EE569 HW#4

Student Name: Shi Lin Chen
Student ID: 2991911997
Student Email: shilinch@usc.edu
Submission Date: Mar 22, 2020

Problem1: Texture Analysis and Segmentation

I. Motivation and Abstract

Texture Analysis could analyze the textures in image. There are several kinds of analysis in this area, such as texture classification and texture segmentation. People utilize texture analysis to process some features in an image in order to implement more advance technique in image signal processing. In this problem, we are asked to implement 2 tasks (classification and segmentation).

II. Approach and Result

Laws Filter

Laws filters are used to analyze textures in the images. These filters are generated by multiplying by all of 5 1D kernels.

Table.1 1D Kernel for 5x5 Laws Filters

Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5 (Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

Therefore, there are total 25 laws filter with size 5*5, which are L5L5, E5E5, S5S5, W5W5, R5R5, L5E5, L5S5 … and so on. Each kernel could detect one certain pattern in one direction. When apply these 25 filters to the images, we will get response images for each filter. These filtered images indicate the features of the given image.

L5L5	L5E5	L5S5	L5W5	L5R5
E5L5	E5E5	E5S5	E5W5	E5R5
S5L5	S5E5	S5S5	S5W5	S5R5
W5L5	W5E5	W5S5	W5W5	W5R5
R5L5	R5E5	R5S5	R5W5	R5R5

Pre-Processing: Subtract Local Mean of the pixels

This operation subtracts the local average from each pixel value to get new pixels. The local average is calculated by averaging the pixels value within a fixed window around given pixels. This pre-processing purpose is to reduce the effect of luminance varies on subsequent result using laws filters.

After understanding the laws filters, we could apply these filters to both train and test images to get the 25-dimensional feature vectors. The feature dimension was reduced from 25 to 15 first by averaging the symmetric filter pairs like L5E5/E5L5, E5S5/S5E5 … and so on. Then the feature dimension could be reduced from 15 to 3 using PCA (Principal Component Analysis) and K-means clustering was applied to these 3-D feature vector to obtain the grouping. For the second part, laws filters are applied to the image and energy features are computed using window approach. Then we need to normalize the energy features which could be computed simply divided by L5L5 kernel. Before doing PCA and K-mean, we need to standardize the feature vector by this formula:

$$x_i' = \frac{x_i - \bar{x}}{std(x)}$$

Then we could apply PCA to reduce dimension and K-mean to classify the test images!

As for the Texture Segmentation, we could observe that the giving image combing 6 different texture by eyes. We are asked to segment the textures by using laws filters and k-means. First, we need to apply laws filter to the given image and calculate the local mean for every pixel in order to reduce the effect of luminance. Then computer the energy of the filtered image and normalize the vectors with law filter L5L5. Finally, we could form the feature vector of every pixels (450*600,15). After finishing the feature

vector, set k to 6 in k-means algorithm and feed all vectors into k-means algorithm, then we could get a segmentation result. To illustrate segmentation results, the result image is created with pixels in same cluster assigned to one gray value, and total 6 different gray values, which are (0, 51, 102, 153, 204, 255), given in assignment.

To be noticed, k-means is an unsupervised training, and SVM and RF are supervised training. SVM and RF method could be used in matlab built-in function. The procedure is almost the same except the training part, we have to extract to features first and feed the feature vectors to SVM and RF classifier to get the labels.

K-means method:

```
[coeff,score] = pca(F_norm,'NumComponents',3);
cluster = kmeans(score,4);
```

where F_norm is 15-D feature vectors of given test images.

SVM & RF function in matlab:

```
SVMModel = fitcecoc(F_train_norm,L);
LabelSVM = SVMModel.predict(F_norm);

TreeModel = TreeBagger(50,F_train_norm,L);
LabelTree = TreeModel.predict(F_norm);
```

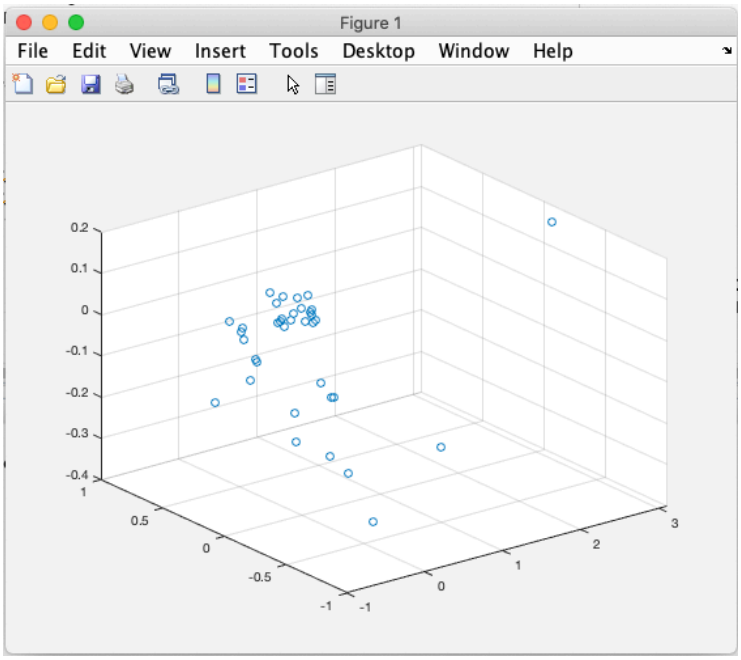
Result:

1.a.2

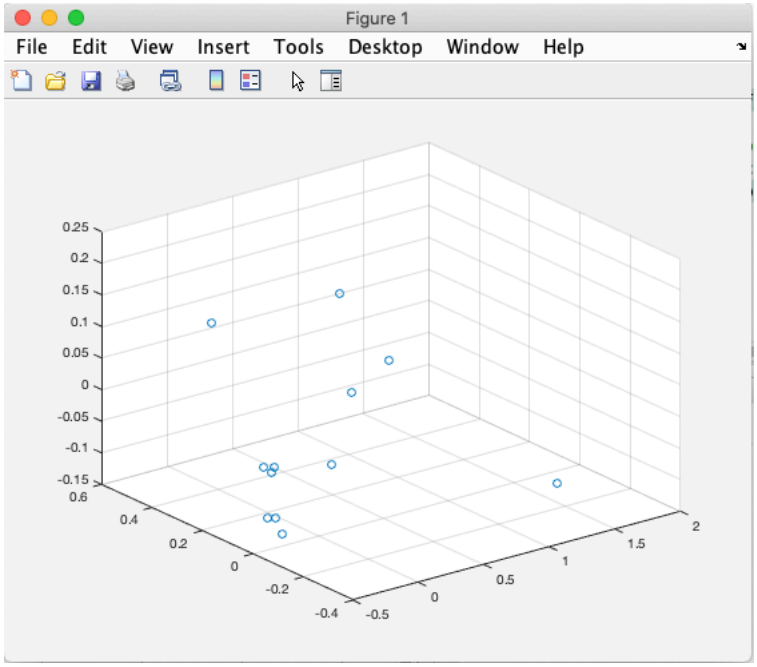
Standard deviation of feature vector of 12 test images.

```
stdtest =

  Columns 1 through 10

  664.0640  373.7199  264.4873  259.3167  708.1051  258.3505  144.4401  116.9949  101.1502  122.1600

  Columns 11 through 20

  185.6793  122.4285  104.4392   88.8777   95.9265  225.9771  121.2062  103.1534   88.0728  105.1613

  Columns 21 through 25

  591.3975  152.2984  127.3321  118.0217  247.8718
```

1.a.3 plot 3-D feature vector



PCA 3D-feature space (train)



PCA 3D-feature space (test)

1.b.(1.2)

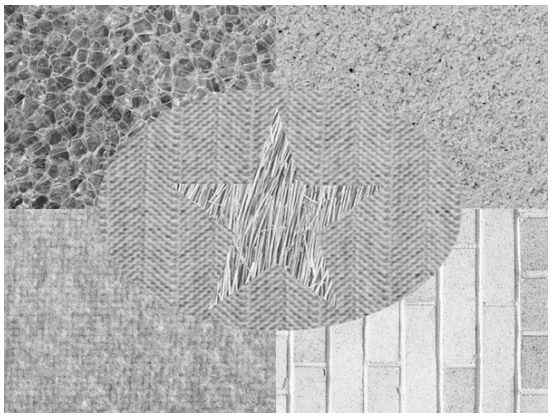
	GT	k-means	k-means (with PCA)	Random forest (with PCA)	SVM (with PCA)
Test1	1	1	1	1	1
Test2	2	1	1	3	1
Test3	2	4	4	4	4
Test4	3	4	4	3	4
Test5	4	4	4	1	1
Test6	1	1	1	1	1
Test7	3	3	3	3	1
Test8	4	4	4	1	4
Test9	4	4	4	4	4
Test10	3	3	3	3	1
Test11	2	2	2	2	2
Test12	1	1	1	1	1
Error Rate		0.25	0.25	0.333	0.5833

	GT	Random Forest (15-D)	SVM (15-D)
Test1	1	1	1
Test2	2	2	1
Test3	2	2	4
Test4	3	3	3
Test5	4	4	4
Test6	1	1	1
Test7	3	2	3
Test8	4	4	4
Test9	4	4	4
Test10	3	3	3
Test11	2	2	3
Test12	1	1	1
Error Rate		0.083	0.25

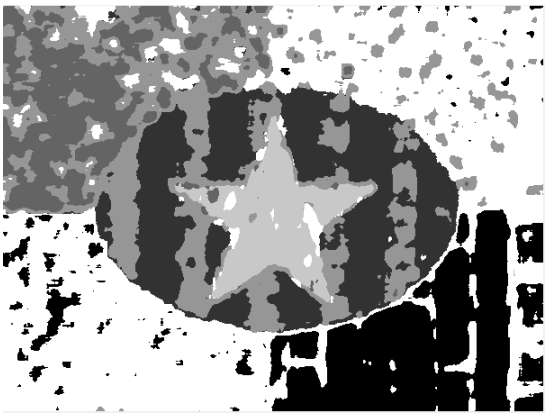
Note: 1 = grass, 2 = blanket, 3 = brick, 4 = rice

1.C. Texture Segmentation

For 14-D feature vector:

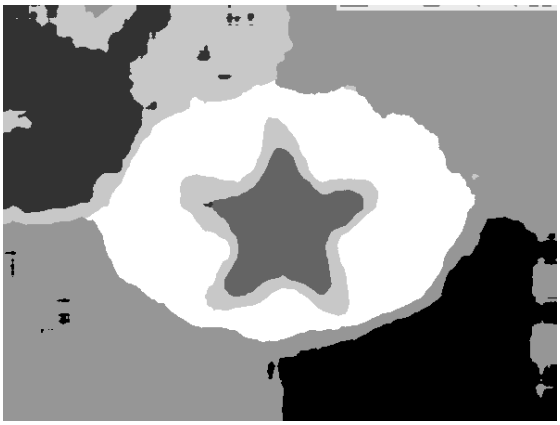


Original Image



Window Size 7 * 7

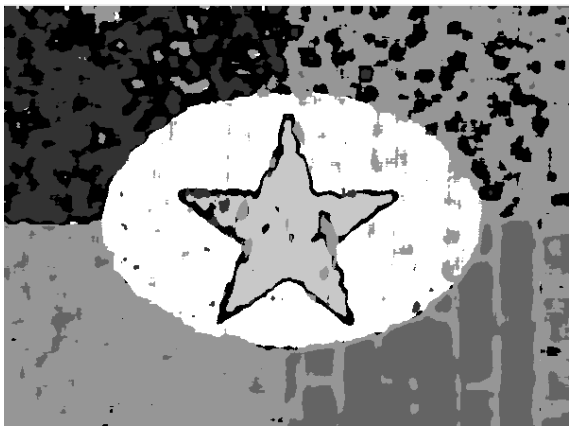
Window Size 25*25



Window Size 41 * 41

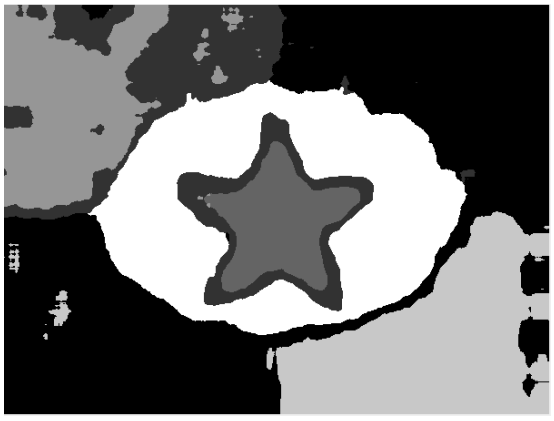
For 3-D feature vector: (14D->3D using PCA)

Window Size 7 * 7

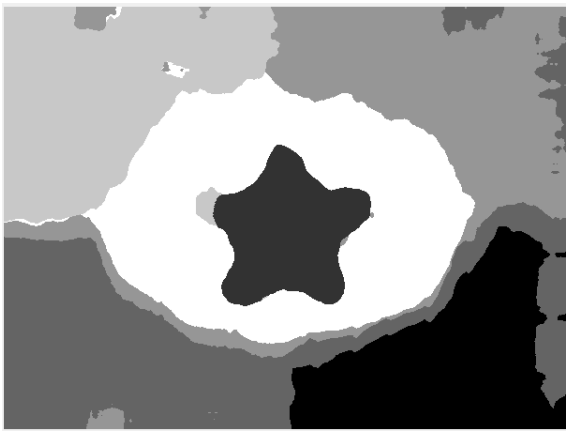


Window Size 13 * 13

Window Size 25 * 25



window Size 35 * 35

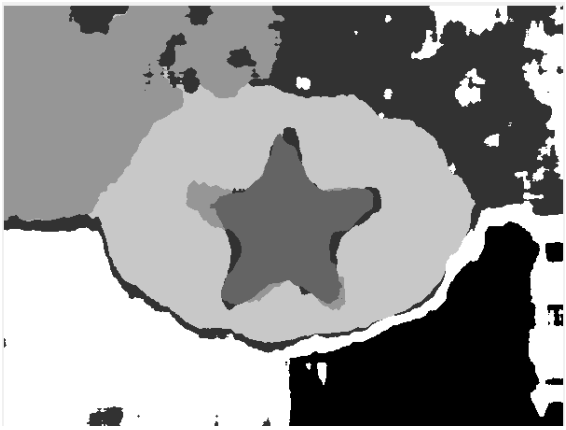


Window Size 51 * 51

*PCA – reduce to different dimension with window size 35 * 35*

14-D -> 3D

14-D -> 6D



14-D -> 9D

Not using PCA (14-D with window size 35*35)

III. Discussion

a.(2.) – discriminant power

In theory, the feature dimension with minimal standard deviation should has the strongest discriminant power, since it makes data out of cluster more obvious to identify.

On the other hand, the feature dimension with maximal standard deviation should has the weakest discriminant power. I calculate the standard deviation of feature vector without normalizing and standardizing, the result shows that filter W5W5 has the strongest discriminant power and L5R5 has the weakest discriminant power.

b.(1.) – unsupervised training

The result and error rate could be viewed at the last part, we could see k-means actually has less error rate when either 15-D feature vector or using PCA to reduce to 3-D feature vector. On the other hand, I got better result when Random Forest and SVM with 15-D feature vector than 3-D vector. Since k-means will choose different centroids when calculating the distances in classifying which will influence error rate when choosing good or bad centroids, I got different result when implement k-means method several times.

(2.) – supervised training

The error rate of using random forest with and not with PCA dimension reduction are both lower than the error rate of using SVM.

c. – texture segmentation

My texture segmentation was done by using different window size and different dimensions of feature vectors. It is observed that the segmentation result got better with increasing window size. However, if the window size gets too big, the result might not be as good because the different region of texture may merge.
As for the dimension, I apply PCA to reduce to dimension from 14-D to 3, 6, and 9 D, but the results are really similar to each other. I think the influence of window size is much greater than the dimension.

d. – Advance Texture Segmentation

1. PCA result is provided above, and the results of 3-D, 6-D, and 9-D seem don't have a lot of different comparing with the result created by original 14-D feature vector.

2. In order to merge the small holes in one texture, I think we could apply large window size (not larger than the original texture size) mask to filter the texture segmentation pixels value. Compare the amount of the label and find out which texture it is, then set other pixels to the same label.

3. Boundary could be handled in the beginning (before implementing texture segmentation.) My thought is to scan the whole image first and record the boundary pixels positions. After implementing segmentation, we could use the boundary position we just stored in the beginning to set the boundary in the image. Due to the time lack, I haven't implemented this thought.

Problem2: Image Feature Extractor

I. Motivation and Abstract

Feature extraction could represent the image information using less dimensional, and the features could be easily compared with others, in order to find whether there is a good match or not. People could apply this technique in different areas of image processing, such as object identification or image matching.

II. Approach and Result

1. Read two pictures.
2. Find the key points of both images
3. The key-point with the largest scale is picked in Husky_3 and its closest neighboring key-point is found in Husky_1.

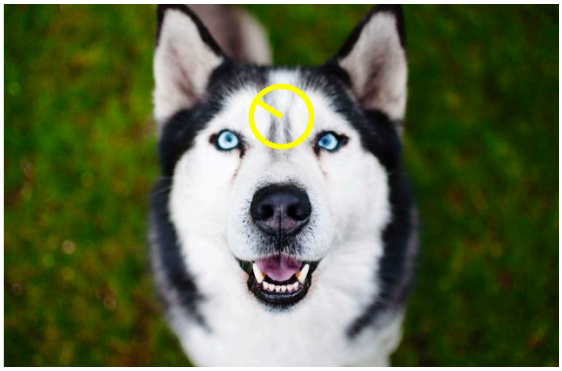
SIFT:

1. Scale-space extrema detection
2. Key point localization
3. Orientation assignment
4. Key point descriptor
5. Key point matching

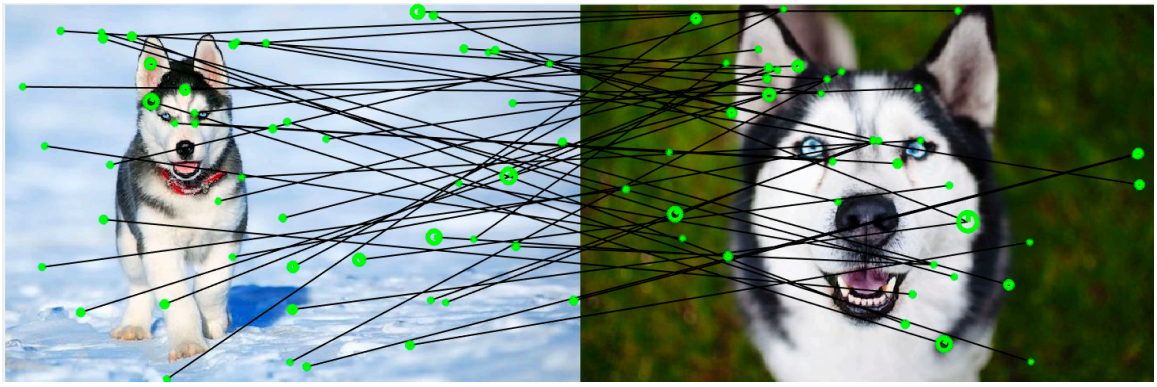
Key points between two images are matched by identifying their nearest neighbors
Programming Reference: https://www.vlfeat.org/index.html

Result:

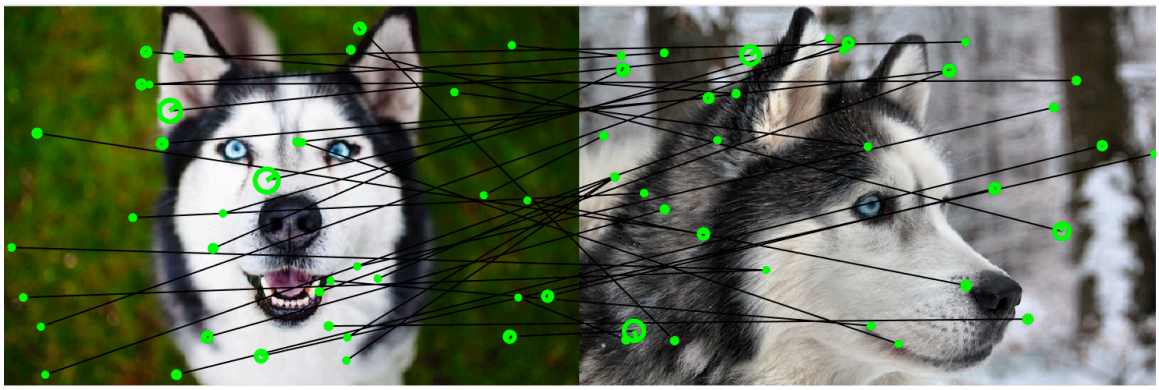
b.(1.)



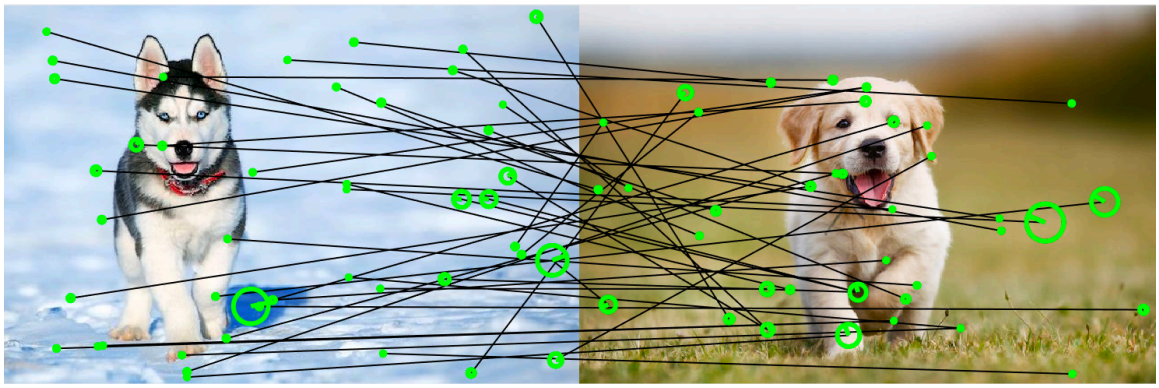
2.



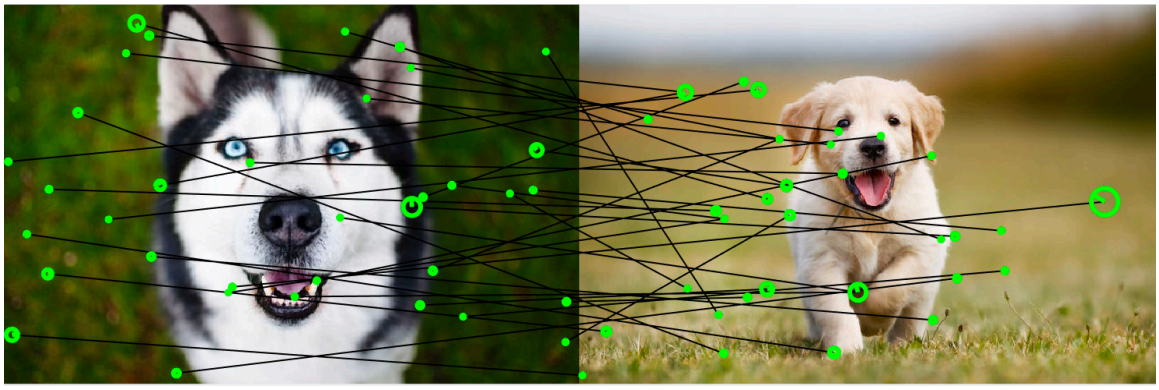
Husky_1 and Husky_3



Husky_3 and Husky_2



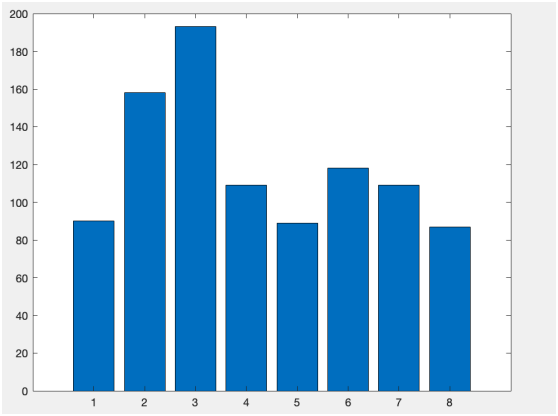
Husky _1 and Puppy_1



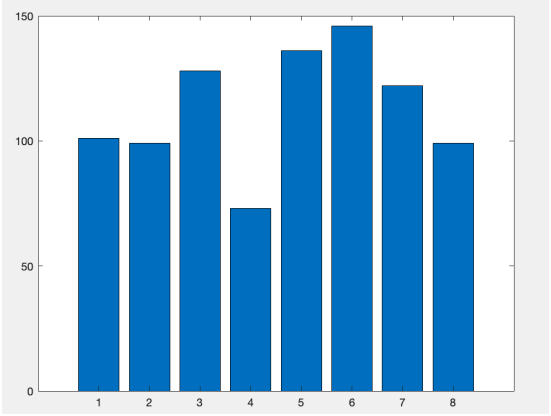
Husky_3 and Puppy_1

(c.) Bag of Words

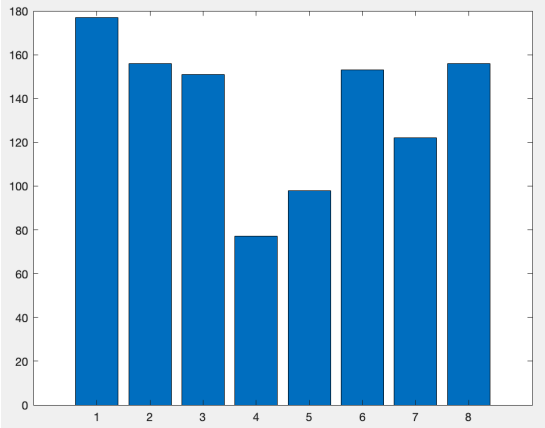
Histogram:



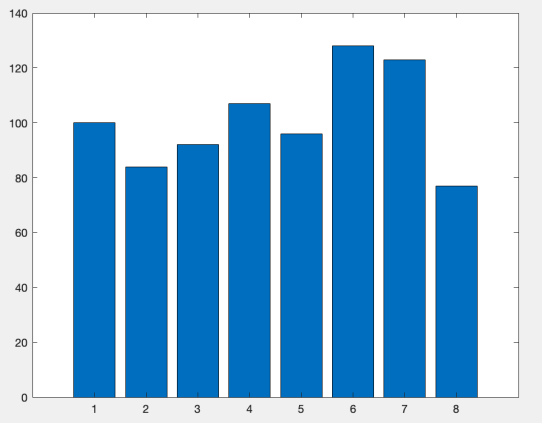
Husky_1



Husky_2



Husky_3



Puppy_1

Codebook – 8 points matching

Husky3 vs husky1								
1	2	6	5	7	3	8		
1	3	8	4	7	2	6		

Fig.1

Husky3 vs husky2								
2	5	3	6	1	7	8		
7	1	6	5	2	8	4		

Fig.2

Husky3 vs Puppy1								
3	8	1	6	7	2	4		
7	4	6	6	2	8	3		

Fig.3

III. Discussion

Part(a.) – [1] reference from paper

Q1:

SIFT is invariant to image scaling, translation, rotation and partially invariant to illumination changes and 3D camera viewpoint.

Q2:

Scaling: SIFT is robust to scaling through multi-scale filtering. SIFT use difference of gaussian, which is an approximation of LoG (Laplacian of Gaussian) to search the stable features in the image across all possible scales. Once the DoG are found, images are searched for local extrema over scale and space. If the pixel is a local extrema, it is a potential key point which basically means it is best represented in that scale.

Rotation: By assigning one or more orientation to each key point of the image based on the local image gradient direction. With these orientation information, rotated features could be compared with original features.

Translation: The local image gradients are measured at the selected scale in the region around every key point. These key points could allow significant levels of local

shape distortion.

Q3:

The feature vector is normalized to unit length. A brightness change in which a constant is added to each image pixel will not change the gradient values, as they are computed from pixel difference. Therefore, the descriptor is invariant to illumination change.

Q4:

Difference of Gaussians approximates Laplacian of Gaussians with less computation. Therefore, the main advantage of using DoG instead of LoG is the speed!

Q5:

Vector size = 8*4*4 + 8*2*2 = 160

Part(b.)

1.

The two largest scale key point seems not matched in two given images. Since SIFT could handle different scales of object. The reason here that the key points are not matched might because the scale between these two huskies is too huge.

2.

The matching points are some parts good but some parts bad in my result. In the pair of Husky_2 and Husky_3, two faces of Huskies are huge enough and the matching result seems better comparing to others. The background might influence the matching result, in some of my matching result, when object is small and the background color are really similar to the object, then the matching points might fail at some place in the images.

3.

From the codebook, we could see the matching part in Fig.1 and Fig.2 are a little different from Fig.3. In Fig.3, there are same label in Puppy corresponding to different labels in Husky. This indicates that there are two different clusters in one image are matched to the same cluster in another image, implying that the matching result are not that good when using SIFT in this two images.