CPS109 - Fall 2018

Assignment 2: Where's Waldo?

template image

search image

Figure 1: Where's Waldo?

The aim of this assignment is to give you practice writing and calling functions, using (nested) loops ... and find Waldo! To do the assignment, you need knowledge up to Chapter 6 in the textbook, *Introduction to Computing and Programming in PYTHON*, by Guzdial and Ericson. The assignment was designed by Dr. Derpanis, who does research in computer vision and teaches the computer vision course, CPS843, which you can enjoy after you have taken the prerequisites: CPS305, MTH210, MTH108 and MTH207).

In this assignment we consider an instance of the *template matching* problem. Template matching is an image processing technique for finding a small part(s) of an image that matches a template image. It can be used in a variety of real world situations, including as a component of a quality control system in manufacturing, a way to autonomously navigate a mobile robot and image manipulation (e.g., blurring an image and extracting edges from an image). Your task is to find Waldo, given by a template image, in a large cluttered image, given by the search image, see Fig. 1.

In the following, we use S[x, y] to denote the scene or 'search' image, where [x, y] represent the pixel coordinates in the search image. We use T[x, y] to denote the template and its coordinates. Both images are assumed to be grayscale.

We then simply move the top-left corner of the template T[over each point (x, y) in the search image and calculate the sum of the absolute differences (SAD) between the pixel values in S[x,y] and T[x', y'] over the area spanned by the template and store this value in the corresponding [x, y] position in the output image. Upon completion of this process, the [x, y] position in the output image with the minimum value corresponds to the best match of the template in the search image.

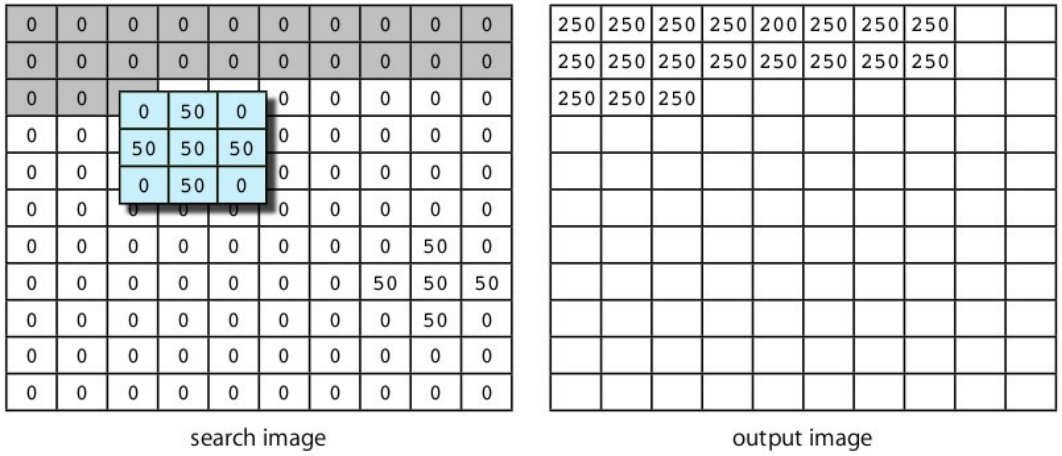

Figure 2: Overview of template matching. The blue image overlaid over the search image represents the template image. The gray search pixels indicate those that have been traversed. The output image is computed by "sliding" the template over the search image from left-to-right, top-to-bottom and computing the SAD match between the portion of the search image under the template image.

In other words, for each position [x, y] in the search image (see Fig. 2 for an overview):

1) Place the template over the search image such that the top left corner of the template is aligned with the current search image position.
2) Compute the match score between the template and search images:
    1) Subtract the luminance of the pixel in the search image from the luminance of the corresponding pixel in the template.
    2) Take the absolute value of the difference and add it to a running sum.
3) Store the sum as the luminance in the output image at position (x, y).

Careful: do not compute the match score around the right and bottom edges of the image where the template would not completely lie within the search image, since there would be index out of bounds exceptions. Instead, set the values of the output image for the undefined regions to some large value. See Figures 3 and 4.

Images: On D2L along with this pdf description file will be images scene.jpg, waldo.jpg, tinyscene.jpg, tinywaldo.jpg. The way the program is described here, the computer can find tiny-waldo in tiny-scene in under 2 minutes, but it would take hours to find waldo in scene. You can work with the tiny versions first, and then if you have time and energy, you can do a more efficient program that works quickly on the original sizes as a bonus.

Your task is to implement:

1) A template matching function, **compareOne(template, searchImage, x1, y1)**, that takes as parameters two images: (i) template and (ii) searchImage, and the coordinates in the search image where the top left pixel of the template is overlayed for matching. The function returns an integer which is the sum of the absolute value of the differences in luminance, as discussed above.
2) A function **compareAll(template, searchImage)** which returns a 2D array containing the computed template match scores for all of the positions in the searchImage. Hint: inside this compare() function, you can initialize the array that the function returns as follows:
   matrix = [[BIG for i in range(W)] for j in range(H)]
   where BIG is some big number, W is the width of the searchImage, and H is the height of the searchImage.
3) A function **find2Dmin(matrix)** that returns the coordinates (minrow, mincol) of the position in array matrix where the value is minimum. Hint: inside find2Dmin() when you are ready to return the minimum row and column, you can do the return as follows:
   return (minrow, mincol)
4) A function **displayMatch(searchImage, x1, y1, w1, h1, color)** that takes as parameters the search image, the position of the target (x1, y1), the width and height of the target (w1, h1), and a color. The function puts a retangle on the searchImage at (x1, y1) so that it would surround the target image (in our case, Waldo). Make the border of the rectangle have width 3 pixels and in the given color.
5) A helper function **grayscale(picture)** that converts a color picture to grayscale. You can use the simple (r + g + b)/3 for the luminance.
6) A function **findWaldo(targetJPG, searchJPG)** that takes in the images and calls your other functions to end up showing the match (using displayMatch()).

Marking rubric (out of 15)

[1 point] Documentation: put comments in your python code and make it readable
[1 point] Coding style: Use of proper indentation, variable names that correspond with their meaning, constants rather than magic numbers (if there are any).
[1 point] Driver function **findWaldo()**.
[6 points] Template matching functions: **compareOne(), compareAll()**.
[3 points] Find minimum match score function, **find2DMin**().
[3 points] Display detected template function, **displayMatch**().
[3 points] Bonus: Write a version of this program that is fast enough to find Waldo in a few minutes in the original size images, scene.jpg and waldo.jpg.

Submit:
Submit on D2L a single python file called a2.py containing your code and your file image with the box around Waldo. If you do the bonus version, then submit a second file a2bonus.py for that one.

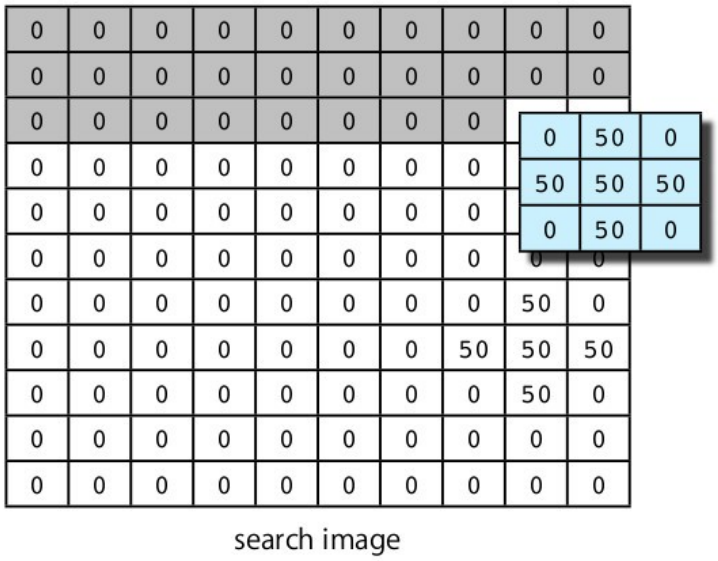

Figure 3: Undefined template match score. The blue image partially overlaid over the search image represents the template image. The gray search pixels indicate those that have been traversed. Since there are template pixels beyond the boundary of the search image, the match score computation is undefined.

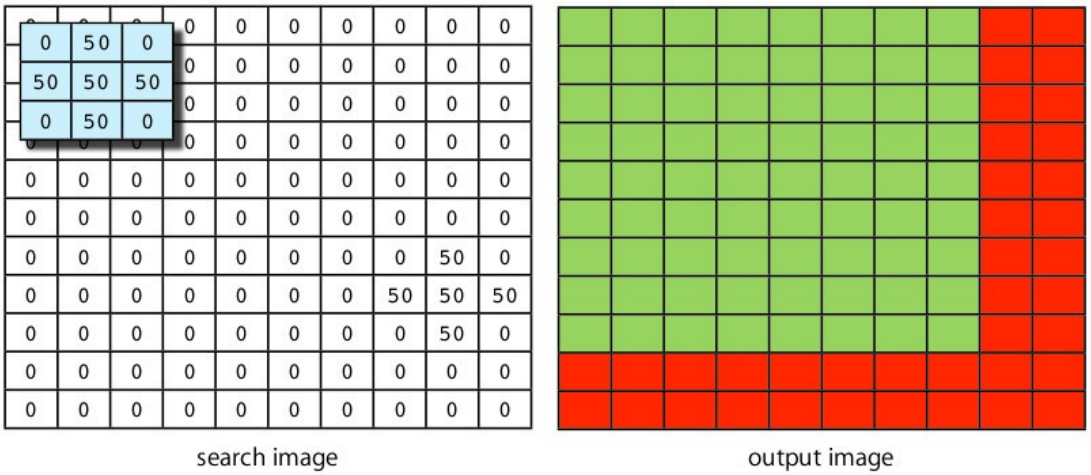

Figure 4: Undefined regions for match computation. The blue image overlaid over the search image represents the template image. The green and red pixels in the output image denote the regions where the match score is defined and undefined, respectively.