2D spatial cross-correlation

This week, we expand what we have learned about 1D spatial cross-correlation into 2 dimensions. The theory and method behind performing cross-correlation in 1 and 2 dimensions is the same. The task is more of a computing challenge, where you will have to start thinking about matrices, rather than 1D arrays.

Images are just 2D matrices of data. Each element of the 2D matric is numeric value, and your screen translates these values into light intensity. If using RGB color, the image is actually three 2D matrices, one matrix of intensity for each pixel color (red, green, blue). We are asking you to do pattern matching for grayscale image data in this class, and so only a single 2D matrix will be used.

You will need to be able to convert RBG images into greyscale for this class. There are many methods to do this. A suggestion is the 'pillow' package for python, or if you want to do it yourself, you can merge the three 2D matrices into a single greyscale 2D matrix by calculating each pixel value, using greyvalue `= 0.2989 R + 0.5870 G + 0.1140 B.` If you want to work with color images, you are allowed, but it will take a bit more effort on your part. This is potentially an extension project and could yield better results than greyscale.

Question 1

Packages such as 'pillow' or 'scipy' can be used.

a) Write a function to load a PNG file into a numpy array.

b) Write a function which turns a 3 x 2D (RGB) image into greyscale.

c) Use your functions to import 1) the rocketman wally (pattern) and 2) the where's wally map (template)

Question 2

Write 2D versions of your 1D spatial cross-correlation functions.

You will have to think about the horizontal **and** vertical offset of the template when scoring a given location in the template. You can either go row by row, or column by column as you shift the pattern through the template.

Report the best *lag* with reference to the top left hand corner of the template. Ie if you find the best location for the pattern is exactly the top left corner, your offset is [0, 0]. If it is one pixel (row in 2D matrix) down, your *lag* is [0, 1] (if using [x, y] coordinate system). 10 pixels to the right and 4 pixels down would be [10, 4].

The ideas behind the scoring of a given location still apply. Compute the elementwise product of the (now) 2D pattern and 2D template slice, normalised by the array energy (matrix energy now) of the pattern and template slice.

Remember to think about how you will deal with the edges of the template, as either zero padding or another method will be necessary.

Question 3 (submission question)

Find the best location for Wally in the Where's wally template using your new functions.