CS 484, Fall 2016
Term Project: Object Recognition

The goal of this project is to develop an object recognition system for indoor and outdoor images. Recognition will be performed using the bag-of-words (also known as bag-of-features, bag-of-visterms) model. You are expected to work in groups of two. Specifications for the components of the object recognition system are given below.

1 Data

The LabelMe (`http://labelme2.csail.mit.edu/`) data set from MIT will be used for the recognition system. A subset of LabelMe that contains 188 indoor and outdoor images will be used for this project (read the README file for data format). The following objects were manually labeled on each image: computer screen (frontal view, 90 examples), keyboard (88 examples), mouse (77 examples), mug (43 examples), car (side view, 53 examples), tree (68 examples), person (33 examples), and building (92 examples). Examples are shown in Fig. 1.

2 Background

We will use the bag-of-words model for object recognition. It was discussed during the lectures but you can also check the bag-of-words tutorial at `http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html` and read the following papers to learn more about the model:

- G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, "Visual Categorization with Bags of Keypoints," European Conference on Computer Vision, 2004.
- L. Fei-Fei, P. Perona, "Bayesian Hierarchical Model for Learning Natural Scene Categories," IEEE Conference on Computer Vision and Pattern Recognition, 2:524–531, June 20–25, 2005.

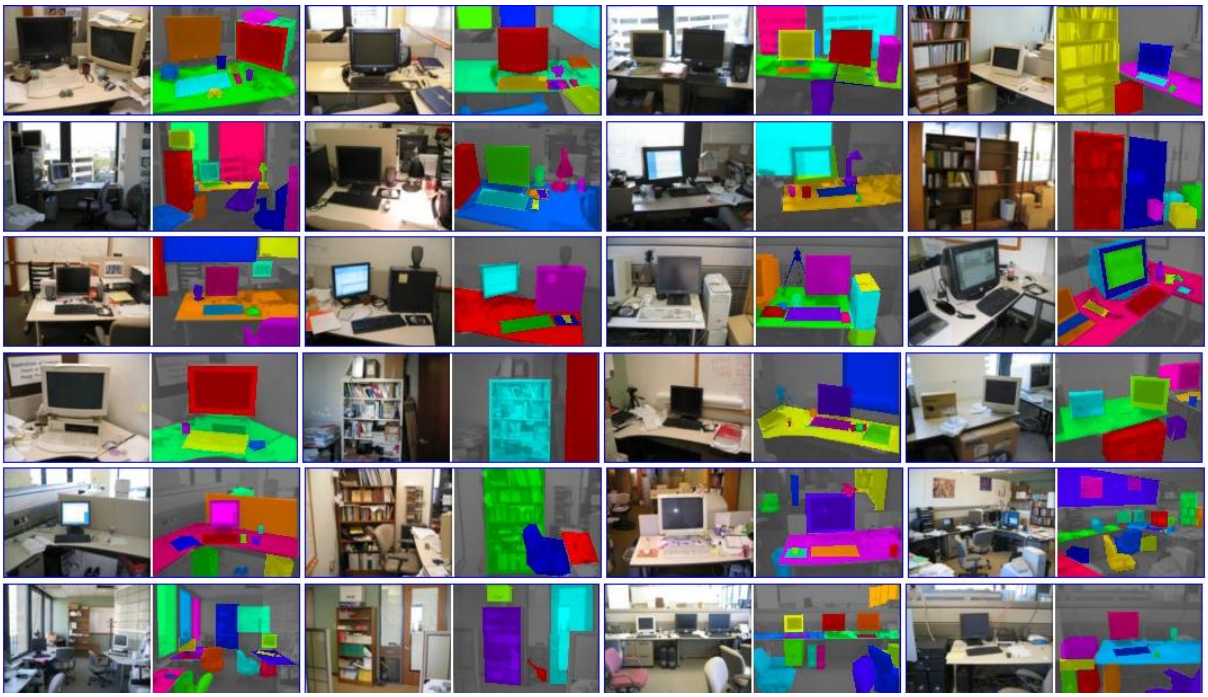
3 Pre-processing

3.1 Modeling the local features

The first step is to model the local features using the Scale-Invariant Feature Transform (SIFT). You are expected to extract these features by an approach called dense sampling where a set of points densely sampled on a regular (uniform) grid is used as the set of interest points. Then, the descriptor of each local feature should be obtained by using the SIFT algorithm (i.e., you will use SIFT only for the descriptor, not as a detector). You can use the VLFeat open source library (`http://www.vlfeat.org/`) to extract the SIFT features.

3.2 Finding the visual words

The next step is the construction of the codebook of visual words. You can use the k-means clustering algorithm for quantizing the local features. Typically, a large number of clusters (e.g., 500, 1000, 2000) has been used in the literature. You need to experiment with the number of clusters with respect to the number of local features obtained in the data. The VLFeat library and Matlab's own toolboxes are good software resources for this step.



(a) Indoor images

(b) Outdoor images

Figure 1: Examples from the LabelMe data set. A mask is available for each object of interest.

4 Training the system

Divide the provided LabelMe data set randomly into two subsets. You can try different random partitions so that the number of examples for each object type is roughly equal in both subsets. Use one of the subsets for training, and the other for testing. Use only the images in the training set for the following step.

Build the bag-of-words model for each object in the training data by computing the histogram of visual words contained within the corresponding object mask. Note that you should use the manually labeled masks (like the ones in Fig. 1) provided as part of the LabelMe data set at this step. At the end of this step, we have the bag-of-words representation and the object label for each object mask in the training set. The code that builds the bag-of-words model for each object mask MUST be your own code.

Using the bag-of-words models and the object labels, train a binary support vector machine (SVM) classifier for each object type. For a particular object type, you can use the examples given for that object as positive samples, and all of the examples given for the rest of the object types as negative samples during training. You can use the LIBSVM library (`https://www.csie.ntu.edu.tw/~cjlin/libsvm/`) for the SVM classifier. LIBSVM has a Matlab interface. Several other good tools are also available for Matlab. You should experiment with the SVM parameters to build effective classifiers.

5 Testing the system

Use the images in the test set for the following steps.

5.1 Finding the segments

Use the normalized cut segmentation algorithm to obtain image segments as candidate objects. Segmentation is a very difficult problem; thus, expect to spend some time experimenting with the parameters to obtain meaningful segments. You can use the normalized cut code at `http://www.timotheecour.com/software/ncut/ncut.html` or at `http://www.timotheecour.com/software/ncut_multiscale/ncut_multiscale.html`.

5.2 Object recognition

Build the bag-of-words model for each segment by computing the histogram of visual words contained within the corresponding segment. Then, for each object type, use the corresponding SVM classifier to compute the probability of each segment to contain an instance of that object. The result of this step is a detection probability map (a separate map for each object type) for the whole image where each pixel gets the probability assigned to its corresponding segment. The code that builds the probability maps from the SVM outputs MUST be your own code.

6 Evaluation

The probability map obtained in the previous step can be converted to a binary detection mask by using a threshold. For a given range of thresholds, produce such binary masks and compare them to the object masks in the test set. The classification performance can be evaluated using receiver operating characteristics (ROC) curves that plot true positive rates (TPR)

$$\text{TPR} = \frac{\text{positives correctly detected}}{\text{total positives}} \tag{1}$$

versus false positive rates (FPR)

$$\text{FPR} = \frac{\text{negatives incorrectly detected}}{\text{total negatives}} \tag{2}$$

for different threshold values. These rates should be computed by comparing and counting pixels in the detection masks and ground truth masks where the pixels belonging to object masks are labeled as positive and the rest of the image is labeled as negative. Aggregate the results from all images, and produce an ROC curve for each object type. The code that performs the evaluation MUST also be your own code.

Submit:

You must submit the final report and the developed code through the online form by the deadline given on the course web page.

1. Report
    - Must be readable and well-organized.
    - Provide proper explanation of the details of the approach, the implementation strategies, the results obtained, and the observations made.
    - Provide examples for dense sampling and resulting SIFT features.
    - Provide examples for the codebook of visual words obtained by clustering.
    - Provide examples for the bag-of-words representations for different objects in the training data.
    - Provide examples for the segmentation results.
    - Provide examples for the bag-of-words representations for different objects in the test data.
    - Provide the detection accuracy for each object type as an ROC curve.
    - Provide examples for the recognition results (e.g., probability maps and binary object maps obtained at certain threshold values) for all object types. You should show examples for both successful and unsuccessful detections.
    - An important part of your report is the discussion where you comment on the performance of your system according to different implementation decisions and parameter settings. You should also analyze the results for individual object types, i.e., which object types were easy and which ones were more difficult, and why you think they were easy/difficult.
    - All figures must have proper captions and must be properly referenced in the text.
    - The report must follow the IEEE Computer Society two-column format as shown on the course web page.
    - Each team member should also provide a written description of her/his own specific contributions to the project, and should include this information as an appendix of the report.
2. Code
    - Provide well-documented code.
    - You are free to use code from other sources, with proper citation, for the parts that do not explicitly indicate that you must write your own code. Each team member should be prepared to answer questions about all parts.