IMS – Project
Bag-of-Words based Image Classification

December 5, 2013

Students should work on this assignment in a group of two. All files should be zipped and sent to **uvaims2013@gmail.com** before 27-12-2013, 23:59 (Amsterdam time). The subject of the email has to be [**IMS2013_FALL**].

1 Introduction

The goal of the assignment is to implement a system for image classification, in other words, this system should tell if there is an object of given class in an image. We will perform four-class (airplanes, motorbikes, faces, and cars) image classification based on bag-of-words approach. The provided data file contains training and test subdirectories for each category (all images are taken from Caltech Vision Group). For each class, test subdirectories contain 50 images, and training subdirectories contain up to 500 images. You have to test your system on all the test images, and train it on at least 50 training images per class (take either the first 50 ones or a random subset of 50). Keep in mind that using more samples in training will (almost certainly) result in better performance. However, if your computational resources are limited and your system is slow, it's OK to use less training data to save time. Please, remember first to debug all your code with a small amount of input images/descriptors and only run your system for the real experiment when you are sure that everything works properly.

2 Bag-of-Words based Image Classification

Bag-of-Words based Image Classification system contains the following steps: (1) Feature Extraction and Description, (2) Building a Visual Vocabulary, (3) Quantize features using visual dictionary (encoding), (4) Representing images by frequencies of visual words, (5) Classification. We will consider each step in detail.

2.1 Feature Extraction and Description

SIFT descriptors can be extracted from key points. As an alternative to key points, SIFT descriptors can be extracted densely. You can use VLFeat functions for dense SIFT and key points SIFT descriptor extraction (http://www.vlfeat.org/).

Moreover, it is expected that you implement all the steps also for RGBSIFT, rgbSIFT and opponentSIFT.

2.2 Building Visual Vocabulary

Here we will obtain visual words by clustering feature descriptors, so each cluster center is a visual word. Take a subset of all training images (this subset should contain images from ALL categories), extract SIFT descriptors from all of these images, and run k-means clustering on these SIFT descriptors to build visual vocabulary. Take about half of train images from each class to calculate visual dictionary (around 1000 images), however, you can also use less images, say 100 from each class. Please, remember first to debug all the code with a small amount of input images and only when you are sure that all code works run it for real. Pre-defined cluster numbers will be the size of your vocabulary. Set the vocabulary size to 400, and experiment with several different sizes (800, 1600, 2000 and 4000).

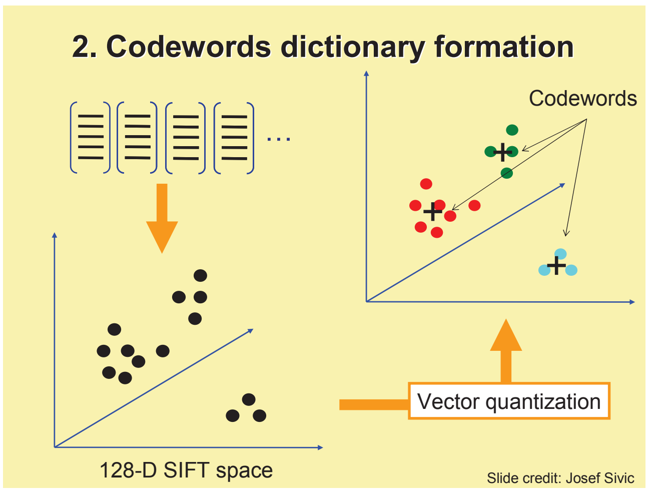

Figure 1: Learning Visual Dictionary. Codewords is another term for visual words.

2.3 Quantize Features Using Visual Vocabulary

Once we have a visual vocabulary, we can represent each image as a collection of visual words. For this purpose, we need to extract feature descriptors (SIFT) and then assign each descriptor to the closest visual word from the vocabulary.

2.4 Representing images by frequencies of visual words

The next step is the quantization, the idea is to represent each image by a histogram of its visual words, see Figure 2 for overview. Check out MATLAB's `hist` function. Since different images can have different numbers of features, histograms should be normalized.

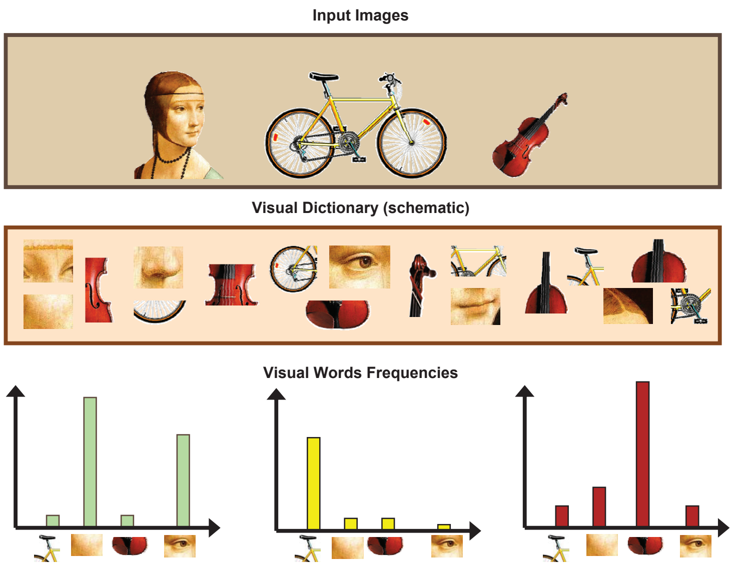

Figure 2: Schematic representation of Bag-Of-Words system.

2.5 Classification

We will train a Support Vector Machine (SVM) classifier per each object class, as a result we will have four binary classifiers. Take images from the training set of the related class (but which you did not use for dictionary calculation), and represent them with histograms of visual words as discussed in the previous section. Please, use at least 50 training images per class (take either the first 50 ones or a random subset of 50) or more, but remember to debug your code first! So if you use the default setting, you should have 50 histograms of size 400. These will be your positive examples. Then, you will obtain histograms of visual words for images from other classes, again about 50 images per class, as negative examples. Therefore, you will have 150 negative examples. Now you can train a classifier. You should repeat it for each class. To classify a new image, you should calculate its visual words histogram as described in Section 2.4 and use the trained support vector machine classifier to assign it to the most probable object class. (Note that for proper SVM scores you need to use cross-validation to get a proper estimate of the SVM parameters. In this assignment you do not have to do this cross-validatation.)

You can use libsvm (http://www.csie.ntu.edu.tw/ cjlin/libsvm/) for the support vector machine.

2.6 Evaluation

To evaluate your system you should take all test images from all classes and rank them based on each binary classifier, so you should classify each test image with each classifier and then sort them based on the classification score. As a result you will have four lists of test images. Ideally, you would have images with airplanes on the top of your list which is created based on your airplane classifier, and images with cars on the top of your list which is created based on your car classifier, and so on.

In addition to the qualitative analysis you should measure the performance of the system quantitatively with the Mean Average Precision over all classes. The Average Precision for a single class c is defines as

$$\frac{1}{m_c} \sum_{i=1}^{n} \frac{f_c(x_i)}{i} \ , \tag{1}$$

where n is the number of images ($n = 50 \times 4 = 200$), m is the number of images of class c ($m_c = 50$), x_i is the i^{th} image in the ranked list $X = \{x_1, x_2, \dots, x_n\}$, and finally, f_c is a function which returns the number of images of class c in the first i images if x_i is of class c, and 0 otherwise. For example, if we want to retrieve R and we get the following sequence: $[R, R, T, R, T, T, R]$, then $n = 7$, $m = 4$, and $AP(R, R, T, R, T, T, R) = \frac{1}{4}\left(\frac{1}{1} + \frac{2}{2} + \frac{0}{3} + \frac{3}{4} + \frac{0}{5} + \frac{0}{6} + \frac{4}{7}\right)$.

3 Deliverables

Students are expected to prepare a report for this project. The report should include analysis of the results for different settings such as key points vs dense sampling, accuracy based on vocabulary size, accuracy based on SIFT descriptor used, accuracy based on number of training samples used and accuracy based on kernel choice for SVM.

A demo function which runs the whole system , should be prepared and submitted with all other implemented functions (excluding vlfeat and libsvm functions!). We do not intend to run your code for all images, however we will be looking at parts of it. Please, make a simple document with four ranked lists of test images as discussed in Section 2.6. This document should also contain all your settings (size of visual vocabulary, number of positive and negative samples, and so on), Average Precision per class, and Mean Average Precision.

The code should be clearly commented, explaining what each line is doing. Also, you should submit a readme file including your names and student identification numbers. i.e:

`John Smith 12345`

All files should be zipped and sent to **uvaims2013@gmail.com** before 27-12-2013, 23:59 (Amsterdam time). Subject of the email has to be **[IMS2013_FALL]**.