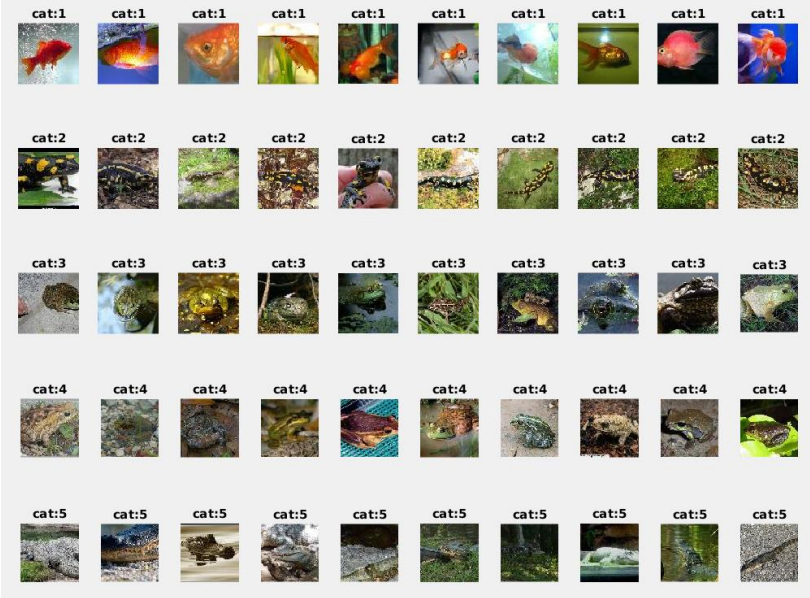
Student Name:_____HONGKUN JIN_____, Student ID: _____16220459_____

Homework-2:

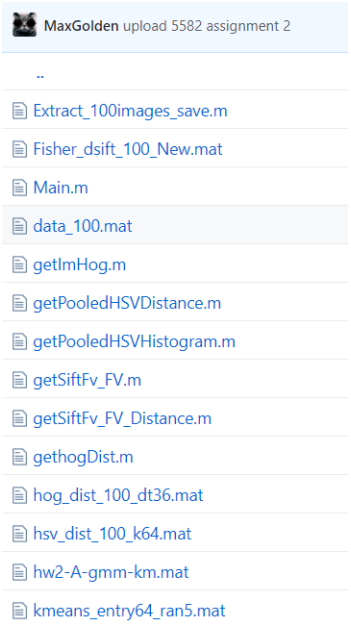
Tiny ImageNet Image Retrieval with Handcrafted features.

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[]:



My GitHub Code Link for this assignment2:

https://github.com/MaxGolden/Personal_Blogs/tree/master/5582IAR/Assignment2/Sub_source



[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]

Before the functions building, I used the code to extract 100 images from training dataset and save the images as our dataset.
One simple code to save the .mat:

```
save data_100.mat;
```

So, the data_100.mat will be our dataset for the following questions.

function [h]=getPooledHSVHistogram(im, codebook, pooling)

1.1 K-means model building

I created another function to train the `centers` as codebook for the HSV function.

```
load('data_100.mat');
random_5 = randi(100, [5, 1]);
random_5_train = [];
for i = 1:5
    im = ims{random_5(i)};
    [r, h, j] = size(im);
    hsv_im = rgb2hsv(im);
    x = double(reshape(hsv_im, [r * h, j]));
    random_5_train = [random_5_train; x];
end
[~, bins] = kmeans(random_5_train, 64);

save kmeans_entry64_ran501.mat bins
```

So, the bins(centers) was saved in kmeans_entry64_ran501.mat, then I could just load and use.

1.2 getPooledHSVHistogram function

The purpose for this function is that it could calculate the HSV histogram based on one image, codebook and pooling method input. The code as below:

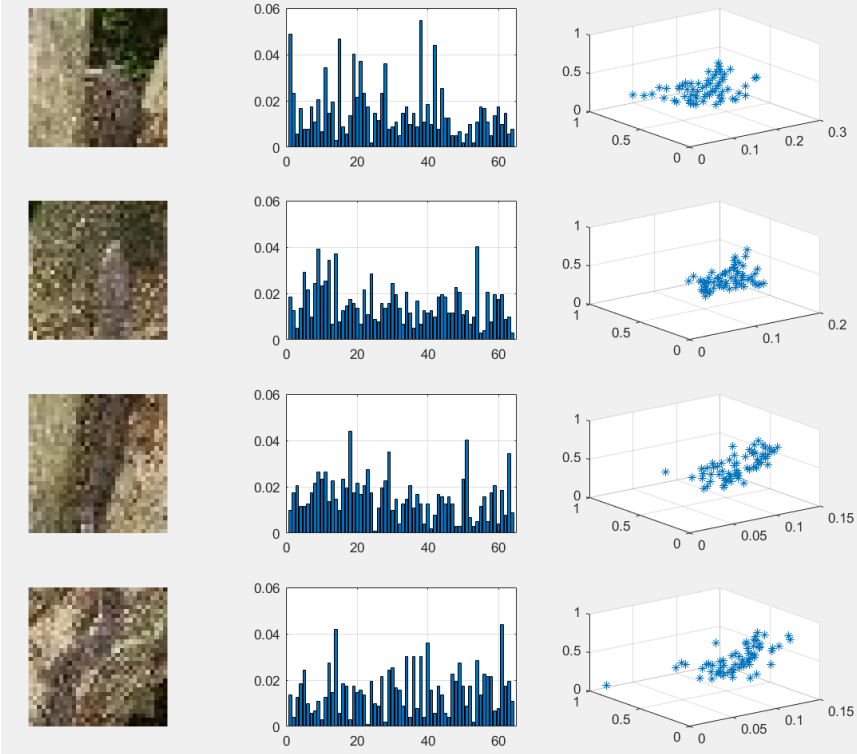
```
pool_hist = [];
num = 0;
for i = 1:pooling(1)
    for j = 1: pooling(2)
        num = num + 1;
        new_im = im_mat(((i - 1) * 32 + 1):i * 32, ((j - 1) * 32 + 1): j * 32, :);   % row then column
        r_pool = r / pooling(1);
```

```
        c_pool = c / pooling(2);

        hsv_im = rgb2hsv(new_im);
        x = reshape(hsv_im, [r_pool * c_pool, 3]);

        % data points and bin
        [n, ~] = size(x);
        [m, ~] = size(bins);
        % compute the n_data_points x n_bins distance matrix
        dist = pdist2(x, bins);
        [~, pixel_bin_offs] = min(dist');
        % Compute the distribution
        for k = 1:m
            h(k) = length(find(pixel_bin_offs == k));
        end
        h = h / sum(h);
        pool_hist = [pool_hist; h];
    end
end
```

For these codes, which is the same with the assignment 1. Easy to complete. The sample results look like this:



It is very obvious that the 2x2 HSV histogram showing in this picture.
And the output h is a 4x64 matrix.

function [dist]=getPooledHSVDistance(im1, im2)

1.3 HSVDistance function

In this part, I calculate the two images HSV histogram and then get their distance by using `pdist2`
Code:

```
h1 = getPooledHSVHistogram(im1, codebook, pooling);
h2 = getPooledHSVHistogram(im2, codebook, pooling);

[n1, m1, k]=size(h1);
[n2, m2, k]=size(h2);

x1 = reshape(h1, n1*m1, k);
x2 = reshape(h2, n2*m2, k);

d1 = mean(min(pdist2(x1, x2)));
d2 = mean(min(pdist2(x2, x1)));
dist = min(d1, d2);
```

[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]
function [hog]=getImHog(im)

2.1 getImHog function

I used the vl_hog to calculate the hog in this function. Based on the question, I still set the pooling size is 2x2 and output of hog is 2x2 cell structure also.
The code:

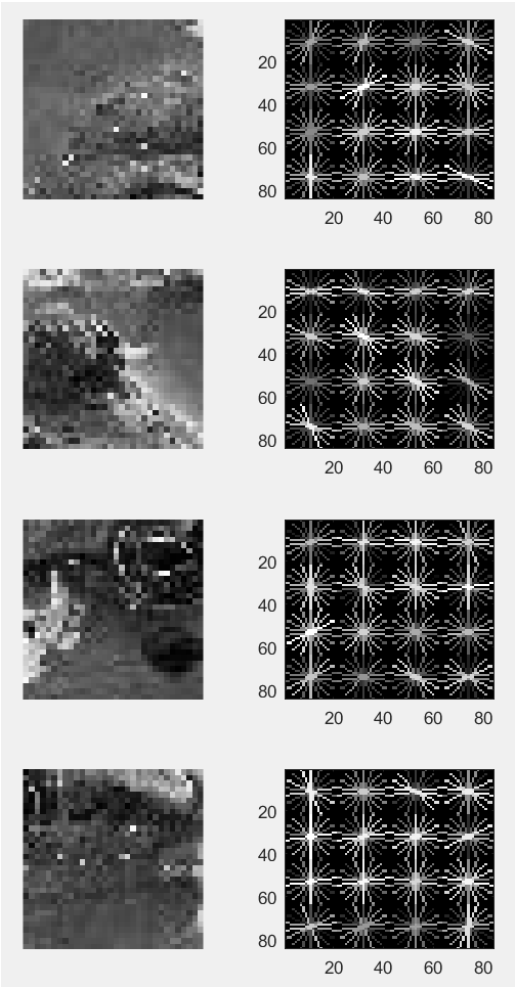
```
pooling = [2, 2];
im_mat = im{1, 1};
[r, c, ~] = size(im_mat);
cellSize = 8;
hog_cell = {};
n = 0;

for i = 1:pooling(1)
    for j = 1: pooling(2)
        n = n + 1;
        r_pool = r / pooling(1);
        c_pool = c / pooling(2);
```

```
        new_im = im_mat(((i - 1) * r_pool + 1):i * r_pool, ((j - 1) * c_pool + 1):
j * c_pool, :);   % row then column
        gray_im = rgb2gray(new_im);
        single_im = im2single(gray_im);

        hog = vl_hog(single_im, cellSize, 'verbose', 'variant', 'dalaltriggs');
        hog_cell{i, j} = hog;
    end
end
```

I also ploted the HOG feture for a sample image:



And the output is 2x2 cell, each cell is a 4x4x36 matrix.

function [hog]=getHogDist(h1, h2)

2.2 getHogDist function

Same as the distance calculation in the question 1, the code:

```
h1 = getImHog(h1);
h2 = getImHog(h2);

[n1, m1, k]=size(h1);
[n2, m2, k]=size(h2);

x1 = reshape(h1, n1*m1, k);  % transform 2x2 cell to 1x4 cell structure
x2 = reshape(h2, n2*m2, k);  % transform 2x2 cell to 1x4 cell structure
dist = [];
for i = 1:4
    cell1 = x1{i}; cell2 = x2{i};
    [n, m, d]=size(cell1);

    cell_r1 =  reshape(cell1, n*m, d);
    cell_r2 =  reshape(cell2, n*m, d);

    d1 = mean(min(pdist2(cell_r1, cell_r2)));
    d2 = mean(min(pdist2(cell_r2, cell_r1)));

    dist = [dist; min(d1, d2)];
end
hog = mean(dist);
hog1 =min(dist);
```

[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]
create a functions:
function [d]=getSiftFv(sift, A, gmm)

3.1 Load GMM and A from `hw2-A-gmm-km.mat`

```
load hw2-A-gmm-km.mat;
```

Load the data we need, the GMM and A.

3.2 Filtering by gaussian

```
    im_gray_single = single(rgb2gray(im_mat));
    h0 = fspecial('gaussian', 3, 1.5);
    % smoothed for dense sift
    im0 = imfilter(im_gray_single, h0);
```

3.3 Calculate the dsift by using vl_dsift

```
    [~, sift2] = vl_dsift(im0, 'step', 2, 'size', 3);
```

3.4 Function getSiftFV.m

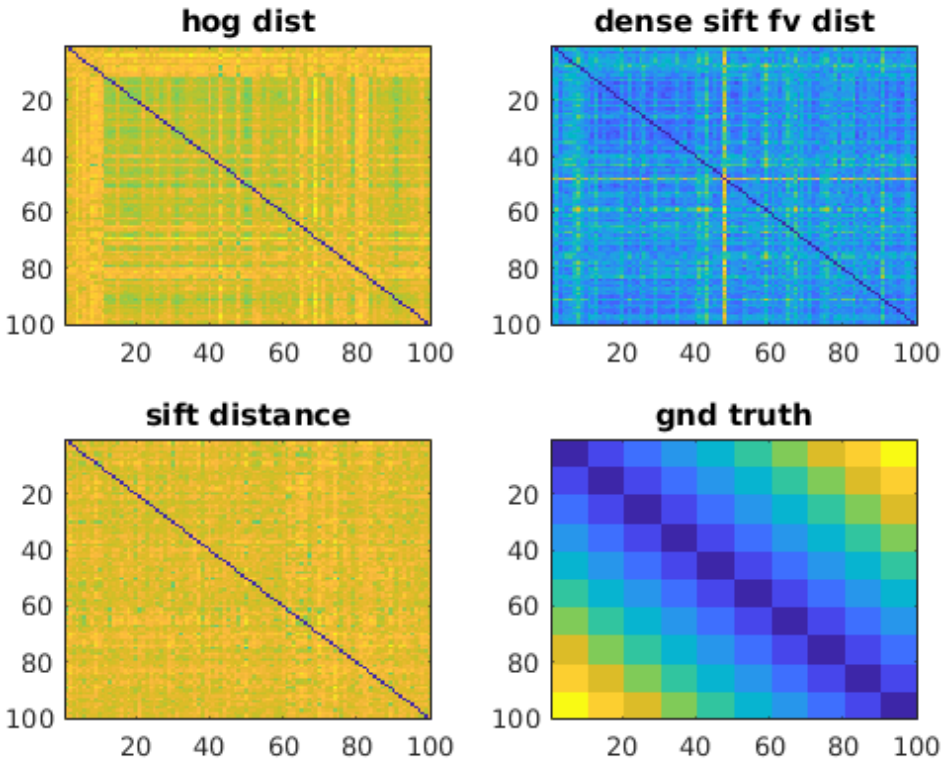
Based on the code from Dr. Li, we used the kd = 16 and nc = 32, to get the dense SIFT by using Fisher Vector.

The code:

```
kd = 16; kd_indx = 1; nc = 32;
% fisher vec
n=1;
dsift_fv = zeros(1, kd*nc);

fv = vl_fisher(A(1:kd, :) * double(sift2), gmm(kd_indx, 2).m, gmm(kd_indx, 2).cov,
gmm(kd_indx, 2).p);
dsift_fv(1, :) = fv(1:kd * nc)';
fprintf('\n %d sift ', 1)
```

[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:



4.1 2x2 pooled HSV feature

Calculate the distance map for these all 100 images

The code:

```
hsv_raw = [];
for i = 1:100
    hsv_colum = [];
    for k = 1:100
        hsv_dist = getPooledHSVDistance(ims(i), ims(k));
        hsv_colum = [hsv_colum; hsv_dist];
    end
    hsv_raw = [hsv_raw, hsv_colum];
end
save hsv_dist_100_k64.mat hsv_raw
```

From the code we can see that it will save the distance map for HSV when it finishes all images calculation. And the output `hsv_raw` is a 100x100 matrix. We will plot the dist map based on that.

4.2 HoG feature

Distance matrix:

Same thing as the HSV feature distance calculation.

The code:

```
hog_raw = [];
for i = 1:100
    hog_colum = [];
    for k = 1:100
        hog_dist = gethogDist(ims(i), ims(k));
        hog_colum = [hog_colum; hog_dist];
    end
    hog_raw = [hog_raw, hog_colum];
end
save hog_dist_100_dt36.mat hog_raw
```

Also, save the HOG distance to local and prepare for distance map ploting.

4.3 Fisher Vector aggregated dense SIFT feature

Same thing:

```
load hw2-A-gmm-km.mat;
Fisher_dsift_raw = [];
for i = 1:100
    Fisher_dsift_colum = [];
    for k = 1:100
        Fisher_dsift_dist = getSiftFv_FV_Distance(ims(i), ims(k), A, gmm);
```

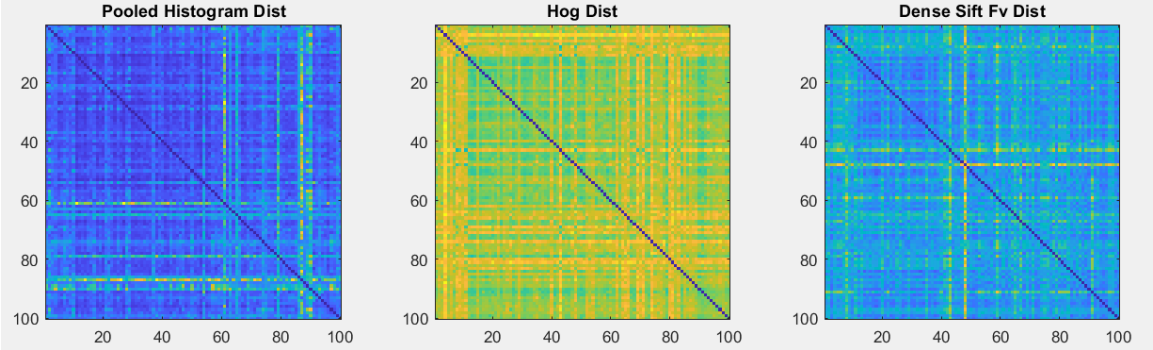
```
        Fisher_dsift_colum = [Fisher_dsift_colum; Fisher_dsift_dist];
    end
    Fisher_dsift_raw = [Fisher_dsift_raw, Fisher_dsift_colum];
end
save Fisher_dsift_100_New.mat Fisher_dsift_raw
```

And saved distance matrix.

4.4 Plot distance map for these three feature types.

The code:

```
subplot(1, 3, 1); imagesc(hsv_raw); title('Pooled Histogram Dist');
subplot(1, 3, 2); imagesc(hog_raw); title('Hog Dist');
subplot(1, 3, 3); imagesc(Fisher_dsift_raw); title('Dense Sift Fv Dist');
```



4.5 TPR-FPR ROC

4.5.1 HSV Performance

```
labels_hsv = ones(1, 100);
scores_hsv = zeros(1, 100);
for i =1:10
    for k = (10*i-9):10*i
    hsv_raw_1 = hsv_raw;
    hsv_raw_1(k, k) = max(hsv_raw_1(:, k));
    [value, index] = min(hsv_raw_1(:, k));
    scores_hsv(1, k) = value;
    if index <= 10*(i-1) || index > 10*i
        labels_hsv(1, k) = -1;
    end
    end
en
```

Get the minimal distance for each image comparing with others, except itself. And get the labels and scores for ploting the ROC by using vl_roc.

4.5.2 HOG Performance

```
labels_hog = ones(1, 100);
scores_hog = zeros(1, 100);
for i =1:10
    for k = (10*i-9):10*i
    hog_raw_1 = hog_raw;
    hog_raw_1(k, k) = max(hog_raw_1(:, k));
    [value, index] = min(hog_raw_1(:, k));
    scores_hog(1, k) = value;
    if index <= 10*(i-1) || index > 10*i
        labels_hog(1, k) = -1;
    end
    end
end
```

Same thing as the HSV.

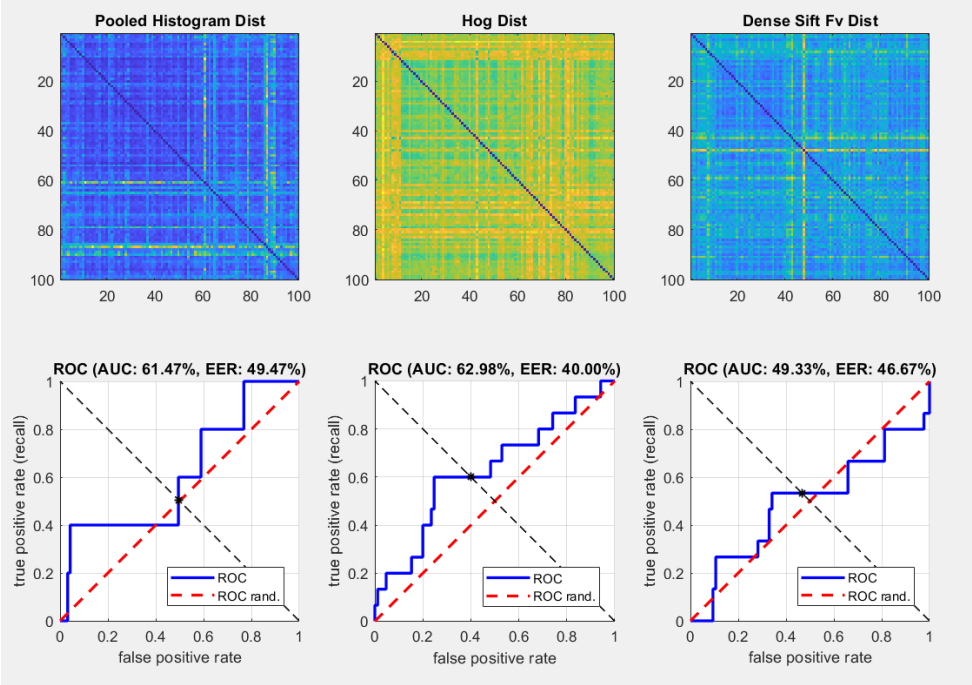
4.5.3 Distance of Fisher Vector aggregated dense SIFT feature Performance

```
labels_Fisher_dsift = ones(1, 100);
scores_Fisher_dsift = zeros(1, 100);
for i =1:10
    for k = (10*i-9):10*i
    Fisher_dsift_raw_1 = Fisher_dsift_raw;
    Fisher_dsift_raw_1(k, k) = max(Fisher_dsift_raw_1(:, k));
    [value, index] = min(Fisher_dsift_raw_1(:, k));
    scores_Fisher_dsift(1, k) = value;
    if index <= 10*(i-1) || index > 10*i
        labels_Fisher_dsift(1, k) = -1;
    end
    end
end
```

4.6 Plot the distance map and ROC together

```
subplot(2, 3, 1); imagesc(hsv_raw); title('Pooled Histogram Dist');
subplot(2, 3, 2); imagesc(hog_raw); title('Hog Dist');
subplot(2, 3, 3); imagesc(Fisher_dsift_raw); title('Dense Sift Fv Dist');
subplot(2, 3, 4); vl_roc(labels_hsv, scores_hsv);
subplot(2, 3, 5); vl_roc(labels_hog, scores_hog);
subplot(2, 3, 6); vl_roc(labels_Fisher_dsift, scores_Fisher_dsift);
```

And the plot:



[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]

The way I want to calculate the the w1, w2, w3 for these features is that I will scale them first by using mean. And fuse them together with their auc weight based on the three features.
The code:

5.1 Get the mean of the distance matrix for each feature

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

5.2 Get the AUC from the ROC

```
[~, ~, p_hsv]= vl_roc(labels_hsv, scores_hsv);
[~, ~, p_hog]= vl_roc(labels_hog, scores_hog);
[~, ~, p_fis]= vl_roc(labels_Fisher_dsift, scores_Fisher_dsift);
```

5.3 W 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;
```

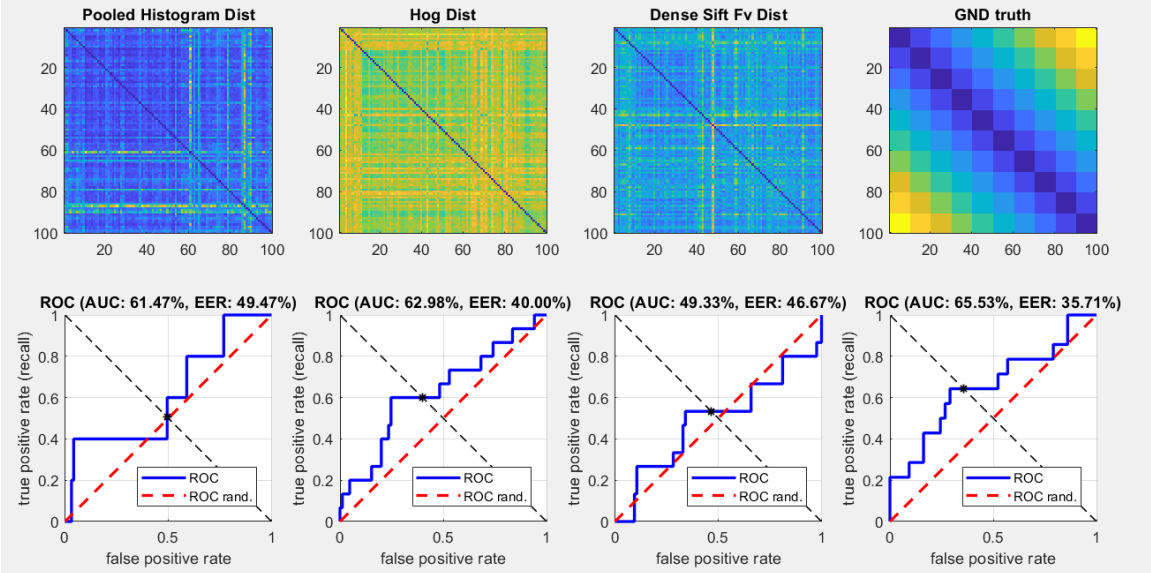
The w1 = 0.4823 w2 = 0.5457 and w3 = -0.028

5.4 Get the final fuse model:

Code:

```
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);
```

The plot:



Thank you!

Edited by Hongkun Jin