EE569 Digital Image Processing

HOMEWORK # 4

Aagam Shah

USC ID-8791018480

USC Email – aagamman@usc.edu

Submission Date – March 22, 2020

Problem 1: Texture Analysis and Segmentation

Texture analysis is basically used to characterize the different types of textures present in an image by analyzing the texture qualities of a region of interest. Based upon this analysis of texture we categorize different regions in an image which is nothing but known as Texture Segmentation. So, basically in this problem it is asked to do the texture analysis by performing the texture classification followed by texture segmentation. Texture analysis finds its application in areas of interest such as satellite imaging, biomedical, computer vision, industrial inspection, etc.

(a) Texture Classification – Feature Extraction:

1. ABSTRACT & MOTIVATION

Texture classification is the process in which variety of textures present in an image are categorized by implementing the Texture Classification algorithm. Texture Classification technique is a two-stage process. The first stage is the training stage and the second stage is the realization stage. In the first stage we train the model for each type of texture content present in an image. Using this model, we classify the texture labels of test images which is nothing but the realization stage i.e., the second stage of texture classification. Various types of textures present in an image are analyzed in order to do improve the further processing required for image segmentation.

2. APPROACH & PROCEDURE

The flow of the Texture Classification algorithm can be discussed as follows:

Step 1: The first step for **Feature Extraction** is the **Pre-processing** step in order to reduce the illumination effects. In this step we basically first convert the input image in the double data format for the calculation convenience. This double formatted image now serves as the new input image. To remove the illumination effects from the given input image we need to eliminate the DC component from it. This can be done by calculating the mean of an entire image (averaging all the pixels) and then subtracting this mean from each pixel of the given input image. The reason behind performing this step is to eliminate the high frequency component that might affect the feature extraction of an image. The basic formula to remove this DC component can be given as follows:

New_input_image (i,j) = Input_image (i,j) – (Mean of all the pixels of an input image)

Step 2: The next step is we perform the **Boundary Extension** for each test as well as train images. In order to work with 5x5 Law Filters we need to extend two rows and two columns on all the sides of an input image.

Step 3: Then we generate twenty-five 5x5 Laws Filters formulated by the tensor product of the five 1-dimensional kernels used for feature extraction. These twenty-five 5-dimensional kernels are applied to all the boundary extended test as well as train images and all these kernels are convolved with the input images in order to calculate the energy feature vector for an input image. The five 1-D Law Filters can be given as follows:

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]

The twenty-five 5x5 Laws Filters can be obtained by vector multiplication as:
BF = K^T * K where K can be any of the five 1-D kernels. All the twenty-five pairs can be given as follows:

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

Each pair has its own significance of feature detection. Each kernel is responsible for pattern detection in different directions and orientations.

Step 4: In this step we perform the **Feature Averaging** operation. So basically, here we average out the energy of all the image pixels for each feature vector for every input image by generating a response map after performing the convolution of the boundary extended image and the Laws Filters. As the response map will contain both the positive as well as the negative values in it so we take the absolute value of it for averaging. The formula for calculating the energy for each filter response is as follows:

$$Energy = \frac{1}{Image\ Width\ \times Image\ Height}\sum_{i=0}^{N}\sum_{j=0}^{M}(I(i,j))^2$$

So, following in a similar fashion we get 25-D feature vector for each input image.

Step 5: Now, in order to reduce the computational complexity, we can combine the symmetric pairs of the feature vectors and average them out such as, we can reduce the 25-D energy feature vector to 15-D energy feature vector. This can be done by combining the below given pairs and taking the average of them and representing them as a single Law Filter. The reduced 15-D pairs can be given as follows:

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

Step 6: Now here we perform the **Feature Reduction** operation. So, in this step we further reduce the dimension of the Feature vector from 15-D to 3-D using the **Principal Component Analysis (PCA)**. The goal behind the usage of principal component analysis (PCA) technique is to reduce the computational complexity and to work with the data of reduced dimension in order to make it easy to understand and analyze. PCA can be performed either by calculating the eigen value and eigen vectors or by SVD (singular vector decomposition). The steps to perform PCA are as follows:

(i) Firstly, we consider the Final energy matrix of (n x m) dimensions, where n = number of features and m = number of images.

(ii) Then we standardize the above matrix using the inbuilt *zscore* command of MATLAB and store the new standardized matrix in X (Feature Matrix).

(iii) Next, we calculate the covariance matrix by multiplying the transpose of the Feature matrix with the Feature Matrix and then normalizing it by dividing it by the total number of images (m).

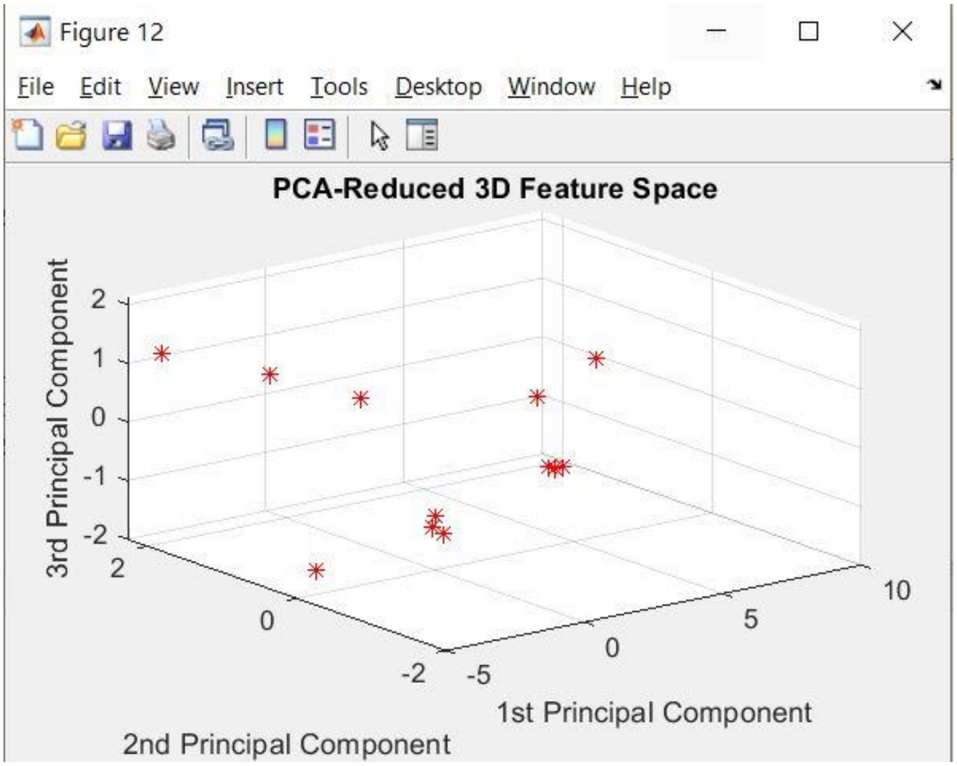
(iv) In this step we compute the SVD of the covariance matrix to get the Singular Matrix and the left and the right singular vectors.

(v) Here we perform the dimensionality reduction i.e., reducing the 15-D vector to 3-D. This is done by first sorting the singular values in descending order and corresponding to that we consider only those left singular vectors. So, here we consider only the first three columns of the U matrix (left singular matrix).

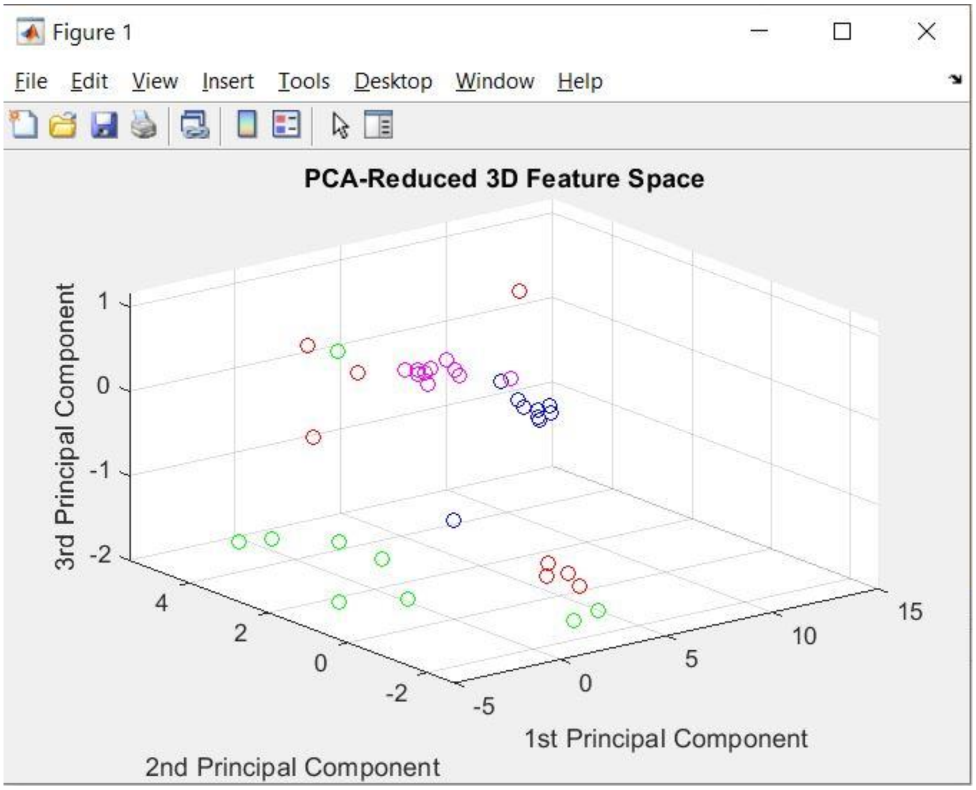
(vi) Lastly, in the Transformation step we obtain the reduced dimensionality matrix (3-D) by multiplying the transpose of the reduced U matrix with the transpose of the X (feature matrix).

3. EXPERIMENTAL RESULTS

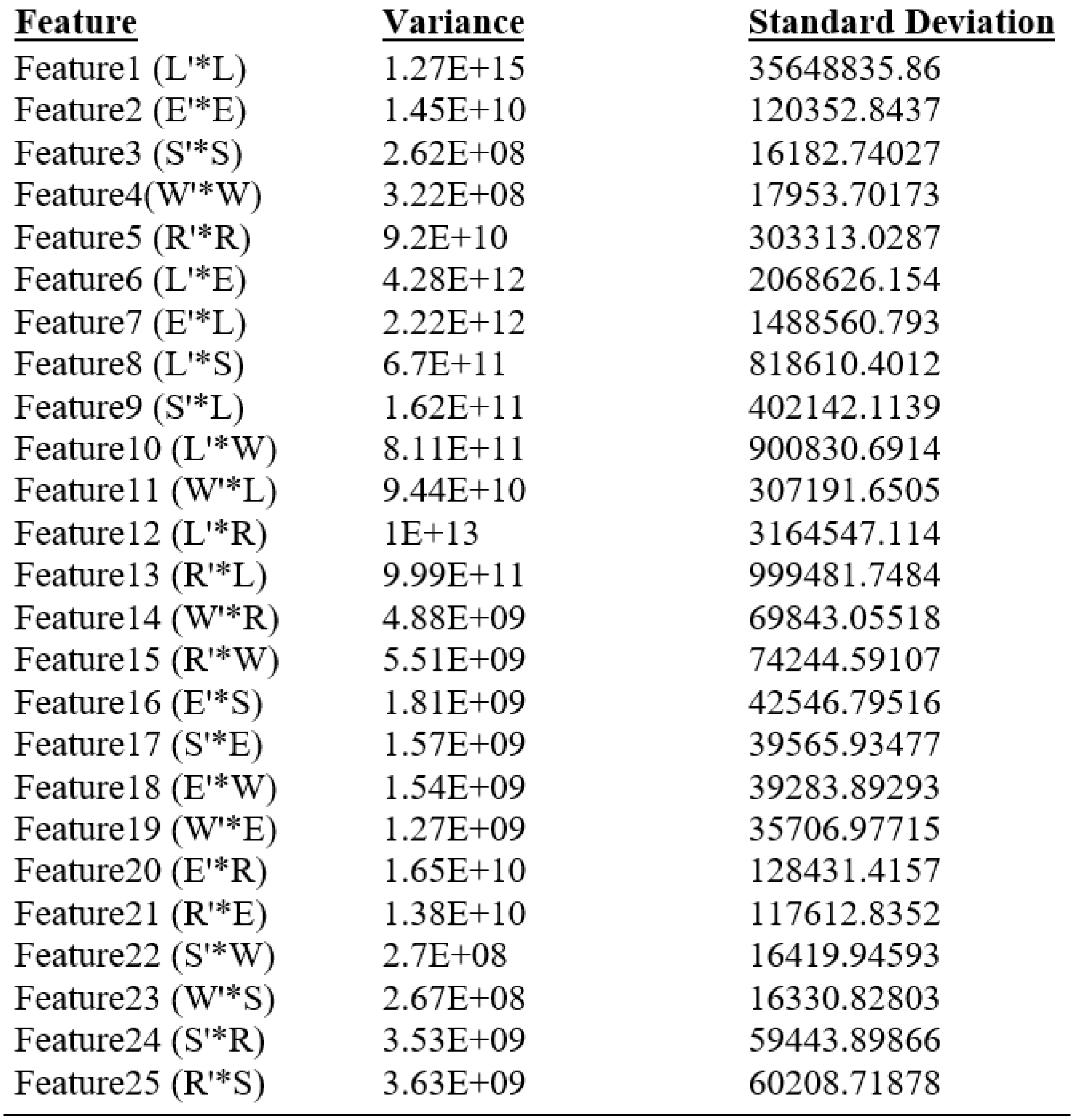
PCA Plot for test dataset:



PCA Plot for train dataset:



4. DISCUSSION



Feature	Variance	Standard Deviation
Feature1 (L'*L)	1.27E+15	35648835.86
Feature2 (E'*E)	1.45E+10	120352.8437
Feature3 (S'*S)	2.62E+08	16182.74027
Feature4(W'*W)	3.22E+08	17953.70173
Feature5 (R'*R)	9.2E+10	303313.0287
Feature6 (L'*E)	4.28E+12	2068626.154
Feature7 (E'*L)	2.22E+12	1488560.793
Feature8 (L'*S)	6.7E+11	818610.4012
Feature9 (S'*L)	1.62E+11	402142.1139
Feature10 (L'*W)	8.11E+11	900830.6914
Feature11 (W'*L)	9.44E+10	307191.6505
Feature12 (L'*R)	1E+13	3164547.114
Feature13 (R'*L)	9.99E+11	999481.7484
Feature14 (W'*R)	4.88E+09	69843.05518
Feature15 (R'*W)	5.51E+09	74244.59107
Feature16 (E'*S)	1.81E+09	42546.79516
Feature17 (S'*E)	1.57E+09	39565.93477
Feature18 (E'*W)	1.54E+09	39283.89293
Feature19 (W'*E)	1.27E+09	35706.97715
Feature20 (E'*R)	1.65E+10	128431.4157
Feature21 (R'*E)	1.38E+10	117612.8352
Feature22 (S'*W)	2.7E+08	16419.94593
Feature23 (W'*S)	2.67E+08	16330.82803
Feature24 (S'*R)	3.53E+09	59443.89866
Feature25 (R'*S)	3.63E+09	60208.71878

- Technically speaking, the feature with the least variance or standard deviation has the strongest discriminant power because the data points in the cluster are easily identifiable and are less spread in the feature space i.e., (S5S5).
- On the other hand, the feature with the maximum variance or standard deviation has the weakest discriminant power because the data points in the cluster are not easily identifiable as they are more widely spread in the feature space thus making it difficult to categorize that it will belong to which cluster i.e., (L5L5).

(b) Advanced Texture Classification – Classifier Explore:

EE569 Digital Image Processing

1. ABSTRACT & MOTIVATION

Advanced Texture Classification is the improvised version of the basic texture classification technique. Here in this approach we implement the machine learning algorithms which aid in self learning and automatic classification of textures of an image. It is basically the application of artificial intelligence systems and advanced approach of texture classification. I implemented it by both unsupervised as well as supervised learning processes. Unsupervised learning algorithm includes k-means clustering technique and the supervised learning algorithm includes the Random Forest Classifier (RF) and Support Vector Machine (SVM) Technique. Both the approaches are explained in detail in the following section.

2. APPROACH & PROCEDURE

The flow for the Advanced Texture Classification is as follows:
Step 1: The first step for Feature Extraction is the **Pre-processing** step in order to reduce the illumination effects. In this step we basically first convert the input image in the double data format for the calculation convenience. This double formatted image now serves as the new input image. To remove the illumination effects from the given input image we need to eliminate the DC component from it. This can be done by calculating the mean of an entire image (averaging all the pixels) and then subtracting this mean from each pixel of the given input image. The reason behind performing this step is to eliminate the high frequency component that might affect the feature extraction of an image. The basic formula to remove this DC component can be given as follows:

New_input_image (i,j) = Input_image (i,j) – (Mean of all the pixels of an input image)

Step 2: The next step is we perform the **Boundary Extension** for each test as well as train images. In order to work with 5x5 Law Filters we need to extend two rows and two columns on all the sides of an input image.

Step 3: Then we generate twenty-five 5x5 Laws Filters formulated by the tensor product of the five 1-dimensional kernels used for feature extraction. These twenty-five 5-dimensional kernels are applied to all the boundary extended test as well as train images and all these kernels are convolved with the input images in order to calculate the energy feature vector for an input image. The five 1-D Law Filters can be given as follows:

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]

The twenty-five 5x5 Laws Filters can be obtained by vector multiplication as:
BF = K^T * K where K can be any of the five 1-D kernels. All the twenty-five pairs can be given as follows:

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

Each pair has its own significance of feature detection. Each kernel is responsible for pattern detection in different directions and orientations.

Step 4: In this step we perform the **Feature Averaging** operation. So basically, here we average out the energy of all the image pixels for each feature vector for every input image by generating a response map after performing the convolution of the boundary extended image and the Laws Filters. As the response map will contain both the positive as well as the negative values in it so we take the absolute value of it for averaging. The formula for calculating the energy for each filter response is as follows:

$$Energy = \frac{1}{Image\ Width\ \times Image\ Height} \sum_{i=0}^{N} \sum_{j=0}^{M} (I(i,j))^2$$

So, following in a similar fashion we get 25-D feature vector for each input image.

Step 5: Now, in order to reduce the computational complexity, we can combine the symmetric pairs of the feature vectors and average them out such as, we can reduce the 25-D energy feature vector to 15-D energy feature vector. This can be done by combining the below given pairs and taking the average of them and representing them as a single Law Filter. The reduced 15-D pairs can be given as follows:

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

Step 6: Now here we perform the **Feature Reduction** operation. So, in this step we further reduce the dimension of the Feature vector from 15-D to 3-D using the **Principal Component Analysis (PCA)**. The goal behind the usage of principal component analysis (PCA) technique is to reduce the computational complexity and to work with the data of reduced dimension in order to make it easy to understand and analyze. PCA can be performed either by calculating the eigen value and eigen vectors or by SVD (singular vector decomposition). The steps to perform PCA are as follows:

(vii) Firstly, we consider the Final energy matrix of (n x m) dimensions, where n = number of features and m = number of images.

(viii) Then we standardize the above matrix using the inbuilt *zscore* command of MATLAB and store the new standardized matrix in X (Feature Matrix).

(ix) Next, we calculate the covariance matrix by multiplying the transpose of the Feature matrix with the Feature Matrix and then normalizing it by dividing it by the total number of images (m).

(x) In this step we compute the SVD of the covariance matrix to get the Singular Matrix and the left and the right singular vectors.

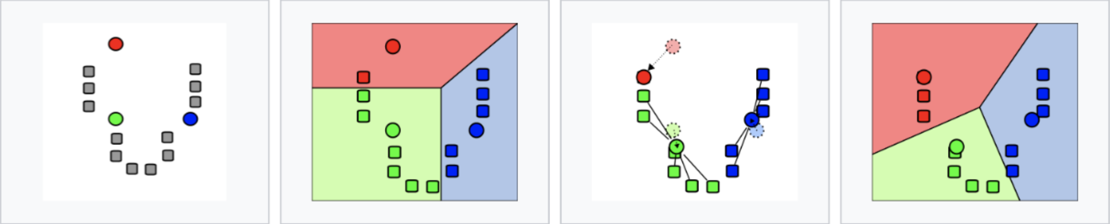
(xi) Here we perform the dimensionality reduction i.e., reducing the 15-D vector to 3-D. This is done by first sorting the singular values in descending order and corresponding to that we consider only those left singular vectors. So, here we consider only the first three columns of the U matrix (left singular matrix).

(xii) Lastly, in the Transformation step we obtain the reduced dimensionality matrix (3-D) by multiplying the transpose of the reduced U matrix with the transpose of the X (feature matrix).

Step 7: Now for Advanced Texture Classification we perform the unsupervised and supervised learning. Unsupervised learning is performed by K-means clustering, while the supervised learning is carried out by Random Forest (RF) and Support Vector Machine (SVM) classification techniques.

Step 8: Let us discuss about the **K-means algorithm** here. Basically, K-means algorithm is an unsupervised machine learning algorithm which is used to categorize the high dimensional data in n number of clusters without the help of ground truth labels. The algorithm is basically a two-stage algorithm. The initial k-means algorithm will initialize the random center of the cluster by choosing the n different values as its center. The second stage of the algorithm or the modified version also known as the k++ means algorithm which will initialize the center of the first cluster as the mean of data value then for the next cluster it will choose its center value as the farthest data value from the previous cluster center. Due to the randomness and the unsupervised learning process k-means algorithm will yield different results every-time we run, so in order to get the best output classification we run the k-means clustering algorithm several times to choose the best output as out final result. The overview flow of the k-means clustering algorithm can be given as follows:

Demonstration of the standard algorithm



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Step 9: Now in this step we perform the supervised learning by using the 3-D feature vector of training images to train the **Random Forest Classifier (RF)** and **Support Vector Machine (SVM)** and then using these trained classifiers we predict the labels for the test dataset images. Here, we are implementing the RF and SVM techniques by using the MATLAB inbuilt functions.

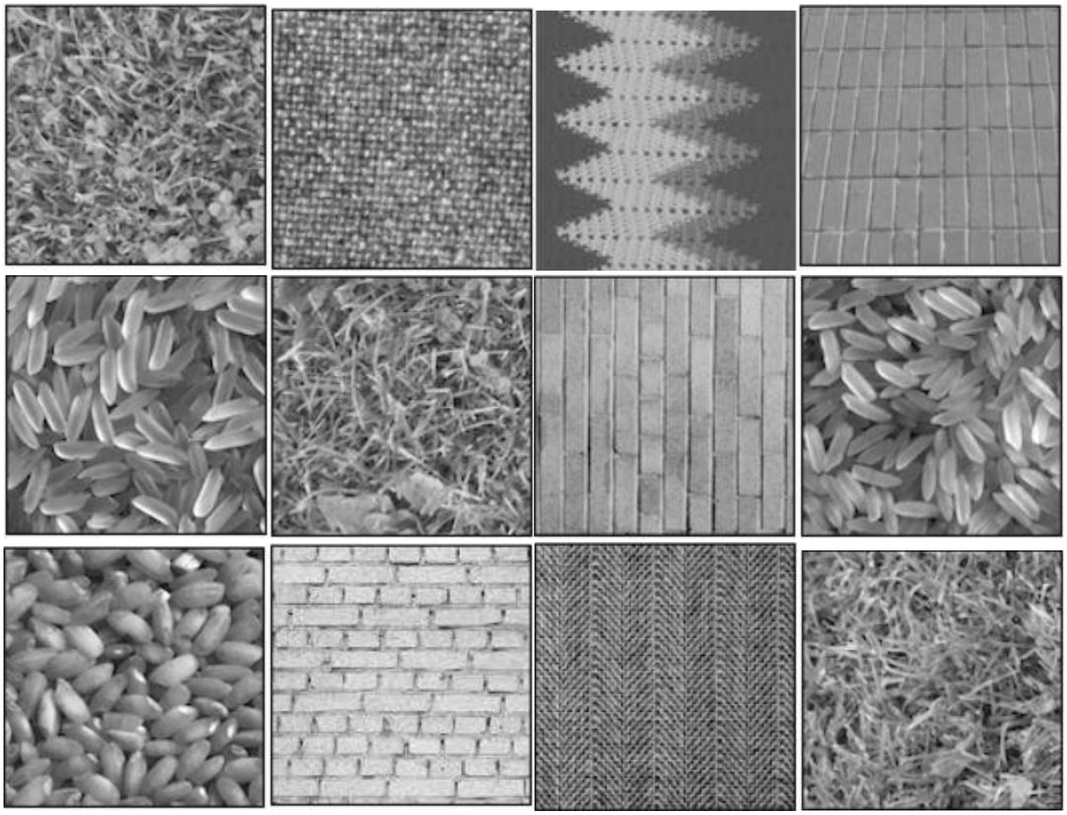
I implemented the Random Forest Classifier using the *TreeBagger* function which creates the bag of decision trees using the loaded samples of data. As from the name of the classifier itself the *TreeBagger* randomly selects the predictors at decision split. The *TreeBagger* basically works on the principle of generation of in-bag data sample by oversampling of the classes with high misclassification rate and similarly undersampling of the classes with less misclassification rate. However, for the out-bag data samples have high misclassification rate with less detections while less misclassification rate for more detections.

I also implemented the Support Vector Machine (SVM) classifier using the inbuilt MATLAB command. In this approach I initially declared the labels for all training images dataset and using that I trained my classifier using the train dataset. Upon this supervised machine learning algorithm of Support Vector Machine classifier now I use it to predict the labels of the test images and based on which I later on compute the misclassification error rate.

3. EXPERIMENTAL RESULTS

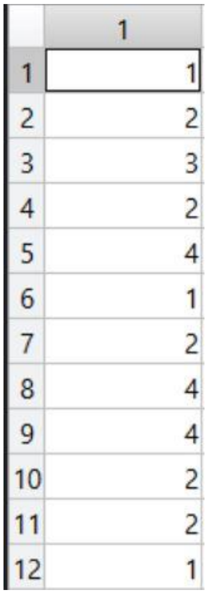
As per observation the classification of the test images can be summarized as follows:

Label	**Cluster of test image**
1 (Blanket)	2,3,11
2 (Brick)	4,7,10
3 (Grass)	1,6,12
4 (Rice)	5,8,9



Textures of Raw test images (1-12)

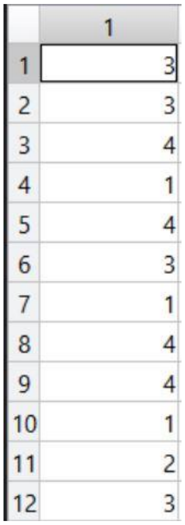
K_means Clustering for 15-D:



	1
1	1
2	2
3	3
4	2
5	4
6	1
7	2
8	4
9	4
10	2
11	2
12	1

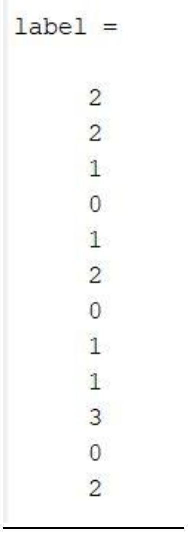
K_means Clustering for 3-D:

EE569 Digital Image Processing



	1
1	3
2	3
3	4
4	1
5	4
6	3
7	1
8	4
9	4
10	1
11	2
12	3

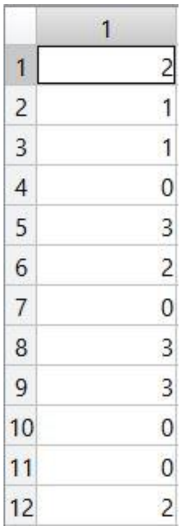
SVM output for 3-D:



```
label =

     2
     2
     1
     0
     1
     2
     0
     1
     1
     3
     0
     2
```

Random Forest Classifier for 3-D:



	1
1	2
2	1
3	1
4	0
5	3
6	2
7	0
8	3
9	3
10	0
11	0
12	2

4. DISCUSSION

In case of unsupervised learning, i.e., K means clustering the reduced dimension i.e., 3D features give almost same error rate as that of the 15D because of the low dimension dataset. Actually, the 3D dataset should give less error rate as compared to that of the 15D ideally. So, for 15-D, K-means clustering the error rate is (2/12) *100 = 16.667% and for 3-D, K-means clustering the error rate is also (2/12) *100 = 16.667%.
As seen by the eyes on observation the test images of Blanket, Brick, Grass and Rice texture resemble to corresponding test images as mentioned in the tabular form above. Based on which we can compute the classification error rate.
In case of supervised learning, i.e., 3-D feature for SVM classification we train the classifier by the training dataset and the use it for the prediction of the test images and then compute the error rate which is found that we get the 4 misclassified points so the error rate is (4/12) * 100 = 33.333%. Similarly, for 3-D feature for Random Forest classifier we train the classifier by the training dataset and the use it for the prediction of the test images and then compute the error rate which is found that we get the (1/12) * 100 = 8.333%. Hence, we observe that the Random Forest Classifier give less misclassification error and gives more accurate prediction of data as compared to that of the SVM technique. The reason may be that the RF approach uses binary tree classification method where the decision is taken at each node which increases the accuracy rather than the SVM approach which uses the computational method of hyperplanes.

(c) Texture Segmentation:

1. ABSTRACT & MOTIVATION

Segmentation in simpler terms is categorizing of the one or more regions in an image based on the similarities or dissimilarities of one or more characteristics or features. Texture Segmentation is basically the segmentation of textures present in an image based on spatial arrangement of pixels of varied intensity levels. Here we perform the region-based texture segmentation. Texture Segmentation is done based on partitioning of regions based on the texture characteristics of an image. The segmenting of textures in this approach is taken care of by the k-means clustering algorithm.

2. APPROACH & PROCEDURE

The only difference in the approach of texture segmentation as compared to that of the texture classification is that here in texture segmentation each pixel of image is segmented into different categories on the basis of energy rather than entire image. The rest of the logic and idea behind algorithm remains the same. The detailed explanation of the approach can be given as follows:

Step 1: The first step for Feature Extraction is the **Pre-processing** step in order to reduce the illumination effects. In this step we basically first convert the input image in the double data format for the calculation convenience. This double formatted image now serves as the new input image. To remove the illumination effects from the given input image we need to eliminate the DC component from it. This can be done by calculating the mean of an entire image (averaging all the pixels) and then subtracting this mean from each pixel of the given input image. The reason behind performing this step is to eliminate the high frequency component that might affect the feature extraction of an image. The basic formula to remove this DC component can be given as follows:

New_input_image (i,j) = Input_image (i,j) – (Mean of all the pixels of an input image)

Step 2: The next step is we perform the **Boundary Extension** for an input image. In order to work with 5x5 Law Filters we need to extend two rows and two columns on all the sides of an input image.

Then we generate twenty-five 5x5 Laws Filters formulated by the tensor product of the five 1-dimensional kernels used for feature extraction. These twenty-five 5-dimensional kernels are applied to all the boundary extended test as well as train images and all these kernels are convolved with the input images in order to calculate the energy feature vector for an input image. The five 1-D Law Filters can be given as follows:

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]

The twenty-five 5x5 Laws Filters can be obtained by vector multiplication as:
BF = K^T * K where K can be any of the five 1-D kernels. All the twenty-five pairs can be given as follows:

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

Step 4: In this step we generate the response map by finding the average energy of each feature vector. So basically here, we perform the **Feature Averaging** operation. So basically, here we average out the energy of all the image pixels for each feature vector for every input image by generating a response map after performing the convolution of

the boundary extended image and the Laws Filters. As the response map will contain both the positive as well as the negative values in it so we take the absolute value of it for averaging.
So, following in a similar fashion we get 25-D feature vector for each input image.

Step 5: Now, in order to reduce the computational complexity, we can combine the symmetric pairs of the feature vectors and average them out such as, we can reduce the 25-D energy feature vector to 15-D energy feature vector. This can be done by combining the below given pairs and taking the average of them and representing them as a single Law Filter. The reduced 15-D pairs can be given as follows:

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

This step yields us 15 different response maps each corresponding to a particular averaged energy feature vector for the given input image.

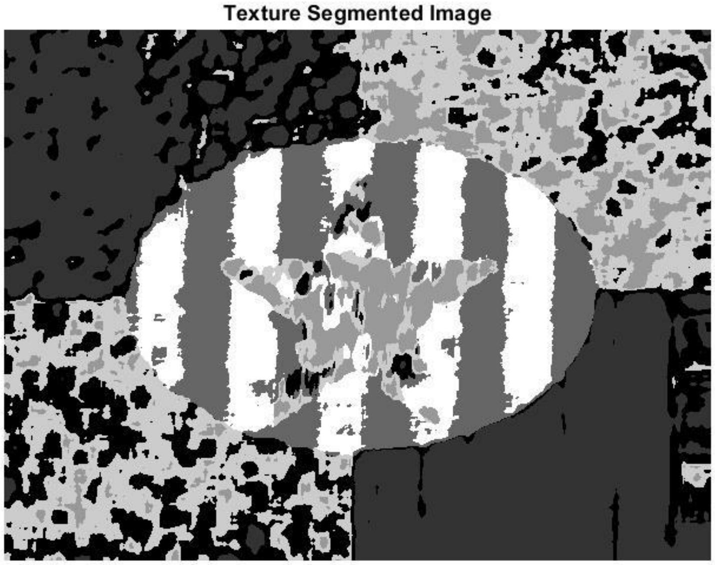
Step 6: Next we compute the **Energy Feature** using the window-based approach. In this approach we basically compute the energy measure of each pixel of each response map image obtained from the above step which will yield us a 15-D feature vector for each pixel for each response map image. Thereby this will basically generate a stack of 15 response maps of size of the input given image and each response map corresponding the one of the 15 masks generated by five 1-D kernels. Here we can vary the size of the window and check for the best desired output result. I treat each response map as a separate image and apply boundary extension condition in general (for n = any value of window size) to each response map image and compute the Energy Feature vector for each pixel for each response map image to generate the corresponding Energy Feature Matrix.

Step 7: The next following step is the **Energy Feature Normalization.** Basically, all the masks obtained from the five 1-D kernels have zero mean except the L5L5 pair. Also, feature extracted by the L5L5 filter does not play any important role either in texture classification or texture segmentation. So, we use this filter to normalize the other 14 masks. Therefore, basically here now we normalize each pixel of each energy feature matrix i.e., 14 by L5L5 filter and also put a counter to count the number of pixels for which we are operating that is basically computing the total number of pixels present in an image.

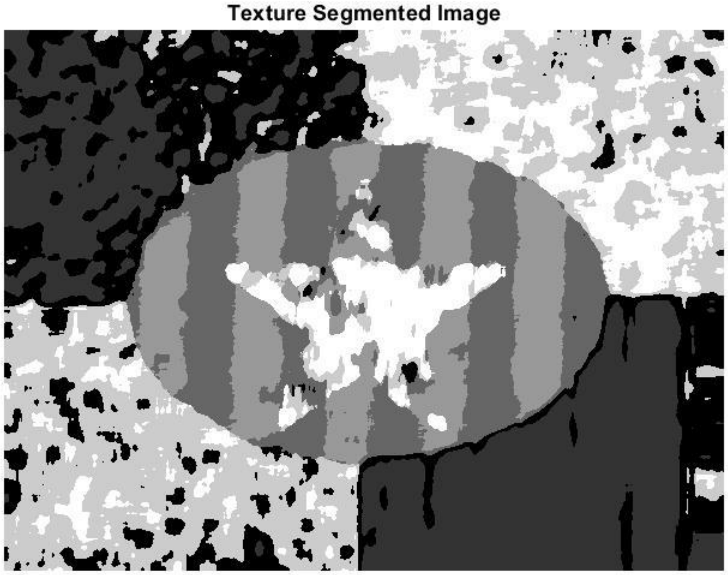
Step 8: In this final step here we perform the **Segmentation** of the composite texture images using the K-means algorithm. As here we have 6 textures so the output image should have 6 gray levels where each type of gray level will represent a particular type of texture. So, here I am using the 6 gray levels as (0, 51, 102, 153, 204, 255) which are basically equally spaced from each other. Each of these gray levels will represent a particular type of texture segmented region. Here, we again convert back the image from double format to 8-bit unsigned integer for the purpose of display.

3. EXPERIMENTAL RESULTS

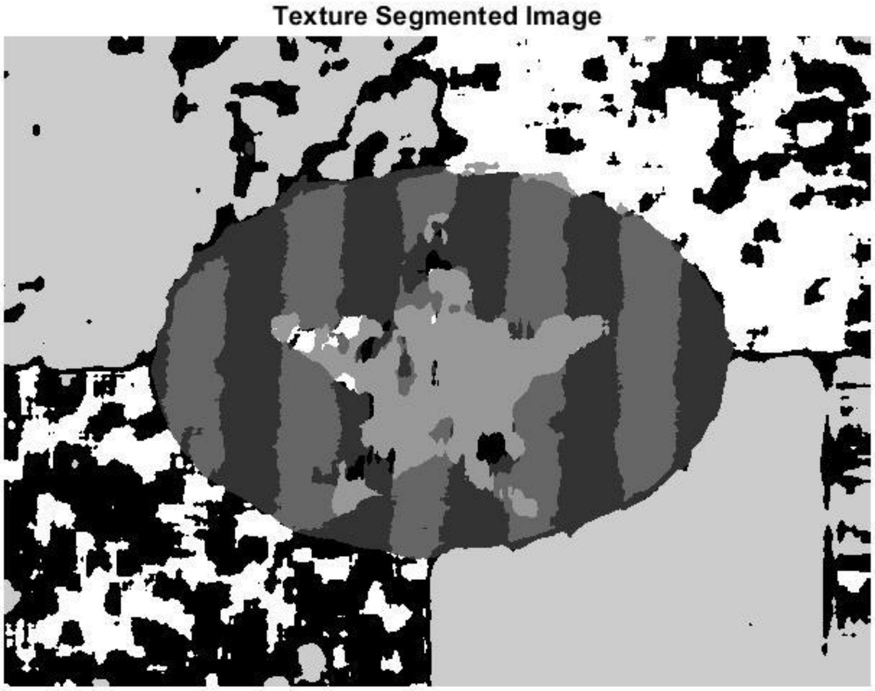
Texture Segmented Image with Window Size = 13



Texture Segmented Image with Window Size = 15



Texture Segmented Image with Window Size = 17



4. DISCUSSION

As we can observe that the we do not get exactly the correct texture segmentation results as desired which implies that the Law Filters Energy implementation is not that powerful. So, if we increase the window size and make it larger, we get more clean and distinct results, because the larger the window size then it considers the bigger patterns of textures into consideration. But this makes the process more computationally intensive and time consuming. So, the window size should neither be too small nor too big but in the optimum range which can yield the proper optimum and desired results. Because also the too large window can cover many regions of textures thereby lastly making it difficult to create texture segmentation. In my case the window size of 15 yields me the optimum output.

(d) Advanced Texture Segmentation:

1. ABSTRACT & MOTIVATION

Advanced Texture Segmentation is based on the implementation of advanced techniques in addition to the above approach of Texture Segmentation. Here we can improvise the results by implementing the Principle Component Analysis for feature reduction or doing post processing technique of merging holes or enhancing the boundaries of two adjacent regions by taking into consideration the textures of those nearby regions. As it can be concluded that texture segmentation is more complex than texture classification because in texture segmentation the various types of textures have to be identified which are present at the boundaries of two adjacent regions along-with the identification of the textures of each particular region. The major application of the Advanced Texture Segmentation is the Object Recognition.

2. APPROACH & PROCEDURE

The only difference in the approach of advanced texture segmentation as compared to that of the advanced texture classification is that here in texture segmentation each pixel of image is segmented into different categories on the basis of energy rather than entire image. The rest of the logic and idea behind algorithm remains the same. The detailed explanation of the approach can be given as follows:

Step 1: The first step for Feature Extraction is the **Pre-processing** step in order to reduce the illumination effects. In this step we basically first convert the input image in the double data format for the calculation convenience. This double formatted image now serves as the new input image. To remove the illumination effects from the given input image we need to eliminate the DC component from it. This can be done by calculating the mean of an entire image (averaging all the pixels) and then subtracting this mean from each pixel of the given input image. The reason behind performing this step is to eliminate the high frequency component that might affect the feature extraction of an image. The basic formula to remove this DC component can be given as follows:

New_input_image (i,j) = Input_image (i,j) – (Mean of all the pixels of an input image)

Step 2: The next step is we perform the **Boundary Extension** for an input image. In order to work with 5x5 Law Filters we need to extend two rows and two columns on all the sides of an input image.

Then we generate twenty-five 5x5 Laws Filters formulated by the tensor product of the five 1-dimensional kernels used for feature extraction. These twenty-five 5-dimensional kernels are applied to all the boundary extended test as well as train images and all these kernels are convolved with the input images in order to calculate the energy feature vector for an input image. The five 1-D Law Filters can be given as follows:

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]

The twenty-five 5x5 Laws Filters can be obtained by vector multiplication as:
BF = $\mathbf{K^T}$ * K where K can be any of the five 1-D kernels. All the twenty-five pairs can be given as follows:

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

<u>Step 4:</u> In this step we generate the response map by finding the average energy of each feature vector. So basically here, we perform the **Feature Averaging** operation. So basically, here we average out the energy of all the image pixels for each feature vector for every input image by generating a response map after performing the convolution of the boundary extended image and the Laws Filters. As the response map will contain both the positive as well as the negative values in it so we take the absolute value of it for averaging.
So, following in a similar fashion we get 25-D feature vector for each input image.

<u>Step 5:</u> Now, in order to reduce the computational complexity, we can combine the symmetric pairs of the feature vectors and average them out such as, we can reduce the 25-D energy feature vector to 15-D energy feature vector. This can be done by combining the below given pairs and taking the average of them and representing them as a single Law Filter. The reduced 15-D pairs can be given as follows:

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

This step yields us 15 different response maps each corresponding to a particular averaged energy feature vector for the given input image.

<u>Step 6:</u> Next we compute the **Energy Feature** using the window-based approach. In this approach we basically compute the energy measure of each pixel of each response map image obtained from the above step which will yield us a 15-D feature vector for each pixel for each response map image. Thereby this will basically generate a stack of 15 response maps of size of the input given image and each response map corresponding the one of the 15 masks generated by five 1-D kernels. Here we can vary the size of the window and check for the best desired output result. I treat each response map as a separate image and apply boundary extension condition in general (for n = any value of window size) to each response map image and compute the Energy Feature vector for each pixel for each response map image to generate the corresponding Energy Feature Matrix.

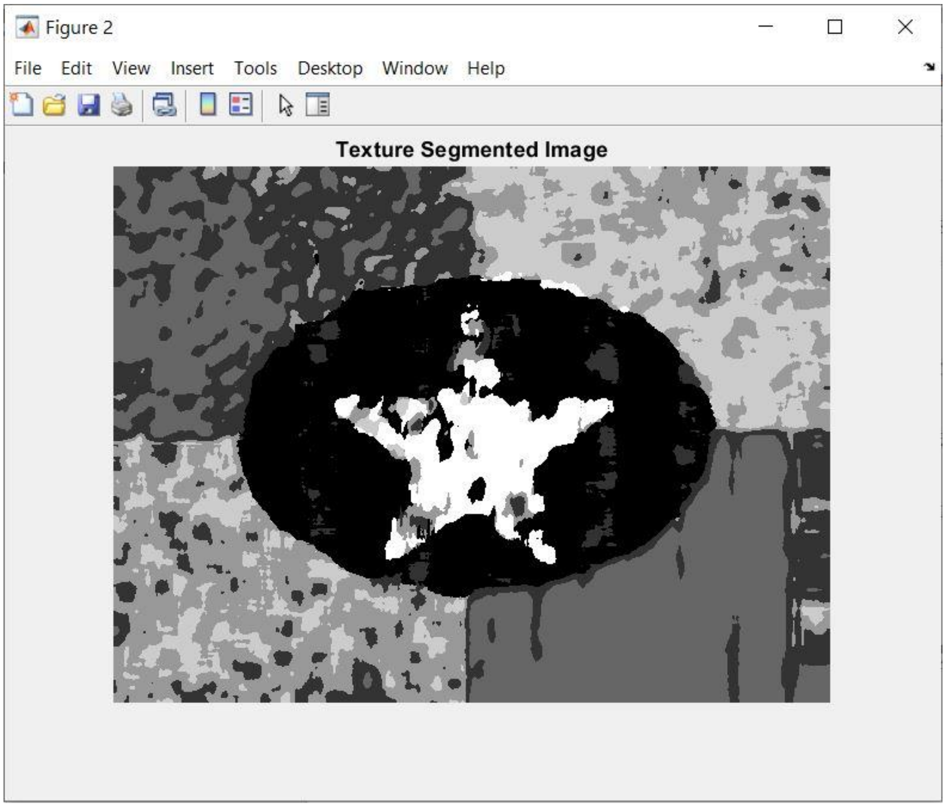
<u>Step 7:</u> The next following step is the **Energy Feature Normalization.** Basically, all the masks obtained from the five 1-D kernels have zero mean except the L5L5 pair. Also, feature extracted by the L5L5 filter does not play any important role either in texture classification or texture segmentation. So, we use this filter to normalize the other 14 masks. Therefore, basically here now we normalize each pixel of each energy feature matrix i.e., 14 by L5L5 filter and also put a counter to count the number of pixels for which we are operating that is basically computing the total number of pixels present in an image.

<u>Step 8:</u> Now moving on to the **Advanced Segmentation** part to reduce the artifacts and obtain a more desirable segmented image I followed the idea described given below. So, in order to get good segmentation results for the complicated texture image I implemented at first the Principle Component Analysis for dimension reduction because in order to work with less dimensions is more easy and manageable with less complexity and discarding the unwanted features. Later on I also implemented the post processing technique by using the dilation and erosion operation thereby using the inbuilt command of MATLAB of *imfill* holes to merge the small holes to get a better segmented result. Another idea to improve the result is we can enhance the boundaries of the two nearby regions by analyzing the texture properties of the two regions taken into consideration.

<u>Step 9:</u> In this final step here we perform the segmentation of the composite texture images using the K-means algorithm. As here we have 6 textures so the output image should have 6 gray levels where each type of gray level will represent a particular type of texture. So, here I am using the 6 gray levels as (0, 51, 102, 153, 204, 255) which are basically equally spaced from each other. Each of these gray levels will represent a particular type of texture segmented region. Here, we again convert back the image from double format to 8-bit unsigned integer for the purpose of display.

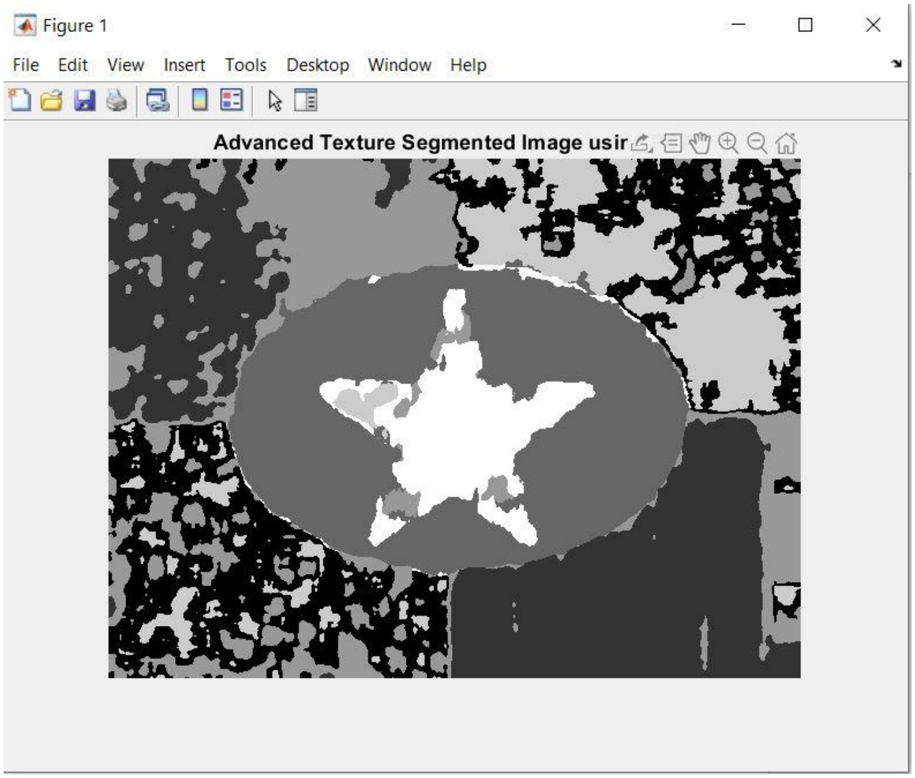
3. <u>EXPERIMENTAL RESULTS</u>

Texture Segmented Image with Window Size = 15



Texture Segmented Image

Advanced Texture Segmented Image with Window Size = 15



Advanced Texture Segmented Image usir

4. DISCUSSION

In Advanced Texture Segmentation after implementing PCA we get better results than before but not the very accurate results, because the reduced dimension though makes the computation easy but not the most optimum so it's better to determine the singular values through the SVD process.

I used *imfill* in order to fill in the holes via dilation, but this idea does not contribute much in the segmentation results because it doesn't make sense to do so without knowing the small regions in prior.

K-means clustering is done finally to improvise the results, but as it is unsupervised machine learning algorithm so to get the desired output result in the first run is hardly possible, so we run the program several times in order to get the data points more converge to the centroids and resemble to a particular cluster thereby achieving the proper segmentation results.

The result of the Advanced Texture Segmentation keeps only the prominent regions as we use the hole filling algorithm and the boundary regions are more enhanced as compared to that of the Texture Segmentation thus segmenting regions more accurately than the previous method.

Problem 2: Image Feature Extractors

(a) Salient Point Descriptor:

1. ABSTRACT & MOTIVATION

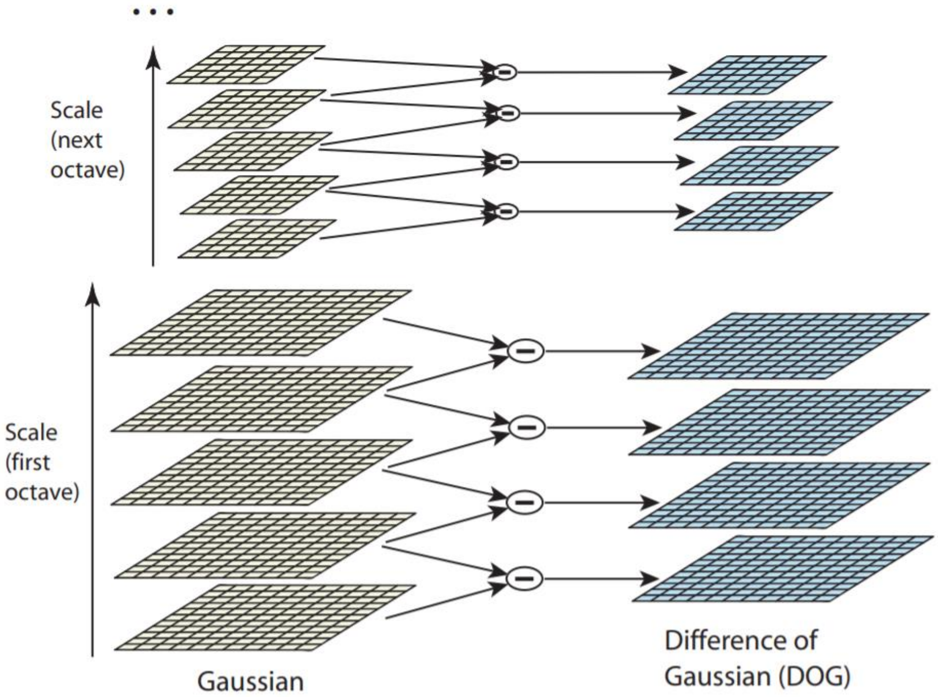
Salient Point Descriptor extraction plays and important role in digital image processing as well computer vision applications. This Salient point descriptor finds its applications in the areas of pattern recognition, object detection, image stitching, tracking and navigation, etc. in order to extract the useful information required for further processing. This is the basic technique used for SIFT and SURF feature extraction. Also, the scale invariant and the rotational invariant characteristics of the extracted features owe to the robustness of the SIFT and SURF algorithms.

2. APPROACH & PROCEDURE

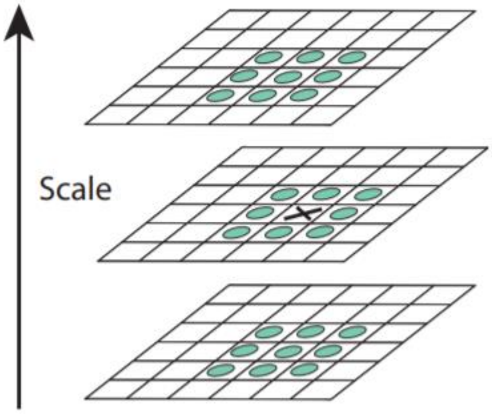
The main idea of using this Salient Point Extraction is to get the good optimum features, because a better classification accuracy is achievable only when we have good features other wise if the features are themselves not properly classified then the classifier won't get trained accurately which in turn will fail to provide the desired output. Here are the steps used for extraction of the key-descriptors of an object in an image using the SIFT algorithm is as follows:

(i) Scale-Space Extrema Detection:

The idea behind this step is to locate the potential key descriptors of an object in an image. In order to create an input image scaling and rotational invariant we basically use the Scale-Space Extrema Detection technique. It is implemented by applying Gaussian kernels in octave fashion to an input image. These octaves are also known the pyramid of images. The main idea of implementing this technique using the Gaussian kernels because they are capable of blurring the image information without blurring the edges or the boundaries of an image. So, now at first the pyramid of images is obtained by blurring an input image by different powers of k multiplied by sigma. If we increase the value of the sigma then the area of region taken into consideration is increased, thereby sigma acting as the scaling factor. So as per the paper the optimum value of the sigma is chosen as 1.6. So after obtaining the first octave we obtain the next octave by blurring the input image again after resampling it using the bilinear interpolation technique and the octave is down sampled by a factor of 2 and the also it is smoothened by a factor of k which is 1.414 as given in the paper.



The blob detection of various sizes in an image can be achieved by Laplacian of Gaussian. The small corners can be detected by the low sigma value of the Gaussian kernel and the large corners can be detected by the higher sigma values of the Gaussian kernel. The Gaussian kernels and it's derivatives are responsible for the smoothening of the scale space analysis. Followed by the Gaussian kernels now we move on to the Difference of Gaussian (DoG) which is obtained by calculating the difference of two consecutive (LoG's) of the scale space octvaes. As the computation of the DoG is easier as compared to that of the LoG we use DoG to find the local extremas of an image over space and scale and if the pixel has a local extrema than it is nothing but the potential key point. In simple terms it means that if the center point which can be either the minimum or maximum when it is compared to the other 26 neighboring points then it is considered as a potential key point.



LoG can be approximated to DoG by the given Heat equation.

Heat equation: $\frac{\partial G}{\partial \sigma} = \sigma^2 \nabla^2 G$

$$\frac{\partial G}{\partial \sigma} = \frac{G(x,y,k\sigma) - G(x,y,\sigma)}{k\sigma - \sigma}$$

Difference of Gaussian $= G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G$

Also,

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y),$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2}$$

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma). \end{aligned}$$

(ii) Key Point Localization:

After deciding the key points, they need to be now redefined by eliminating the outliers from an image. These outliers can be either edges or low contrast points.

As these outliers may have some redundant information which need to be eliminated with the help of DoG. The steps for elimination are as follows:

- In order to eliminate these outliers, we use the Taylor Series Expansion for scale space extremas where the threshold is set to 0.03 which means that the points with intensity values less than the threshold are discarded, given by the following equation:

$$D(\hat{X}) = D + \frac{1}{2}\frac{\partial D^T}{\partial X}\hat{X}$$

- In order to eliminate the redundant edges we implement the Hessain matrix which is basically used to detect the curvatures and discard those edges which does not comprise any of the salient feature in it although being a part of an edge. As the Hessain matrix comprises of the larger as well as the smaller Eigen values in it so we take the ratio of the largest eigen value to that of the smallest in order to discard the unwanted edge points.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

If the ratio of $\frac{(Dxx + Dyy)^2}{DxxDyy - Dxy^2} > \frac{(r+1)^2}{r}$ where r=10 for SIFT, then discard the candidate key-points.

- So now we take the derivative of DoG as

$$D(X) = D + \frac{\partial D^T}{\partial X}X + \frac{1}{2}X^T\frac{\partial^2 D}{\partial X^2}X$$

where the location of the maxima or minima is given as:

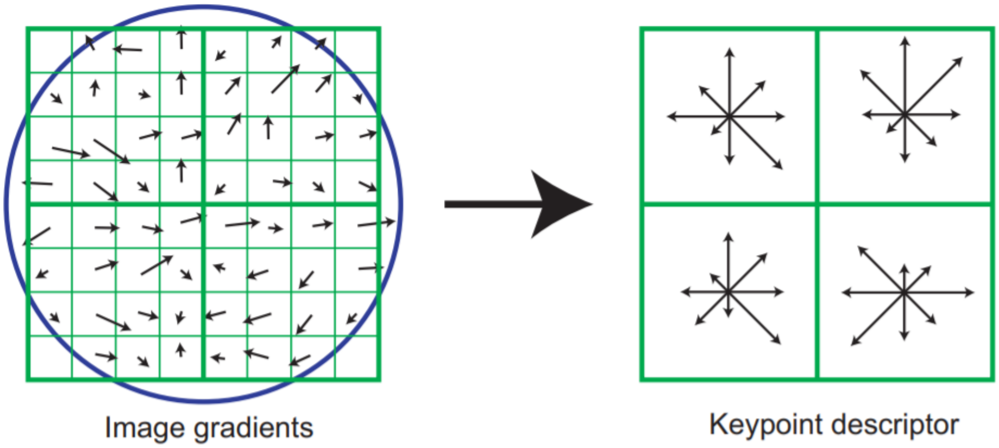
$$\hat{X} = -\frac{\partial^2 D^{-1}}{\partial X^2}\frac{\partial D}{\partial X}$$

The Value of D(X) at minima/maxima must be large $| D(X) | >$ threshold

So basically, this the way the edges and the points of low contrast are discarded, and the points of interest are retained.

(iii) Orientation Assignment:

In order to make an image rotation invariant the value or orientation at each key point is taken which can be obtained by the neighboring point of interest. This can be achieved by first choosing the neighboring key point based on the scale and then calculating the gradient magnitude as well as the direction. So, for the variation of every 10° orientation the magnitude is computed which corresponds to the 1 bin of gradient magnitude. So, in this way there will be 36 bins for 360° rotations. The orientation is then weighted by the gradient magnitude and the Gaussian window of size 4x4. Now we plot the histogram of gradient magnitude vs 36 bins followed by the calculation of the maximum peak of the histogram. So, if there is any peak whose value is higher than the 80% of the maximum value then we consider it for the orientation computation. So, this process basically generates the key points with different orientations but of the same scale.



(iv) Key Point Descriptors:

The last step is to create the key point descriptors. As given in the paper that the histogram plotted in the above step is effective in reducing the illumination effects and the differences in the angles. So basically, here along the key point orientation, we have 16x16 neighborhood around the pixel of interest, which is divided into 4x4 blocks and then the histogram of each block is calculated to generate the 8 bins of gradient orientations. So, in this manner there will be 16 such blocks on whom we perform the same operation. So, in this way we obtain (16x8) bins of histogram for every 16x16 neighborhood. Concluding, we can say that every key point is 128-D vector and it can be reduced further for easy analysis by Principle Component Analysis.

3. DISCUSSION

(i) From the paper abstract, the SIFT is robust to scaling, translation, rotation and geometric modifications as well as invariant to illumination and affine/3D projections.

(ii) SIFT achieves its robustness by as mentioned in the above answer by scaling, translation, rotation, affine/3D projection and illumination invariance. Here I explain each one of them in detail.

Scaling: The Gaussian kernel is used for smoothening which yields good results. The Pyramid of an image with different scale is build and only the key points with relevant values in every level are retained.

Translation: As seen that the gradient information regarding the local region is given by the vector, which is in turn utilized for further calculations. So basically, I mean to say that the position of this feature vector does not affect the outputs of the further computations which means that even if the key point is translated to another new position the gradient information regarding the local region won’t change.

Rotation: In order to make an image rotation invariant the value or orientation at each key point is taken which can be obtained by the neighboring point of interest. This can be achieved by first choosing the neighboring key point based on the scale and then calculating the gradient magnitude as well as the direction. So, for the variation of every 10° orientation the magnitude is computed which corresponds to the 1 bin of gradient magnitude. So, in this way there will be 36 bins for 360° rotations. The orientation is then weighted by the gradient magnitude and the Gaussian window of size 4x4. Now we plot the histogram of gradient magnitude vs 36 bins followed by the calculation of the maximum peak of the histogram. So, if there is any peak whose value is higher than the 80% of the maximum value then we consider it for the orientation computation. So, this process basically generates the key points with different orientations but of the same scale.

Illumination: At first, to make SIFT invariant to the illumination changes the gradient magnitude is used to conceal the features. Secondly, the robustness is increased by setting the threshold for the gradient magnitudes for each pixel. If the value of the gradient magnitude is higher than the threshold than it is eliminated else if the value of the gradient magnitude is less than the threshold then it is retained. Thereby, it remains invariant to the illumination changes.

Affine/3D Projection: As SIFT features are tolerant to affine/3D projections and noise so they are invariant to these changes and can bear the small number of shifts in these properties. So basically, when these SIFT features conceals the information of the local gradient in a vector, the bin size of 30° is used for orientation along-with a scale factor of 2. So, as a result the segmentation has larger amount of step width thereby giving a high amount of fault tolerance capability.

(iii) **Illumination:** At first, to make SIFT invariant to the illumination changes the gradient magnitude is used to conceal the features. Secondly, the robustness is increased by setting the threshold for the gradient magnitudes for each pixel. If the value of the gradient magnitude is higher than the threshold than it is eliminated else if the value of the gradient magnitude is less than the threshold then it is retained. Thereby, it remains invariant to the illumination changes.

(iv) The advantages that SIFT uses Difference of Gaussians instead of Laplacian of Gaussians is because DoG nearly approximates LoG and also as explained earlier in the above section that DoG requires less computation and complexity as compared to that of the LoG. So, as a result DoG reduces the execution time of an algorithm.

(v) The SIFT's output vector size in the original paper is 128 if only (4x4) grids is considered or else it is 160 if (4x4) as well as (2x2) grids are considered. It can be explained as follows, as there are two different types of scale i.e., 4x4 and 2x2 grid of locations with the radius of 8 pixels from the center location. So, (8x4x4) =128, (8x4x4) + (8x2x2) = 160.

(b) Image Matching:

1. ABSTRACT & MOTIVATION

The main idea behind Image Matching to extract the key points of one image and to match it with the key points of another image. This is done by using the Salient Key Point Descriptors using the SIFT or SURF techniques. This algorithm finds its applications in the pattern or object recognition and computer vision applications.

So, basically after the extraction of the key features, the nearest neighbors are identified to perform the image matching operation. But it might happen that the two neighbors may be close to the point of interest or the test point, which may be due to the noise. So, in such cases we take the ratio of the first best and the second distance neighbor. If the ratio turns out to be higher than the threshold which is set to 0.8 then that match is discarded. Due to which only 5% matches can be considered, and the rest 95% matches are eliminated.

After the extraction of key points using the SIFT or SURF techniques the matching of key points can be done by either the Brute Force Matcher or the Flann Matcher.

- ***Brute Force Matcher:***

  The idea behind the working principle of the Brute Force Matcher is that the descriptors of one feature from 1 dataset are matched with the descriptors of all the rest of the features from the other dataset with the use of L2 norm and then the feature having the least or the closest distance is used for matching purpose. This method is computationally very intensive.

- ***Flann Matcher (Fast Library for Approximate Nearest Neighbors):***

  This technique basically is used for high dimensional data or large dataset as it is quick as compared to the Brute Force Matcher and computationally less intensive.

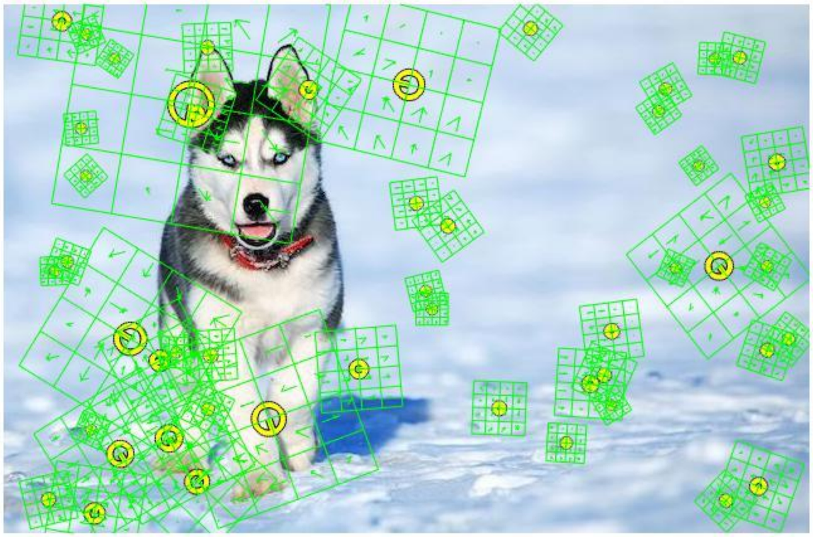
2. APPROACH & PROCEDURE

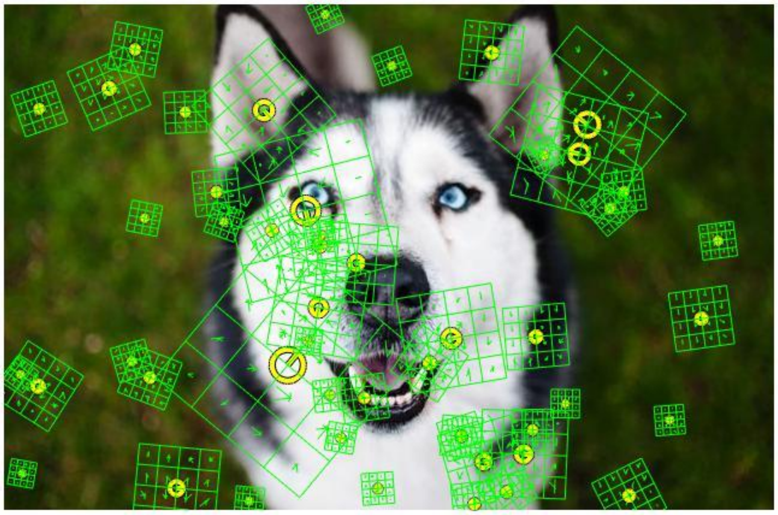
The procedure for Image Matching can be described as follows:

Step 1: To find a match between two images or objects we need to determine the key features and the discriminant power of it will give the efficiency of matching.
Step 2: SIFT technique is used to extract the key features and descriptors.
Step 3: Nearest neighbors are used as key points to calculate the minimum Euclidean distance and then computing the largest L2 norm and then the ratio of the first closest neighbor to the second closet is computed and that particular match is retained or discarded based on the set threshold value.

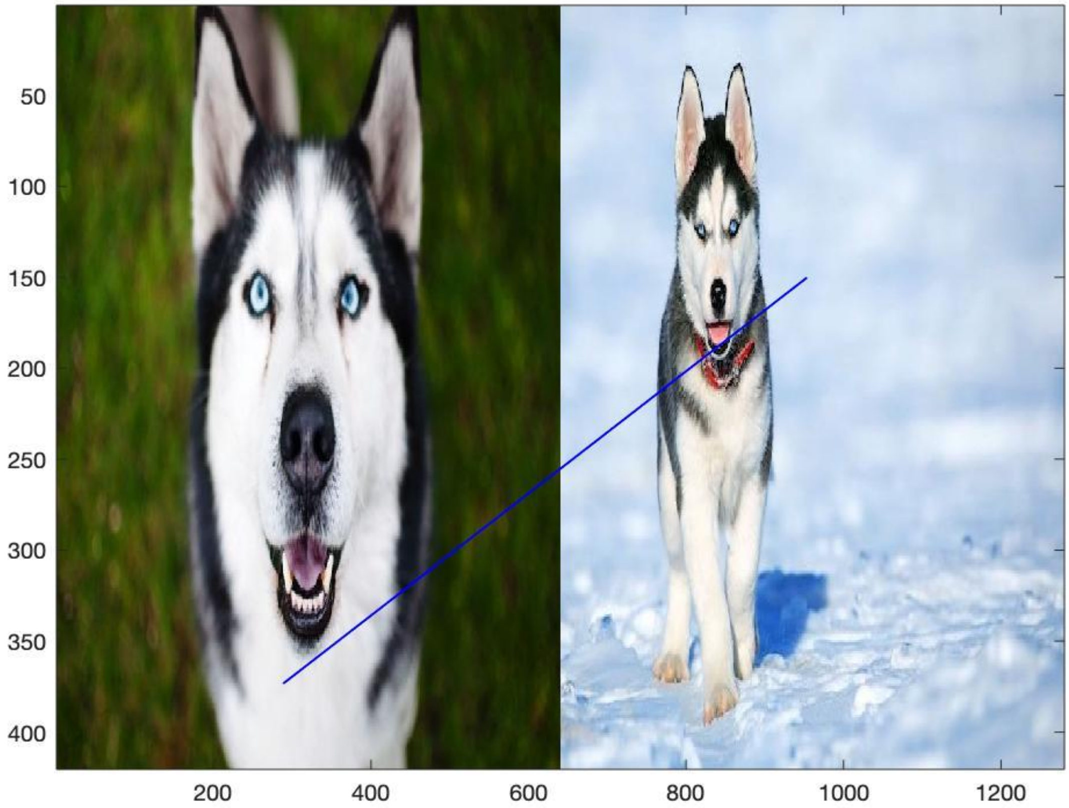
Step 4: Then using the Bin approach as described above we can easily identify the nearest neighbors of high probability. The match having the least Euclidean distance between the two images is retained.
Step 5: Finally, the corresponding retained matches of the key points and descriptors are plotted.

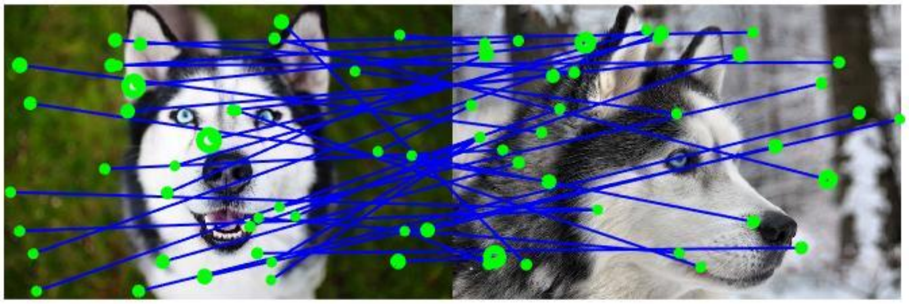
3. EXPERIMENTAL RESULTS



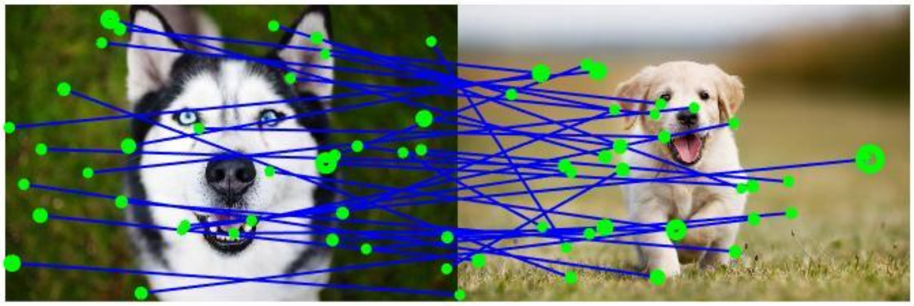


EE569 Digital Image Processing

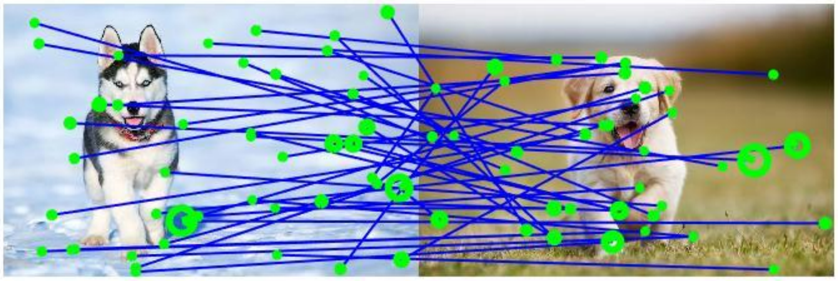




Husky_3 and Husky_2



Husky_3 and Puppy_1



Husky_1 and Puppy_1

4. DISCUSSION

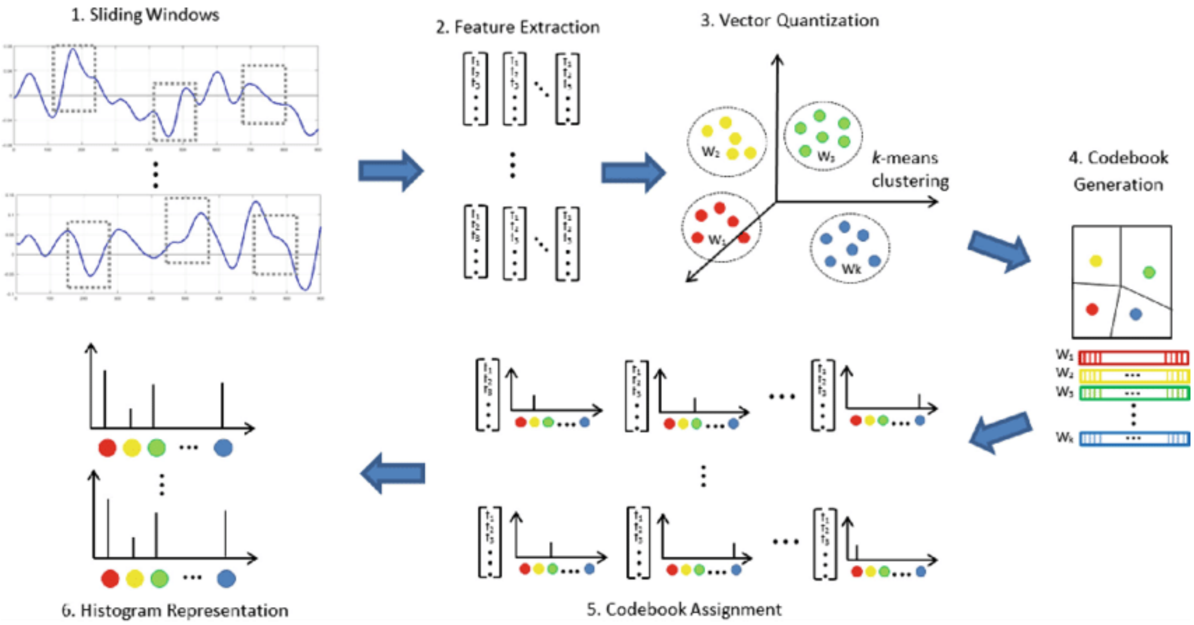
There are a lot number of key points found in the pair of images taken two at a time out of the given four images. But it is not that difficult to find it because of the robustness of the SIFT features as discussed in the above section. The local regions which are same belong to the same region which can easily be justified by the eyes, but the images which does not resembles same implies that the SIFT technique would work for it. If both the key points are of same size implies that they have the same scaling factor. There are basically 524, 884, 373 and 585 key descriptors for the Husky_1, Husky_2, Husky_3 and Puppy_1 image respectively. Also, SIFT calculates the feature vectors in the counterclockwise direction which are not same as that of the gradient orientation. Also, the feature vector is similar for the nearest pair of key points.

The reason for the success or the failure of the algorithm in few cases can be explained in the sense that failure for feature matching may occur due to the images taken at different camera viewing angles or improper scene structure estimation or it can be due to the inappropriate range of images. The success rate of feature matching is basically due to the invariance of the parameters or the robustness of the SIFT technique which up-to a great extend maps the various key features among the given two test images using various types of approaches.

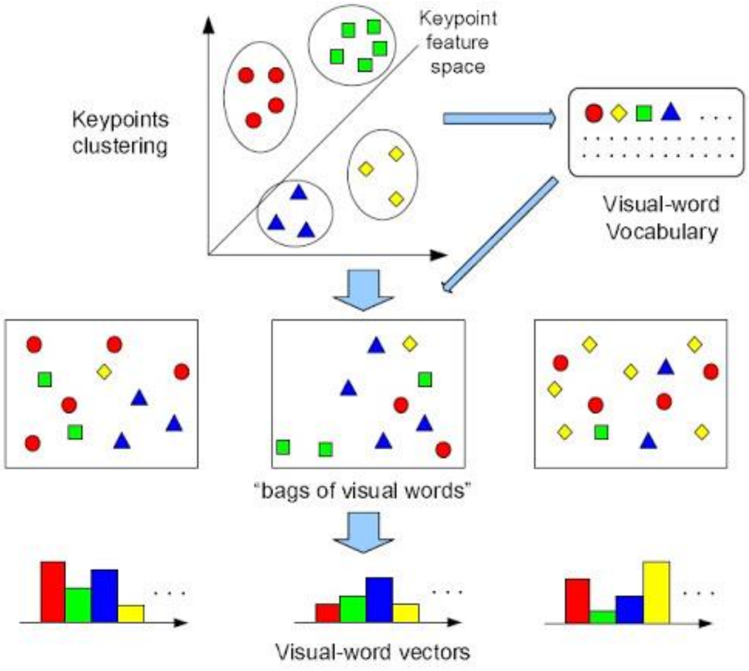
(c) Bag of Words:

1. ABSTRACT & MOTIVATION

Bag of words is a type of image classification technique. The input image is categorized into small images and it is stored in dictionary. Here in this approach the image features are basically treated as the words. These image features are then used to build vocabularies in order to show each image as frequency of histogram of features. Bag of visual words is basically the vector of frequency of words which is expressed as histogram of each word. The major application of this approach is for object or pattern recognition implementations.



Precisely,

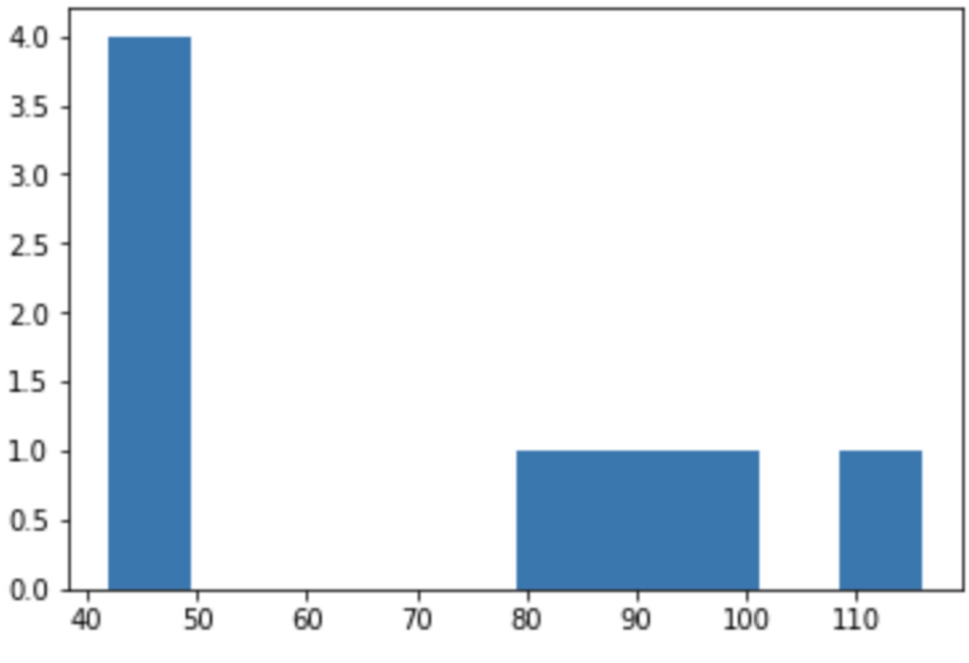


2. APPROACH & PROCEDURE

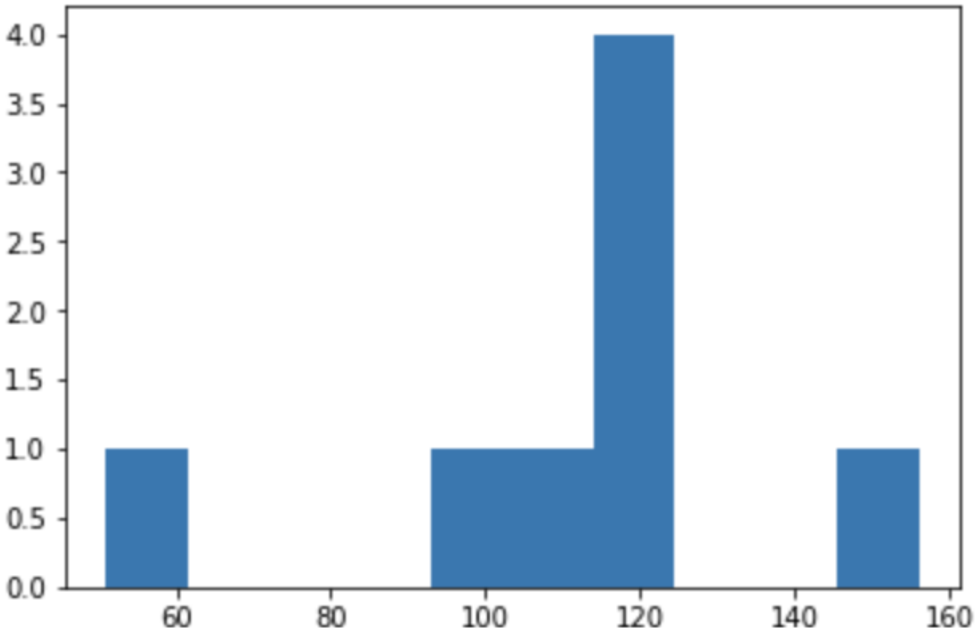
The in general approach for the Bag of Words can be explained as follows:

Step 1: First we load the training images in order to extract the key features and descriptors from each of the image using the SIFT technique.
Step 2: Then after extraction of the SIFT features we stack them into a single set.
Step 3: Next with the aid of K-means clustering algorithm when performed over the set of feature vectors obtained in the previous step, we now find the centroids of each cluster of bags of words to create the vocabulary which will consist the best feature.
Step 4: After finding the centroid coordinates we assign the code name to each centroid.
Step 5: Next we compute the Euclidean distance of each image with the vocabularies by generating the histogram of frequency of events of descriptors of the image.
Step 6: Finally, we compute the absolute difference between the training dataset and the test dataset in correspondence to the 8 bins and then decide that which test image resembles more closely to which training image from the training dataset based on K-means clustering nearest neighbor.

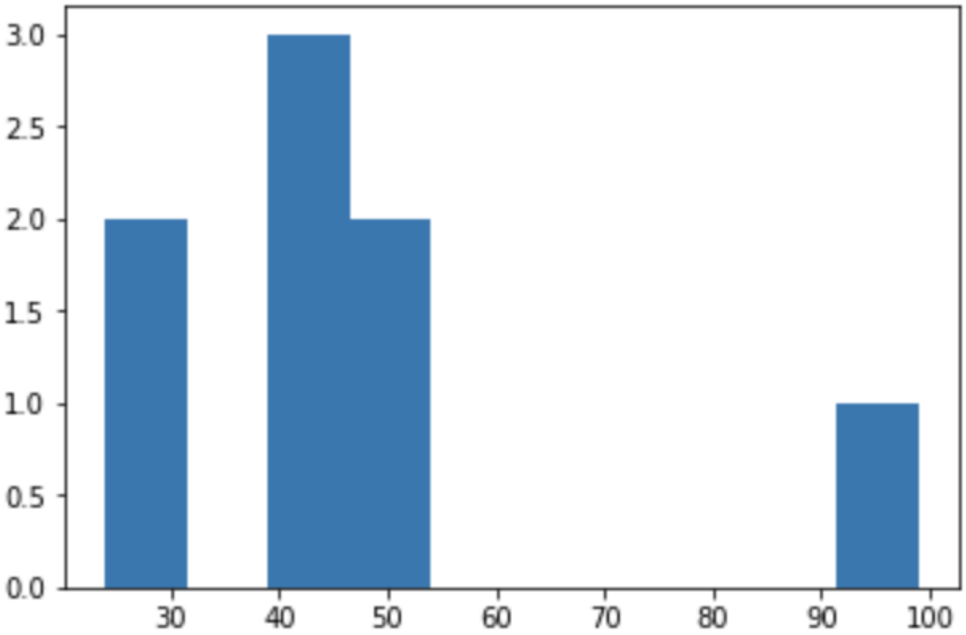
3. EXPERIMENTAL RESULTS



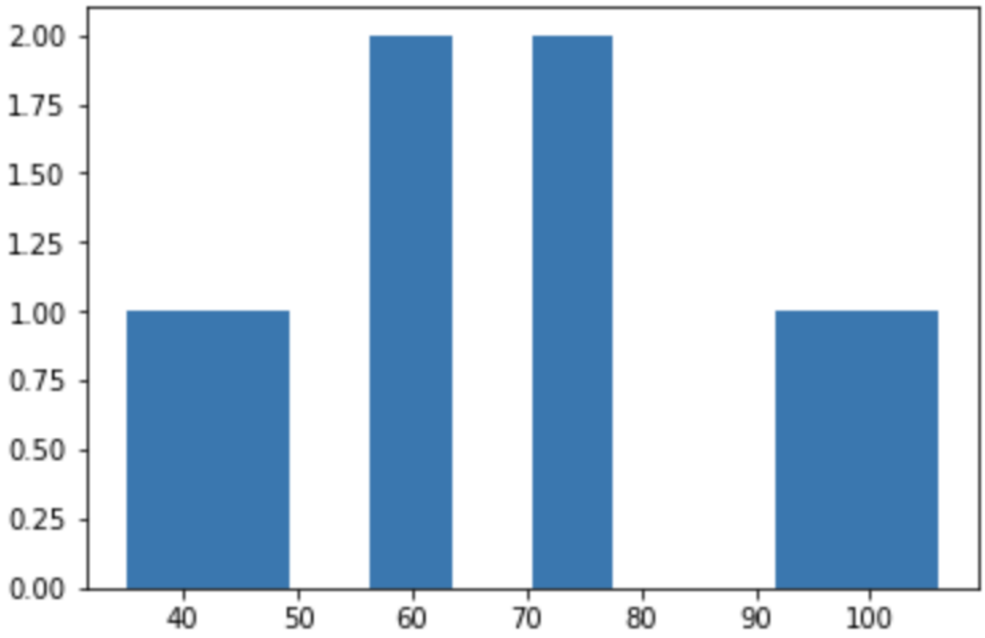
Husky_1



Husky_2



Husky_3



Puppy_1

4. DISCUSSION

Firstly, we obtain different results for different scaling factors, because this is due to the similarity of SIFT parameters in every run. So basically, when I scale up the image the smaller patterns turn larger and thus thereby, we obtain different results.
Secondly, as we can observe that on implementing the Bag of Words, we observe that the histogram intersection is maximum for Husky_3 and Husky_1. I observed that there is the maximum match of key points or the maximum histogram intersection is found for the Husky_3 and Husky_1 pair followed by the Husky_3 and Puppy_1 and the least match or intersection is found for the Husky_3 and the Husky_2 image which may be due to the front faces of Husky getting more matched key descriptors in Husky_3 and Husky 1 while in Husky_3 and Puppy_1 although having different faces but represent the same front face feature and on the other hand the Husky_3 and Husky_2 both having different face orientation have the least match.

References:

1. Digital Image Processing, Second Edition, Rafael Gonzalez, Richard E Woods, Pearson Education.
2. Digital Image Processing, Fourth Edition, William K.Pratt, Wiley-Interscience Publication.
3. Discussion Notes

4. Class Notes
5. David G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, 60(2), 91-110, 2004.
6. OpenCV for SIFT technique
7. Wikipedia