TECHNICAL UNIVERSITY OF CRETE

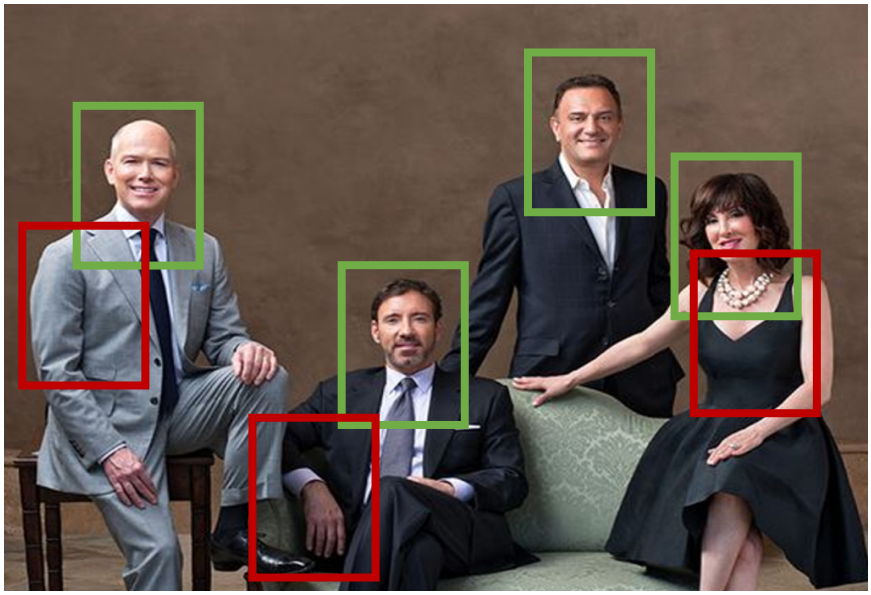
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

CS418: COMPUTER VISION

Instructor: Effrosyni Doutsi

Assignment 3: Detection

Due date: Exam Day



This assignment is adapted from CVRL works.

In this assignment, you will develop an object detector based on gradient features and sliding window classification. You will be provided by some images to be tested and a skeleton concerning how to build your code.

Description of the Steps you need to follow

1. **Image Gradient:** Write a function that takes a grayscale image as input and returns two arrays the same size as the image, the first of which contains the magnitude of the image gradient at each pixel and the second containing the orientation.

[mag,ori] = mygradient(I)

Your function should filter the image with the simple **x- and y-derivative filters**. Once you have the derivatives you can compute the **orientation** and **magnitude** of the gradient vector at each pixel. You should use imfilter with the 'replicate' option in order to nicely handle the image boundaries.

Visualize the resulting gradient magnitude and orientation using imagesc. Make sure and use colorbar and a non- grayscale colormap in order to make the visualization better. The colormap jet for the

gradient magnitude, and colormap hsv for the orientation values are suggested. Also use title to put informative titles on your plots. Include a plot of both the magnitude and the orientation in your writeup for an example image of your choosing.

2. **Histograms of Gradient Orientations:** Write a function that computes gradient orientation histograms over each 8x8 block of pixels. Your function should bin the orientation into **9 equal sized bins** between -pi/2 and pi/2. The input of your function will be an image of size HxW. The output should be a three-dimensional array ohist whose size is (H/8)x(W/8)x9 where ohist(i,j,k) contains the count of how many edges of orientation k fell in block (i,j).

To determine if a pixel is an edge, we need to choose some threshold. It would be suggested using a threshold that is a tenth the maximum gradient magnitude in the image (i.e. **thresh = 0.1*max(mag(:))**). Since each 8x8 block will contain a different number of edges, you should **normalize** the resulting histogram for each block to sum to 1 (i.e., sum(ohist,3) should be 1 at every block location).

It is suggested that your function loops over the orientation bins. For each orientation bin you'll need to identify those pixels in the image whose magnitude is above the threshold and whose orientation falls in the given bin. You can do this easily in MATLAB using logical operations in order to generate an array the same size as the image that contains 1s at the locations of every edge pixel that falls in the given orientation bin and is above threshold. To collect up pixels in each 8x8 spatial block you can use the function im2col(..,[8 8],'distinct'). The im2col function will automatically pad out the image to a multiple of 8 which is convenient.

3. **Detection:** Write a function that takes a **template** (see the step 4) and an **image** and returns the top detections found in the image. Your function should have the prototype

```
[x,y,score] = detect(I,template,ndet)
```

where **ndet** is the number of detections to return. In your function you should first compute the histogram-of-gradient-orientation feature map for the image, then correlate the template with the feature map. Since the feature map and template are both three dimensional, you will want to filter each orientation separately and then sum up the results to get the final response. This final response map will be of size (H/8)x(W/8).

When constructing the list of top detections, your code should implement non-maxima suppression. You can do this by sorting the responses in descending order of their score. Every time you add a detection to the list to return, check to make sure that the location of this detection is not too close to any of the detections already in the output list. You can compute overlap by computing the distance between a pair of detections and checking that the distance is greater than say 70% of the width of the template.

Your code should return the locations of the detections in terms of the original image pixel coordinates (so if your detector had a high response at block (i,j) then you should return (8*i,8*j) as the pixel coordinates).

4. **Detection Script:** The provided demo detection script loads in a training image and a test image and has the user click on one or more patches of the training image. The script then builds an average **template** using the histogram feature map. Finally the script calls your detect function using this average template to detect objects in the test image.

You should modify the provided demo detection script to add two additional features. First, you should make the width and height of the template a variable parameter so that you can easily change it to detect different size objects. You should specify the template size in blocks (e.g. the provided script assumes the template is 16x16 blocks). This will require a handful of changes to values that are hard-coded in the demo script. Second, you should modify the script to incorporate negative training examples and produce a final template which is the difference of positive and negative averages. Negative examples can either be generated by having the user click or by providing a negative training image which contains no instances of your object and sampling random locations.

Write a brief discussion of where the detector works well and when it fails. Describe some ways you might be able to make it better.

Turning in the Assignment

Your submission should consist of the following:

- All your code and output images **in a single zip file**.
- A brief report **in a single PDF file** with all your results and discussion.

Academic integrity: Feel free to discuss the assignment with each other in general terms. Coding should be done a product of each individual team. If you use existing material, be sure to acknowledge the sources in your report.