CVPR 2018, Projects assignments

Felice Andrea Pellegrino

December 30, 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).

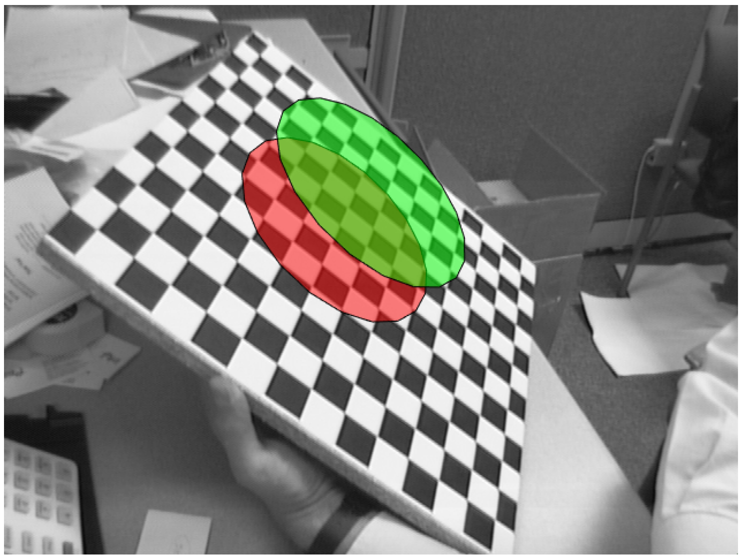

Figure 1: Superimposing a cylinder to a calibration image.

Project 1

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

1. calibrate using Zhang procedure, i.e. find the intrinsic parameters K and, for each image, the pair of R, t (extrinsic);
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 from 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 set 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) containing a face. 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 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 others (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] (you can download the code from the HOGgles website `http://www.cs.columbia.edu/~vondrick/ihog/`) to understand the reason.

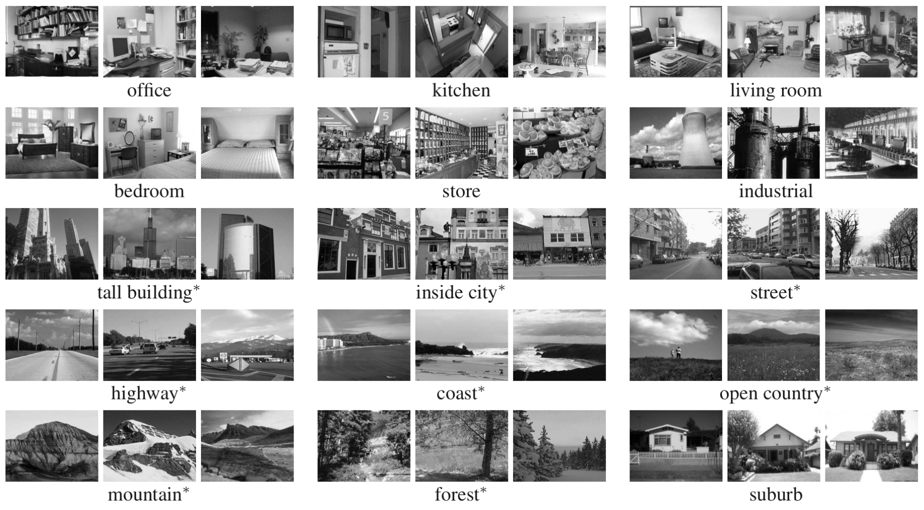

Figure 3: Examples of images from each of the 15 categories of the provided dataset (the same as [Lazebnik et al., 2006]).

Project 3

This project requires the implementation of an image classifier based on the bag-of-words approach. The provided dataset (from [Lazebnik et al., 2006]), contains 15 categories (office, kitchen, living room, bedroom, store, industrial, tall building, inside city, street, highway, coast, open country, mmountaing, forest, suburb), and is already divided in training set and test set. Samples are shown in Fig. 3.
You are expected to:

1. build a visual vocabulary:
   - sample many (10K to 100K) SIFT descriptors from the images of the training set (you can sample on a grid in the scale-space, instead of using a detector);
   - cluster them using k-means (the choice of the number of clusters is up to you, and you should experiment with different values, but you could start with a few dozens);
   - collect (and save for future use) the clusters' centroids which represent the k 128-dimensional visual words.
2. Represent each image of the training set as a normalized histogram having k bins, each corresponding to a visual word; a possibility is to perform a rather dense sampling in space and scale; another possibility is to use the

SIFT detector to find the points in scale-space where the descriptor is computed. In any case, each computed descriptor will increase the value of the bin corresponding to the closest visual word.

3. Employ a nearest neighbor classifier and evaluate its performance:
   - compute the normalized histogram for the test image to be classified;
   - assign to the image the class corresponding to the training image having the closest histogram.
   - repeat for all the test images and build a confusion matrix.
4. Train a multiclass linear Support Vector Machine, using the one-vs-rest approach (you will need to train 15 binary classifiers having the normalized histograms as the input vectors and positive labels for the "one" class and negative for the "rest.")
5. Evaluate the multiclass SVM:
   - compute the normalized histogram for the test image to be classified;
   - compute the real-valued output of each of the SVMs, using that histogram as input;
   - assign to the image the class corresponding to the SVM having the greatest real-valued output.
   - repeat for all the test images and build a confusion matrix.
6. Optionally, you could train the SVM using a generalized Gaussian kernel (Lecture 8, page 69) based on the χ^2 distance;
7. optionally, you could train the SVM using a generalized Gaussian kernel based on the earth mover's distance;
8. optionally, you could use soft assignment to assign descriptors to histogram bins. Each descriptor will contribute to multiple bins, using a distance-based weighting scheme (see [Van Gemert et al., 2008]);
9. optionally, you could add some spatial information by taking inspiration from the spatial pyramid feature representation of [Lazebnik et al., 2006] (a simple approach could be building an extended descriptor, containing the histogram of the whole image plus the histograms of the 4 quadrants and so on up to a desired level);
10. optionally, you could implement the spatial pyramid kernel described in [Lazebnik et al., 2006].

References

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). 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.

[Lazebnik et al., 2006] Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *null*, pages 2169–2178. IEEE.

[Van Gemert et al., 2008] Van Gemert, J. C., Geusebroek, J.-M., Veenman, C. J., and Smeulders, A. W. (2008). Kernel codebooks for scene categorization. In *European conference on computer vision*, pages 696–709. Springer.

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