Computer Vision **CS 543 / ECE 549**

Homework 5
Due April 20, 2015

Answer the following questions and explain all solutions. Numbers in parentheses give maximum credit value.

1. Face Recognition (50%)

In this assignment you will implement the Eigenface and Fisherface methods for recognizing faces. You will be using face images from the Yale Face Database B where there are 10 faces under 64 lighting conditions. Using your implementation, you will evaluate the ability of the algorithm to handle lighting conditions of the probe images that differ from those in the training images.

You can download the data for this assignment at http://www.cs.ucsd.edu/classes/sp05/cse152/faces.zip

For more information on the Yale Face Database, see
http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html

Relevant papers:
http://www.cs.tau.ac.il/~shekler/Seminar2007a/PCA%20and%20Eigenfaces/eigenfaces_cvpr.pdf
http://www.cs.columbia.edu/~belhumeur/journal/fisherface-pami97.pdf

A. Recognition with Eigenfaces (PCA)

Step 1. Take each 50 × 50 pixel training image and vectorize into a 2500-dimensional vector. Then perform principal component analysis (PCA) on the entire set of training image vectors, retaining the first *d* principal components. Use the trick (discussed in class) for avoiding the full 2500x2500 covariance matrix. The *d* eigenvectors (when reshaped and displayed as images) are the Eigenfaces. You will display the top eigenfaces as images in your report.

Step 2. Now, for each of your training images, project to the d-dimensional Eigenspace. Once this is done, classification will be performed by nearest neighbor with the L2 (Euclidean distance) metric in the Eigenspace.

Step 3. Evaluate your algorithm on the frontal pose of the ten people in the Yale Face Database B. These images have been cropped and aligned for you. From the set of all ten individuals, we will consider 5 subsets, indexed as below:

Set 1. person*01.png to person*07.png
Set 2. person*08.png to person*19.png
Set 3. person*20.png to person*31.png
Set 4. person*32.png to person*45.png
Set 5. person*46.png to person*64.png

We have provided code "readFaceImages.m" to read in the images and store the person labels and subset numbers for each image.

What to report:

1) Train your Eigenface algorithm with $d = 9$ and $d = 30$ on all images in subset 1 (70 images). Then, evaluate your algorithm on subsets 1-5, and report the error rates in a table (error rate for each subset). For subset 1, you would expect perfect recognition because it is used for training. Include pseudocode (or equations) for the Eigenface recognition algorithm. (15%)
2) Display the top 9 eigenvectors (eigenfaces) after training. I suggest using `subplot(3,3,k)` with `imagesc` and `axis image, axis off, colormap gray` and save the figure as a png. (5%)
3) For both $d = 9$ and $d = 30$, display one original and the corresponding reconstructed face from each subset. The reconstructed face is constructed from the mean vector and from the eigenfaces and their coefficients. I suggest using `subplot(2, 5, k)` to display all five original and reconstructed faces in one figure. (5%)

B. Recognition with Fisherfaces (FLD)

Extend your Eigenface algorithm to perform the Fisherface algorithm discussed in class. First, project the training images via Eigenfaces to a $n - c$ dimensional space (n=number of images, c=number of different people); second, apply Fisher's Linear Discriminant to obtain $c - 1$ dimensional feature vector for each image.

What to report:

1) Evaluate the Fisherface algorithm. Train your Fisherface algorithm with $c = 10$ and $c = 31$ on all images in subset 1, and classify subsets 1-5 as in Part A. Report the error rates in a table (error rate for each subset). Use the same table as in Part A. Include pseudocode or equations for the Fisherface algorithm. (15%)
2) Now retrain PCA and FLD using both subset 1 and subset 5 as training data. Test on all subsets. (10%)

Your final table should look something like this:

Method (train set)	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
PCA (S1)					
FLD (S1)					
PCA (S1+S5)					
FLD (S1+S5)					

Notes:

1) You may not use built-in PCA functions like `pcacov` or `princcomp`. Use `eig` for computing the PCA and FLD subspaces. Be careful about whether the first or last columns correspond to the largest eigenvalues, as it depends on the way that you call `eig`.
2) You should pre-normalize each face by subtracting its mean and dividing by its standard deviation, as this will lead to better performance.
3) I encourage you to explore further on your own. Try different sized subspaces, or try training on different subsets and see how it impacts the others. Look at the smallest eigenvectors or at reconstruction error as the number of eigenvectors vary. You don't have to include these extra explorations in your report.

Acknowledgement: The data and basic design of this homework problem are from David Kriegman.

2. Scene Categorization (50 pts)

In the supplemental material, we have supplied images with 8 outdoor scene categories[1]: coast, mountain, forest, open country, street, inside city, tall buildings and highways. The dataset has been split into a train set (1888 images) and test set (800 images), placed in `train` and `test` folders separately. The associated labels are stored in `gs.mat`, for example, label id of `42.jpg` in the training folder corresponds to `train_gs(42)`. Its actual label name will be `names{train_gs(42)}`. Your task is to implement features, train a classifier on the training set, and evaluate the classifier on the test set.

1. Image categorization using color histograms and nearest neighbor classifier (15 pt):
   - Implement a function to compute the color histogram of an image
   - Implement spatial pyramid and use spatial pyramid of color histogram as your image representation
   - Use nearest neighbor classifier to categorize
2. Bag of Words model and nearest neighbor classifier (10 pt)
   - Implement K-means clustering algorithm to compute visual word dictionary (write your own kmeans)
   - Use the included SIFT word descriptors included in `sift_desc.mat` to build spatial pyramid of visual words as your image representation
   - Use nearest neighbor classifier to categorize them
3. Choose one or more ways to further improve your performance (15 pt). This can include but is not limited to:
   - Use a discriminative classifier (e.g., SVM).
   - Use fisher vector encoding, 2nd order pooling, or other ways of pooling SIFT features.
   - Incorporate additional features.
4. Describe your design decisions and report results (10 pt)
   - Color histogram & Pipeline: Your quantization/binning method and parameters, levels of pyramid, K for KNN classifier, etc.
   - K-Means & BoW: Number of visual words, K-means stopping criterion, your way to combine BoW and color histogram, etc.
   - Improvement: The dimensionality of your new features, running time of your classifier, computing distance measure, etc. Make sure you provide enough description on what you have you done and what external resources you used.
   - Results: You should show results for at least three models: color histogram, BoW, and your improved model. For each of the methods you implemented, display the confusion matrix (e.g. textttimagesc(confusion_matrix)); also, report overall categorization accuracy (the percentage of test images that are correctly categorized).

Tips:

- If your feature computation is time consuming, you may consider caching them as your features compute. For K-means clustering, for speed reasons, we suggest using a low number of data points and clusters. For example, you may use 10,000 SIFT descriptors to create a 100 word codebook. You may get about 51% accuracy using color histograms and 1-NN classifier with L2 distance. With improvements, it is possible to achieve more than 80% accuracy. However, you will not judged primarily by your results.
- You should come up with reasonable parameters and design decisions on your own (e.g., don't ask what K to use, what bin size, what color space, etc.).
- You may use built-in function `hist` and any existing classifier code.

[1] `http://people.csail.mit.edu/torralba/code/spatialenvelope/`

3. Image classification using deep convolutional neural network [Extra credit up 10pts]

Convolutional neural networks have recently demonstrated impressive performance in visual recognition tasks. For this problem, you will apply a pre-trained CNN model from the ImageNet classification task (http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf) to your own photos. You can use the ConvNet library in MATLAB http://www.vlfeat.org/matconvnet with one of the pre-trained models http://www.vlfeat.org/matconvnet/pretrained to classify your own photos (these could be your own photos or ones that you download from online that you think would make interesting tests – don't use standard datasets).

Include in your writeup:

- Three images that were correctly classified (the main object in the image is identified). Show each image and report the top-5 predicted classes and their scores.
- Three images that were incorrectly classified (i.e., the true class is not in the top-5 predicted classes). Show each image and report the top-5 predicted classes and their scores.
- Describe at least three factors that would make object classification in this setting (trained on ImageNet to classify images of objects into one of thousands of categories) difficult.