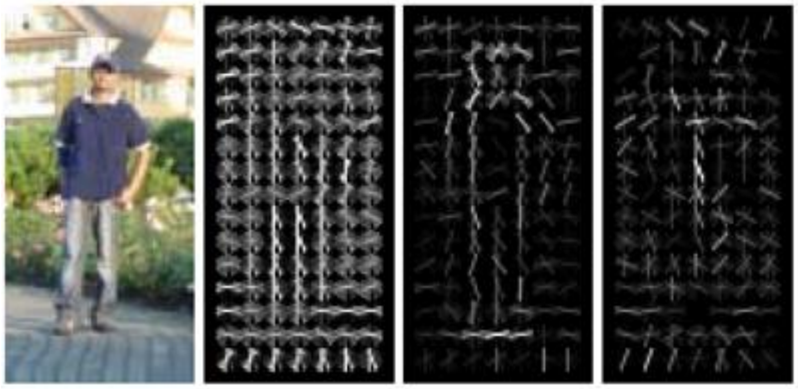
Δ05 Computer Vision 2020-2021
Assignment 4 – Face detection with a sliding window
Due Date: Wednesday, June 23, 2020, 11.59pm

Overview



The sliding window model is conceptually simple: independently classify all image patches as being object or non-object. Sliding window classification has long been a dominant paradigm in object detection and for one object category in particular -- faces -- it is one of the most noticeable successes of computer vision for category-level detection. For example, modern cameras and photo organization tools have prominent face detection capabilities. These success of face detection (and object detection in general) can be traced back to influential works such as Rowley et al. 1996 and Viola-Jones 2001, and forward to recent successes with convolutional neural networks for object detection like Faster R-CNN. You can look at these papers for suggestions on how to implement your detector. However, for this project you will be implementing the simpler (but still very effective!) sliding window detector of Dalal and Triggs 2005. Dalal-Triggs focuses on representation more than learning and introduces the SIFT-like Histogram of Gradients (HoG) representation (pictured to the right). You will not be asked to implement HoG. You will be responsible for the rest of the detection pipeline -- handling heterogeneous training and testing data, training a linear classifier, and using your classifier to classify millions of sliding windows at multiple scales. Fortunately, linear classifiers are compact, fast to train, and fast to execute. A linear SVM can also be trained on large amounts of data, including mined hard negatives.

Details and starter code

The following is an outline of the stencil code, which uses the VLFeats library (the binary of version 0.9.21 is included in the resources):

- proj4.m. The top level script for training and testing your object detector. If you run the code unmodified, it will predict random faces in the test images. It calls the following functions, many of which are simply placeholders in the starter code:

- get_positive_features.m (you code this). Load cropped positive trained examples (faces) and convert them to HoG features with a call to vl_hog.
- get_random_negative_features.m (you code this). Sample random negative examples from scenes which contain no faces and convert them to HoG features.
- classifier training (you code this). Train a linear classifier from the positive and negative examples with a call to vl_trainsvm.
- run_detector.m (you code this). Run the classifier on the test set. For each image, run the classifier at multiple scales and then call non_max_supr_bbox to remove duplicate detections.
- evaluate_detections.m. Compute ROC curve, precision-recall curve, and average precision. You're not allowed to change this function.
- visualize_detections_by_image.m. Visualize detections in each image. You can use visualize_detections_by_image_no_gt.m for test cases which have no ground truth annotations (e.g. the class photos).

Creating the sliding window, multiscale detector is the most complex part of this project. It is recommended that you start with a *single scale* detector which does not detect faces at multiple scales in each test image. Such a detector will not work nearly as well (perhaps 0.3 average precision) compared to the full multi-scale detector. With a well-trained multi-scale detector with small step size you can expect to match the papers linked above in performance with average precision above 0.9.

If you are interested in extending the algorithm (not mandatory for the assignment but recommended if you would like to do research in computer vision), you may add contextual reasoning to your classifier by one of the following approaches:

a. Learn probable locations of faces given scene statistics, in the spirit of Contextual priming for object detection, Torralba.
b. Use typical arrangements of groups of faces as in Understanding Images of Groups of People by Gallagher and Chen.

Data

The choice of training data is critical for this task. Face detection methods have traditionally trained on heterogeneous, even proprietary, datasets. As with most of the literature, we will use three databases: (1) positive training crops, (2) non-face scenes to mine for negative training data, and (3) test scenes with ground truth face locations.

You are provided with a positive training database of 6,713 cropped 36x36 faces from the Caltech Web Faces project. This subset has already filtered away faces which were not high enough resolution, upright, or front facing. There are many additional databases available For example, see Figure 3 in Huang et al. and the LFW database described in the paper. You are free to experiment with additional or alternative training data for extra credit.

Non-face scenes, the second source of your training data, are easy to collect. You are provided with a small database of such scenes from the SUN scene database. You can add more non-face training scenes, although you are unlikely to need more negative training data unless you are doing hard negative mining for extra credit.

A benchmark for face detection is the CMU+MIT test set. This test set contains 130 images with 511 faces. The test set is challenging because the images are highly compressed and quantized. Some of the faces are illustrated faces, not human faces. For this project, we have converted the test set's ground truth landmark points in to bounding boxes. We have inflated these bounding boxes to cover most of the head, as the provided training data does. For this reason, you are arguably training a "head detector" not a "face detector" for this project.

Copies of these data sets are provided with your starter code in the folder named “Assignment resources”.

Write up

In the report, please include the following:

- Describe your algorithm and any decisions you made to write your algorithm a particular way.
- Include the precision-recall curve and average precision (AP) of your final classifier and any interesting variants of your algorithm.
- Show and discuss the results of your algorithm.

Credits

This project was originally created by Prof. James Hays of Georgia Tech and modified by Prof. Kristen Grauman of University of Texas at Austin, whose description and code is used in this project.

- You should submit the code in MATLAB and a report in pdf iin a single zip file.
- Please send your assignments to cnikou@uoi.gr with the subject:
    - LastName_D05_Assignment_4
- The name of your zip file should be the same as the email subject.