Assignment 1 Report

Kexin Ding

1. Problem Description

We try to design an easy and effective algorithm to find an object in an image. For the dataset, There are different sets of object images. The “big image” for each set is different, but for the object images in the same set, they have the same “big image.” For the algorithm, we need to make sure it finishes all object detection in a limited time and matches as a correct position(coordinate) in the big image as we can.

For this correct position, it can be an approximate or an exact coordinate of the top-left point(the start point). Then we will calculate the similarity between the region we found and the ground truth to get an evaluation value. And if the similarity is larger than the set threshold, the test code will evaluate our detection is successful in this object image and move on to the next object image.

2. Detailed Solution

Object detection algorithm:

1) Load a big image and object image as two matrixes
2) Transfer them from RGB image to greyscale image
3) Sampling the greyscale big image and object image on the same scale. Moreover, in this step, we need to set different sampling parameters for big images in a different size. Like a large size image will have a smaller sampling degree parameter(it means sampling more). And a smaller image will have a larger sampling degree parameter(it means sampling less).
(this step can decrease the running time so that we can pass all test samples)
4) Traversing each pixel point(as a start point) and cropping a matrix which has the same size as the object image matrix.
5) Calculating the value of the correlation between the object image matrix and cropped matrix
6) Stopping searching for this object image when we find the value of correlation is larger than 0.9
7) Return the coordinate of the start point of the cropped matrix(Important!!! We need to transfer the coordinate to the original scale before we return it.)
8) Repeat (1) to (7) until the algorithm passes all test samples or the running time larger than the limited time.

3. Evaluation

Euclidian distance will be calculated between actual object location and predicted location on holdout test sets. Moreover, running time can also be evaluation metrics for the algorithm. If the running time is too long, the code will stop when the running time near to the limited time and this case cause we fail to test all samples.

Hence finding a tradeoff between running time, scale degree, and the value of correlation can be an essential issue in this assignment.

4. Result

This is the best result I got.

```
>> main
1,1 - alg(2.523129e+01,1.009251e+01) vs gt(24,9) - 1.015625 sec -> total pt [1]
1,2 - alg(3.027754e+01,8.578638e+01) vs gt(30,85) - 1.781250 sec -> total pt [2]
1,3 - alg(8.410429e+01,1.850294e+01) vs gt(84,18) - 2.546875 sec -> total pt [3]
1,4 - alg(6.728343e+00,2.287637e+02) vs gt(6,228) - 3.296875 sec -> total pt [4]
1,5 - alg(5.046257e+00,2.590412e+02) vs gt(4,258) - 4.125000 sec -> total pt [5]
1,6 - alg(3.027754e+01,2.792262e+02) vs gt(29,278) - 4.984375 sec -> total pt [6]
1,7 - alg(2.186712e+01,5.264929e+02) vs gt(21,526) - 5.671875 sec -> total pt [7]
1,8 - alg(3.532380e+01,4.642557e+02) vs gt(34,464) - 6.453125 sec -> total pt [8]
1,9 - alg(0,0) vs gt(76,448) - 7.203125 sec -> total pt [8]
1,10 - alg(1.295206e+02,4.945332e+02) vs gt(129,494) - 8.046875 sec -> total pt [9]
2,1 - alg(1.230769e+02,1.153846e+02) vs gt(122,112) - 12.750000 sec -> total pt [10]
2,2 - alg(3.038462e+02,5.653846e+02) vs gt(301,562) - 15.562500 sec -> total pt [11]
2,3 - alg(1.423077e+02,2.034615e+03) vs gt(139,2031) - 26.812500 sec -> total pt [12]
2,4 - alg(400,2.115385e+03) vs gt(399,2113) - 29.171875 sec -> total pt [13]
2,5 - alg(150,1.303846e+03) vs gt(146,1301) - 35.937500 sec -> total pt [14]
2,6 - alg(300,1.230769e+03) vs gt(298,1227) - 38.703125 sec -> total pt [15]
2,7 - alg(1.311538e+03,1.153846e+02) vs gt(1310,109) - 41.625000 sec -> total pt [15]
2,8 - alg(1.315385e+03,2.769231e+02) vs gt(1312,273) - 43.500000 sec -> total pt [15]
2,9 - alg(9.807692e+02,6.307692e+02) vs gt(978,629) - 44.843750 sec -> total pt [16]
2,10 - alg(7.615385e+02,9.576923e+02) vs gt(761,955) - 47.140625 sec -> total pt [17]
3,1 - alg(6.026963e+01,6.344171e+01) vs gt(58,61) - 47.437500 sec -> total pt [18]
3,2 - alg(1.522601e+02,8.564631e+02) vs gt(150,854) - 47.703125 sec -> total pt [19]
3,3 - alg(4.060270e+02,8.406027e+02) vs gt(405,839) - 48.046875 sec -> total pt [20]
3,4 - alg(3.806503e+02,1.078509e+02) vs gt(381,105) - 48.484375 sec -> total pt [21]
3,5 - alg(6.122125e+02,5.392546e+02) vs gt(610,536) - 48.796875 sec -> total pt [22]
3,6 - alg(2.727994e+02,5.709754e+01) vs gt(272,53) - 49.156250 sec -> total pt [23]
3,7 - alg(4.504362e+02,4.091990e+02) vs gt(450,406) - 49.484375 sec -> total pt [24]
3,8 - alg(6.566217e+02,7.232355e+02) vs gt(656,722) - 49.828125 sec -> total pt [25]
3,9 - alg(4.440920e+01,5.804917e+02) vs gt(44,579) - 50.187500 sec -> total pt [26]
3,10 - alg(3.330690e+02,2.569389e+02) vs gt(332,255) - 50.468750 sec -> total pt [27]
```