Samwoo Seong
7953-6137-66
samwoose@usc.edu

EE569 Homework #4
March 22, 2020

1.

1.1 Abstract and Motivation

Textures in images are not mathematically defined. However, if we use Laws filters, we can characterize one texture of an image. In other words, we can extract features from the texture images. Plus, by assigning one texture (e.g. grass, rice, brick, or blanket) to the image, we can construct a train data set (i.e. Data set that has features and a corresponding label). Now, this train data set can be used for texture classification task. Similarly, we can extract energy feature and apply K-means algorithm for texture segmentation task in one image.

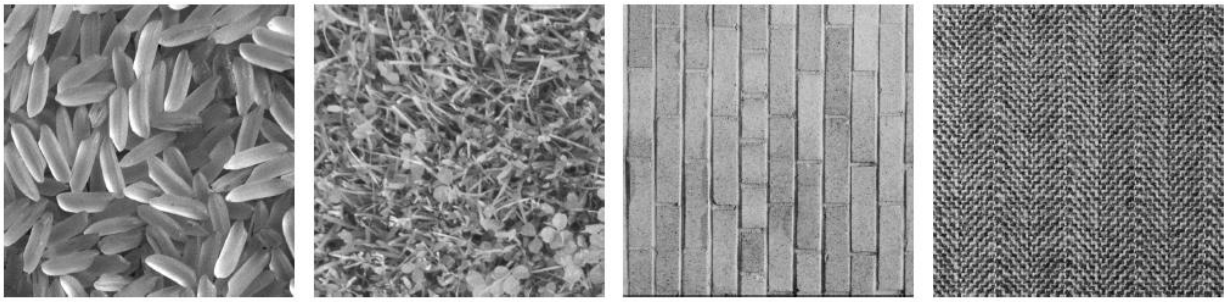

Fig 1. Texture examples (rice, grass, brick, and blanket from left to right) [1]

1.2 Approach and Procedures

(a)

1.

To extract features from an image, we can use 25 Laws filters. Each filter is able to capture different characteristic of the image in frequency domain such as low and high frequency components and more. Specific procedure can be shown as step-by-step procedure.

Step1. Construct 25 Laws filters by 5 1-D kernels given using tensor product.
e.g. filter1 = L5⊗L5, filter2 = L5⊗E5, … , filter24 = R5⊗W5, filter25 = R5⊗R5

Step2. Take convolution on 36 train images and 12 test images with the 25 filters
It will generate 25-D feature vector for each pixel of image. Hence, we will end up with (image length)*(image height) feature vectors per image.
e.g. 128x128 = 16384 feature vectors per image

Fig 2. Step-by-step procedure of feature extraction with 25 Laws filters.

2.
Based on 16384 feature vectors we computed above, we can actually construct train data set and test data set for texture classification task by averaging 16384 25-D feature vectors per image. Furthermore, we can reduce its dimension to 15-D since feature values computed by certain filters are very similar to each other.
Step-by-step procedure is shown below. (The certain pairs are included in the procedure)

Step1. Average 16384 feature vectors feature value wise. It still maintains its previous dimension, 25-D. Hence, we will end up one 25-D feature vector per image. When averaging a feature vector, we need to use asolute feature values.

Step2. Average feature vectors from such pairs below. This is because each feature values of averaged feature vector will be very simiar due to taking asolute operation to feature values at step1. It will lead us to 15-D feature vector per image.
e.g. Pairs needed to be averaged.
(L5E5,E5L5), (E5S5,S5E5), (L5S5,S5L5), (E5W5,W5E5), (L5W5,W5L5), (E5R5,R5E5), (L5R5,R5L5), (S5W5,W5S5), (W5R5,R5W5), (S5R5,R5S5)

Step3. Repeat step1 and 2 for 36 train images and 12 test images.
We will obtain 36 feature vectors for train data set and 12 feature vectors for test data set. Here, each feature vector has 15 dimension.

Fig 3. Step-by-step procedure of feature averaging

3.
We can go even further. In traditional machine learning, relationship between dimensionality of feature vector and number of data points in data set plays a key role. As rule of thumb, to overcome overfitting during machine learning process, it would be better to keep the relationship, number of data >> (3~10) x (dimensionality of feature vector). We can achieve this by the principle component analysis (PCA). The PCA provides orthogonal directions that give the greatest variance in data [2]. By using the directions, we can project each data point to the new feature space with reduced dimension. In other words, dimension of feature vectors can be reduced. In this practice, we reduce feature vector dimension from 15-D to 3-D. Therefore, we can satisfy the relationship mentioned above between number of data points and dimensionality of feature vector. E.g. 36 train data points > (3~10) x 3 = 9 to 30
Step-by-step procedure is given below.

Step1. Apply the principle component analysis to 15-D feature vector.

Step2. Repeat step1 for all feature vectors computed at the previous procedure.

Step3. Plot the 3-D feature vectors in the new feature space.

Fig 4. Step-by-step procedure of feature reduction.

(b)

We have obtained total 4 data sets, train data set & test data set with 15-D, and train data set & test data set with 3-D. Based on these data sets, we can perform unsupervised and supervised learning to carry out texture classification task.

1.

In practice, unsupervised learning becomes handy when we need to label a data set which labels for each data are partially unknown. Additionally, unsupervised learning is also known as clustering task meaning it can be used to cluster groups that has similar characteristics. In this problem, we are going to perform K-means clustering to classify test images to one of classes such as grass, brick, rice, or blanket.

Step-by-step procedure can be summarized as follows.

Step1. Contruct X_test variable. It is consist of 12 feature vector without labels. Dimension of the variable is 12x15 for 15-D feature vector.

Step2. Set number of clusters to 4 (i.e., there are four classes.)

Step3. Perfrom K-means clustering using kmeans function in MATLAB. The function parameters are X_test variable and number of clusters.

Step4. Calculate error rate based on human-eyes observation.
e.g. result of data point1:
label by eyes = grass,
lable by K-means = grass
$\Rightarrow$ correctly classified
$\Rightarrow$ Otherwise, incorrectly classified.
Error rate = 1 – accuracy
, where accuracy = # of data points correctly classified / # of all data points

Step5. Repeat step1, 2, 3, and 4 for 3-D feature vector.

Fig 5. Step-by-step procedure of unsupervised learning

2.
With train data set with 3-D feature vectors and corresponding labels, we can perform supervised learning. This task is useful to develop the machine learning model that will be used for offline classification task on embedded system that can recognize certain objects. As we discuss above, to reduce effect of overfitting (it often happens when number of data points is not bigger than dimensionality of feature vector), we will use 3-D feature vector. For the classifier, we are going to use Support Vector Machine, and Random Forest which perform great in general.
Step-by-step procedure is provided below.

Step1. Construct X_train and Y_train variable.
X_train is consist of 36 3-D feature vectors. Hence, its dimension is 36x3.
Y_train is consist of labels of the feature vectors. Each integer label corresponds to texture type such as blanket: 1, brick: 2, grass: 3, and rice: 4.
We have 9 grass images, 9 brick images, 9 rice images, and 9 blanket images.

<Support Vector Machine>
Step2. Using built-in function train a predictor(a.k.a machine learning model). Parameters of function are X_train, and Y_train. You can choose several kernel options.

Step3. Using the predictor, predict labels of test data points.

Step4. Compute error rate. Formular for error rate is the same that used in Unsupervixed Learning.

Fig 6. Step-by-step procedure of SVM

Step1. Construct X_train and Y_train variable.
X_train is consist of 36 3-D feature vectors. Hence, its dimension is 36x3.
Y_train is consist of labels of the feature vectors. Each integer label corresponds to texture type such as blanket: 1, brick: 2, grass: 3, and rice: 4.
We have 9 grass images, 9 brick images, 9 rice images, and 9 blanket images.

<Random Forest>
Step2. Using built-in function train a predictor. Need to choose number of trees of the predictor. Parameters of function are X_train, Y_train, and number of trees. Depending on the number of tree, the model could have slightly better performance because predictor with many tree tends to overcome overfitting issue.

Step3. Using the predictor, predict labels of test data points.

Step4. Compute error rate. Formular for error rate is the same that used in Unsupervixed Learning.

Fig 7. Step-by-step procedure of Random Forest

(c)
We have learned each Laws filter can capture different characteristics from an image. If we think a little bit deeper, we can find out characteristics of each pixel with Laws filters in one image. Using this property, we can also perform texture segmentation task that we will try to distinguish multiple textures in one image. Specifically, to capture each pixel's feature, we will use energy feature and normalization approach. After extracting features of all pixel, we perform K-means algorithm to determine what texture each pixel is.
Step-by-step procedure is given below.

Step1. Extract feature of each pixel by convolution operations with 15 Laws filters. Output of this step is 15 128x128 matrix.

Step2. Calculate energy of each pixel with window approach. Output of this step will have the same dimension, however value at (row,col) represents energy of the pixel.

Step3. Normalize values of feature vectors with values computed by L5L5 filter. In other word, each feature vector, [x1,x2,...,x15] becomes [1,x2/x1,...,x15/x1].

Step4. Remove 1 in all feature vectors. Dimension of feature vectors become 14-D

Step5. Apply K-means clustering to determine texture of each pixel.

Step6. Based on the index computed by K-means algorithm we can assign different gray scale for each pixel.
e.g. index 1: 0, index 2: 51, index 3:102, index 4:153, index 5:204, index 6:255

Step7. Display the image generated at Step6.

Fig 8. Step-by-step procedure of texture segmentation

(d)

Even though we could extract energy features of each pixel with Laws filters, it is still possible that the segmentation result is not well performed. This is because data points also known as feature vectors are not purely separable meaning different textures are lying on the same region in feature space. This happens when feature vector, itself, does not 100% represent the texture class properly. This is why feature extraction task is important and not very trivial in machine learning field. However, we can still slightly improve our result by PCA and post processing.

For PCA, it provides more separability amongst data points sometimes. Furthermore, to make our result look better, we can manually find location of some dots that are not properly assigned after K means clustering.

Step-by-step procedure is shown as follows.

Step1. Recude dimension of the feature vectors computed at problem 1 (c) to 3-D

Step2. Apply K-means clustering to determine texture of each pixel.

Step3. Based on the index computed by K-means algorithm we can assign different gray scale for each pixel.
e.g. index 1: 0, index 2: 51, index 3:102, index 4:153, index 5:204, index 6:255

Step4. Display the image generated at Step3.

Step5. Find locations where have small holes (i.e. incorrectly assigned texture) and manually post process thoese small holes to be correct texture.

Step6. Display the image generated at Step5.

Fig 9. Step-by-step procedure of texture segmentation Enhancement.

1.3 Experimental Results (Partially open source is used)

(a)

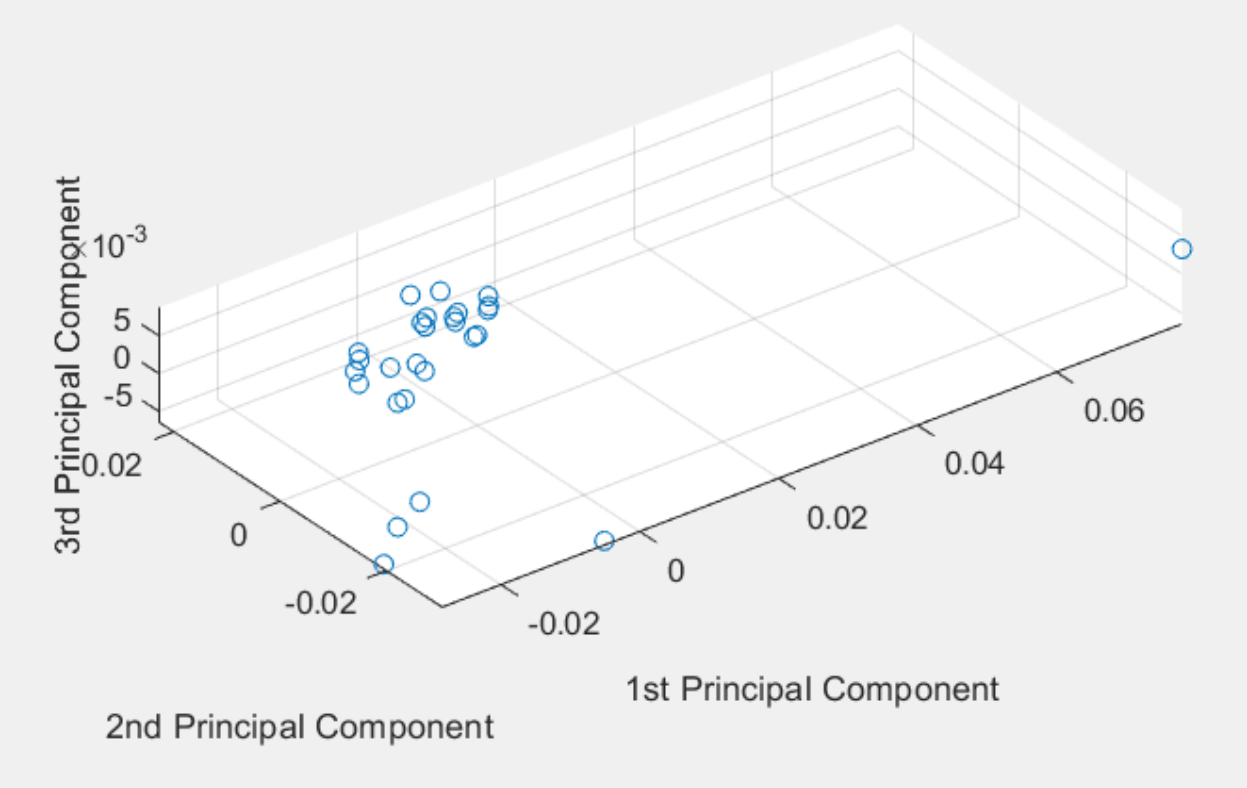

Fig 10. Plot of 3-D train data feature vectors

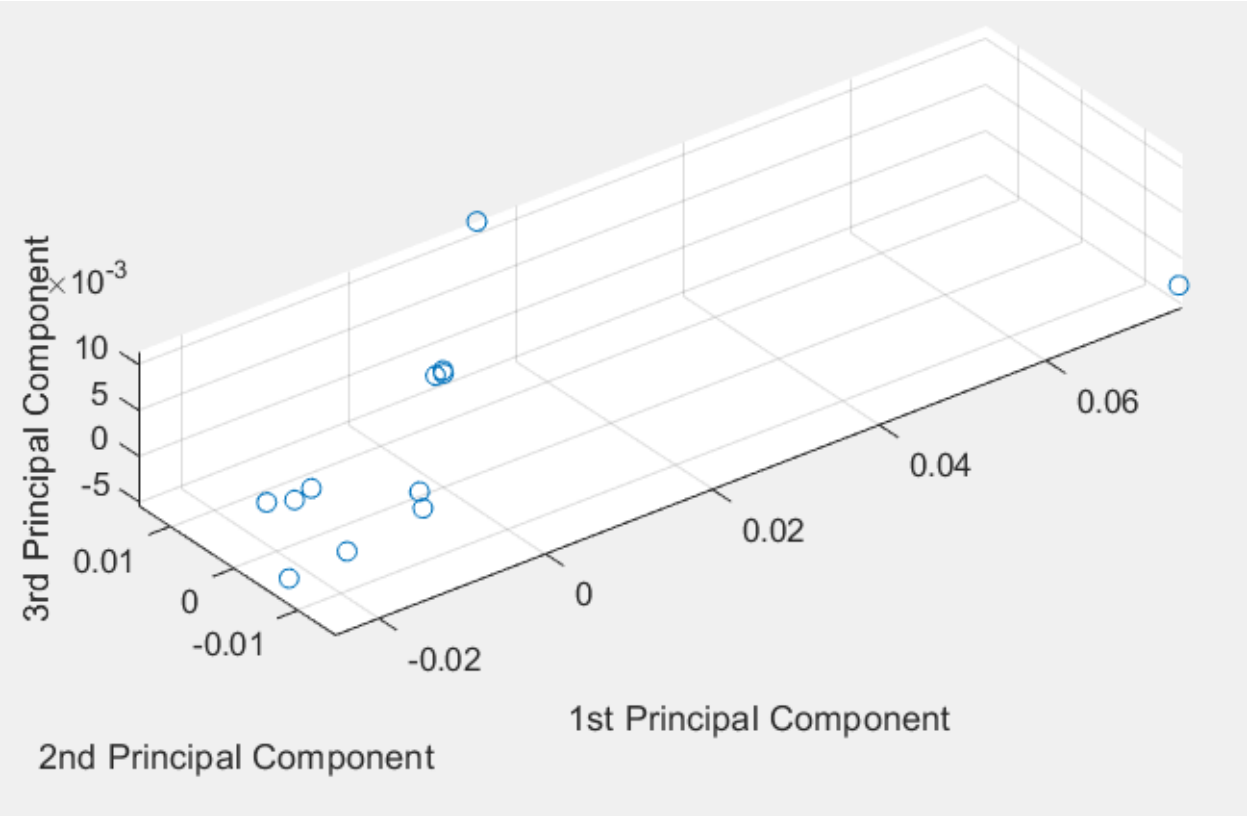

Fig 11. Plot of 3-D test data feature vectors

(b)

	15-D feature vector	3-D feature vector
Error Rate 1st	0.8056	0.5278
Error Rate 2nd	0.8056	0.6389
Error Rate 3rd	0.4167	0.5278
Error Rate 4th	0.8056	0.4444
Error Rate 5th	0.8056	0.4167

Table 1. Error rates by K-means algorithm

	SVM	Random Forest
Error Rate 1st	0.7500	0.6667
Error Rate 2nd	0.7500	0.5833
Error Rate 3rd	0.7500	0.7500
Error Rate 4th	0.7500	0.5833
Error Rate 5th	0.7500	0.6667

Table 2. Error rates by SVM and Random Forest

(c)

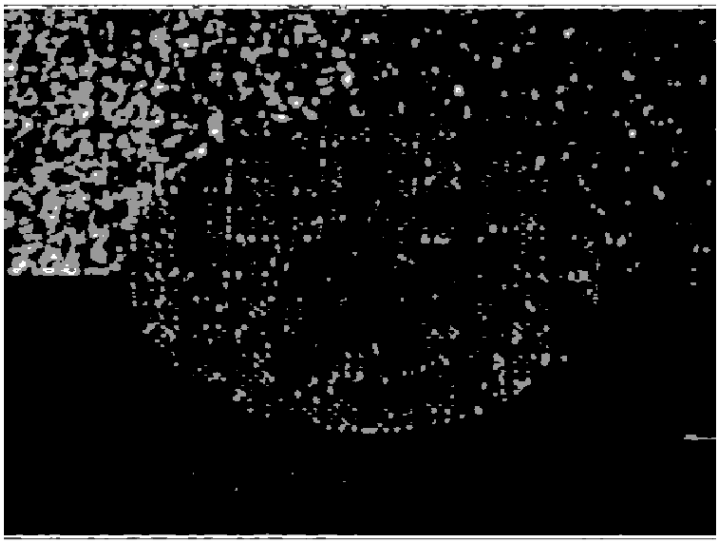

Fig 12. Segmentation image by window size 5

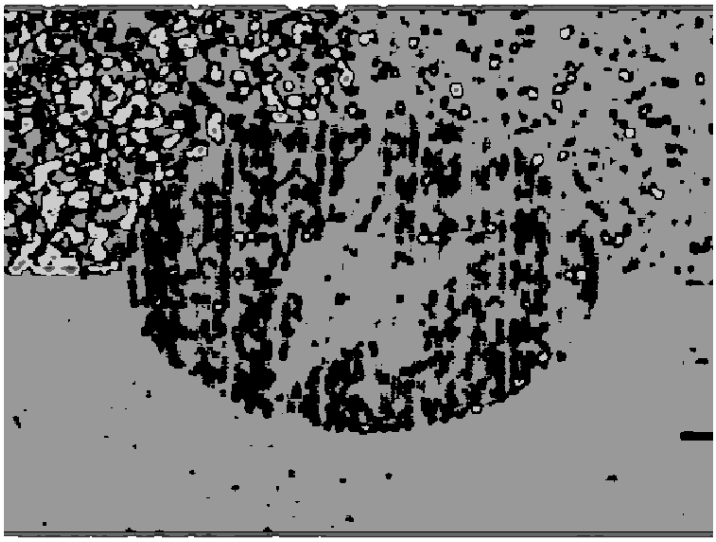

Fig 13. Segmentation image by window size 7

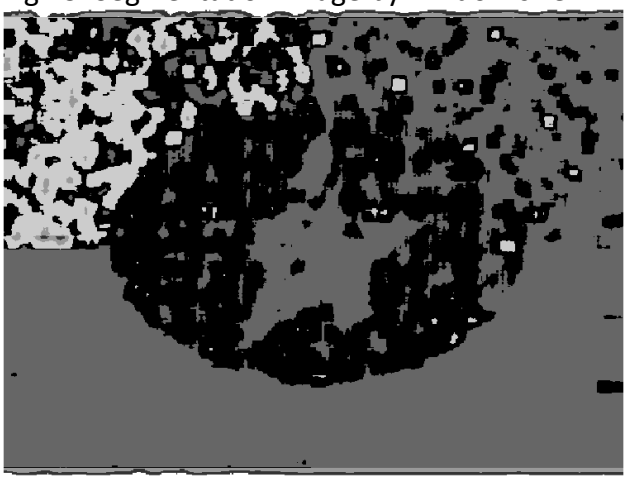

Fig 14. Segmentation image by window size 13

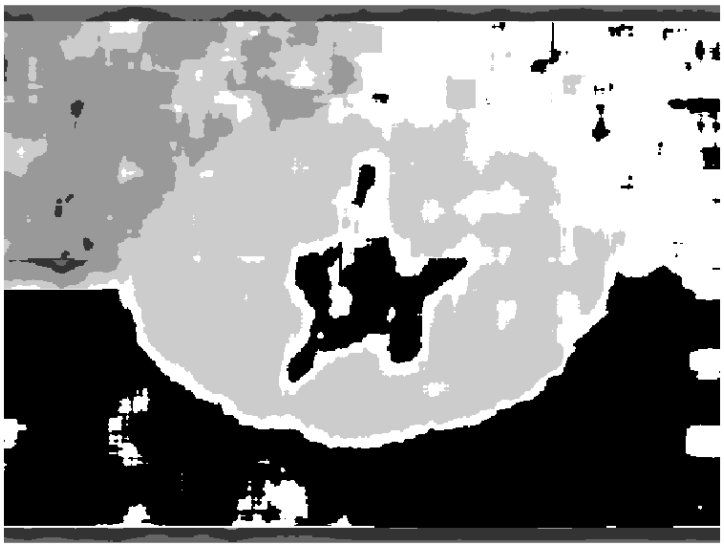

Fig 15. Segmentation image by window size 25

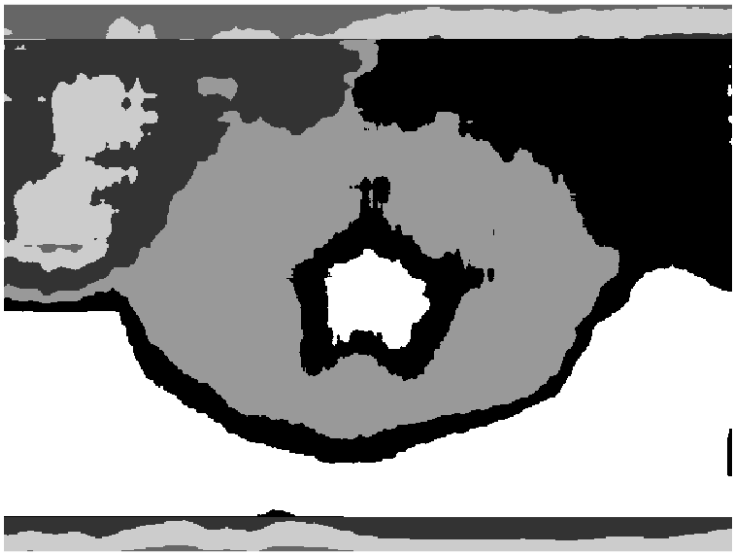

Fig 16. Segmentation image by window size 55

(d)

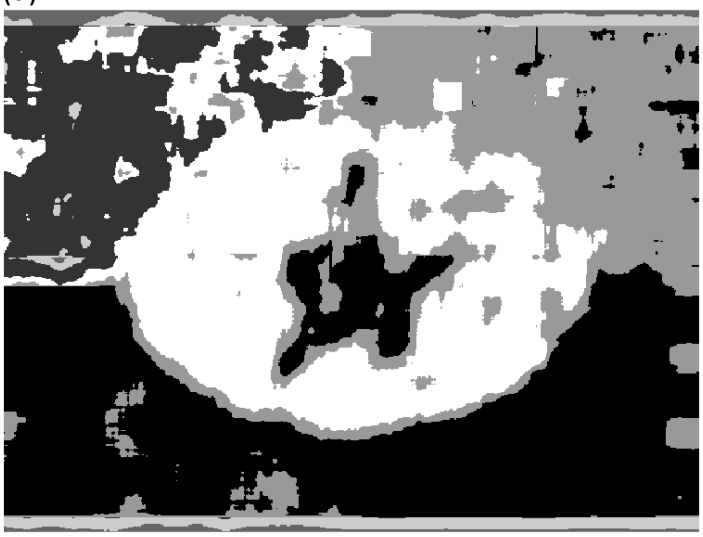

Fig 17. Segmentation image by window size 25 and PCA

1.4 Discussion

(a)

As we can observe in the plot by PCA below, there are four groups we can divide by glance. These groups of feature vectors will be highly likely classified to the separated group by machine learning algorithm. We will discuss the result at (b).

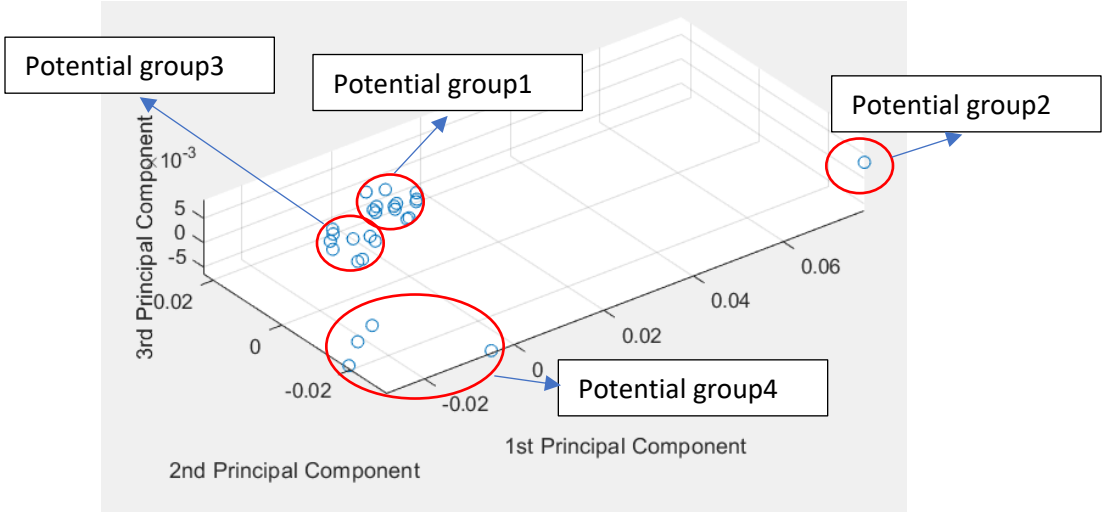

Fig 18. Potential groups in the plot generated by PCA.

(b)

For the unsupervised learning task, because of randomness of K means algorithm at initialization step, error rates keep changing. However, in general error rates with 15-D feature vectors tend to much higher than error rates with 3-D feature vectors. This means dimension reduction of feature vector help the clustering algorithm cluster unlabeled data points better. In other words, by PCA, each data point become more separable in this case.

For the supervised learning task, as we see in the table 2, prediction results by support vector machine are very consistent this is because the SVM model I used in this problem doesn't have any random process during optimization process. On the other hand, the results by random forest has some change on each run. This is because this algorithm includes random process while it grows multiple decision tree. However, its worst case is the same as SVM's result (To achieve this tendency, we need to use enough number of trees. In this case I use 60 trees). It means random forest algorithm can outperform SVM or perform equally as SVM does in general. In this problem, I didn't employ cross validation step to decide hyperparameters of each model.

(c)
We can observe that resulting image of texture segmentation gets relatively better to human eyes as window size increases. This is because the larger window can capture energy of texture better. However, this is not a general rule. As it is shown above in Fig 16 and 17, the resulting image by window size 55 seems worse than the resulting image by window size 25. It indicates that too large window size can contaminate ability of window to capture energy of different textures.

(d)
In this practice, PCA provides more separability among feature vectors meaning PCA generate more distance amongst potential clusters. It is an expected result as we have seen in problem 1 (b) at unsupervised learning practice.

2.
2.1 Abstract and Motivation
When it comes to recognition tasks such as object recognition, and face recognition, not all information is contributed for human to recognize an object or multiple object in an image. Any object has certain patterns and our brain remember them as features of the object. Later, we can recognize the same object in different scales and different angles by recalling the feature, we have found in the object. For machine as known as computer, we can extract features in several ways. In this problem, we are going to extract the feature with Scale-Invariant Feature Transform (SIFT). This feature is called Salient points meaning they are most important and noticeable points in an image. They are also well matched in environment where the scale and angle of object has changed by certain amount.

2.2 Approach and Procedures
(b)
In this problem, we want to observe that features extracted by SIFT are invariant to scale change. To achieve this goal, we can apply SIFT to Husky_1 and Husky_3 respectively. Then, compare the found features from each image and observe pairs of SIFT features of two images. We expect most pairs of the features are located on similar place. E.g. features that represents dog's eyes in Husky_3 are supposed to be close to features that represents dog's eyes in Husky_1.
Step-by-step procedure can be shown below.

Step1. Convert RGB image to grayscale image

Step2. Compute feature frame & descriptor of the image using vl_sift function in MATLAB.
e.g. [featureFrame_Husky3,descriptor_Husky3] = vl_sift(husky3Gray)

Step3. Repeat 1&2 for Husky1, Husky2, and Puppy1 images

Step4. Use nearest neighbor search to find corresponding SIFT pairs of following image pairs.
(Husky3 & Husky1), (Husky3 & Husky2), (Husky3 & Puppy1), (Husky1 & Puppy1)

Step5. Display corresponding SIFT pair with feature frames on the image using vl_plotframe function in MATLAB.

Fig . Step-by-step procedure of image matching

(c)
As we have learned, we can extract salient points (feature vectors of an image) from an image with SIFT. We can combine these extracted feature vectors with K means clustering and construct a set of codewords for the image. This set of codewords is a representation of the image and it is also known as bag of words. Each “word” corresponds to a label induced by K means algorithm. In this practice, we are going to use 8 clusters.
Step-by-step procedure can be summarized as follows.

Step1. Convert RGB image to grayscale image

Step2. Compute feature frame & descriptor of the image using vl_sift function in MATLAB.
e.g. [featureFrame_Husky3,descriptor_Husky3] = vl_sift(husky3Gray)

Step3. Repeat 1&2 for Husky1, Husky2, and Puppy1 images

Step4. Cluster descriptors(a.k.a SIFT feature vectors) of each image applying K means algorithm.

Step5. To find corresponding word(label) in other images such as Husky 1, 2 and Puppy 1 to word(label) in Husky3, calculate distance between each word. You will get 8 distances for each word in Husky 1, 2 and Puppy 1 cases. Find the word that gives shortest distance. Do this process for all 8 words in Husky 1, 2 and Puppy 1 cases. You will end up with "conversion table" that tells a word from Husky 1, 2 and Puppy 1 corresponds to which word in Husky3.

Step6. Convert word in Husky 1, 2 and Puppy 1 cases with "conversion table"

Step7. Plot a histogram of word for each image.

Fig . Step-by-step procedure of constructing a bag of words

2.3 Experimental Results (Partially open source is used)
(b)-1

	Index	Orientation in radian
Key-point with the largest scale in Husky_3	708	0.6583
Closest neighboring key-point In Husky_1	903	-3.0304

(b)-2
(i) Husky_3 and Husky_1
Number of matching points = 39

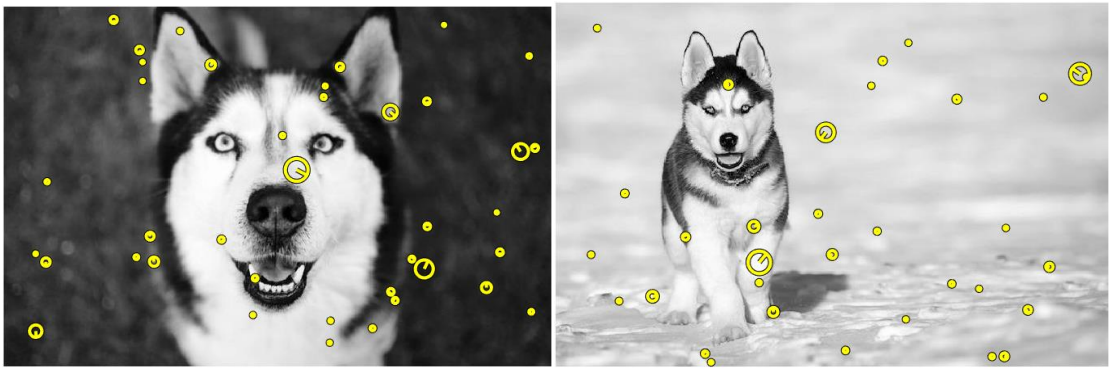

Fig . corresponding SIFT pairs found by vl_ubcmatch in MATLAB

(ii) Husky_3 and Husky_2
Number of matching points = 37

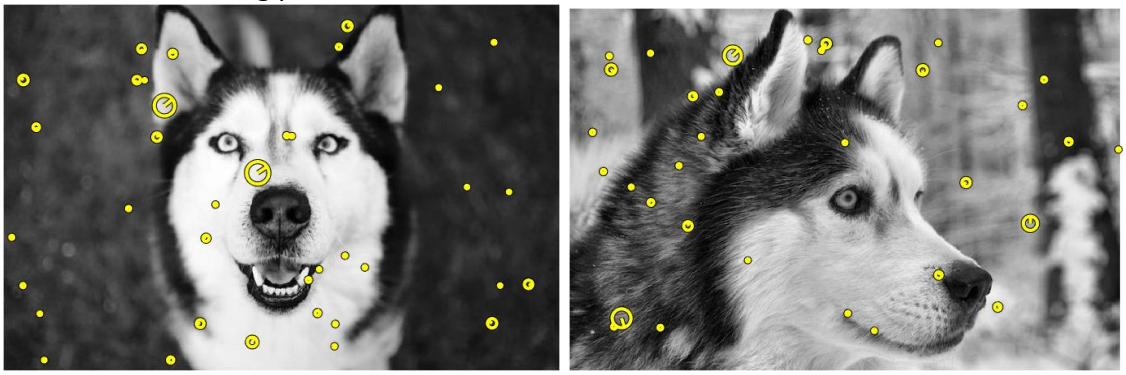

Fig . corresponding SIFT pairs found by vl_ubcmatch in MATLAB

(iii) Husky_3 and Puppy_1
Number of matching points = 36

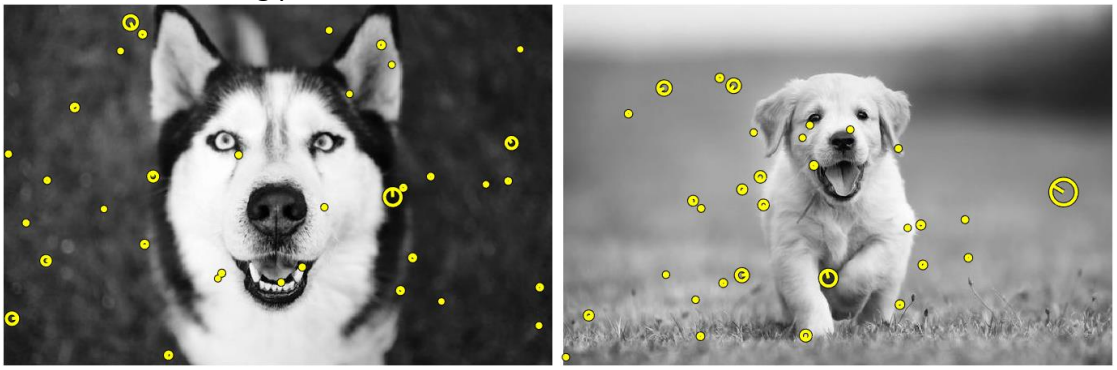

Fig . corresponding SIFT pairs found by vl_ubcmatch in MATLAB

(iv) Husky_1 and Puppy_1
Number of matching points = 46

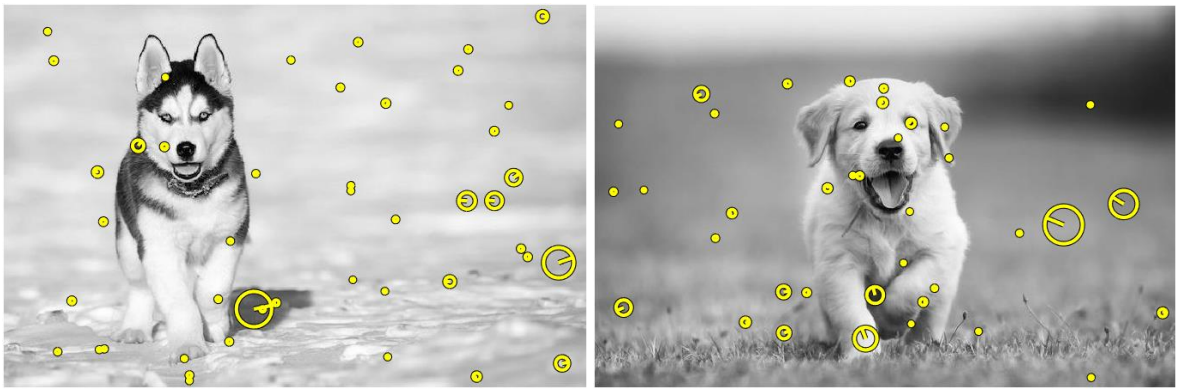

Fig . corresponding SIFT pairs found by vl_ubcmatch in MATLAB

(c)

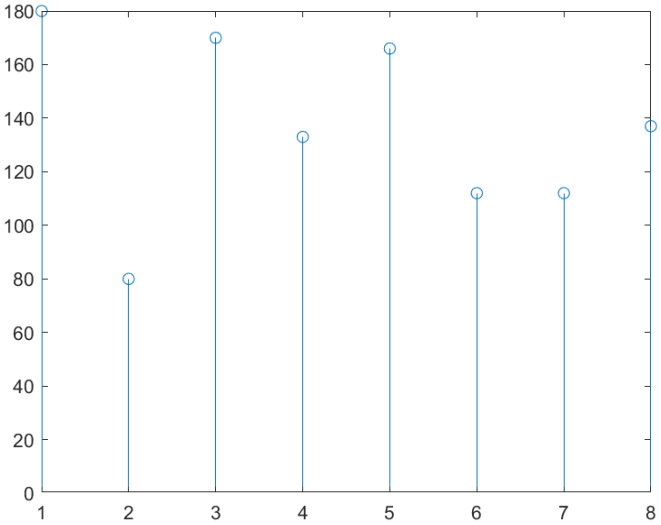

Fig. Bag of words for Husky 3 based on Husky_3

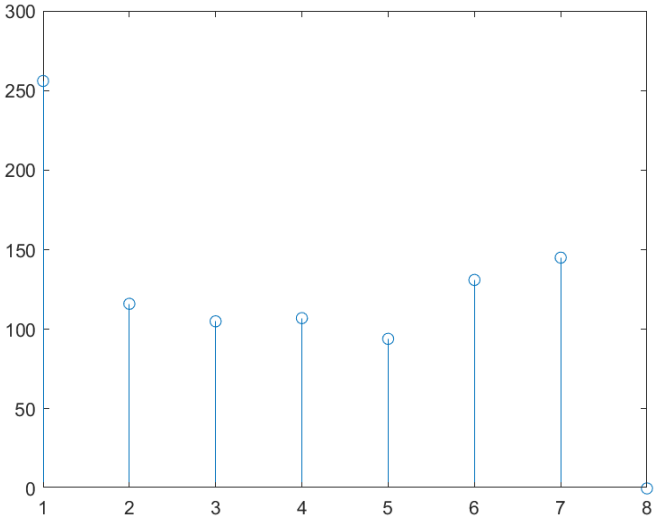

Fig. Bag of words for Husky 1 based on Husky_3

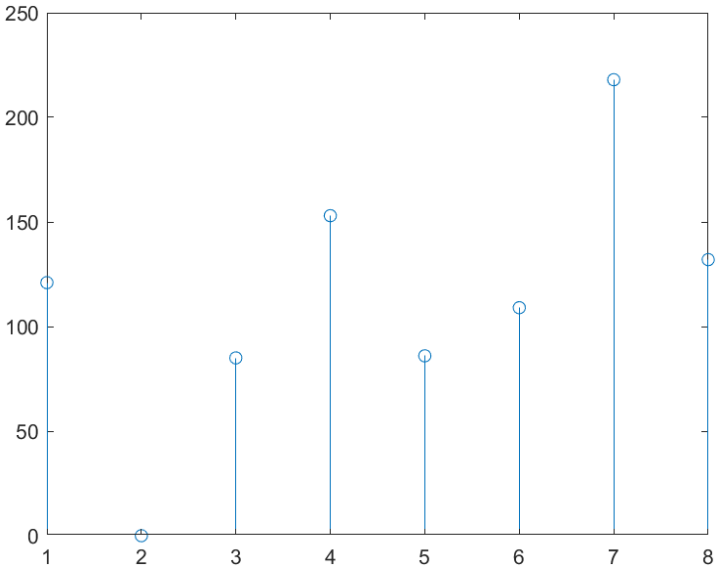

Fig. Bag of words for Husky 2 based on Husky_3

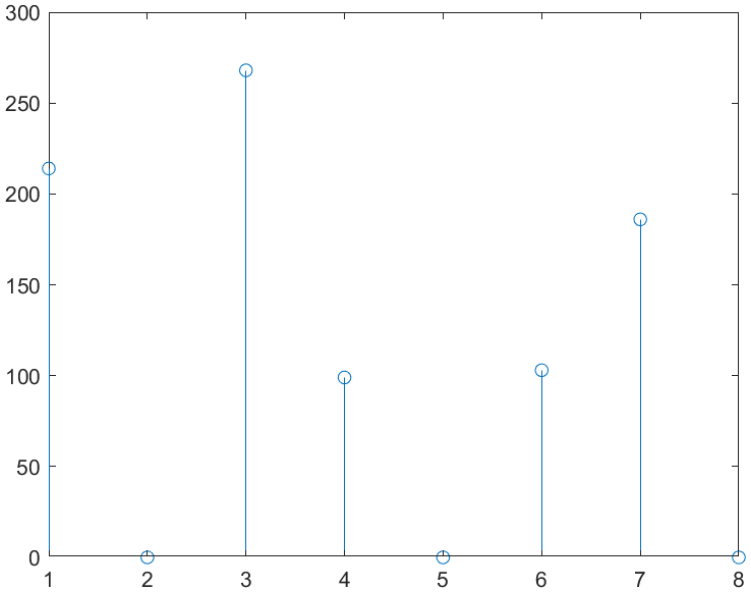

Fig. Bag of words for Puppy 1 based on Husky_3

2.4 Discussion

(a)

1. features extracted by SIFT are robust to a substantial range of affine distortion, 3D viewpoint change, and change of scale & angle. [3]
2. For each feature, operation in SIFT assign scale, orientation, and location. This provides robustness to each feature. [3]
3. In SIFT, it computes local image gradients in the region around each key-point at the certain scale. These are represented as a form that is robust to change in illumination. [3]
4. DoG is an approximation of LoG that is close enough. With DoG, we can use computation resource more efficiently. [3]
5. 4x4x8 = 128 element feature vector for each key-point

(b)

1. Orientations of matching points are quite close to each other as we expected. This is because significant feature in each photo is oriented in similar angler. E.g. dog's nose, eyes, mouth, etc.
2. For finding and displaying SIFT pairs task, we can pay attention to number of matching pairs. First of all, for Husky_3 & Puppy_1 pair, it has the least number of matching points (36) because of there is relatively big difference in terms of scale and they are totally different objects. Second, for Husky_3 & Husky2 pair, it has 37 matching points. It wins the third place because two same objects have big difference in angle. Husky_3 and Husky_1 wins the second place since they have only difference in scale. Features extracted by SIFT is relatively more robust to change in scale. Last, Husky1 & Puppy1 wins the first place with matching points 46. This is tricky property of SIFT. They have many matching points compared to other pairs since it has similar scale, angle, and background even though they are different objects.

(c)

In bag of words practice, we can observe that centroid 2, 5, and 8 do not appear in Puppy_1, but their frequencies are quite high in Husky_3. This is because there are scale difference and they are different object. For the Husky_2 and Husky_3. Only centroid 2 does not appear in Husky_2. This is caused by difference in angle, but they are the same objects. Between Husky_1 and Husky_3, only centroid 8 is not observed. This could be caused by scale difference. Other centroids are well distributed in general.

Additionally, the results of histograms can be different on each run because of randomness of K means algorithm.

References

[1] C.-C. Jay Kuo, “EE 569: Homework #4,” [Course Material]. Available for students in the course:

https://courses.uscden.net/d2l/le/content/17205/viewContent/301652/View?ou=17205

[2] Min Zhang, “EE569_Disuccsion_WK8_030620,” [Course Material]. Available for students in the course:

https://courses.uscden.net/d2l/le/content/17205/viewContent/302004/View?ou=17205

[3] David G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, 60(2), 91-110, 2004.