CVPR 2018, Projects assignments

Felice Andrea Pellegrino

December 20, 2018

Introduction

The student is required to complete one of the following projects. Students can work in group; in that case, however, each group's member is responsible of the whole project and is expected to be aware of all the details of the work done by the group. The choice of the environment/language is free. A report with problem statement, description of the approach, implementation choices, results and references must be written and sent to the instructor (*fapellegrino@units.it*) along with a compressed folder containing the code (or, preferably, a reference to a repository where the code can be accessed).

Project 1

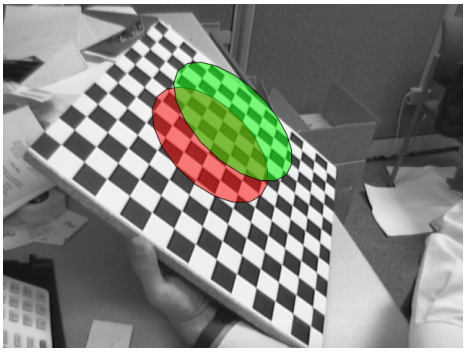

Figure 1: Superimposing as cylinder to a calibration image.

This project is based on the first lab lecture (see the Matlab starter code and images in Moodle). Things to do:

1. apply the code as it is for calibration;

2. choose one of the calibration images and compute the total reprojection error (Lecture 2, page 62) for all the grid points (adding a figure with the reprojected points);

3. add **radial distortion compensation** (Lecture 2, page 70) to the basic Zhang's calibration procedure;

4. compute the total reprojection error with radial distortion compensation and make a comparison to the one without compensation;

5. superimpose an object (for instance, a cylinder as in Fig. 1), to the calibration plane, in all the images employed for the calibration;

6. optionally, you could print a calibration object and apply the previous steps to a set of images acquired with your own smartphone, webcam or digital camera (thus, calibrating your device).

Figure 2: Images fro the Caltech dataset.

Project 2

This project requires the implementation of a sliding window face detector using histograms of oriented gradients (HoG)[Dalal and Triggs(2005)], see also Lecture 8, page 21.
The provided training dataset contains 6713 cropped 36×36 faces from the Caltech Web Faces project. A sample is reported in Fig. 2. Non-face scene *are not provided* but are easy to collect. As a benchmark, the CMU+MIT test is provided. This test set contains 130 images with 511 faces. The ground truth for the test set is provided in a text file containing, for each face, a row with the

name of the image and the coordinates of two opposite corners of the bounding box (x top left, y top left, x bottom right, y bottom right). For example:

```
voyager.jpg 117 37 143 69
```

You are expected to:

1. load the training positive instances and compute the HoG descriptor (it is recommended to use the VLFeat library and, in particular, the function `vl_hog`);
2. sample random negative instances from non-face images and compute the corresponding HoG descriptors;
3. train a linear SVM using the collected positive and negative feature vectors (a possibility is to use `vl_trainsvm` from VLFeat);
4. run the classifier on the test set at a single scale;
5. run the classifier on the test set at a multiple scales (by rescaling the test image multiple times);
6. evaluate the performance in terms of precision and recall based on the provided ground truth. In order to do so:
   - suppress multiple overlapping detections: if a center of a detected bounding box is in the interior of another detection, keep the detection having higher confidence (real-valued output of the linear classifier) and discard the other (if you use Matlab, you may want to use the function `selectStrongestBbox`);
   - consider a detection as a true positive if the detected bounding box overlaps at least 50% with the ground truth box (if you use Matlab, you may want to use `bboxPrecisionRecall`).
7. optionally, you could retrain the classifier using the false positives of the first round as negative examples (as in [Dalal and Triggs(2005)]);
8. optionally you could run the detector on other images than the provided test set;
9. optionally, if you spot strange detections, you could use the technique in [Vondrick et al.(2013)Vondrick, Khosla, Malisiewicz, and Torralba] (you can download the code from the HOGgles website `http://www.cs.columbia.edu/~vondrick/ihog/`) to understand the reason.

References

[Dalal and Triggs(2005)] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, pages 886–893. IEEE, 2005.

[Vondrick et al.(2013)Vondrick, Khosla, Malisiewicz, and Torralba] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013.