INTRODUCTION TO COMPUTER VISION

ASSIGNMENT 3

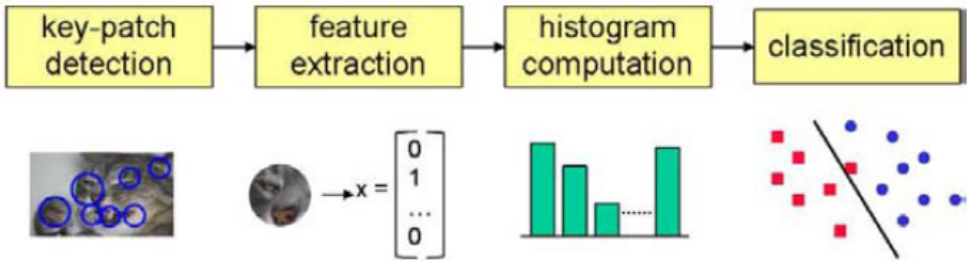
Antonio J. Rodriguez-Sanchez, 20-Jan-2016

Deadline: 17-Feb-2016

These assignment can be done in groups of 2 people

(10 points) Construct a basic object recognition system.

1. GOAL: Develop a system that can distinguish airplanes from cars. The object recognition system will be given an image and its output is to classify as containing an airplane or a car.
    a. The system will follow the process explained in lecture 11 (slide 68), namely:



2. THE FEATURES:

    You will use three features:
    a. The energy of Gabor filter responses.
    b. Scale Invariant Feature Transform (SIFT).
    c. HoG (Histogram of Gradients).

    For (a) you can use the code from assignment 1. For (b) and (c) you can download or use code from Matlab or OpenCV, these are popular feature descriptors that are in those libraries or available online.

3. THE DATASET:

    a. Download the image database with 2 classes of objects from the course website, this basic image database is obtained from Caltech 256 (http://www.vision.caltech.edu/Image_Datasets/Caltech256/). The 2 classes we are going to work on are faces (123 images) and cars (123 images). The category airplanes contain only images of airplanes, the category car contain images that have cars.

b. Learning how to use a classifier. We will use one of two classifiers: Adaboost and SVM. For Adaboost I provide the code and an example below. For SVM you have to look for a library and learn how to use it. Choose the one of your preference, if programming in Matlab, I recommend Adabost (you can use the code below). If OpenCV, I recommend SVM.
c. Read the images and assign.
    - Consider that the images are at different sizes and some are colored, others are black and white.
    - You must transform all images to the same size of 240x159, a useful function for this purpose is "imresize".
    - Colored pictures must be transformed to black and white, a useful function to achieve this is "rgb2gray" (in Matlab). Another more classical way is to use the color luminances of each channel:

```
0.2990*img(:,:,1)+0.5870*img(:,:,2)+0.1140*img(:,:,3)
```

    - To create a random set use "randperm".
    - Usually images are normalized regarding contrast, this can be done easily with just one line of code such as:

```
img2 = uint8((double(img)-double(min(min(img))))*(255/double(max(max(img))-
min(min(img)))));
```

4. OBJECT REPRESENTATION: feature extraction using Gabor filters/SIFT/HoG.

   For Gabor: We will use a very simple feature extraction technique in which we extract the edges at some orientation using a Gabor filter and then downsampling for efficiency
   a. Apply a Gabor filter to each image, use the energy of the Gabor function provided in the first assignment. You may use for example:
      - the real part and one orientation (e.g. 0 degrees) or
      - the energy of the Gabor at all orientations or
      - the preferred orientation at the specified pixel, etc.
      - Describe in the report what you used and how it works.

      You can reduce the size of the feature vector by computing a histogram of responses/feature values. E.g. If you are using the orientation of the filters, group them into a histogram. In this case, use increments of 10-15 degrees for the Gabor orientations.
   b. Transform each image feature into an N vector (E.g. 50 or150 elements). This is necessary because both, Adaboost and SVM work with sets of 1D vectors. A useful function for this is "reshape" (Matlab).
   c. You must end with a training set composed of 150 images (75 corresponding to cars and 75 corresponding to airplanes and a test set of the same size. These sets must be organized into arrays, in which each column corresponds to the image features (N features in total) and each row to each image (150 images for the training set and 106 for the test set):

	Feat. 1	Feat. 2	Feat. 3	Feat. 4	Feat. 5	Feat. 6	Feat. 7	.....	F. N
Image 1									
Image 2									
Image 3									
.....									
Image 150									

Repeat for SIFT and HoG.

5. LEARNING: Training a classifier. Download the code for a classfier. You can use the Adaboost code from the course website (in Matlab), or any other method, e.g. SVM (library libsvm for C++). You only need to use a classifier of your choice.

   For Adaboost (method seen in class), below it is explained how to use it.
   a. Use Adaboost to train the system with the training set.
      - As a weak learning we are going to use a classification tree:

        ```
        weak_learner = tree_node_w(3);
        ```

        This creates a classification tree with maximum three splits.
      - For training, each training image has a label (face or no face), assign label 1 to the ones containing a face, and label -1 the ones containing background images. Label also the test set, we will need it for later.
      - Then, we call Adaboost to construct the improved classification tree, e.g.

        ```
        [L W] = RealAdaBoost(weak_learner, TrainData, labels, 300);
        ```

        This calls Adaboost with the weak learner created before. "TrainData" corresponds to the training data (the matrix created in step 3d.). "Labels" are the labels for the training set (1 if the image contains a face, -1 if it does not contain a face). 300 is the number of iterations for Adaboost.
        The function returns "L" and "W" which are the learners (different classification trees) and the weights respectively (the higher the value the better the tree is).
   b. The same can be done with an SVM: Download an SVM library or code for Octave or C++ and learn how to use.

6. RECOGNITION: Using classifier in test data.

   We have now a trained system that can distinguish an image depending if there is a face or not, now we have to see if the system performs well with images not included in the training set and this is when we use the test set created before. Again for Adaboost:
   a. Call the learners "L" with their weights "W" created by Adaboost in the previous step to obtain the labels of the images in the test set:

      ```
      ResultR = sign(Classify(L, W, TestData));
      ```

      This will return the sign (+1 or -1) corresponding to the test data using the classification method created in step 4. +1 corresponds to faces, -1 to background. It will return in "ResultR", which is a vector of labels corresponding the test data.
   b. The equivalent can be done with an SVM.

7. RESULTS

At this point we have a classification system that can recognize if an image has a car or an airplane.
We have to measure how good is our system, for this we compare the true labels of the test set with the ones obtained in step 5. The higher the percentage the better, if it is below 50% it means our system is ever worse than using chance (a coin with tails and heads), if it is a bit over 50% it means it is ok, 70-80% good enough and over 90% quite good, if over 95% it is great.
Create a table reporting true positives, false positives, true negatives and false negatives.
Compare the three features: Gabor, SIFT, HoG to see which one performs better at this task. This should be reported in graphs, tables or ROC curves.

Deliverable: Document with the following:

Page 1

a) Introduction on object recognition, the steps to achieve object recognition.

b) The purpose of this assignment. State the *hypothesis*.

Page 2

Your solution: How your solution was implemented and the algorithm you followed to solve the assignment. How the features were created and what are their basis for their creation. Comment on Adaboost and other classification methods such as SVM and how they compare and the differences (you may need to do some research for this).

Page 3

Results: Include the results of the assignment. Provide cases where the method worked and cases where it did provide the wrong answer. Give the success rate (percentage of correctly classified images) and TP/TN/FP/FN.
Compare the results from the Gabor filter, SIFT and HoG for each classifier.

Page 4

Discussion: What you learnt about this assignment and how the method can be improved or extended.

Appendix

Code and anything else you may consider of interest

References

HoG: http://rogerioferis.com/VisualRecognitionAndSearch2014/material/papers/DPMCVPR98.pdf

SIFT: http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf

Libsvm: https://www.csie.ntu.edu.tw/~cjlin/libsvm/

APPENDIX

Code for the Gabor filter

```
% Parameters: n -> Size of filter
%             sigma_y and sigma_x -> Sigma values for x and y coords.
%             theta -> Orientation of the Gabor filter
%             x0, y0 -> Center coordinates
% Output:     G_even -> Even part of Gabor filter
%             G_odd -> Odd part of Gabor filter

function [G_even G_odd] = GaborD(n, sigma_y, sigma_x, theta, pr, x0, y0)

if length(n)>1,
  incx = 2*n/(n(1)-1);
  incy = 2*n/(n(2)-1);
else
  incx = 2*n/(n-1);
  incy = 2*n/(n-1);
end
[X,Y] = meshgrid(-n:incx:n,-n:incy:n);     % this creates x and y coordinates
xp = (X-x0)*cos(theta)+(Y-y0)*sin(theta);  % this controls the orientation
yp = -(X-x0)*sin(theta)+(Y-y0)*cos(theta);

G_even = 1./(2*pi*sigma_x*sigma_y).*exp(-.5*((xp-x0).^2./sigma_x^2+(yp-y0).^2./sigma_y^2)).* ...
  cos(2*pi/(4*sigma_x)*pr.*xp); % Even part of Gabor filter

G_odd = 1./(2*pi*sigma_x*sigma_y).*exp(-.5*((xp-x0).^2./sigma_x^2+(yp-y0).^2./sigma_y^2)).* ...
  sin(2*pi/(4*sigma_x)*pr.*xp); % Odd part of Gabor filter

end
```

Code for Gaussian pyramid

```
function [img] = GaussianPyramid(img,levels)
g_filter = [1 4 6 4 1]/16;

for i = 1:levels
    img = conv2(g_filter,g_filter,img,'same');
    img_size = size(img);
    img = img(1:2:img_size(1),1:2:img_size(2));
end
```