Contents

```
%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