Ayivi Seylom (ayivigu2) **HW5**
CS 543 Computer Vision

1) Face Recognition

A. Recognition with Eigen faces

Method (train set)	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
PCA (S1) d = 9	0	0	0.216	0.654	0.873
PCA (S1) d = 30	0	0	0.042	0.565	0.91

Pseudo code:

```
Train:
  • Load all images from training set   ($NxN$  images)
  • For each image in training set
      o Reshape image ($N^2x1$) and perform mean normalization
      o Add M row vector images obtained to a matrix
  • Compute Eigen vectors for $N^2xM$


Recognition:
  • Load images from the testing set
  • For each image
      o Reshape to row vector and perform mean normalization
      o Convert to reduce dimension image after subtracting mean of training
        examples and use nearest neighbors to find a match the closest match in
        the training set.
```

1)

2)

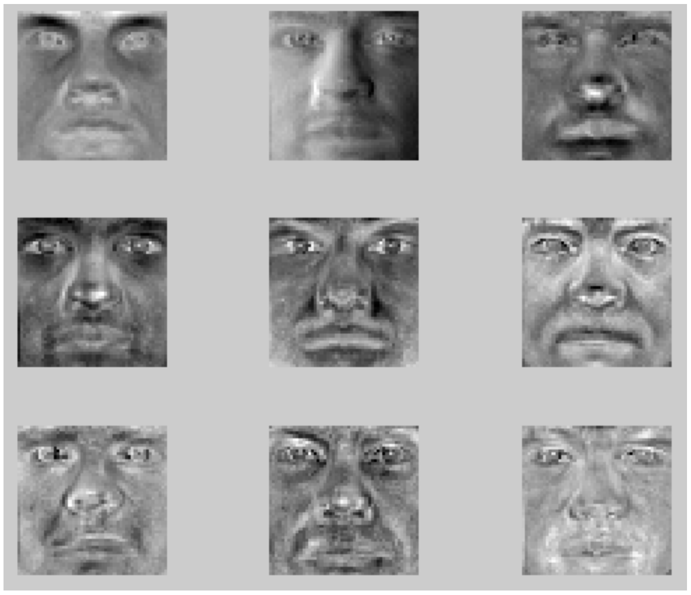

Figure 1 - Eigenfaces

3)

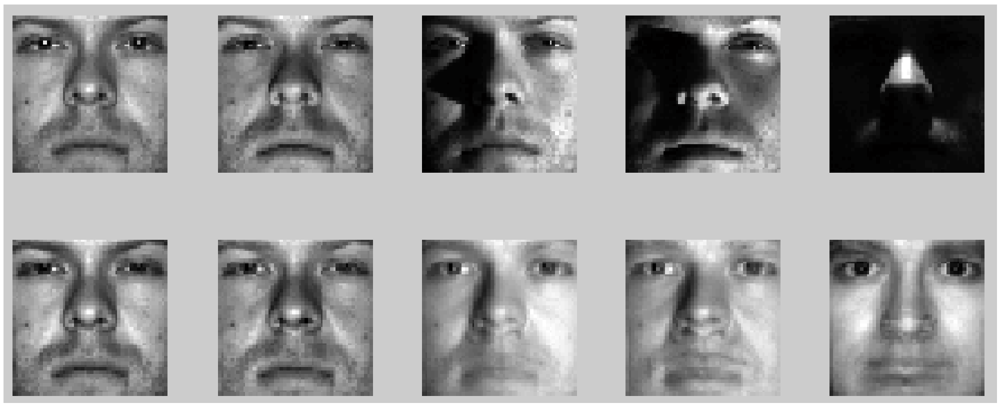

Figure 2: d =9

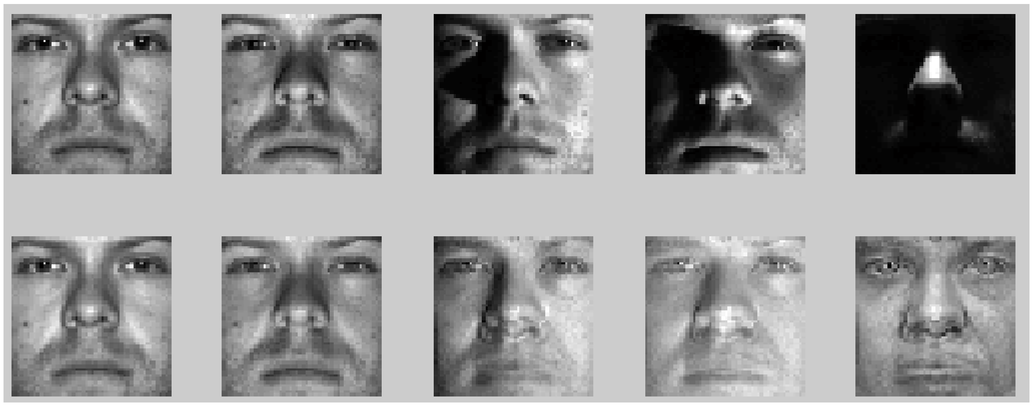

Figure 3: d =30

B. Fisher faces

d=9, c=10

Method (train set)	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
PCA (S1)	0	0	0.225	0.665	0.857
FLD (S1)	0	0	0.008	0.51	0.91053
PCA (S1 + S5)	0	0.16667	0.725	0.69286	0
FLD (S1 + S5)	0	0.025	0.1916	0.23	0

d = 30, c= 31

Method (train set)	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5
PCA (S1)	0	0	0.041	0.56	0.81579
FLD (S1)	0	0	0.01	0.57	0.81
PCA (S1 + S5)	0	0.025	0.19167	0.23571	0
FLD (S1 + S5)	0	0.025	0.1916	0.23	0

Fisher algorithm:

- Mean normalize every image vector and subtract mean of all images.
- Compute the Principal components of the image (similar to the Eigen face step) and discard c vectors in order to keep n-c vectors.
- Compute the between class scatter and the within class scatter
- Project both scatter matrixes obtained onto a lower dimensional space using the Eigen vectors obtained at the beginning.
- Find the optimal weights for the fisher matrix by solving the generalize Eigen problem.

- Given a face, compute a new fisher face by using the new weighed matrix and the mean of the fisher faces from the training set.
- Compute the distance or the reduced face to all reduced training vector faces.
- Choose the nearest neighbor as the assigned face.

2) Scene Categorization

A. Using spatial pyramids.

File: ***runthisPyramid.m***

NOTE: In order to run the file, you will need to update the path to point to the directory containing the script. I was unable to find a way to automatically retrieve it. Sorry for the inconvenience.

Computations take a few minutes to run.

Several methods were investigated in order to assess the influence of multiple levels of pyramid as well as marginal versus joint binning schemes. Below are the steps of the algorithms used to compute the multi-level pyramids:

- For each image
  - Given the number of level divide the image in patches were each patch
    - Divide the image into patches for the level.
    - Compute histograms for each patches
    - Combine patches appropriately for each level.
  - Concatenate all levels of the pyramid with some weighting per level to obtain the image feature vector.

About the weighting:
I used a simple division by 4 ^ (level-1) (no weighting for the whole image level) because each level is divided into 4 patches. This was just for experimenting, as I didn't obtain better than a **60%** error rate with weighting.

I use multiple different combinations of pyramid levels as well as histogram quantization methods which all gave rather similar results. I wasn't able to get higher than **52% accuracy**.

Some of the combinations I have used:

- RGB to HSV with [10, 10, 10] as bins for each of the three channels and 1 levels pyramid : **49.50% accuracy** .

- RGB to HSV with [10,5,10] bins per channel and 1 level pyramid : **52% accuracy**
- RGB to HSV with [10,5,10] bins per channel and 2 level pyramid : **48% accuracy**

Color histogram implementations I experimented with:

1)

- Compute histogram per channel using hist.
- Normalize each channel and concatenate the result of each channel.
- Use the concatenated vector as a feature.
- Concatenate each level of the pyramid without any weighting to obtain the image feature vector.

2)

- Compute all possible bin combinations C, (number of combinations obtained by multiplying the number of bins for each channel)
- The histogram for an image is the combination of histograms for each channels. The feature vectors will have C components per pyramid level

The vector obtained grows quite fast, and unless I could select a large enough number of bins the results was not as different from the concatenation of bins for each channel.

<u>Classification:</u>

I used a 1-nearest neighbor to find the closest match in the training set an assign the image id to it.

As show in my code I computed the closest distance using

- Histogram intersection
- Sum of squared differences,
- Sum of absolute value differences.

I obtained very similar results every time with all measures. Best result: **52% accuracy**.

<u>**B- Using a bag of word.**</u>

File: ***runthisBagOfWords.m***

NOTE: In order to run the file, you will need to update the path to point to the directory containing the script. I was unable to find a way to automatically retrieve it. Sorry for the inconvenience.

<u>K-means algorithm implementation:</u>

- Given a set of data points and a predefined number of clusters:
- do:
    - Randomly select k points to be the initial cluster centers
    - For each data points
        - Compute its distance to every cluster centers
        - Assign the point to the closest cluster center
    - Compute new cluster centers by computing the mean of all points assigned to a particular cluster

  While (clusters are above a threshold or we have executed a certain amount of loop)

<u>Bag of words implementation</u>

We aim to select 10000 randomly selected sift vectors from our training images. Steps used to construct the bag of words are the following:

- Compute the number of sift vector k to gather from each image.
- For each image:
    - Randomize the list of sift descriptor and choose the first k sift vectors
    - Add the selected vectors to the list of vectors to use to build the dictionary
- Run k-means clustering using the vectors sifts gathered and predefined the number of words in the dictionary. In our case, we chose 200. (those are the cluster centers)
- For each image
    - For each sift vector in the image
        - Find the closest cluster center in the vector region and assign the sift vector to the cluster center.
        - Increment the count of the word (assigned cluster) for the image.

To build a pyramid of our bag of words we use an approach similar to the one used for spatial histograms. We divide the image in patches and assign sift vector inside the patch to their closest cluster centers relative to the region.

The file ***buildBagOfWordPyramid.m*** shows my implementation of the patch based clustering.

Using a single 0-level pyramid gave about **48% accuracy** for our 200 words dictionary.

Trying to improve our algorithm:

- Changing the vocabulary to 300-500 words the accuracy jumped to **52.5% accuracy**.
- I was able to find some section of code to improve the computation of the nearest neighbors. My initial implementation was not vectorized. After using that code to compute the nearest clusters for histogram construction the runtime decreased from several minutes to slightly above a minute.
- I used the built-in ***histc*** function to compute the bag of word histogram for each image.
- I did not tried to see the results for more than a single level of pyramid.

3) Object category detection

Any object with a shape that is not constant or depends on positions, camera angle, size, may affect the detection.

- As an example humans or animals could fail the detection since their positions is not always identically defined in a bounding box and they are subject to deformation.
- Some objects can look very different based on viewing angle and scale. For example Using a bag of words could help increase the chance of accurate detection.
- Complex objects may not be detectable unless you can confidently recognize parts of it. In that case, learning to recognize patches (or specific parts of the object) may help.

  We could think about using an adjustable sliding window that could mirror the shape of the object to recognize. Also, as previously mentioned, combining classifiers such as SVM and bag of words could prove helpful.