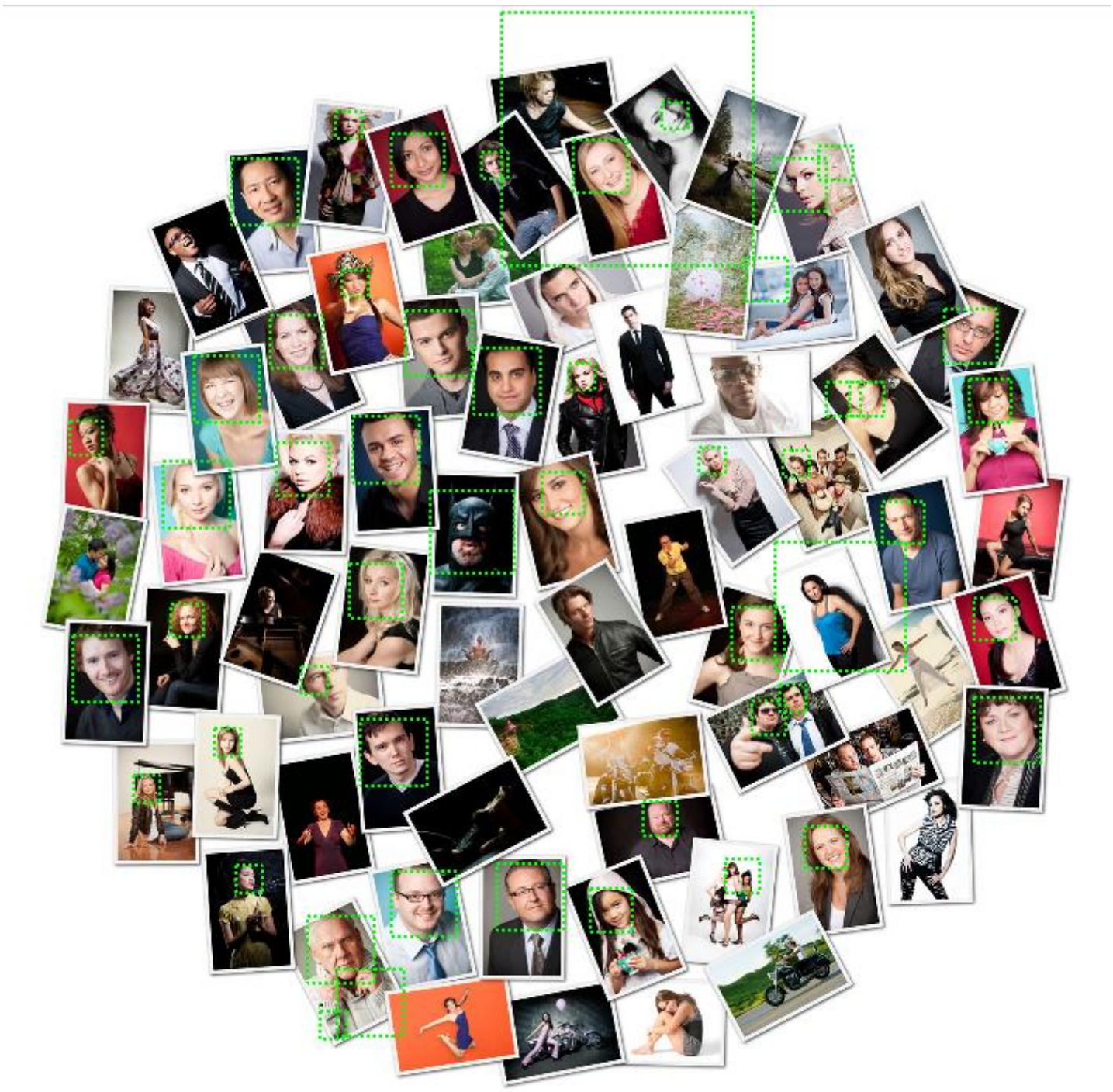
Project: Face Detection with a Sliding window



Overview

The goal of the project is to implement a simple but effective method for detecting faces in an image. My work is mainly based on the technique of [Dalal and Triggs 2005.](#) The technique includes four main components:

1. **A feature descriptor**. First, I need a way to describe an image region with a high-dimensional descriptor. For this project, I use Histogram of Oriented Gradient (HoG) feature representation. The essential thought behind the Histogram of Oriented Gradient (HoG) descriptors is that local object appearance and shape within an image can be described by the distribution of intensity gradients or edge directions. The HoG descriptor is thus particularly suited for human detection in images.
2. **A learning method**. Next, I need a way to learn to classify an image region (described using HoG) as a face or not. For this, I use support vector machines (SVMs) with linear kernel and a large training dataset of image regions containing faces (positive examples) or not containing faces (negative examples).
3. **A sliding window detector**. Using the SVM classifier, I can now tell if a fixed size (in my implementation, 36 * 36) image region is a face or not. The next step is to run the classifier as a sliding window detector on an input image in order to detect all faces of a single scale in that image. In order to detect faces at multiple scales, I also need to run the sliding window detector at multiple scales of the original image.
4. **Non-maxima suppression**. Now there are lots of candidate face regions of multiple scales which are overlapped. The final step is to find the best detections within a neighborhood by selecting the strongest responses given by the classifier.

Implementation Pipelines

Getting training data

The positive examples are pre-cropped images of human faces of size 36 * 36. I then extract HoG features (with HoG cell size = 4 in my implementation) from each image as a 2511 ((36 / 4) * (36 / 4) * 31) dimensional vector (as shown in Figure.2). In my implementation, I use more than 6,700 positive features.

The negative examples are not off-the-shelf. So I get some images without faces (e.g. animal photos, scenery photos, etc.), and randomly sample square regions of size 36 * 36 at multiple scales, i.e. scale the image to different factors and randomly sample 36 *36 regions from it, and then extract HoG features of these sampled regions (as shown in Figure.3). In my implementation, I get more than 35,000 negative features from around 200 non-face photos. It turns out taking negative samples at multiple scales can be more representative, and thus can improve the accuracy of the face detection.

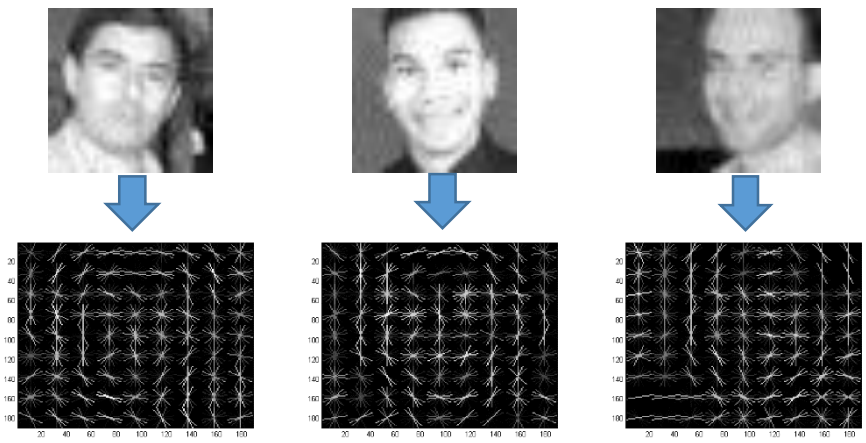


Figure.2 Visualization of positive HoG features

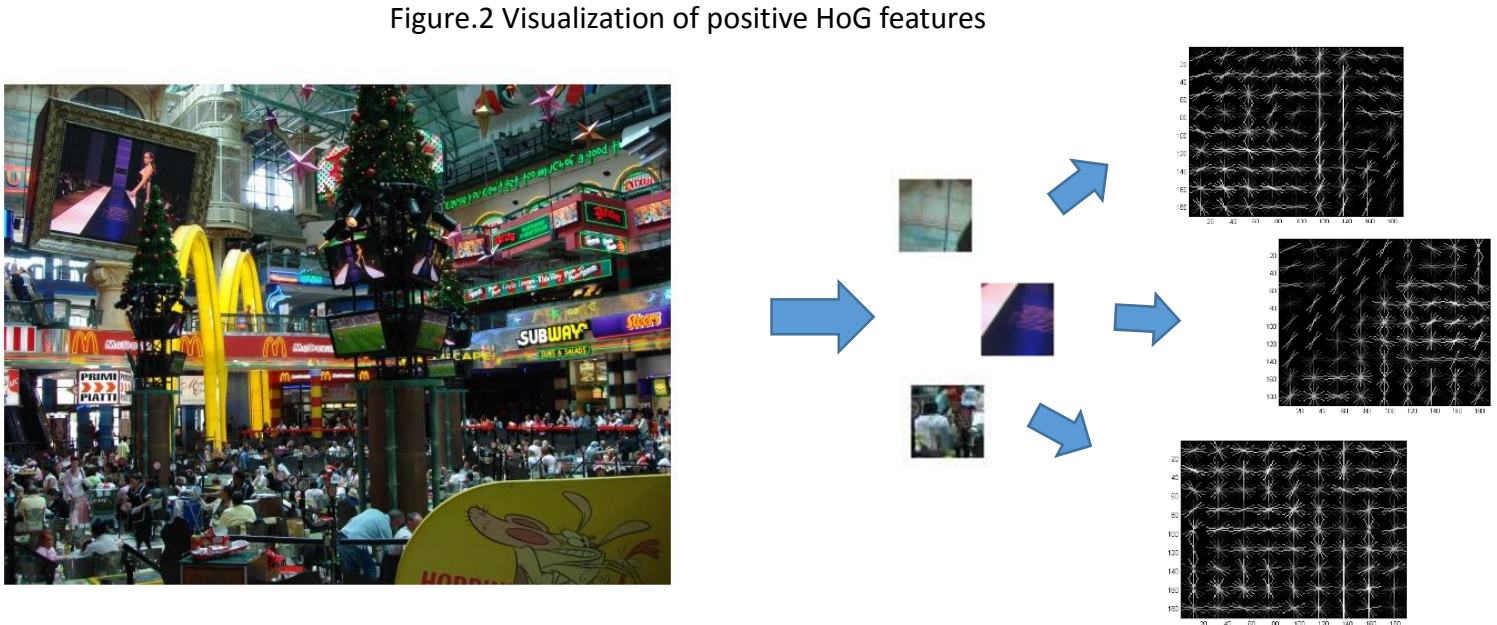


Figure.3 Visualization of negative HoG features

From Figure.2 and Figure.3, we can clearly see the edge and shape of objects in images are maintained by HoG features.

Training Linear SVM

In this section, I train a linear Support Vector Machine (SVM) classifier on the above extracted positive HoG features and negative HoG negatives. Support Vector Machine (SVM) is one of the state-of-the-art supervised learning algorithm. In general, the SVM can be stated as following:

$$argmin_{w,b,\xi}\left\{\frac{1}{2}\|\vec{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

$$subject\ to: y_i(\vec{w}\cdot\vec{x_i} - b) \geq 1 - \xi_i\ (\forall\ \mathrm{i} = 1, \ldots, \mathrm{n})$$

$$\xi_i \geq 0$$

C is the tuning parameter. Intuitively, C trades off the penalty for wrong classifying and margin width. For my implementation, I choose C to be 0.0001.

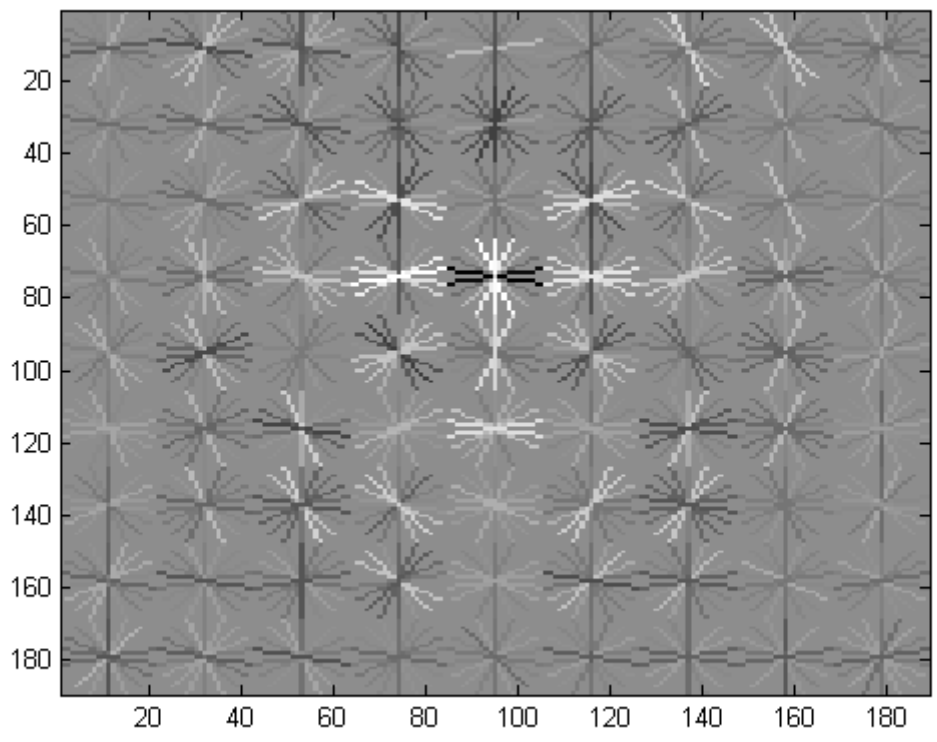


Figure.4 Visualization of learned HoG template

From Figure.4, we can see the learned HoG template resembles the shape of human face very much, including locations of eyes, eyebrows, nose and mouth.

Running the sliding window detector

The next step is to implement the sliding window detector. The main idea is to move the sliding window detector over every possible 36 * 36 patch of

the image, extract HoG features, and use formerly trained SVM classify to classify. To be able to detect faces other than size of 36 * 36, I resize the original image to different factors and repeat the window sliding process. (Notice: the sliding window size is fixed at 36 * 36, what's change is the scale of the image, not the size of sliding window, because the classifier is only suitable for 36 * 36 patches, otherwise classifiers for different size patches are needed.) Figure.5 is a simple illustration of the sliding window detection process.

There are 3 tuning parameters: the step size to move the sliding window, the scale factor to resize the image, and the confidence threshold of a candidate face. Intuitively, with finer step size and scale factor, there is less possibility to escape face, while it will require more computational efforts and considerably slow down the detection process. In my implementation, I choose step size to be 4 (the same as the hog cell size) and scale factor to be 0.8. With higher confidence threshold, there will be less detections, and lower confidence threshold leads to more detections. In my implementation, I set confidence threshold at 0.7.

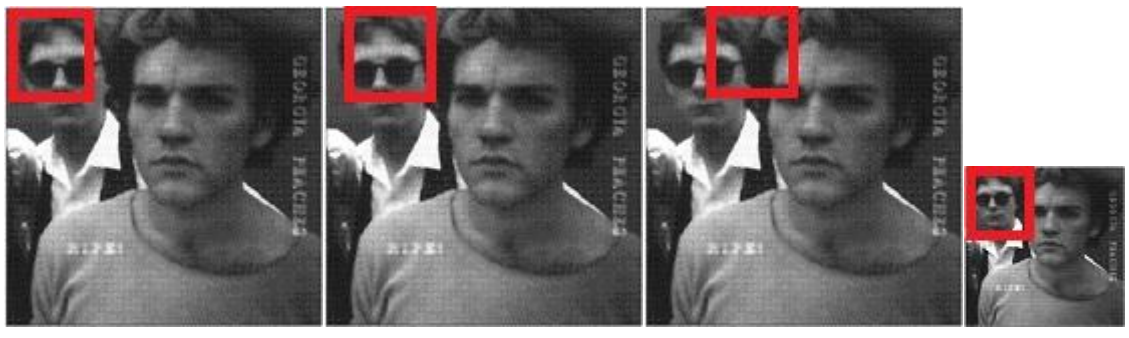


Figure.5 Illustration of sliding window detector

Non-maxima Suppression

The final step is to implement the non-maxima suppression algorithm. After the above steps, there are many face candidates detected. However, many of them are overlapped across scales (shown in figure.6 left). The non-maxima suppression step is to suppress detections within neighborhoods to one, according to their response given by the classifier, and only keep the detection with the strongest response. (as shown in figure.6 right)

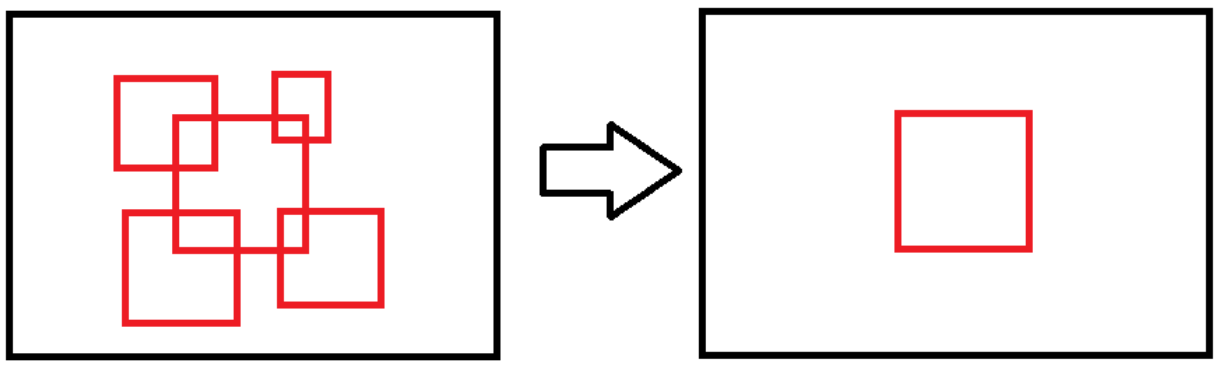


Figure.6 Illustration of non-maxima suppression

Test performance on test datasets

To test the performance of my implementation of face detection system, I need some test images which are unseen at the training stage. The test datasets consists of images whose ground truth face locations are known (manually labelled by human). I consider using the precision-recall curve as the metric of performance (Figure.7).

Precision = $\frac{TP}{TP+FP}$ Recall = $\frac{TP}{TP+FN}$ where TP = True Positive, FP = False Positive, FN = False Negative

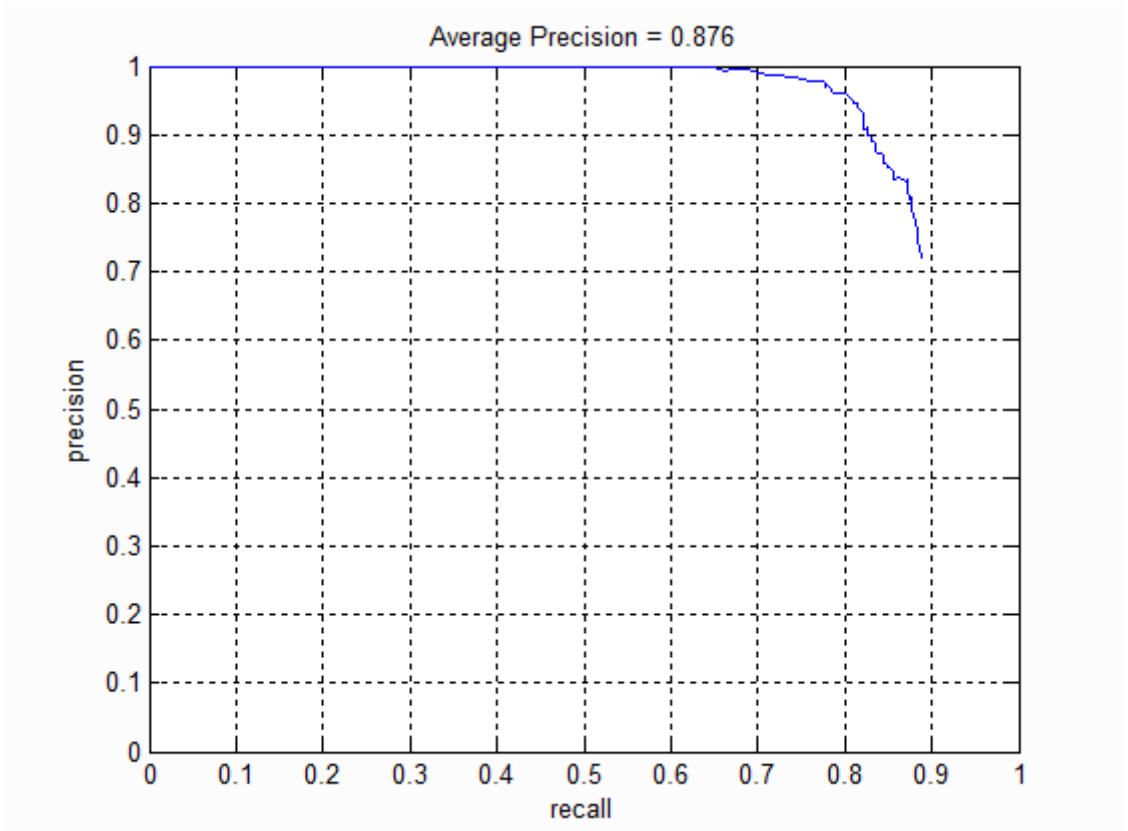


Figure.7 Precision-Recall Curve

Code Outline

code/

vlfeat/ - An open source library implementing some popular computer vision algorithms. I use vl_hog and vl_svmtrain from this library to extract HoG features and train the linear SVM.

proj.m - The top level script guiding the training and testing process of my sliding window detection system.

get_positive_features.m - Load cropped positive face examples and convert them to HoG feature representation.

get_random_negative_features.m - Load non-face images and randomly sample negative examples from them and convert them to HoG features.

run_detector.m - Run the trained classifier on the test set.

report_accuracy.m - Report the accuracy of SVM classifier.

evaluate_detections.m - Plot the ROC curve (i.e. precision-recall curve). (**from the 2010 Pascal VOC toolkit)

visualize_detections_by_image.m - Visualize detections in images whose ground truth face locations are known. (**from the 2010 Pascal VOC toolkit)

visualize_detections - Visualize detections in image whose ground truth are unknown.

non_max_supr_bbox.m - Non-maxima suppression algorithm.

sliding_window_detector.m - implements the multiple scale sliding window algorithm.

face_detection.m - The final interface to use the face detection system. Usage: face_detection('image_name'). Type: '' help face_detection '' for more detail usages.

data/

caltech_faces/ - 36 * 36 cropped face images.

extra_test_scenes/ - images with unknown ground truth face locations.

test_scenes/ - test images with ground truth face locations manually labelled.

train_non_face_scenes/ - non-face images. This is where negative examples sampled from.

visualizations/ - This directory saves results of detected images

Sample Results

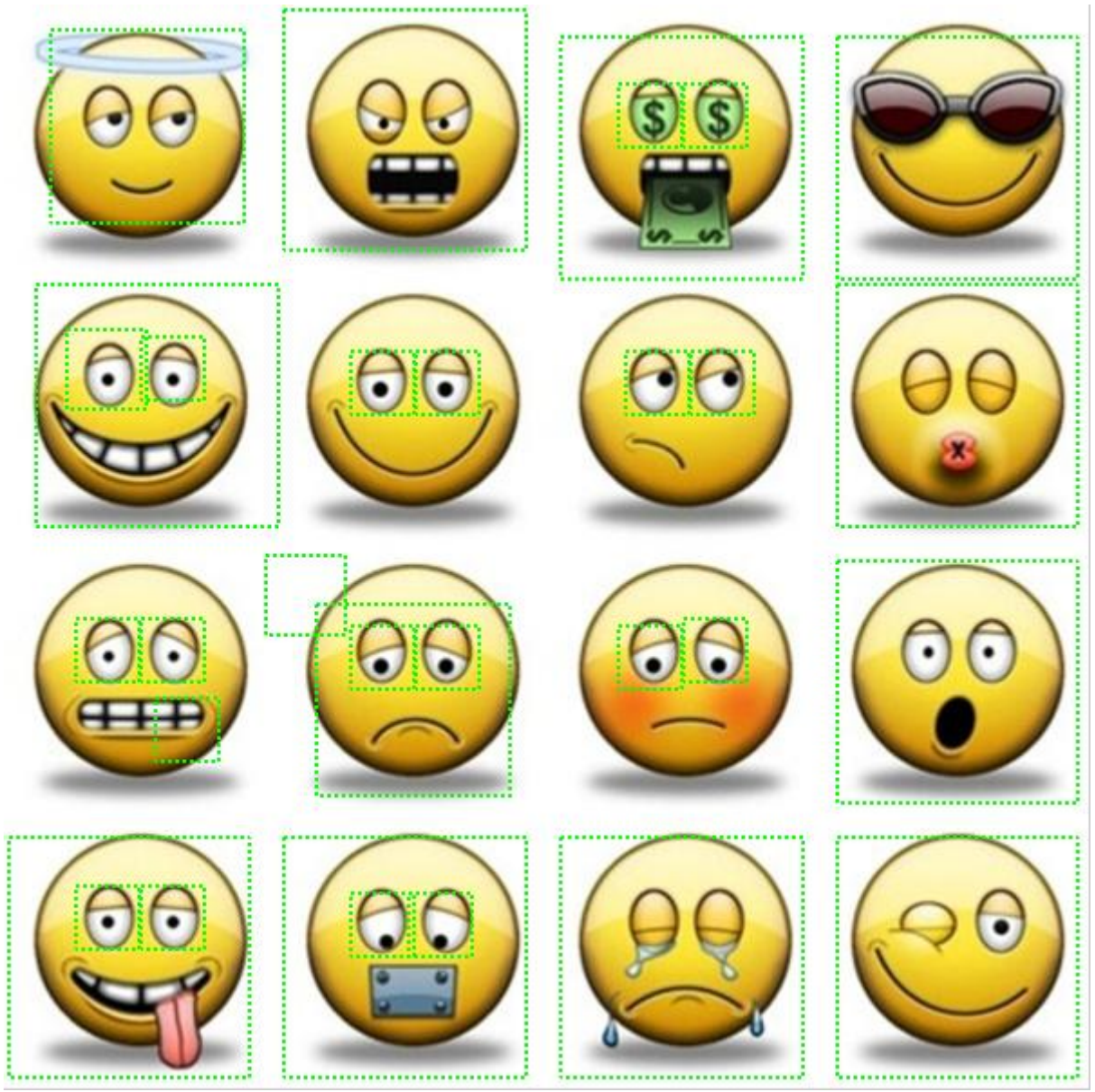
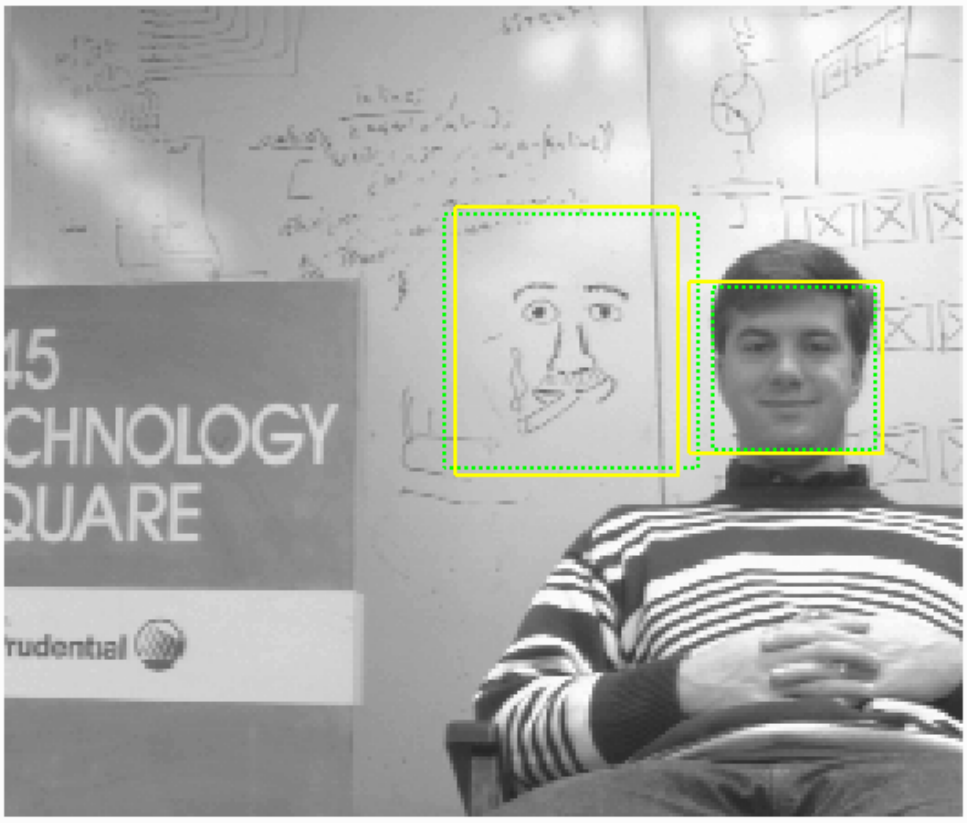
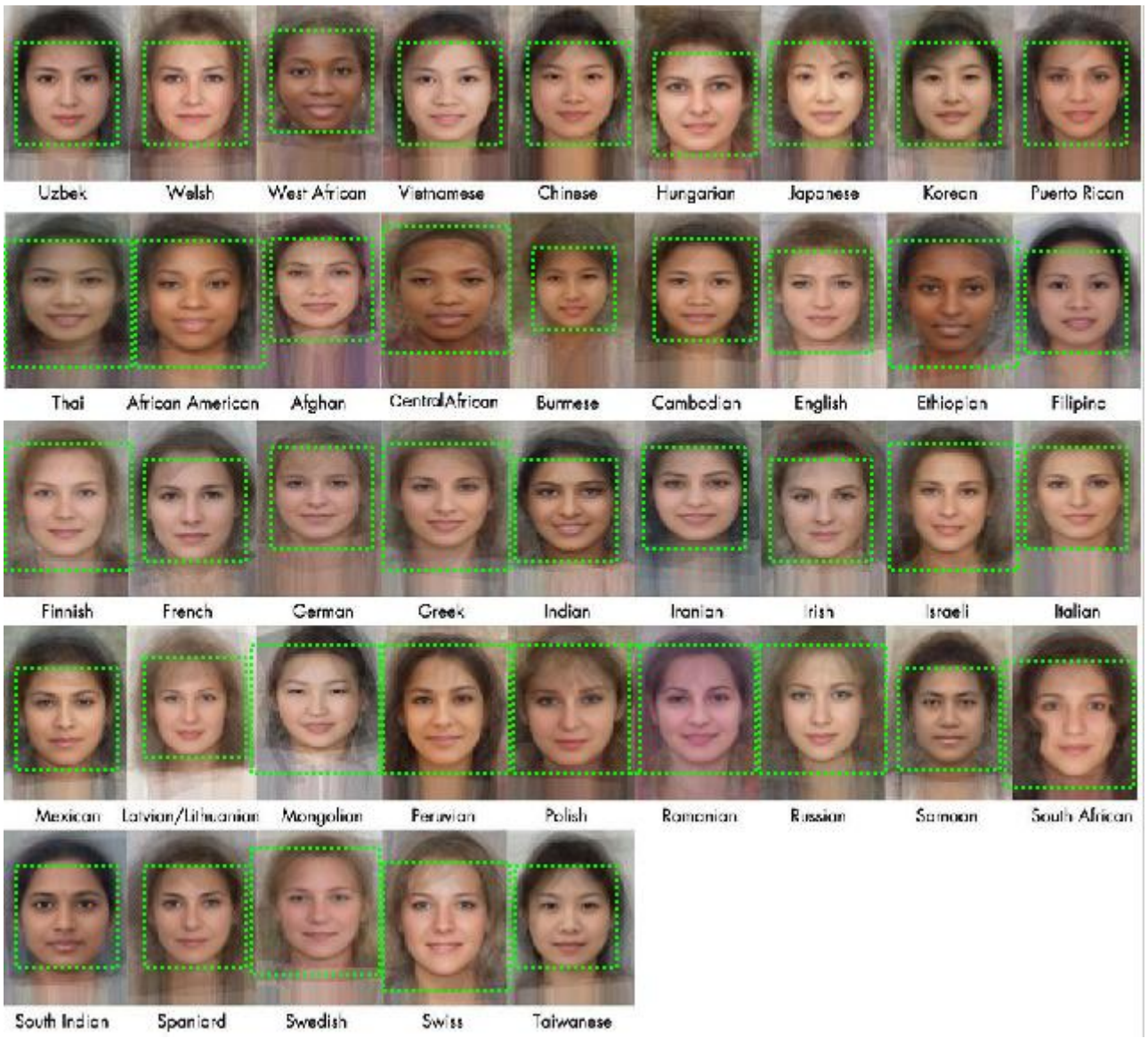


image: "torrance.jpg" (green=true pos, red=false pos, yellow=ground truth), 2/2 found





References

- P.Viola and M.Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. CVPR 2001.
- N.Dalal and B.Triggs. Histograms of Oriented Gradients for Human Detection. CVPR 2005.
- VL Feat Matlab reference
- PASCAL VOC 2007 person detection dataset