ROBOTICS & CV PROJECT 4

Bag of Words & Eigenmethods Recognition

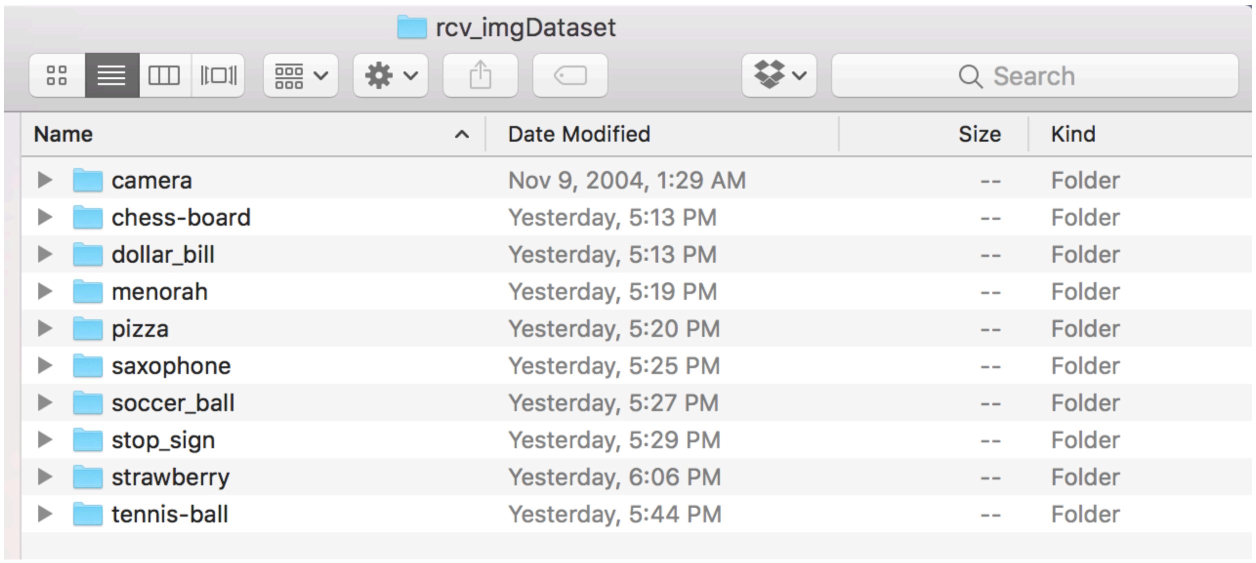
Niral Shah

Professor K. Dana | 12/06/16

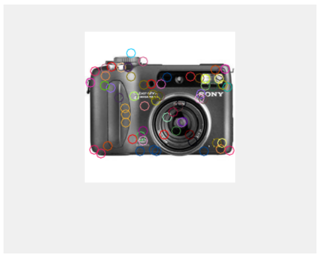
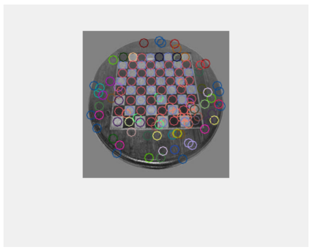
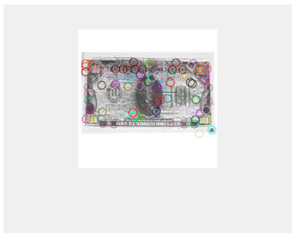
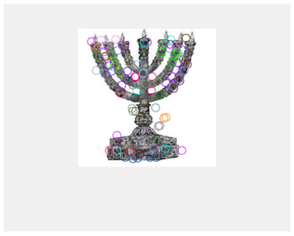
I. Bag of Words Project:

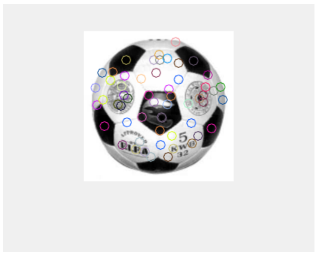
Dataset:

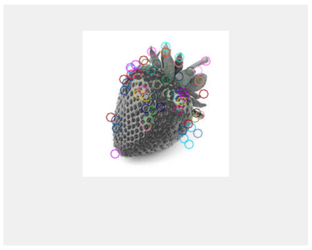
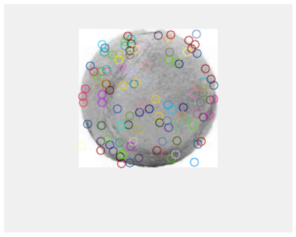
- These the ten classes are from the Caltech 101 Object Library (50 images each)


rcv_imgDataset

Name	Date Modified	Size	Kind
camera	Nov 9, 2004, 1:29 AM	--	Folder
chess-board	Yesterday, 5:13 PM	--	Folder
dollar_bill	Yesterday, 5:13 PM	--	Folder
menorah	Yesterday, 5:19 PM	--	Folder
pizza	Yesterday, 5:20 PM	--	Folder
saxophone	Yesterday, 5:25 PM	--	Folder
soccer_ball	Yesterday, 5:27 PM	--	Folder
stop_sign	Yesterday, 5:29 PM	--	Folder
strawberry	Yesterday, 6:06 PM	--	Folder
tennis-ball	Yesterday, 5:44 PM	--	Folder

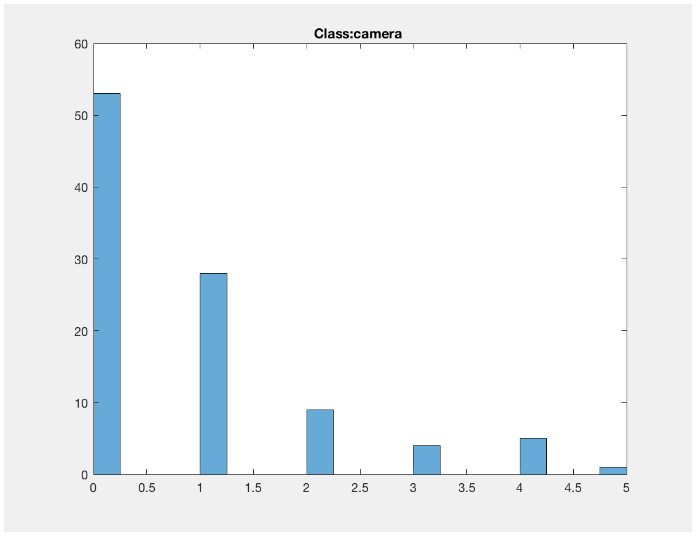


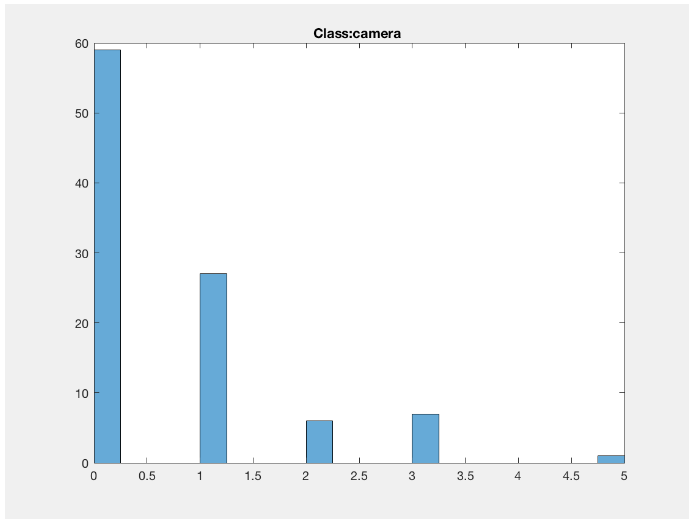



Comparing Visual Words:

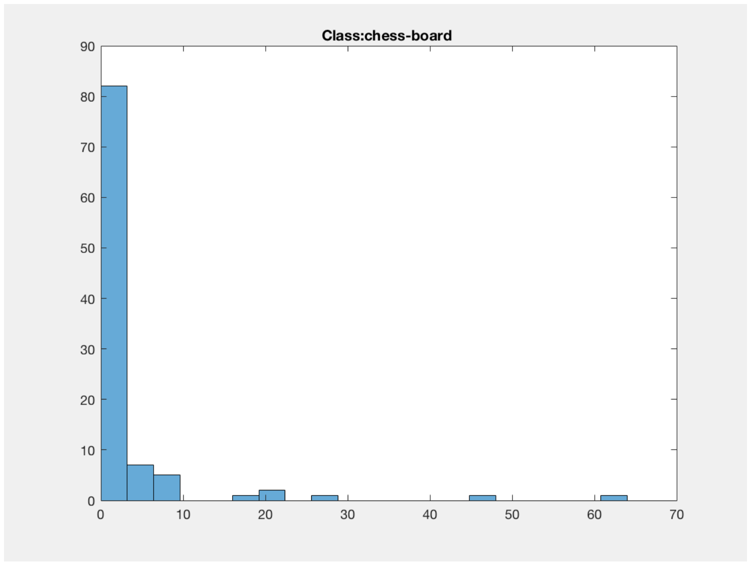
Once a visual-word dictionary was built, with K = 100, a hard-weighting scheme was used to construct histograms for each image in the training set, these are the visual words for each class. To demonstrate how two images from the same class have similar histograms the likeness is shown below:

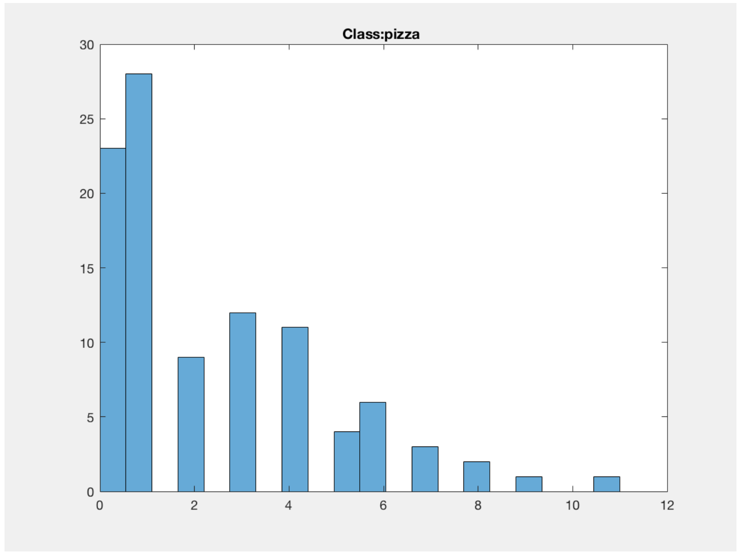
Histograms of two images of a camera:





Compare the above histograms to those of a chess-board and a pizza-pie

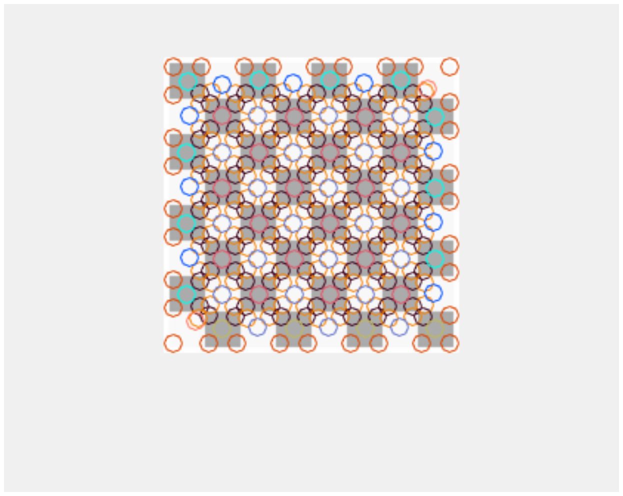


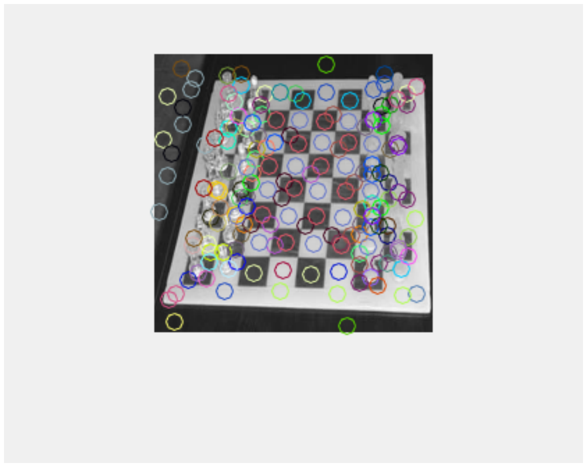


Evaluating Visual Words

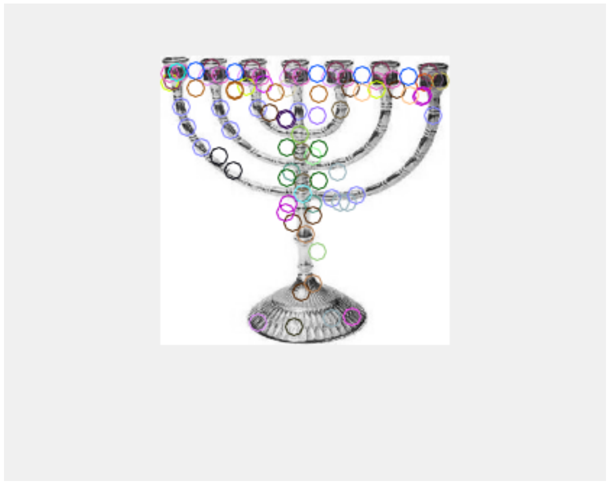
In order to tell if the visual words actually have any consistent meaning to them, it should be observed that similar features are marked by the visual words for each class of images. This was observed by using K distinguishable colors, where each color represents a different visual word. Below are some examples of images with their visual-words plotted on-top of the image:

Chessboard:

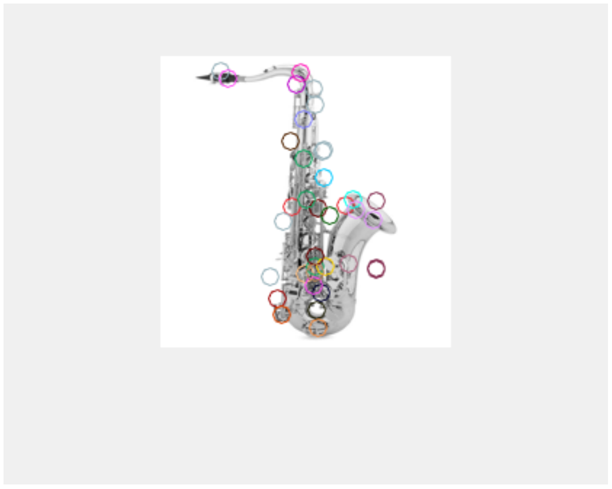




Menorah:



Saxophone



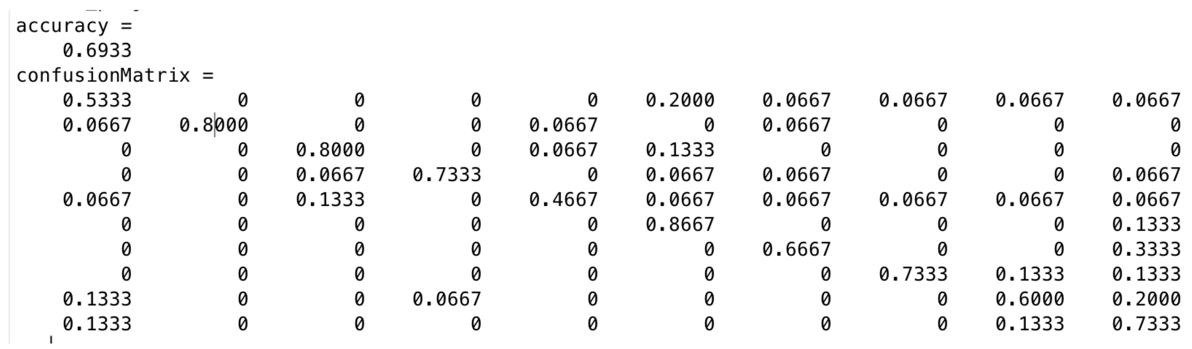
As you can see like features are consistently mapped with the same visual word. One easy to observe example would be for the color pink. It appears the pink is associated with a darker color features in the image. Thus, in the chess board it picks up the darker squares, for a saxophone the reed and for a menorah the actual candle holder.

Confusion Matrix Statistics & Results:

Overall had a **69% accuracy** over five-splits**,** recognizing objects in the different classes. As the confusion matrices show below, several classes of images were very tricky in accurately identifying and similar visual words found in each class contributed to misclassification. In particular, it seems the camera class and saxophone, were easily mis-classified and brought the overall recognition rate down. This may be due to a variety of factors including object orientation, differences in the images, the existence of background that create extra interest points. Furthermore, the results below are from choosing the number of words in the Visual Dictionary, k to be 100. In addition, a hard-weighting scheme was used. The soft-weighting scheme as suggested by the professor, brought in a lot of noise and made it tougher to recognize objects. Through experimentation, it was observed K values greater than 100 had no noticeable improvements, and K values below increase mis-classification. Also the visual dictionary was created from a subset of the training images, about 20%.

Split 1:

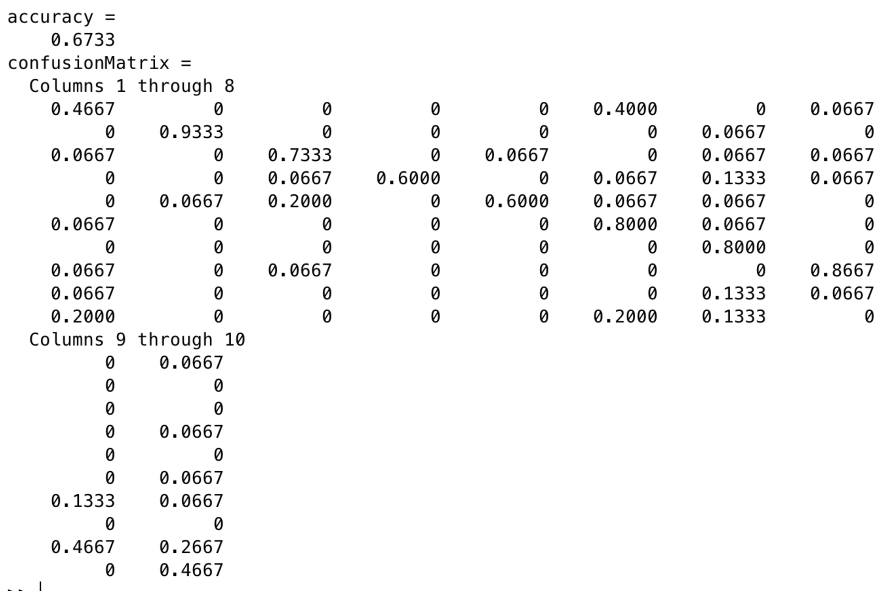
69.33 % - Accuracy



```
accuracy =
    0.6933
confusionMatrix =
    0.5333         0         0         0         0    0.2000    0.0667    0.0667    0.0667    0.0667
    0.0667    0.8000         0         0    0.0667         0    0.0667         0         0         0
         0         0    0.8000         0    0.0667    0.1333         0         0         0         0
         0         0    0.0667    0.7333         0    0.0667    0.0667         0         0    0.0667
    0.0667         0    0.1333         0    0.4667    0.0667    0.0667    0.0667    0.0667    0.0667
         0         0         0         0         0    0.8667         0         0         0    0.1333
         0         0         0         0         0         0    0.6667         0         0    0.3333
         0         0         0         0         0         0         0    0.7333    0.1333    0.1333
    0.1333         0         0    0.0667         0         0         0         0    0.6000    0.2000
    0.1333         0         0         0         0         0         0         0    0.1333    0.7333
```

Split 2:

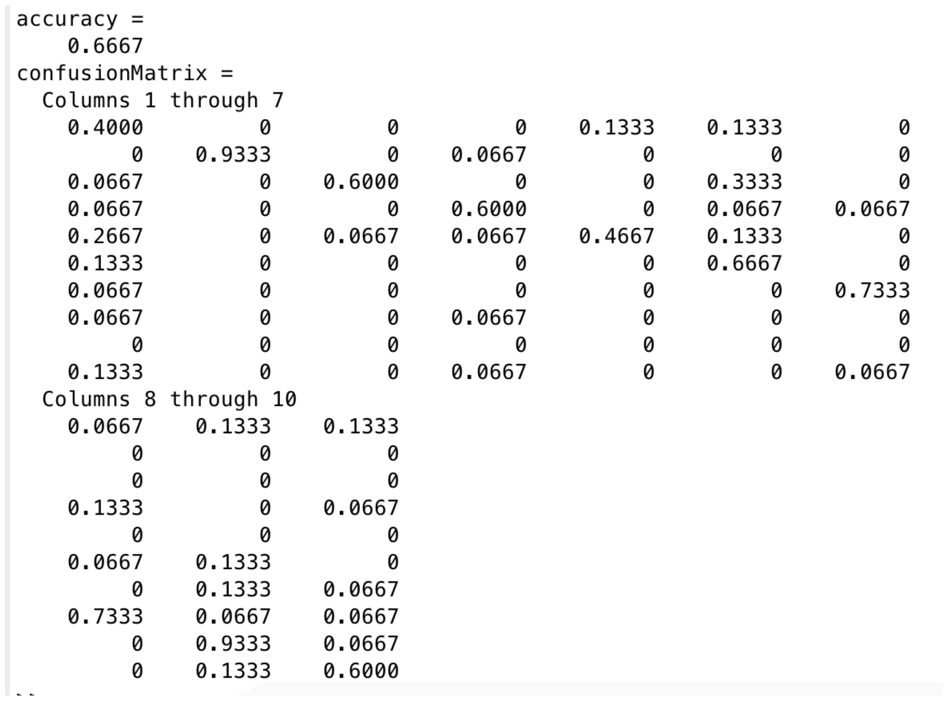
67.733 % - Accuracy



```
accuracy =
    0.6733
confusionMatrix =
  Columns 1 through 8
    0.4667         0         0         0         0    0.4000         0    0.0667
         0    0.9333         0         0         0         0    0.0667         0
    0.0667         0    0.7333         0    0.0667         0    0.0667    0.0667
         0         0    0.0667    0.6000         0    0.0667    0.1333    0.0667
         0    0.0667    0.2000         0    0.6000    0.0667    0.0667         0
    0.0667         0         0         0         0    0.8000    0.0667         0
         0         0         0         0         0         0    0.8000         0
    0.0667         0    0.0667         0         0         0         0    0.8667
    0.0667         0         0         0         0         0    0.1333    0.0667
    0.2000         0         0         0         0    0.2000    0.1333         0
  Columns 9 through 10
         0    0.0667
         0         0
         0         0
         0    0.0667
         0         0
         0    0.0667
    0.1333    0.0667
         0         0
    0.4667    0.2667
         0    0.4667
```

Split 3:

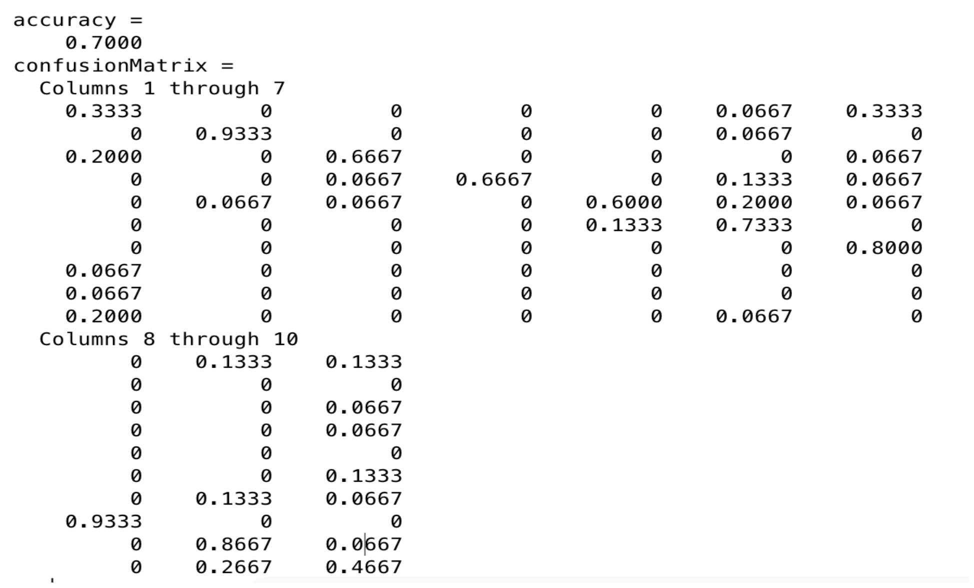
66.67 % - Accuracy



```
accuracy =
    0.6667
confusionMatrix =
  Columns 1 through 7
    0.4000         0         0         0    0.1333    0.1333         0
         0    0.9333         0    0.0667         0         0         0
    0.0667         0    0.6000         0         0    0.3333         0
    0.0667         0         0    0.6000         0    0.0667    0.0667
    0.2667         0    0.0667    0.0667    0.4667    0.1333         0
    0.1333         0         0         0         0    0.6667         0
    0.0667         0         0         0         0         0    0.7333
    0.0667         0         0    0.0667         0         0         0
         0         0         0         0         0         0         0
    0.1333         0         0    0.0667         0         0    0.0667
  Columns 8 through 10
    0.0667    0.1333    0.1333
         0         0         0
         0         0         0
    0.1333         0    0.0667
         0         0         0
    0.0667    0.1333         0
         0    0.1333    0.0667
    0.7333    0.0667    0.0667
         0    0.9333    0.0667
         0    0.1333    0.6000
```

Split 4:

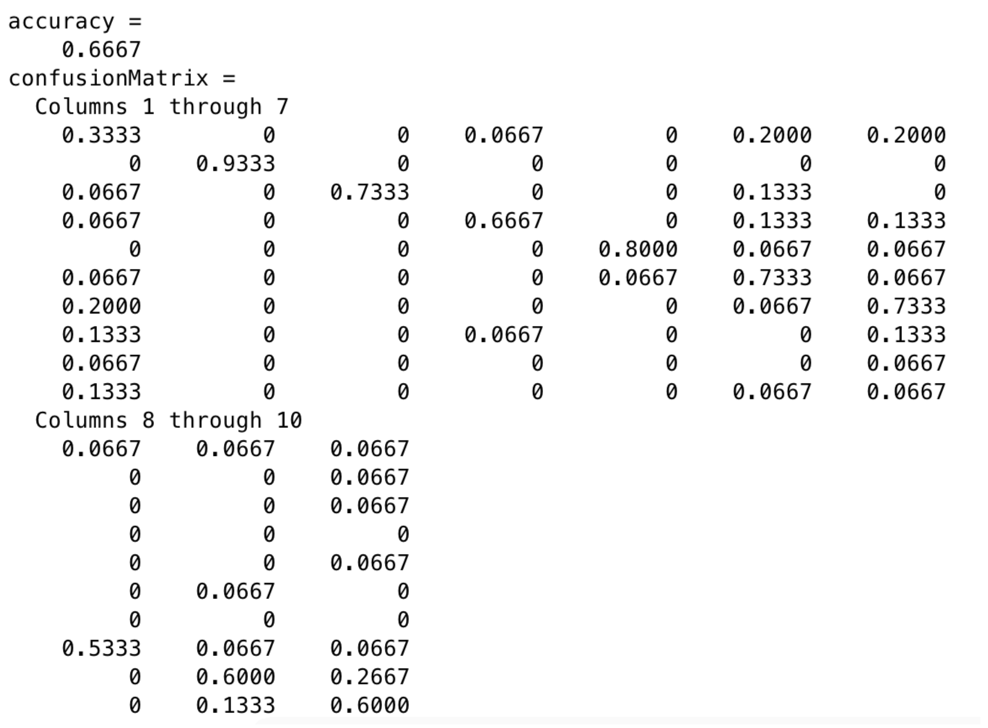
70% - Accuracy



```
accuracy =
    0.7000
confusionMatrix =
  Columns 1 through 7
    0.3333         0         0         0         0    0.0667    0.3333
         0    0.9333         0         0         0    0.0667         0
    0.2000         0    0.6667         0         0         0    0.0667
         0         0    0.0667    0.6667         0    0.1333    0.0667
         0    0.0667    0.0667         0    0.6000    0.2000    0.0667
         0         0         0         0    0.1333    0.7333         0
         0         0         0         0         0         0    0.8000
    0.0667         0         0         0         0         0         0
    0.0667         0         0         0         0         0         0
    0.2000         0         0         0         0    0.0667         0
  Columns 8 through 10
         0    0.1333    0.1333
         0         0         0
         0         0    0.0667
         0         0    0.0667
         0         0         0
         0         0    0.1333
         0    0.1333    0.0667
    0.9333         0         0
         0    0.8667    0.0667
         0    0.2667    0.4667
```

Split 5:

66.67 % Accuracy



```
accuracy =
    0.6667
confusionMatrix =
  Columns 1 through 7
    0.3333         0         0    0.0667         0    0.2000    0.2000
         0    0.9333         0         0         0         0         0
    0.0667         0    0.7333         0         0    0.1333         0
    0.0667         0         0    0.6667         0    0.1333    0.1333
         0         0         0         0    0.8000    0.0667    0.0667
    0.0667         0         0         0    0.0667    0.7333    0.0667
    0.2000         0         0         0         0    0.0667    0.7333
    0.1333         0         0    0.0667         0         0    0.1333
    0.0667         0         0         0         0         0    0.0667
    0.1333         0         0         0         0    0.0667    0.0667
  Columns 8 through 10
    0.0667    0.0667    0.0667
         0         0    0.0667
         0         0    0.0667
         0         0         0
         0         0    0.0667
         0    0.0667         0
         0         0         0
    0.5333    0.0667    0.0667
         0    0.6000    0.2667
         0    0.1333    0.6000
```

Contents

- Visual Word Dictionary
- Hard-weighting
- Soft-Weighting:
- BOW Classification:

```
%Niral Shah
% RCV Bag of Words Implementation


% read in data set
setDir  = fullfile(toolboxdir('vision'),'visiondata','imageSets');
imds = imageDatastore('/Users/niralshah/Desktop/rcv_imgDataset/','IncludeSubfolders',true,'LabelSource',...
    'foldernames');

[trainingSet,testSet] = splitEachLabel(imds,0.7,'randomize');

% Look at a subset of the training set to create Visual Word Dictionary
subtrainingSet = splitEachLabel(trainingSet,0.2,'randomize');
K = 100;
```

Visual Word Dictionary

```
global_descriptor = [];

for i =1:length(subtrainingSet.Files)
    [I, fileinfo] = readimage(subtrainingSet,i);
    %fileinfo.Label

    [x,y,z] = size(I);
    if(z ~= 1)
        I = rgb2gray(I) ;
        I = imresize(I,[200,200]);
    else
        I = imresize(I,[200,200]);
    end
    %figure;
    %imshow(I)
    %title(['Class:' char(fileinfo.Label) ' '])

    [f,d] = vl_sift(single(I));
    global_descriptor = [global_descriptor; d'];

end

[idx,C] = kmeans(double(global_descriptor),double(K));
```

```
centroids = C'; % Have K (128 x 1) centroids
```

Hard-weighting

for loop to compare descriptors to find the closest visual word label do this for every descriptor and increment to find kx1 histogram of labels

```
labels = grp2idx(trainingSet.Labels);
vw_histogram = [];
for i =1:length(trainingSet.Files)
    [I, fileinfo] = readimage(trainingSet,i);
    [x,y,z] = size(I);
    if(z ~= 1)
        I = rgb2gray(I) ;
        I = imresize(I,[200,200]);
    else
        I = imresize(I,[200,200]);
    end


    colors = distinguishable_colors(K);
    [feat,des] = vl_sift(single(I));
    [xlen,ylen] = size(des);
    hg_training_image = zeros(1,K);
    iPts = [];

    for j = 1:ylen
       % distance = sqrt(sum((double(centroids)-double(des(:,j))),1).^2)
        [IDX,D] = knnsearch(double(des(:,j)'),centroids');
        [val,index]= min(D);
        hg_training_image(index) = hg_training_image(index) + 1;
        iPts = [iPts; feat(1,j) feat(2,j) colors(index,:)];
    end
    vw_histogram = [vw_histogram;hg_training_image];

%  Code for K Distinguishable Colors Plots:
%     colors1 = [iPts(:,3) iPts(:,4) iPts(:,5)]
%     figure;
%     imshow(I);
%     hold on;
%     scatter(iPts(:,1), iPts(:,2),[],colors1);
%     hold off;
%     pause(1);
%     I = getframe(gcf);
%     imwrite(I.cdata, ['pic' num2str(i) '.png']);
%     close;

%     Code to show histogram for each Image in the Training Set
%       figure;
%       histogram(hg_training_image,20);
%       title(['Class:' char(fileinfo.Label) ' ']);
%       I = getframe(gcf);
%       imwrite(I.cdata, ['histogram' num2str(i) '.png']);
%       close;
end
```

Soft-Weighting:

Ultimately not used as results were worse with soft-weighting

```
% for loop to compare descriptors to find the closest visual word label
% do this for every descriptor and increment to find kx1 histogram of
% labels
% labels = grp2idx(trainingSet.Labels);
% vw_histogram = [];
%
% for i =1:length(trainingSet.Files)
%     [I, fileinfo] = readimage(trainingSet,i);
%
%
%     [x,y,z] = size(I);
%     if(z ~= 1)
%         I = rgb2gray(I) ;
%         I = imresize(I,[200,200]);
%     else
%         I = imresize(I,[200,200]);
%     end
%
%
%
%     [feat,des] = vl_sift(single(I));
%     [xlen,ylen] = size(des);
%     hg_training_image = zeros(1,K);
%
%     for j = 1:ylen
%        % distance = sqrt(sum((double(centroids)-double(des(:,j))),1).^2)
%         [IDX,D] = knnsearch(double(des(:,j)'),centroids');
%         dsum = zeros(1,8);
%         indices = zeros(1,8);
%
%         for k = 1:8 % find the weights and the indices
%             [val,index]= min(D);
%             dsum(k) =  val;
%             indices(k) = index;
%             D(index) = 10^5; % set smallest element to very high value,
%                              %to allow to find the next smallest
%         end
%
%         for k = 1:8 % add the weights to the histogram
%             weight = 1- dsum(k)/sum(dsum);
%             hg_training_image(indices(k)) = hg_training_image(indices(k))+weight;
%         end
%
%     end
%     vw_histogram = [vw_histogram;hg_training_image];
%
% %     figure;
% %     histogram(hg_training_image,20);
% %     title(['Class:' char(fileinfo.Label) ' ']);
%
% end
%
```

BOW Classification:

```
class_label = [];
true_label = grp2idx(testSet.Labels);
output = [];
for i =1:length(testSet.Files)
    [I, fileinfo] = readimage(testSet,i);


    [x,y,z] = size(I);
    if(z ~= 1)
        I = rgb2gray(I) ;
        I = imresize(I,[200,200]);
    else
        I = imresize(I,[200,200]);
    end

    [feat_t,des_t] = vl_sift(single(I));
    [xlen,ylen] = size(des_t);
    hg_test_image = zeros(1,K);

% hard -weighting:
     for j = 1:ylen % build VW histogram
        [IDX,D] = knnsearch(double(des_t(:,j)'),centroids');
        [val,index]= min(D); %- hard-weighting


        % my own implementation of nearest-neighbors:
            %distance = sqrt(sum((double(centroids)-double(des_t(:,j))).^2,1));
            %[value1, index1] = min(distance);
            % results matched that of knnsearch

         hg_test_image(index) = hg_test_image(index) + 1; %- hard-weighting
    end


 % soft-weighting:
%     for j = 1:ylen % build VW histogram
%         [IDX,D] = knnsearch(double(des_t(:,j)'),centroids');
%         %[val,index]= min(D); - hard-weighting
%         dsum = zeros(1,8);
%         indices = zeros(1,8);
%
%         % soft -weighting:
%         for k = 1:8 % find the weights and the indices
%             [val,index]= min(D);
%             dsum(k) =  val;
%             indices(k) = index;
%             D(index) = 10^5; % set smallest element to very high value,
%                              %to allow to find the next smallest
%         end
%
%         for k = 1:8 % add the weights to the histogram
%             weight = 1- dsum(k)/sum(dsum);
%             hg_test_image(indices(k)) = hg_test_image(indices(k))+weight;
%         end
%
%
%         % my own implementation of nearest-neighbors:
%             %distance = sqrt(sum((double(centroids)-double(des_t(:,j))).^2,1));
%             %[value1, index1] = min(distance);
%             % results matched that of knnsearch
%
%        %  hg_test_image(index) = hg_test_image(index) + 1; - hard-weighting
%     end
%


    % compare histogram from training set
    [ids, euc_dist] = knnsearch(hg_test_image,vw_histogram);

    % own implementation of nearest neighbors- correct results
    %distance = sqrt(sum(((vw_histogram)-repmat(hg_test_image,length(vw_histogram),1)).^2,2));
    %[value1, index1] = min(distance)

    [val,index]= min(euc_dist);
    class_label = [class_label;labels(index)];
end
```

```
output = [class_label true_label];
accuracy = 1- length(find(class_label ~= true_label))/length(class_label)
```

```
accuracy =
    0.6533
```

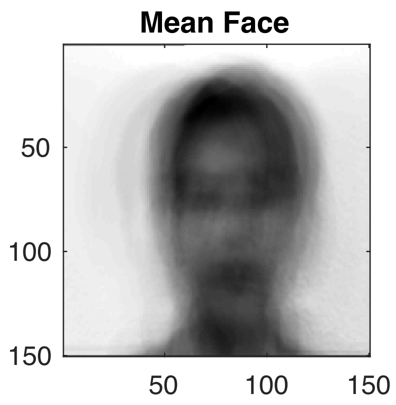
```
confusionMatrix = zeros(10,10);

for i= 1:length(true_label)
     confusionMatrix(true_label(i),class_label(i)) = confusionMatrix(true_label(i),class_label(i))+1;
end

for j = 1:length(confusionMatrix)
   cum_sum = sum(confusionMatrix(j,:));
   confusionMatrix(j,:) = confusionMatrix(j,:)./cum_sum;
end

confusionMatrix
```

```
confusionMatrix =
  Columns 1 through 7
    0.4667         0         0         0         0    0.0667    0.1333
         0    0.9333         0         0         0         0    0.0667
         0         0    0.8000         0         0    0.1333    0.0667
         0         0    0.0667    0.6000    0.0667    0.0667    0.2000
         0    0.0667    0.0667    0.0667    0.5333         0    0.0667
    0.0667         0    0.0667    0.0667    0.0667    0.5333         0
         0         0         0         0         0         0    0.8667
    0.1333         0         0    0.0667         0         0         0
    0.2667         0         0         0         0         0    0.0667
    0.1333         0         0         0         0    0.1333    0.0667
  Columns 8 through 10
    0.0667    0.0667    0.2000
         0         0         0
         0         0         0
         0         0         0
    0.1333    0.0667         0
    0.0667    0.0667    0.0667
         0    0.0667    0.0667
    0.6000    0.0667    0.1333
         0    0.5333    0.1333
         0         0    0.6667
```

Published with MATLAB® R2016b

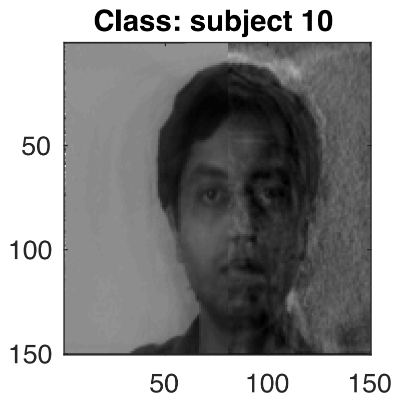
II. Eigenfaces Project

The Yale Face-Dataset was used. Recognition was performed on ***15 classes***, each with 11 photos each.



To check eigen-faces are accurate, try to reconstruct an image from training data:

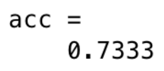
Reconstructed Face: **Original Face:**



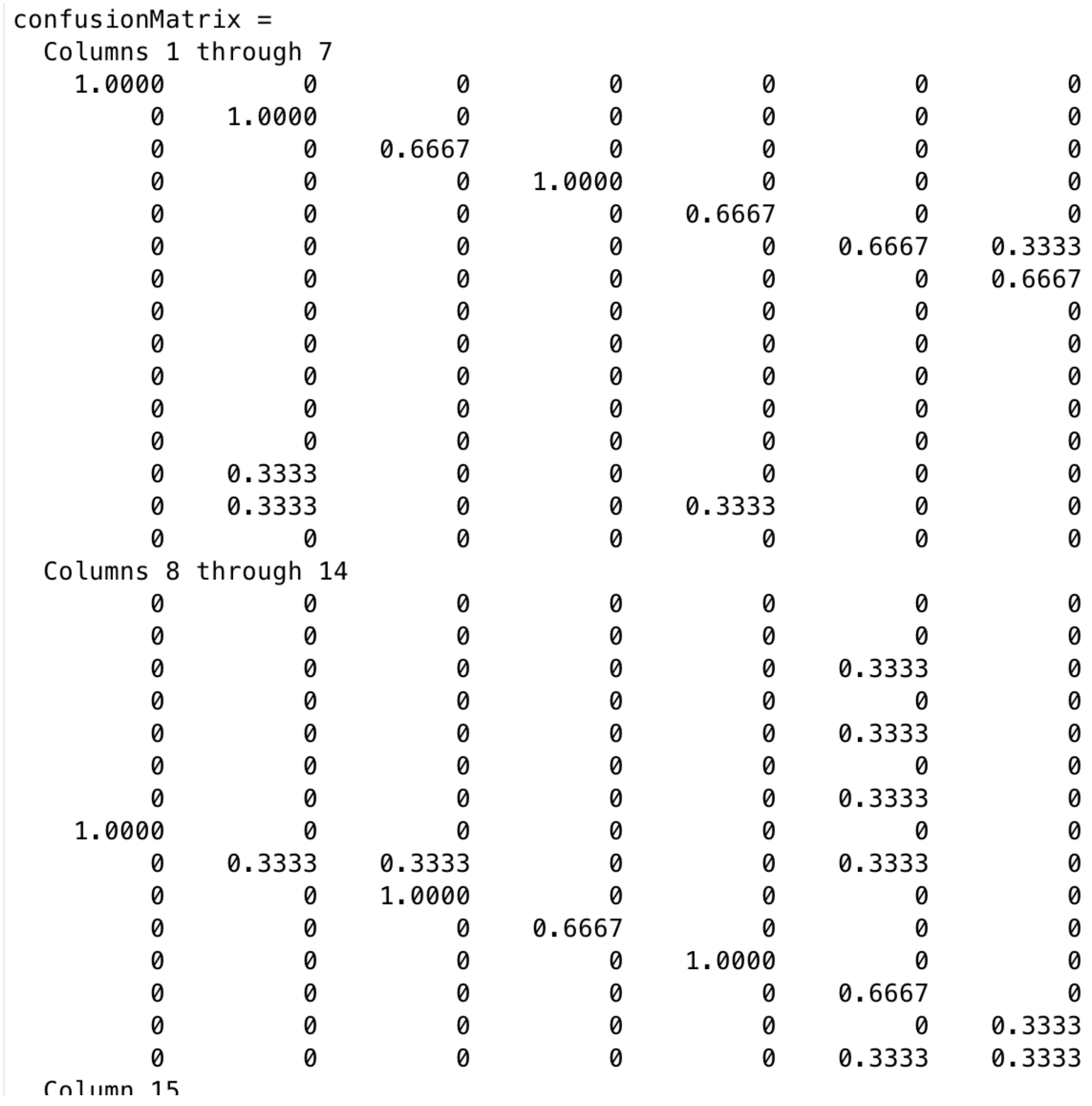
Average Recognition Rate *(over 5 (70-30) splits):* **80.0 % Accuracy**
Of the 15 classes, it was observed in the confusion matrices, that certain classes, like Subject 1 had consistently high accuracy. While others like class 14 and 15 had trouble and often were mis-categorized for individuals of similar skin-tones and ethnicities. This may have something to do with using nearest-neighbors for classification, because if even one image is slightly closer in euclidian distance it may result in choosing the wrong label. Furthermore, it was observed that choosing K that was slightly greater than 50% of the size of the basis of the Eigenfaces, in this case K was 12,000 captured enough information to give an average of 80% accuracy. It can be predicted that increasing K or choosing more Eigenfaces would have improved the recognition rate.

Split 1:

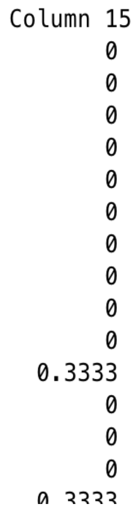
73.33% Accuracy



```
acc =
    0.7333
```



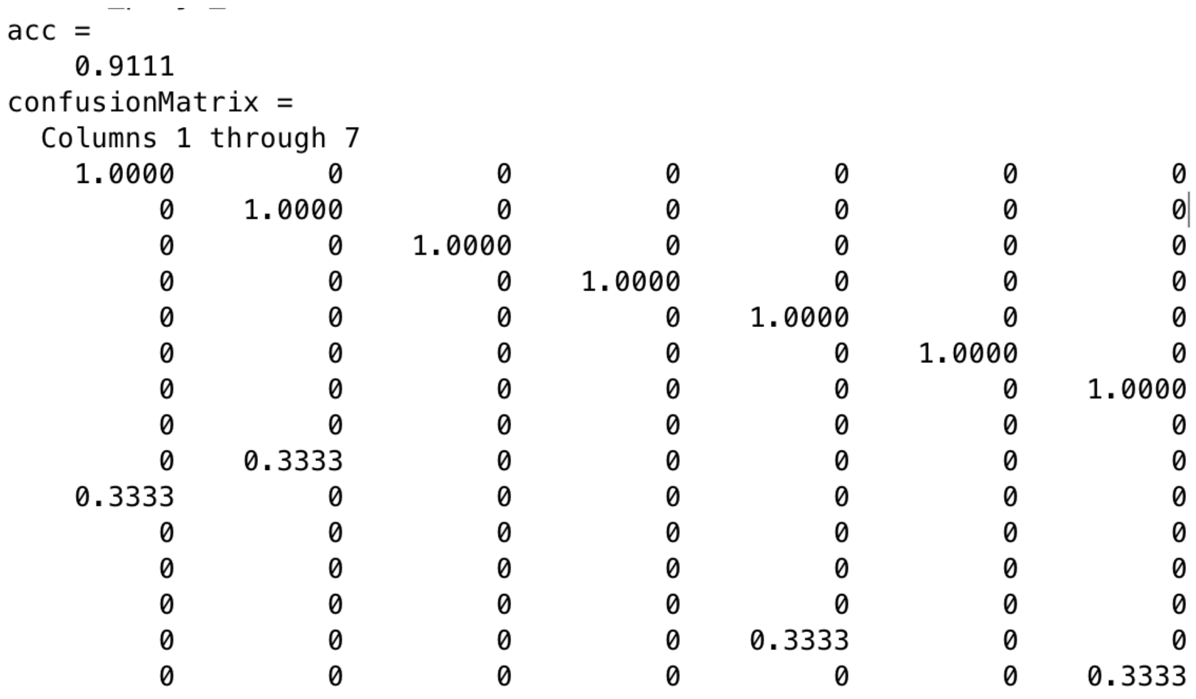
```
confusionMatrix =
  Columns 1 through 7
    1.0000         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    0.6667         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    0.6667         0         0
         0         0         0         0         0    0.6667    0.3333
         0         0         0         0         0         0    0.6667
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0    0.3333         0         0         0         0         0
         0    0.3333         0         0    0.3333         0         0
         0         0         0         0         0         0         0
  Columns 8 through 14
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0    0.3333         0
         0         0         0         0         0         0         0
         0         0         0         0         0    0.3333         0
         0         0         0         0         0         0         0
         0         0         0         0         0    0.3333         0
    1.0000         0         0         0         0         0         0
         0    0.3333    0.3333         0         0    0.3333         0
         0         0    1.0000         0         0         0         0
         0         0         0    0.6667         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    0.6667         0
         0         0         0         0         0         0    0.3333
         0         0         0         0         0    0.3333    0.3333
  Column 15
```



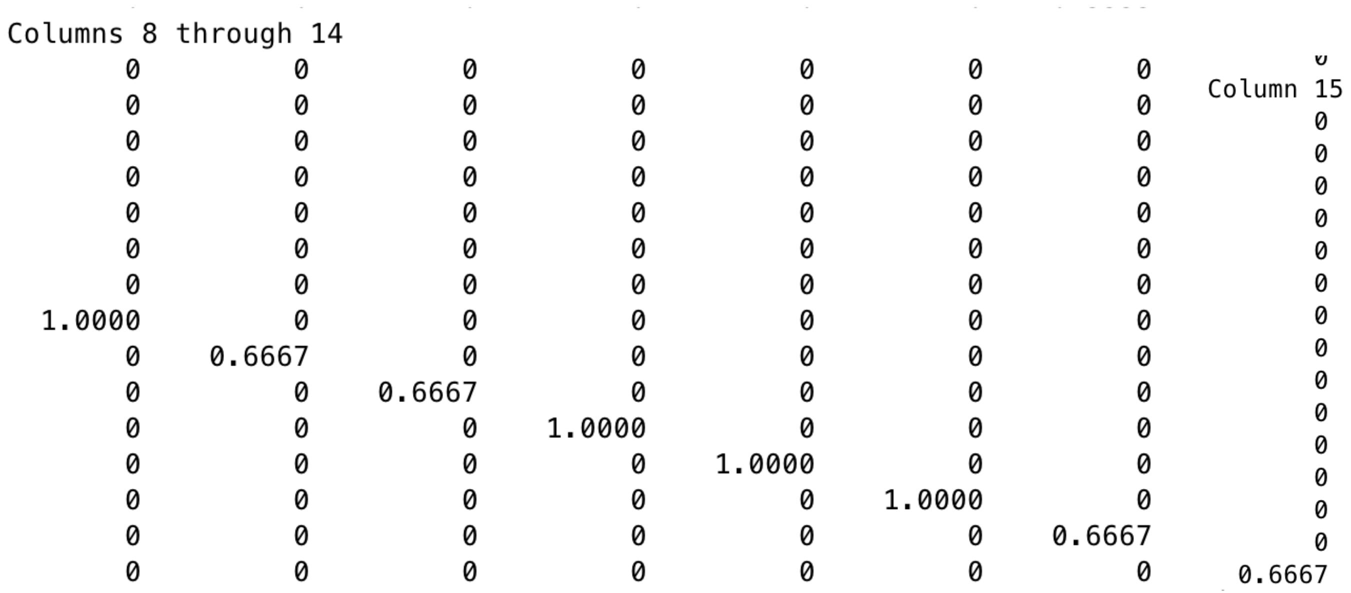
```
Column 15
        0
        0
        0
        0
        0
        0
        0
        0
        0
        0
   0.3333
        0
        0
        0
   0.3333
```

Split 2:

91.11 % Accuracy:



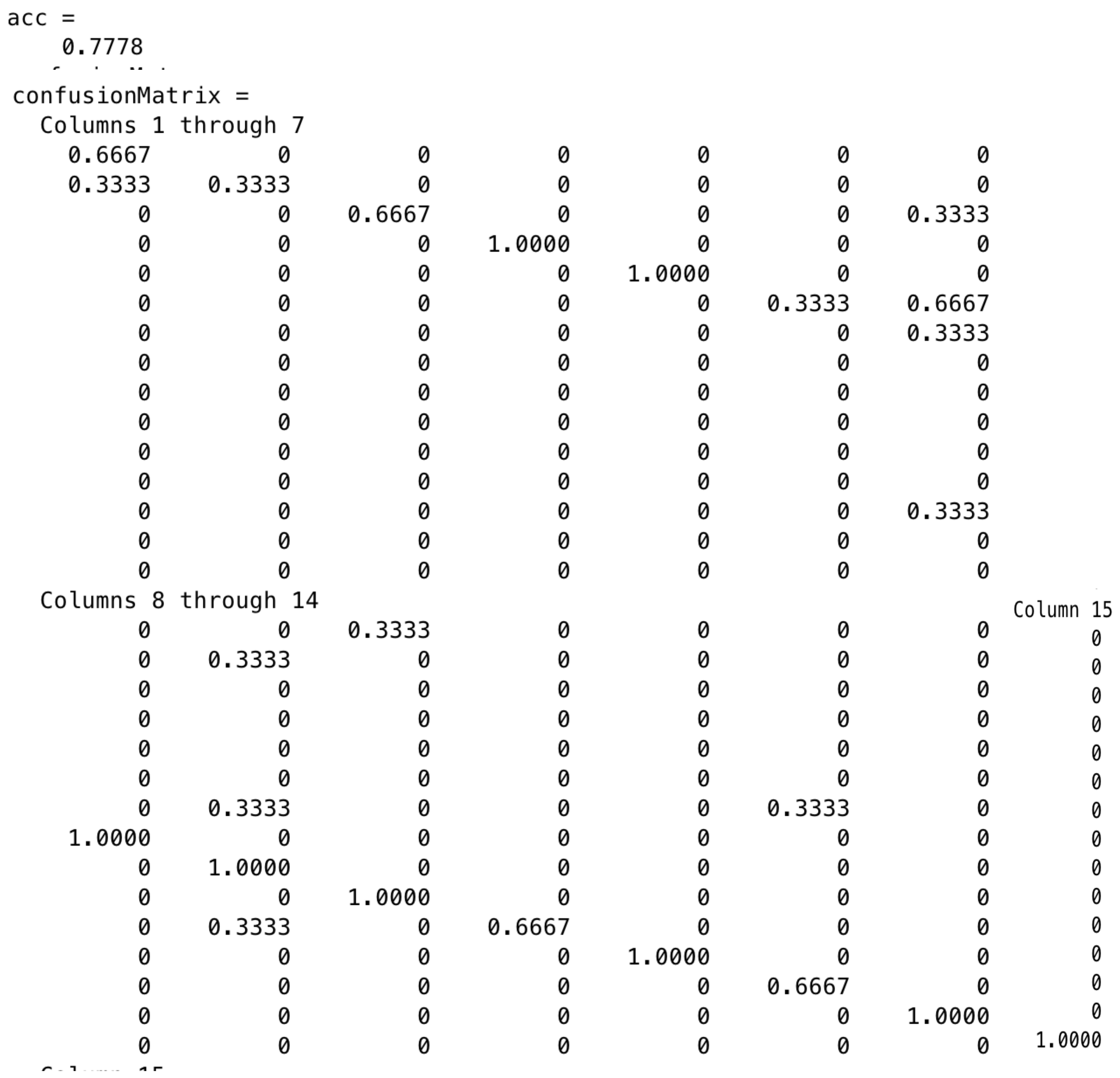
```
acc =
    0.9111
confusionMatrix =
  Columns 1 through 7
    1.0000         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0
         0    0.3333         0         0         0         0         0
    0.3333         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0    0.3333         0         0
         0         0         0         0         0         0    0.3333
```



```
 Columns 8 through 14                                                     Column 15
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
    1.0000         0         0         0         0         0         0         0
         0    0.6667         0         0         0         0         0         0
         0         0    0.6667         0         0         0         0         0
         0         0         0    1.0000         0         0         0         0
         0         0         0         0    1.0000         0         0         0
         0         0         0         0         0    1.0000         0         0
         0         0         0         0         0         0    0.6667         0
         0         0         0         0         0         0         0    0.6667
```

Split 3:

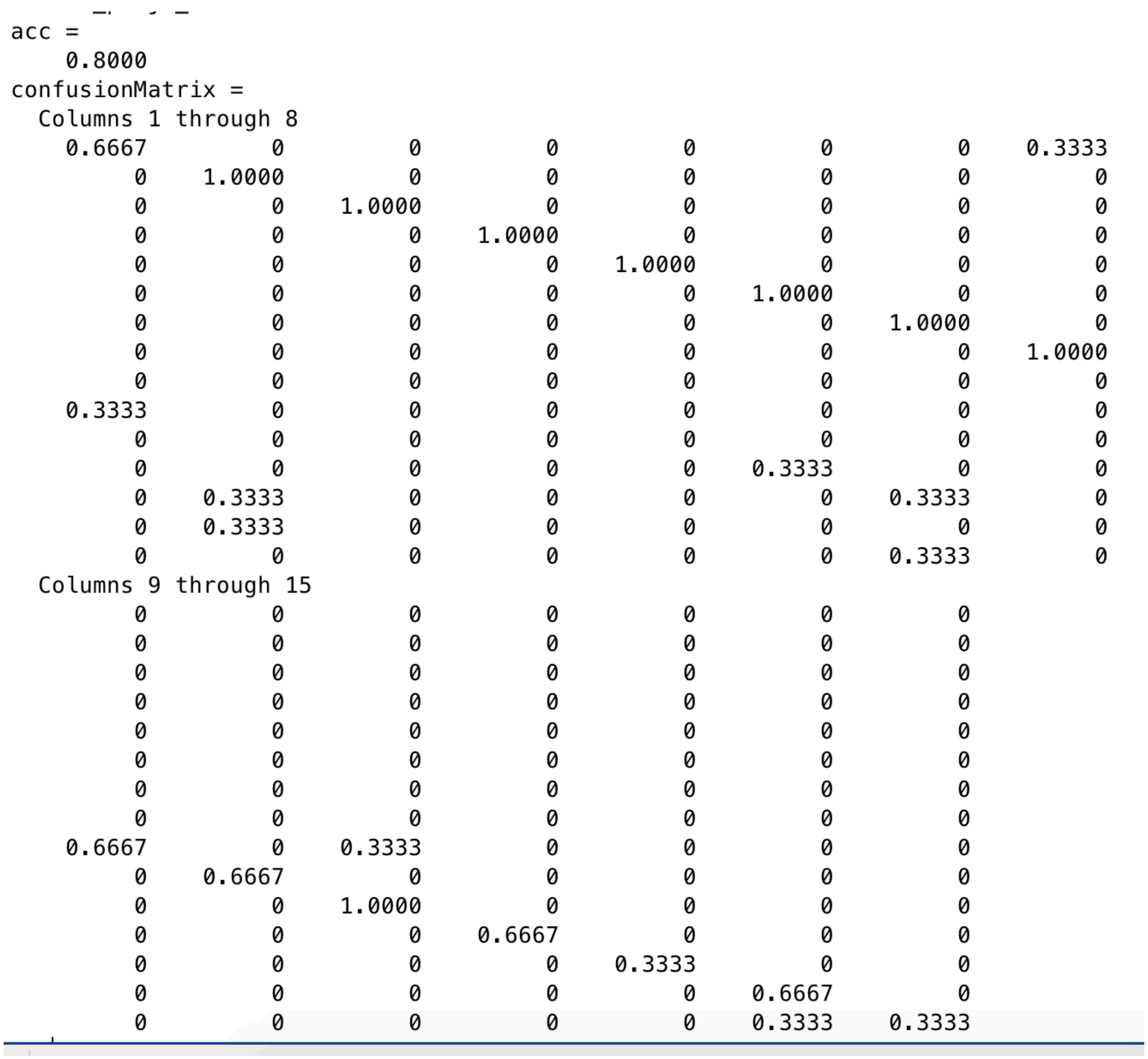
77.78% Accuracy



```
acc =
    0.7778

confusionMatrix =
  Columns 1 through 7
    0.6667         0         0         0         0         0         0
    0.3333    0.3333         0         0         0         0         0
         0         0    0.6667         0         0         0    0.3333
         0         0         0    1.0000         0         0         0
         0         0         0         0    1.0000         0         0
         0         0         0         0         0    0.3333    0.6667
         0         0         0         0         0         0    0.3333
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0    0.3333
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
  Columns 8 through 14                                                  Column 15
         0         0    0.3333         0         0         0         0         0
         0    0.3333         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0    0.3333         0         0         0    0.3333         0         0
    1.0000         0         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0         0
         0         0    1.0000         0         0         0         0         0
         0    0.3333         0    0.6667         0         0         0         0
         0         0         0         0    1.0000         0         0         0
         0         0         0         0         0    0.6667         0         0
         0         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0         0    1.0000
```

Split 4:

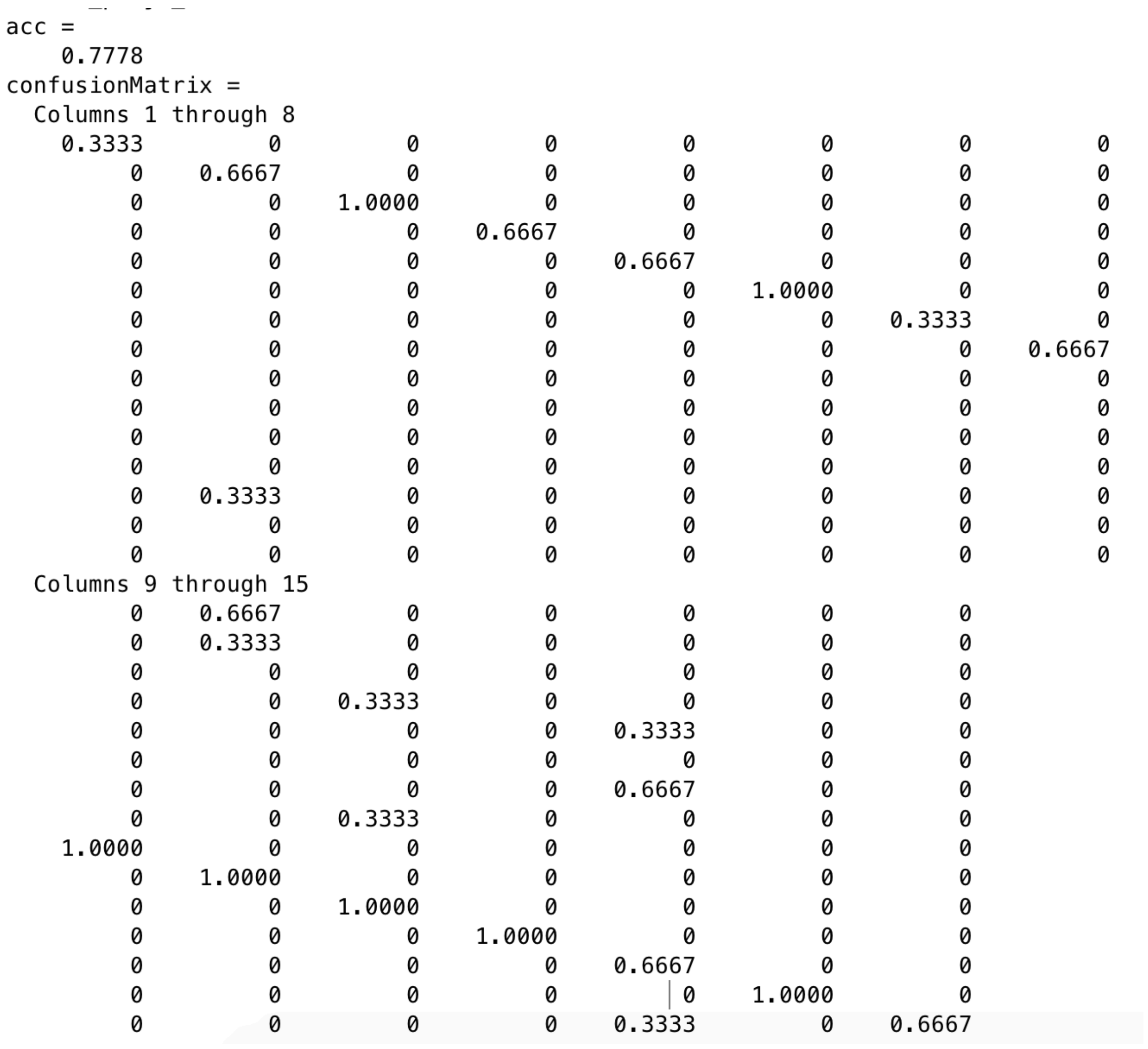
80 % - Accuracy



```
acc =
    0.8000
confusionMatrix =
  Columns 1 through 8
    0.6667         0         0         0         0         0         0    0.3333
         0    1.0000         0         0         0         0         0         0
         0         0    1.0000         0         0         0         0         0
         0         0         0    1.0000         0         0         0         0
         0         0         0         0    1.0000         0         0         0
         0         0         0         0         0    1.0000         0         0
         0         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0         0    1.0000
         0         0         0         0         0         0         0         0
    0.3333         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0    0.3333         0         0
         0    0.3333         0         0         0         0    0.3333         0
         0    0.3333         0         0         0         0         0         0
         0         0         0         0         0         0    0.3333         0
  Columns 9 through 15
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
    0.6667         0    0.3333         0         0         0         0
         0    0.6667         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    0.6667         0         0         0
         0         0         0         0    0.3333         0         0
         0         0         0         0         0    0.6667         0
         0         0         0         0         0    0.3333    0.3333
```

Split 5:

77.78 % - Accuracy



```
acc =
    0.7778
confusionMatrix =
  Columns 1 through 8
    0.3333         0         0         0         0         0         0         0
         0    0.6667         0         0         0         0         0         0
         0         0    1.0000         0         0         0         0         0
         0         0         0    0.6667         0         0         0         0
         0         0         0         0    0.6667         0         0         0
         0         0         0         0         0    1.0000         0         0
         0         0         0         0         0         0    0.3333         0
         0         0         0         0         0         0         0    0.6667
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0    0.3333         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
  Columns 9 through 15
         0    0.6667         0         0         0         0         0
         0    0.3333         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0    0.3333         0         0         0         0
         0         0         0         0    0.3333         0         0
         0         0         0         0         0         0         0
         0         0         0         0    0.6667         0         0
         0         0    0.3333         0         0         0         0
    1.0000         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    0.6667         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0    0.3333         0    0.6667
```

Contents

- RCV eigenface recognition
- Mean Face
- Check if training data can be reconstructed:
- Face Recognition (based on class label)
- Project Requirements:

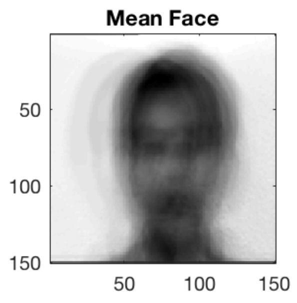
RCV eigenface recognition

Niral Shah 12/03/16

```
setDir  = fullfile(toolboxdir('vision'),'visiondata','imageSets');
imds = imageDatastore('/Users/niralshah/Desktop/yalefaces','IncludeSubfolders',true,'LabelSource',...
    'foldernames');
[trainingSet,testSet] = splitEachLabel(imds,0.7,'randomize');
training_labels = grp2idx(trainingSet.Labels);
test_labels = grp2idx(testSet.Labels);

A = [];
average = zeros(150,150);
for i =1:length(trainingSet.Files)
    [I, fileinfo] = readimage(trainingSet,i);


       [x,y,z] = size(I);
    if(z ~= 1)
        I = rgb2gray(I) ;
        I = imresize(I,[150,150]);
    else
        I = imresize(I,[150,150]);
    end
    average = average + double(I);
    T = reshape(I,[],1);
     A = [A T];
%      figure;
%      imshow(I);
%      pause(2);
%      close;
end
```

Mean Face

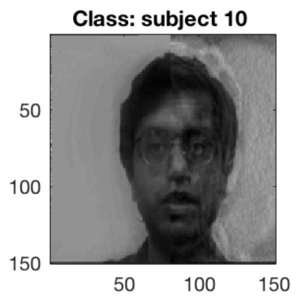
```
average = average/i;

hfig= figure(2);
set(hfig,'Position',[0 0 150 150])
imagesc(average);
colormap gray;
title('Mean Face');
```



```
a = reshape(average,[],1);
A_meansub = double(A)-a;

[U,S,V] = svd(A_meansub);
```

Check if training data can be reconstructed:

```
img = A(:,10); % image to reconstruct
[I, fileinfo] = readimage(trainingSet,10);

k = 12000;
Uk = U(:,1:k);
csum = zeros(length(Uk),1);

for j = 1:k
    csum = csum + (Uk(:,j)'*double(img)*Uk(:,j));
end
csum= csum+a;
new_img = reshape(csum,150,150);

hfig=figure(3);
set(hfig,'Position',[0 0 150 150])
imagesc(new_img);
colormap gray;
title(['Class: ' char(fileinfo.Label) ' ']);
```



Face Recognition (based on class label)

```
output = [];
class_label = [];
labels = grp2idx(trainingSet.Labels);
true_label = grp2idx(testSet.Labels);

for i =1:length(testSet.Files)
    [I, fileinfo] = readimage(testSet,i);
```

```
        [x,y,z] = size(I);
    if(z ~= 1)
        I = rgb2gray(I) ;
        I = imresize(I,[150,150]);
    else
        I = imresize(I,[150,150]);
    end

    T = double(reshape(I,[],1));
    T = T-a;
    csum = zeros(length(Uk),1);
    for l = 1:k
        csum = csum + Uk(:,l)'*double(T)*Uk(:,l);
    end

    [idx,D] = knnsearch(csum',A_meansub');
    [value,index] = min(D);
    [I2, fileinfo2] = readimage(trainingSet,index);
    output = [output;fileinfo.Label fileinfo2.Label];
    class_label = [class_label;labels(index)];
    new_img = reshape(csum,150,150);

%     hfig=figure(3);
%     set(hfig,'Position',[10 10 150 150])
%     imagesc(new_img);
%     colormap gray;
%     title(['Class:' char(fileinfo2.Label) ' ']);
%     pause(2.5);
%     close;
end
```

```
numberCorrect = length(testSet.Files)-length(find(output(:,1) ~= output(:,2)));
doubleCheck = length(testSet.Files)-length(find(class_label(:) ~= true_label(:)));

acc = doubleCheck/length(testSet.Files)
accuracy = numberCorrect/length(testSet.Files);
confusionMatrix = zeros(15);

for i= 1:length(true_label)
     confusionMatrix(true_label(i),class_label(i)) = confusionMatrix(true_label(i),class_label(i))+1;
end


for j = 1:length(confusionMatrix)
   cum_sum = sum(confusionMatrix(j,:));
   confusionMatrix(j,:) = confusionMatrix(j,:)./cum_sum;
end

confusionMatrix

% accuracy with K =10000 is 86.67 %
% accuracy with K =1000 is 86.67%
```

```
acc =
    0.7778
confusionMatrix =
  Columns 1 through 7
    1.0000         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    0.6667         0         0
         0         0         0         0         0    1.0000         0
         0         0         0         0         0         0    0.3333
         0         0         0    0.3333         0         0         0
         0         0         0         0         0         0         0
    0.3333         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0    0.3333         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
  Columns 8 through 14
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0    0.3333         0
         0         0         0         0         0         0         0
         0         0         0         0         0    0.6667         0
    0.6667         0         0         0         0         0         0
         0    1.0000         0         0         0         0         0
         0         0    0.6667         0         0         0         0
         0         0         0    1.0000         0         0         0
         0         0         0         0    0.6667         0    0.3333
         0         0         0         0         0    0.6667         0
         0         0         0         0         0    0.3333    0.6667
         0         0         0         0         0    0.3333    0.3333
  Column 15
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
         0
    0.3333
```

Project Requirements:

```
%Use 5 splits of the data (50-50 split or up to 30-70).
% Provide a confusion matrix for each method in the description of results. Include
% the input and output class names in the confusion matrix. Use ConfusionMatStats
% to provide confusion matrix statistics. Write a paragraph discussing these results.
```

Published with MATLAB® R2016b

VGG Practical – Questions
(Sting-ray version)

Part 1.1.1: Convolution by a single filter

1. The size of the output image is (H-H'+1) x (W-W' +1).
2. Based on the results it seems the filter is excited most by dotted areas, light areas that are surround by darker regions.

Part 1.1.2: Convolution by a filter bank

1. The number of feature channels K in this example is three. This is because there are three separate filters that compose this filter.

2. After running the code, the channel responses do make sense. The first feature channel emphasizes the dots on the sting-ray. The second is horizontal image derivatives, vertical features are emphasized and horizontal features are filtered out. Finally, the third, is vertical image derivatives. This is the opposite of the second, horizontal features are enhanced while vertical features are filtered out.
3. If the input tensor has C feature channels, the third dimension of **w** should also be C.
4. Concatenating along the fourth dimension allows each of the filters to be applied to each feature channel of the tensor.

Part 1.1.3: Convolving a batch of images

1. Looking at the image, it is clear that the first filter will emphasize dotted features, the second filter will emphasize vertical features, and the third filter highlight horizontal features.

Part 1.2: Non-linear activation (ReLU)

1. If all the layers all linear, then eventually the output image would be represented by pixels that are all negative. This makes it harder to visualize the picture properly, as grayscale is between 0 and 1.

Part 1.2.1: Laplacian and ReLU

1. This filter focuses on the image structures where there are sharp transitions between light and dark areas.
2. The Laplacian is negated because we want to invert the colors that are passed from the original filter (make dark pixels, light).

Part 1.2.2: Effect of adding a bias

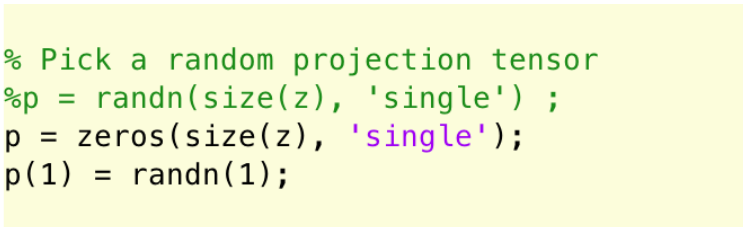
1. It only focuses on areas that are dots (particularly the sting-ray's back).

Part 2: Back Propagation

1. The output derivatives have the same size as the parameters in the network, since each point in the derivative maps to the change in the output based on those input parameters. Each parameter results in its own derivative.

Part 2.1: Backward mode verification

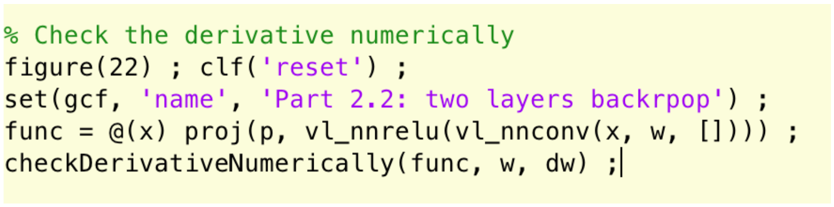
1. Lines 16-19 inside the nested for loop use this expression.
2. The output of the vl_nconv() is project onto p in order to produce a tensor that is the result of a filtered tensor applied to a projection tensor. This make it so that the derivative calculated will be the same as calculated in the backwards mode.
3. To calculate the derivative of the first element of y, the variable p needs to be changed.



```
% Pick a random projection tensor
%p = randn(size(z), 'single') ;
p = zeros(size(z), 'single');
p(1) = randn(1);
```

(This code is changed in exercise2.m, line 38)

4. Code to calculate the derivative with respect to w instead of x:

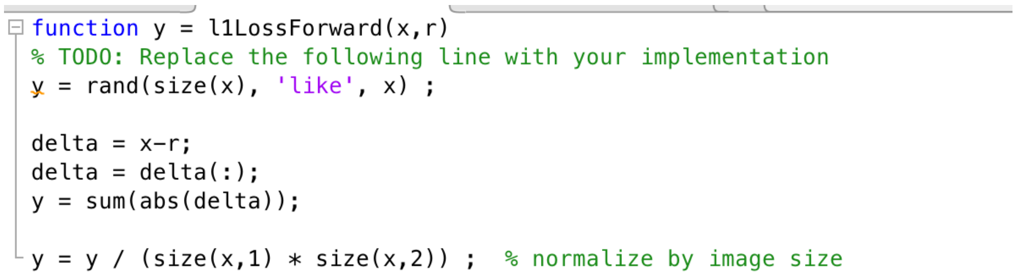


```
% Check the derivative numerically
figure(22) ; clf('reset') ;
set(gcf, 'name', 'Part 2.2: two layers backrpop') ;
func = @(x) proj(p, vl_nnrelu(vl_nnconv(x, w, []))) ;
checkDerivativeNumerically(func, w, dw) ;
```

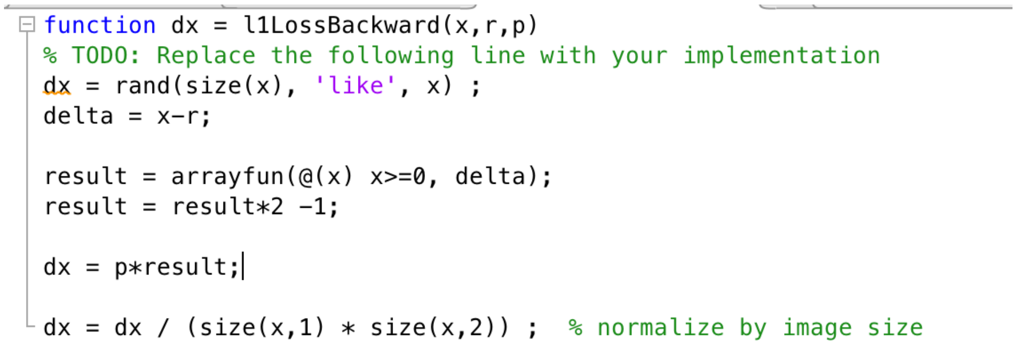
Part 2.2: Backwards Propagation

1. The vl_nnconv() operator receives dy because dy is basically p after going through the ReLU.
2. (optional) skipped.

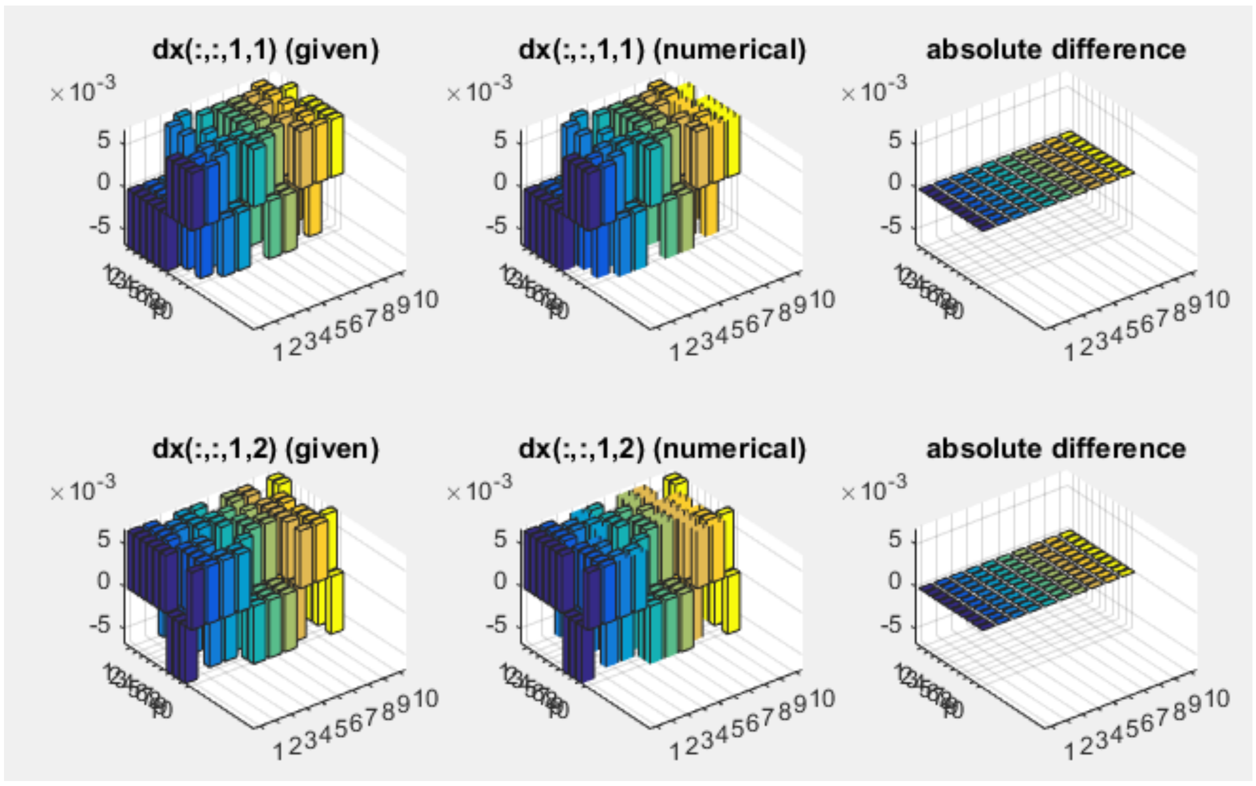
Part 2.3: Design and verify your own layer



```
function y = l1LossForward(x,r)
% TODO: Replace the following line with your implementation
y = rand(size(x), 'like', x) ;

delta = x-r;
delta = delta(:);
y = sum(abs(delta));

y = y / (size(x,1) * size(x,2)) ;  % normalize by image size
```



```
function dx = l1LossBackward(x,r,p)
% TODO: Replace the following line with your implementation
dx = rand(size(x), 'like', x) ;
delta = x-r;

result = arrayfun(@(x) x>=0, delta);
result = result*2 -1;

dx = p*result;

dx = dx / (size(x,1) * size(x,2)) ;  % normalize by image size
```



Part 3: Learning a CNN for text-deblurring

1. The number of training images is 2550, and the number of validation is 902. The resolution is 64x64.
2. The interval [-1,0] is chosen because we are only focused on the features of the image. The [-1:0] scale is is grayscale with -1 being black and 0 is white. Convolution minimizes the blurring between boundaries.