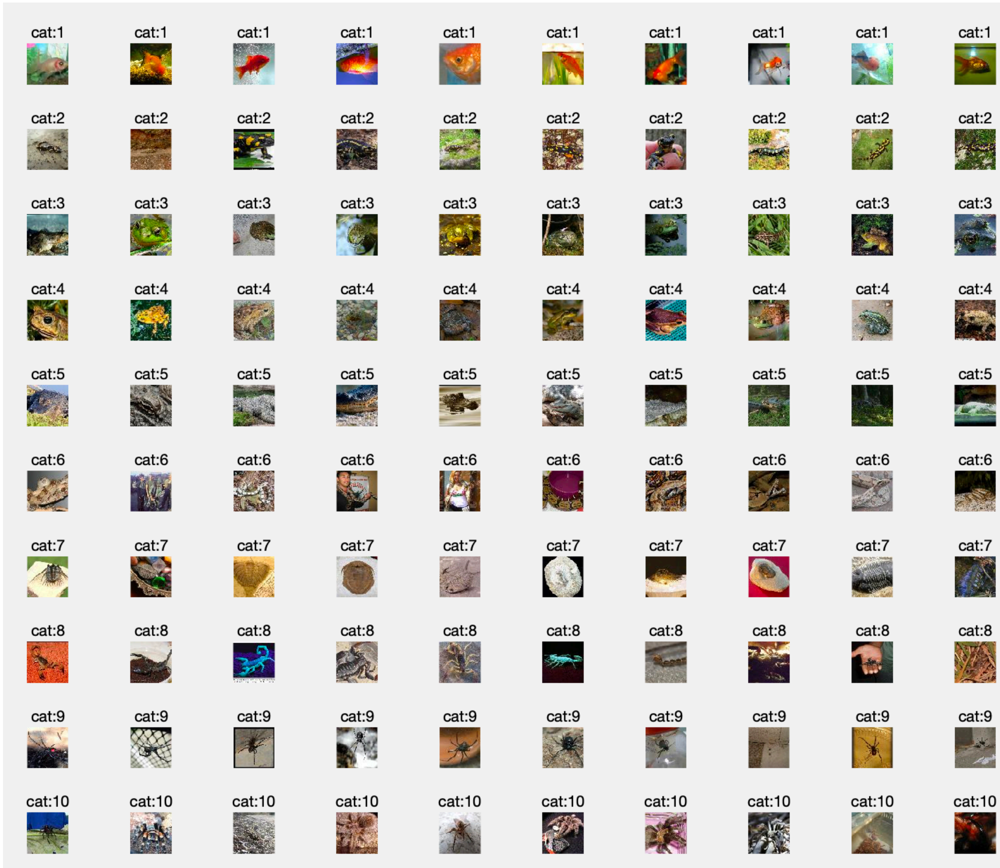
Homework-2: Tiny ImageNet Image Retrieval with Handcrafted features. In this homework we will work with a larger data set, the Tiny ImageNet instead: https://tiny-imagenet.herokuapp.com/ This is a data set with 64x64 sized color images from 200 categories. For HW-2 we only need to deal with 100 images from the first 10 categories, and the following is how to extract the data into ims{} and ids[]:

- First we need to load the image by following code:

```
% find folder with some info into a struct by path
cat_list = dir('tiny-imagenet-200/train');
n_cat = 10; n_img = 10;
n = 0;
ids=zeros(10,9);
% load images one category by one gategory
for k=1:n_cat
    ids(k,:) = cat_list(k+3).name; %when k=1 k+2=3 cat_list will return to a invalid folder ;we can obtain t
    fprintf('\n cat %s', ids(k,:));  %print folder name from ids
    flist = dir(sprintf('tiny-imagenet-200/train/%s/images/*.JPEG',ids(k,:)));
    for j=1:n_img
        n=n+1; %after k loop, n=100
        ims{n} = imread(sprintf('tiny-imagenet-200/train/%s/images/%s',ids(k,:), flist(j).name));
        fprintf('.');
    end
end
% associated labels
ids = kron([1:n_cat], ones(1,n_img))';  %ids[] is the labels of category.
```

we use dir() to find the data and use imread to load the data. Then we can have a clear look about our image.



[1] Use pooled (2x2) color histogram to represent images provide the two following functions for feature extraction and distance computing (codebook can be just fixed 8x4x2 HSV uniform quantization) [20pts]

First step, we need to use Kmeans methods with 64 entries, and then we can get a matrix for centroids as 64*3 (code book).To implement a clustering method directly.

```
imshsv=[]
for k=1:(n_cat*n_img)
    c=cell2mat(ims(k))  %tranfer from cell to matrix
    cc=reshape(c,[64,64,3])
    imshsv(k,:,:,:)=rgb2hsv(cc)
end
imshsv1=reshape(imshsv,[(n_cat*n_img)*64*64,3])
%we put all images from hsv format into kmeans to get the centroids
[~, centers] = kmeans(imshsv1,64)
%calculate histogram for each image
```

Then I create two functions , one is to separate one image into four pools and calculate the histogram with centroids, finally merge them together. The other is calculate two images distance by histogram.

```
getPooledHSVHistogram(im, codebook, pooling):
```

```
function [h] = getPooledHSVHistogram(im, codebook, pooling)
   im=rgb2hsv(im)
   p1=pooling(1) %rows
   p2=pooling(2) %columns
   size1=64/p1
   size2=64/p2
   %seprate into four different areas
   %then reshape them to calculate distance
   part1=im(1:size1,1:size2,1:3)
   part2=im((size1+1:64),1:size2,1:3)
   part3=im(1:size1,(size2+1):64,1:3)
   part4=im((size1+1):64,(size2+1):64,1:3)
   part1=reshape(part1,[32*32,3])
   part2=reshape(part2,[32*32,3])
   part3=reshape(part3,[32*32,3])
   part4=reshape(part4,[32*32,3])
   %we obtain histogram by distance to each centriods
   h1=pdist2(part1,codebook)
   h2=pdist2(part2,codebook)
   h3=pdist2(part3,codebook)
   h4=pdist2(part4,codebook)
```

```
  % remove the inaccurate data
  [~,pixel_bin1] = min(h1');
  [~,pixel_bin2] = min(h2');
  [~,pixel_bin3] = min(h3');
  [~,pixel_bin4] = min(h4');
  for k=1:64
       h1(k) = length(find(pixel_bin1==k));
       h2(k) = length(find(pixel_bin2==k));
       h3(k) = length(find(pixel_bin3==k));
       h4(k) = length(find(pixel_bin4==k));
  end
  h1=h1/sum(h1)
  h2=h2/sum(h2)
  h3=h3/sum(h3)
  h4=h4/sum(h4)
  h=[h1,h2,h3,h4];
nd
```

getPooledHSVDistance(hist1, hist2)

```
function [dist_matrix]=getPooledHSVDistance(hist1, hist2)
d1=mean(min(pdist2(hist1, hist2)));  %we can get a spercific distance number rather than a matrix
d2=mean(min(pdist2(hist2, hist1))); % we select each columns min value and then get average value
dist_matrix=min(d1,d2);
end
```

[2] Compute HoG feature for image, use the average, as well as 2x2 pooled average as texture feature. Use block size of 8 pixel and 2x2 cell structure. {Hint: use rgb2gray(), vl_hog()} [20pts]

The previous steps for hog() are as same as histogram, we also need to separate image into four pooling. According to hog() requirement, the input must be single gray, we need to transform image from RGB to gray ,and single() them.

```
[hogTotal]=getImHog(im,pooling)
```

```
part4=rgb2gray(part4)
sin1=im2single(part1)
sin2=im2single(part2)
sin3=im2single(part3)
sin4=im2single(part4)
hog1=vl_hog(sin1, cellSize, 'verbose', 'variant', 'dalaltriggs');
hog2= vl_hog(sin2, cellSize, 'verbose', 'Variant','DalalTriggs')
hog3= vl_hog(sin3, cellSize, 'verbose', 'Variant','DalalTriggs')
hog4= vl_hog(sin4, cellSize, 'verbose', 'Variant','DalalTriggs')
hogTotal=[hog1,hog2,hog3,hog4]
```

To simplify the question , I merge the hog1-hog4. And then make a distance calculation.

```
        sizeTotal=size(h1)
        size1=sizeTotal(1)
        size2=sizeTotal(2)
        size3=sizeTotal(3)
        im1=reshape (h1,[size1*size2,size3])
        im2=reshape (h2,[size1*size2,size3])
        d1=mean(min(pdist2(im1, im2)));
        d2=mean(min(pdist2(im2, im1)));
        hogDis=min(d1,d2);
end
```

I call these function I created above to test and the code are as below

```
%we put all images from hsv format into kmeans to get the centroids
[~, centers] = kmeans(imshsv1,64)
%calculate histogram for each image
hh=getPooledHSVHistogram(cc, centers, [2,2])
hog1=getImHog(cc,[2,2])
c1=cell2mat(ims(98))  %tranfer from cell to matrix
cc1=reshape(c1,[64,64,3])
hh1=getPooledHSVHistogram(cc1, centers, [2,2])
hog2=getImHog(cc1,[2,2])
hogdist=getHogDist(hog1,hog2)
```

[3] For each image compute its dense SIFT and use Fisher Vector to aggregate. Training SIFT data set for PCA and GMM computation is provided as below, with the training SIFT data at: https://umkc.box.com/s/3shyqe1mkvb6n19arnrusdqstwqms3rs [20pts]

According to professor’s code, we can get a dataset after PCA

hw2-A-gmm-km.mat

hw2-A-gmm-km.mat (MAT 文...

名称	值
A	128x128 double
km	2x3 cell
gmm	2x3 struct

What is sift?

In simple terms, the sift is to extract some interest points(key points) from one image. These point are irrelevant with image’ s scale. We can use these points to detect and recognize objects.

“the image should be pre-smoothed at the desired scale level”

Before we sift , we have to reduce noise ,smoothing the data by convolution.

```
load hw2-A-gmm-km.mat;
d_all = [];
for i = 1:100
     im = ims(i);
     im_mat=cell2mat(ims(i))
     im_gray_single = single(rgb2gray(im_mat));

     %VL_DSIFT() does NOT compute a Gaussian scale space of the image
     %we need to smooth the data before we make it.
     h0 = fspecial('gaussian', 3, 1.5);
     % convolution
     im0 = imfilter(im_gray_single, h0);
     % sift
     [~, sift2] = vl_dsift(im0, 'step', 2, 'size', 3);%sift2 is a 128 x NUMKEYP
     d = getSiftFv(sift2, A, gmm);% obtain by hw2-A-gmm-km
     d_all = [d_all, d];
end
distSIFT1=getSiftFvDis(d_all(:,99),d_all(:,100))
```

we can get a sift2 matrix with 128 x NUMKEYPOINTS . One column represents one descriptor.Then I use Fisher Vector to aggregate as a fuction function [fv]=getSiftFv(sift, A, gmm)

```
function [fv]=getSiftFv(sift, A, gmm)
kd = 16;  nc = 32;
% fisher vec
dsift_fv = zeros(1, kd*nc);
% 2*3=6 gmm matrix on this file and pick one to calculate
fv = vl_fisher(A(1:kd, :) * double(sift), gmm(1, 1).m, gmm(1, 1).cov,  gmm(1, 1).
dsift_fv(1, :) = fv(1:kd * nc);
fprintf('\n %d sift ', 1)
end
```

Finally, we calculate the distance with a function

```
getSiftFvDis(sift1,sift2)
```

```
function [distSIFT] = getSiftFvDis(sift1,sift2)
    dis1=mean(min(pdist2(sift1,sift2)));
    dis2=mean(min(pdist2(sift2,sift1)));
    distSIFT=min(dis1,dis2)
end
```

[4] For the 2x2 pooled HSV feature, HoG feature, Fisher Vector aggregated dense SIFT feature, please compute the n x n distance map between all image pairs, and plot their TPR-FPR separately (hint: use vl_roc). 20pts. Examples of distance map:

We only use for loop to calculate and then store them into a matrix.

In my opinion, to reduce the time complexity , we can only store the data in a half of matrix. Because distance(3,7)=distance(7,3). And then it is easy to use function defined before to calculate.

```
% hsv distance
hsvDis=zeros(100,100);
for i=1:100
    im1=ims{i}
    h1=getPooledHSVHistogram(im1,centers,[2,2])
    hsvDisByRow=zeros(1,100);
    for j=i:100
        im2=ims{j}
        h2=getPooledHSVHistogram(im2,centers,[2,2])
        dis=getPooledHSVDistance(h1,h2)
        hsvDisByRow(:,j)=dis
    end
    hsvDis(i,:)=hsvDisByRow
end
%hog distance
HogDis=zeros(100,100)
for i=1:100
    im1=ims{i}
    h1=getImHog(im1,[2,2])
    hogDisByRow=[]
    for j=i:100
        im2=ims{j}
        h2=getImHog(im2,[2,2])
        dis=getHogDist(h1,h2)
        hogDisByRow(1,j)=dis
    end
    HogDis(i,:)=hogDisByRow
end

%sift distance


for i=1:100
    for j=1:100
        SiftDis(i,j)=SiftDis(j,i);
    end

end
%ground dist
groundDis=pdist2(ids,ids)
```

SiftDis

100x100 double

	1	2	3	4	5	6	7	8
1	0	0.0020	0.0013	0.0014	0.0013	0.0012	0.0014	0.0012
2	0	0	0.0016	0.0022	0.0015	0.0015	0.0015	0.0014
3	0	0	0	0.0013	0.0014	0.0014	0.0013	0.0014
4	0	0	0	0	0.0015	0.0015	0.0013	0.0012
5	0	0	0	0	0	0.0014	0.0015	0.0012
6	0	0	0	0	0	0	0.0014	0.0012
7	0	0	0	0	0	0	0	0.0017

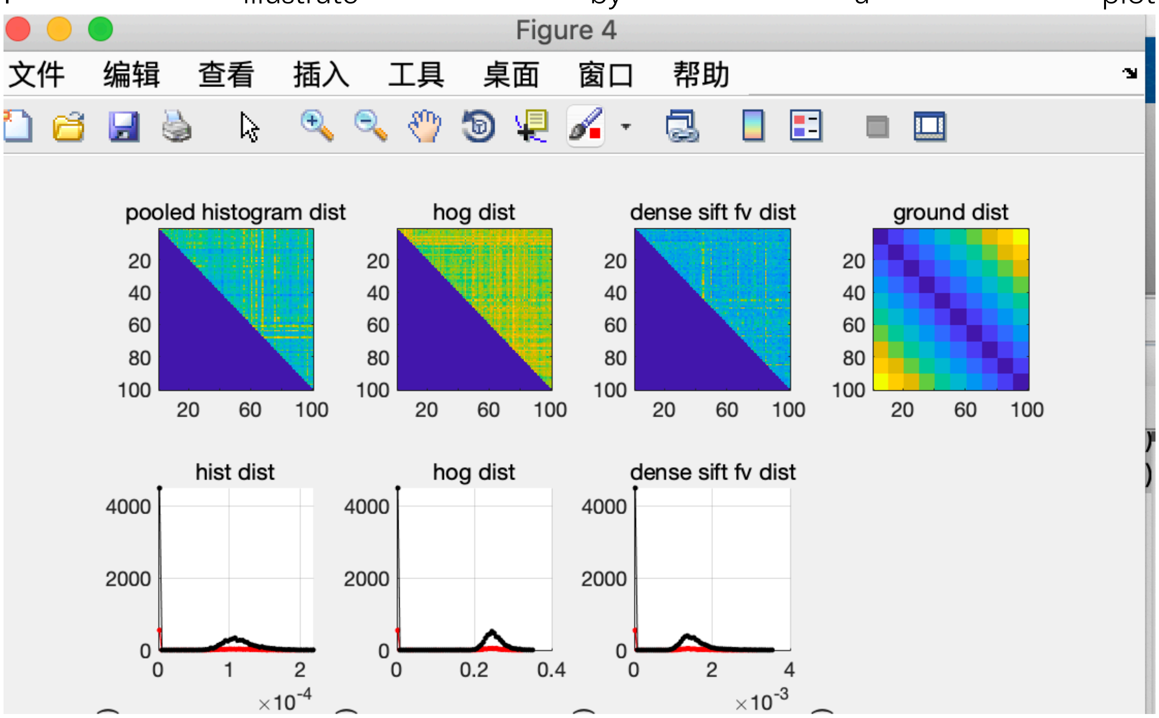
SiftDis groundDis

100x100 double

24	25	26	27	28	29	30	31	32
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
2	2	2	2	2	2	2	3	3
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2
1	1	1	1	1	1	1	2	2

```
%plot distance beween
figure(4);
subplot(3,4,1); imagesc(hsvDis); title('pooled histogram dist');
subplot(3,4,2); imagesc(HogDis); title('hog dist');
subplot(3,4,3); imagesc(SiftDis); title('dense sift fv dist');
subplot(3,4,4); imagesc(groundDis); title('ground dist');
```

I illustrate by a plot

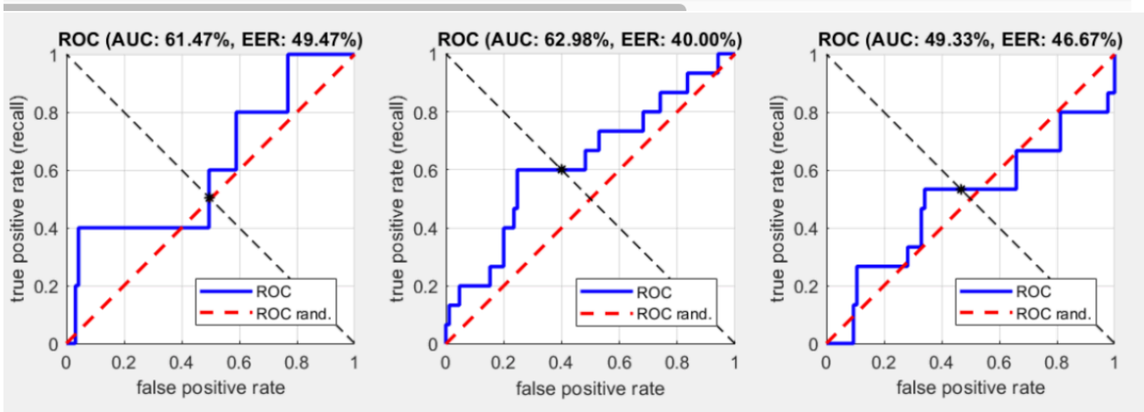


True postive rate and false postive rate we can obtain by vl_roc fuction.

[TPR,TNR] = VL_ROC(LABELS, SCORES) computes the Receiver Operating Characteristic (ROC) curve [1]. LABELS is a row vector of ground truth labels, greater than zero for a positive sample and smaller than zero for a negative one. SCORES is a row vector of corresponding sample scores, usually obtained from a classifier. The scores induce a ranking of the samples where larger scores should correspond to positive labels.

```
% find images in same category by groundDis equals to zero.
% plot ROC
% elements in scores/labels((i-1)*10)+1):i*10 represents category:n
% in my assumption each element compare with its corresponding by vl_roc
labels=ones(100,1)
scores1=zeros(100,1)
scores2=zeros(100,1)
scores3=zeros(100,1)
scores4=zeros(100,1)

for i=1:9
    scores1((((i-1)*10)+1):i*10,1)=hsvDis((((i-1)*10)+1),(((i-1)*10)+1):i*10);
    scores2((((i-1)*10)+1):i*10,1)=HogDis((((i-1)*10)+1),(((i-1)*10)+1):i*10);
    scores3((((i-1)*10)+1):i*10,1)=SiftDis((((i-1)*10)+1),(((i-1)*10)+1):i*10)
    scores4((((i-1)*10)+1):i*10,1)=SiftDis((((i-1)*10)+1),(((i-1)*10)+1):i*10)
end
subplot(3,4,9); hold on; grid on;  title('hist ROC');
vl_roc(labels, scores1);
subplot(3,4,10); hold on; grid on;  title('hog ROC');
vl_roc(labels, scores2);
subplot(3,4,11); hold on; grid on;  title('dense sift ROC');
```



[5] Fuse the distances from different features, and try your own way of finding the best mixing parameters, i.e,

$$d(I_1, I_2) = w_1 d(H_1, H_2) + w_2 d(HoG_1, HoG_2) + w_3(FV_1, FV_2))$$

For images and their Color Histogram, HoG, and FV aggregated dense SIFT features respectively. Justify your choice of weights, and plot TPR-FPR ROC curves for different choices. [20pts]

```
mean_hog = mean2(hog_raw);
mean_hsv = mean2(hsv_raw);
mean_fis = mean2(Fisher_dsift_raw);
```

weight calculation

```
w1 = (p_hsv.auc / (p_hsv.auc+p_hog.auc + p_fis.auc));
w2 = (p_hog.auc / (p_hsv.auc+p_hog.auc + p_fis.auc));
w3 = (p_fis.auc / (p_hsv.auc+p_hog.auc + p_fis.auc));
Fuse_raw = w1*hsv_raw_new + w2*hog_raw_new + w3*Fisher_dsift_raw_new;
```

The w1 = 0.4823 w2 = 0.5457 and w3 = -0.028

```
figure(randi(100))
subplot(2, 4, 1); imagesc(hsv_raw); title('Pooled Histogram Dist');
subplot(2, 4, 2); imagesc(hog_raw); title('Hog Dist');
subplot(2, 4, 3); imagesc(Fisher_dsift_raw); title('Dense Sift Fv Dist');
subplot(2, 4, 4); imagesc(All_labels); title('GND truth');
subplot(2, 4, 5); vl_roc(labels_hsv, scores_hsv);
subplot(2, 4, 6); vl_roc(labels_hog, scores_hog);
subplot(2, 4, 7); vl_roc(labels_Fisher_dsift, scores_Fisher_dsift);
subplot(2, 4, 8); vl_roc(Label_Fuse, Scores_Fuse);
```