COMP 1012: Lab 10

August 10th to August 14th, 2020

Material covered

- Numpy 2d-arrays
- Functions

Question 1 (Required)

In this question, you will be asked to implement 2 functions related to blurring an image. This lab is a continuation of one of the examples from Week 9.

How do you blur an image? There are many ways, but here is an easy way of doing it, known as a *box blur*. Consider a pixel at location (x,y) and assume it has 8 neighboring pixels. This pixel's color is specified by its red, green, and blue components, where each is an integer between 0 and 255. To compute the new R(ed) component for the pixel at location (x,y),denoted by , $Red(x,y)$, you simply compute the average of the red components of the pixel and its 8 neighbors. That is

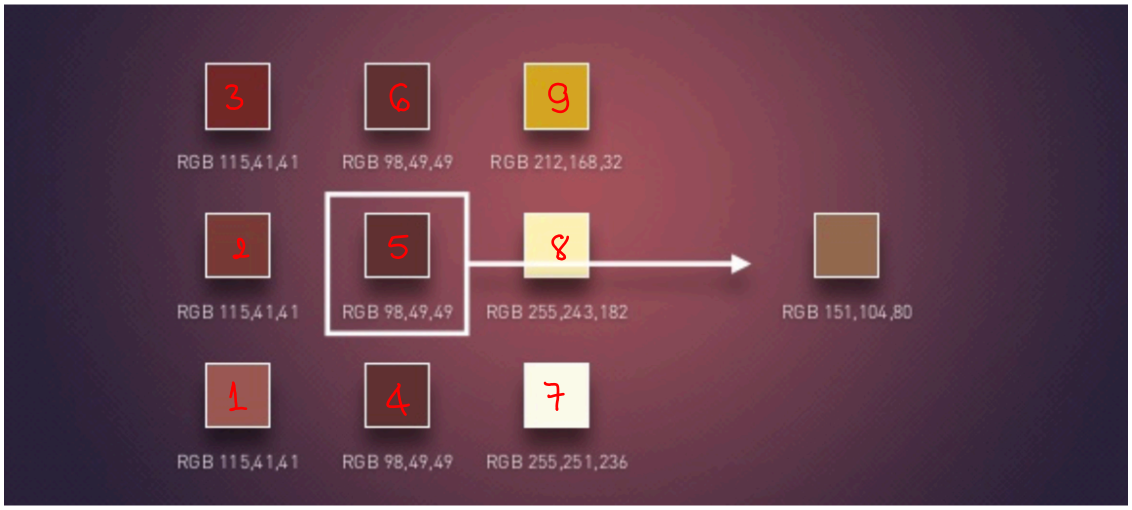
$$Red(x,y) \leftarrow \left(\sum_{i=-1}^{1} Red(x-1,y+i) + \sum_{i=-1}^{1} Red(x,y+i) + \sum_{i=-1}^{1} Red(x+1,y+i) \right) / 9.$$

If you expand this out, you wil get:

$$Red(x,y) \leftarrow (\overset{1}{Red(x-1,y-1)} + \overset{2}{Red(x-1,y)} + \overset{3}{Red(x-1,y+1)} + \overset{4}{Red(x,y-1)} + \underset{5}{Red(x,y)} + \underset{6}{Red(x,y+1)} + \underset{7}{Red(x+1,y-1)} + \underset{8}{Red(x+1,y)} + \underset{9}{Red(x+1,y+1)})/9.$$

You use the same calculation for the G(reen) and B(lue) components of the pixel at location (x,y).

The following figure illustrates the calculation of the new RGB values for a pixel using this formula. You should do the calculation by hand to ensure you understand how the new values are obtained.



In the figure, the pixel in the middle originally has color (98,49,49). After apply the averaging formula given above, its new color is (151,104,80). Here the color is given as a 3-tuple specifying the R,G,B values respectively.

For more information on this blurring technique, see here

What you need to Do

Download the file lab10.py which contains some code that you will complete.

1. Complete the `blur(origImg)` function that will blur the image given by `origImg`. Make a copy (call it `img` for instance) of `origImg` and blur and return the copy. You don't need to blur pixels that are on the edge of the image (those do not have 8 neighbors). You'll need to apply the blurring formula to each pixel in the image. Make sure that when apply the formula, you are using the values from the original image `origImg` to do the computations and storing the results in the copy `img`.

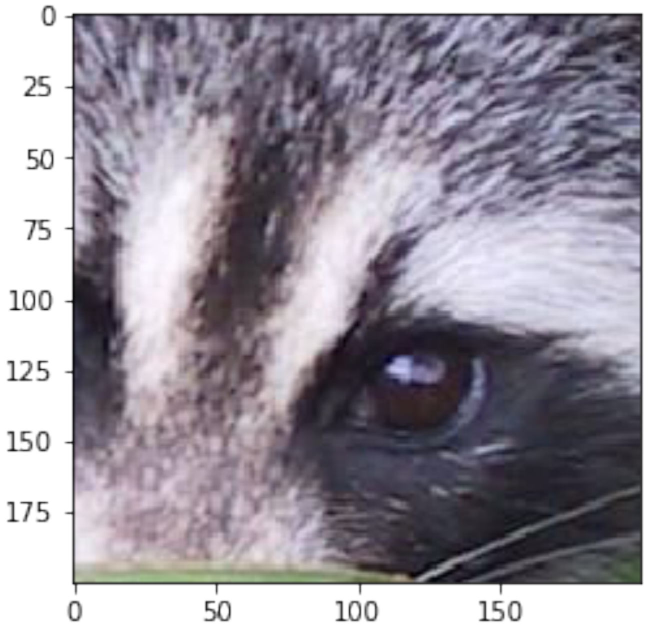
2. Complete the `circleBlur(origImg)` function that will blur the part of the image that is inside the circle of radius given by the minimum of the width and height of the image. Pixels outside the circle should be left alone. Again, make a copy of `origImg` and apply your alterations on the copy and return the copy. Use the function `circle` provided as a guide.

Important As stated above already, you should **NOT** modify the image that is passed into the function. Instead modifications must be on a copy of the image.

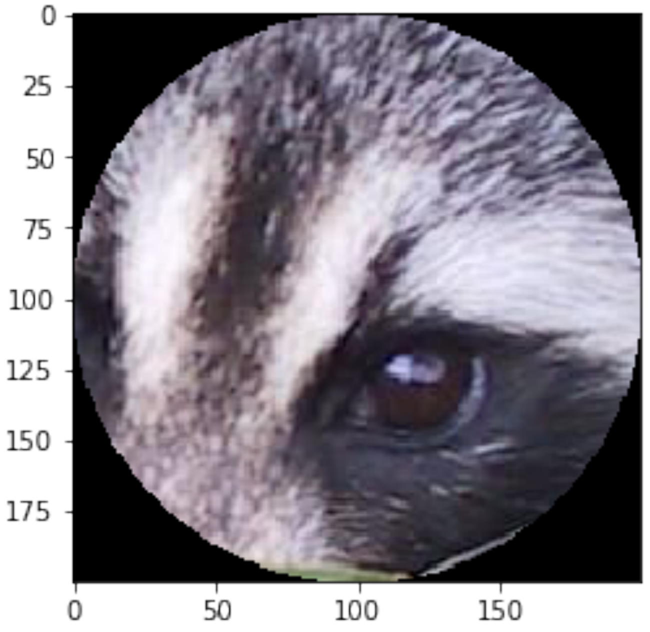
It may be easier to create a third function `blurPixel(img,x,y)` that will return the blurred RBG values for pixel (x,y) in the image `img`.

Output

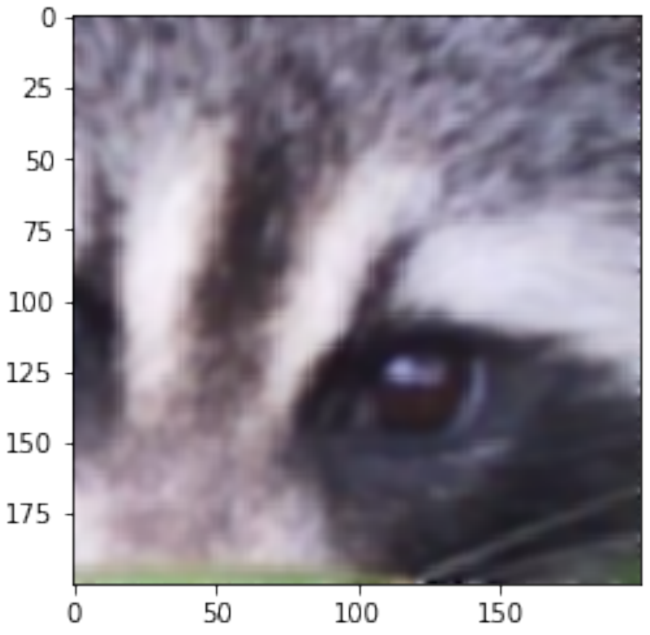
By running `lab10.py`, you should generate 4 images. The first image displayed is the original (clipped) image as give below.



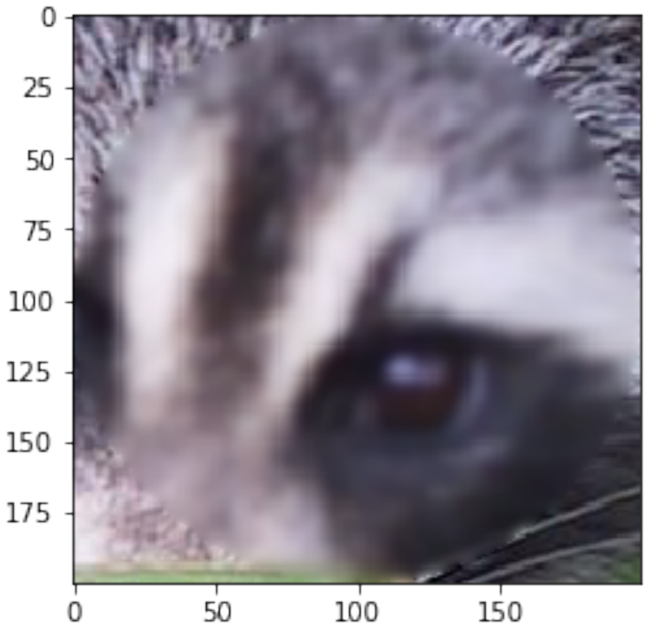
The following image is displayed with the pixels outside the circle blacked out. Note you don’t need to write this function as it is provided to you.



The next image displayed should be your blurred image, which should look like the one below.



Finally, the last image displayed should be the image with the inside of the circle blurred. You should be able to see that the pixels inside the circle are blurred while the pixels outside the circle look like those in the original image.



Note: Depending on the speed of your computer, it made take a bit of time for the blurring to finish. For testing, you can play around with the values of `N` and `M` to adjust the number of times `blur` and `circleBlur` are executed.

Jupyter

- For those of you who want to use a Jupyter notebook, you can copy the code from `lab10.py` into a cell and add the line `%matplotlib inline` at the top of the cell.
    - This will allow you to display images.