desert of the Real, or more green to resemble The Matrix. Add images together. Crop an image. Pixelate a face. Blur an image. The possibilities are endless. Submit the code you are proud of.