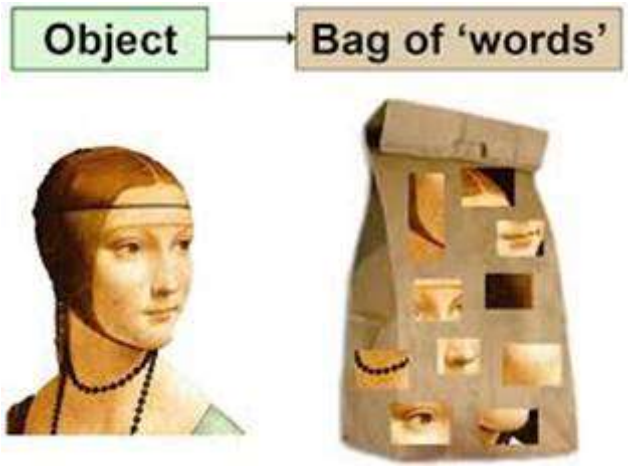
CV Homework Set #2 – Object Recognition using BOW

Due: 1/5/16

Submission: 1) HW will be tested interactively with the checker. Times will be posted.

2) HW code should be submitted to HW checker either via DOK or by email to:
imageprocessinghaifau@gmail.com



Goal: Write an object recognition system based on Bag Of Words (BOW).

General: You will be using the Computer Vision Open Source library **OpenCV**.
You may choose the programming code of your choice (C++, Java, Python, Matlab) to interface with the OpenCV code.
Unfortunately, no technical support will be given in this course. A discussion forum has been opened in the Moodle site for this course.

Grading: Grade will be given based on actively running the program on images.
Some example images you will be tested on will be available for you to test on.
Additional new images will be used for testing as well.
Grade will be based on quality of segmentation, as well as convenient interfacing, organization and documentation of code, ext.

Submission:

Submit all necessary files to run the code.
Submit a **Readme file** explaining how to compile, make, and prepare the code (including

Makefile if necessary).
Submit **Instructions file** explaining how to run the program, (explain inputs, interface,
howto, etc).
Do not forget to put Names and Student I.D. in all files.

HW2 Instructions:

Implement an object recognition system based on Bag Of Words.

Write code that implements the following:

1) Create a DataBase –
   a) Choose a specific object class enter your selection in the doc: https://docs.google.com/spreadsheets/d/12-NX0R_NzaKCOwpV1lwCOjpnYB43xCAWZbOnbWV5isc/edit?usp=sharing

   make sure you choose an object class that has not been chosen by another student.
   Suggestion – choose object class that are pretty well defined as a class but have
   variability.
   E.g. books is NOT a good class since every book cover is completely different.
   E.g. plain white mugs is NOT good since is not variable enough.

   b) Collect images of the object class (I would say 50 images at least). Search in Google images, flicker, object image DBs. Collect 3 levels of complexity for your images: simple (unoccluded object, typical frontal view, large in image, constant background, etc), to most complex (cluttered background, small object, unexpected view point etc).

   c) Label the images – You may label each image as a whole, OR mark a bounding box
   around the object in the image, OR mark the object boundary more exactly. The more effort in your labeling – the better will be the recognition performance.

   e) Collect images that are NON-class objects – these will be used later to build the classifier (step 4).

   f) Note: In the testing stage you will be asked to test on "new" images. So

you should probably collect these images at this stage as well. Make sure you have test images from all 3 classes of complexity. Also some NON-class images.

2) Build visual word dictionary – Use a feature descriptor of your choice (SIFT,HOG,SURF etc) to extract feature vectors from all your DB Images. Cluster the features into K clusters (use K-means, mean-shift or any clustering
method of your choice. K is a parameter (or window size in mean shift etc). This results in K visual words (usually numbered 1..K). Decide how you will represent each word so that later (in Step 3 and 5) new features found in image will be tested against the visual words. Another possible system parameter will be the complexity (dimension) of the feature descriptor.

   The routine created here might be run several times – using different parameters
   and different sets of images and their features.

3) Create frequency histogram (BOW) for each of the images in the DB. This will require writing a routine that determines for a given feature vector, which visual word it represents if at all. The clusters of visual words and/or their representations will be used here. Another system parameter determines when a feature is NOT any visual word (usually a threshold above which the feature is too far from all clusters).

4) Given the histograms (BOWs) of the DB image – build a classifier to distinguish between object and non-object. Build Classifier of your choice (SVM (linear or kernel), Fisher, Cascade etc).

5) Build recognizer – given an image determine if it contains the object or not. This routine will extract features from the image, determine the visual words these features represent, build BOW for the image and then classify the BOW using the classifier build in step 4.

6) Testing – this is the important stage of the project. You will report several recognition testing results. Submit all the testing results in a word or powerpoint document.
   Results are quantified by Precision and Recall values (see slide 96 in lecture

slides – or look online). Plot in ROC plot.
When requested to report performance: plot the ROC curve for the system parameters and then report the best parameter and the actual Precision/Recall Values.

a) Report performance of Object Recognition on the DB images:
   i) performance over ALL DB images
   ii-iv) performance over simple, medium and complex subset of DB images (you can plot the 3 performances in a single ROC plot).

b) Report performance on new test images. You can report separately for simple, medium and complex test cases (thus emphasizing success in simple cases).

c) Show example images that were falsely determined as object (False Positive / False Alarm) and images that were incorrectly classified as NON-object (false negatives / miss).

   Improve your object recognition system (add object images, non-object images or change methods of feature descriptors/clustering/classifications etc).
   Report and compare the improved results.