CHARU SINGH

USCID: 2773417243

EE569

Homework 4

Texture Analysis:

Abstraction and Motivation:

Textures in an image are the features that are used to determine the regions of interests and classifying them in image. They tell us about the spatial arrangement of intensities in an image. Based on the texture of the images, they are classified into classes. Here we are given 36 train images, our goal is to cluster all these images where we will use law filters to capture texture information by extracting the features.

There are two key components of texture analysis:

1. Texture classification: from a given set of textured classes it determines the required texture region.
2. Texture segmentation

Procedure:

1. Here in order to find out the type of texture in a region we have used a quantitative measure of the arrangement of intensities.
2. In order to generate texture features we have used different local kernels. An approach given by Laws is based on texture- energy extraction which measures the variation within the window.
3. Texture energy is computed using following kernels:

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]

Here we will do tensor product of any 2 filters thus obtaining 25 combination of filters.

For example, tensor product of E5 with L5 is given by:

$$\begin{bmatrix} -1 \\ -2 \\ 0 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -4 & -6 & -4 & -1 \\ -2 & -8 & -12 & -8 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 8 & 12 & 8 & 2 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

4. Subtract the global mean in order to ameliorate illumination effects.

5. Here 25 kernels are applied so we will get 25-dimensional feature vector at each pixel of an input image.

6. You will get 25-dimensional feature vector for each image.

7. Calculate energy for 25 filtered images by finding out the absolute value in local neighbor and summing them at each pixel.

$$\text{Energy} = \sum_{x=1}^{Row} \sum_{y=1}^{Col} (\text{abs}(f(x,y)))\text{^}2$$

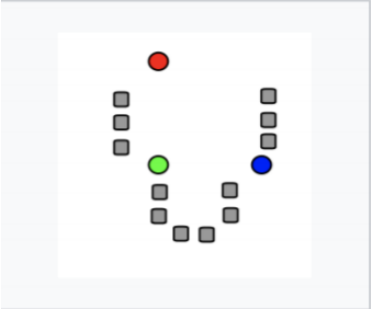
   a) First stack all the images into 3D stacked image.
   b) Find out all filter combinations:
   c) `Do step 4`
   d) After convolution with 25 filters:

e) Get energy for every mask you will get 25d feature vector for each image.

8. Now each image has 25 values of energy which are known as features.
9. Obtain a feature matrix of dimension 36*25.
10. Apply PCA in order to reduce the dimension from 25 to 15 and obtain 36*15 matrix.
11. Perform K means clustering algorithm and give classes /label the images according to clustering.

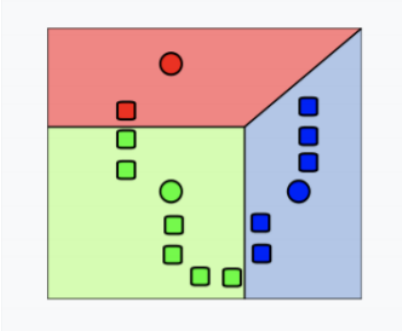
K means clustering: Unsupervised machine learning algorithm

1. Choose number of centroids. Here k=4
   Input: set of points E1…En and place centroids c1…ck at random locations.



2. Repeat until convergence:
   For each point E_i:
   - Find nearest centroid C_j
   - Assign point E_i to cluster j
   - Labelling the sample:

     $L_i = \text{argmin } d(E_i, C_j)$



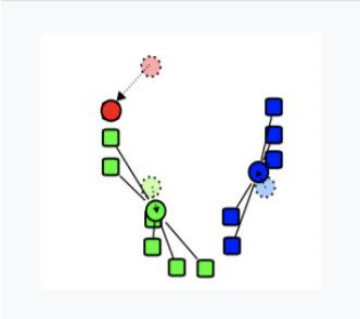
3. From each data point find out the distance from every centroids.

4. Look for minimum distance from the centroid and assign it to that cluster.
   Distance between instance E_i and cluster center C_j is given by:

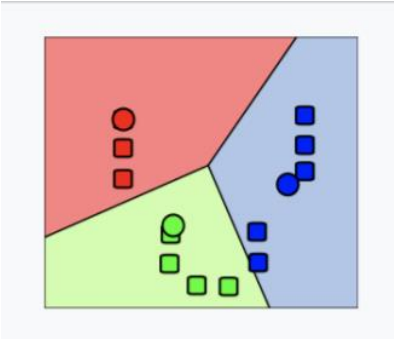
$$d(E_i, C_k) = \sqrt[2]{\sum_{j=1}^{m} (E_{i,j} - C_{k,j})^2}$$

   where i=1…..n and k=1….K

5. Based on new labels centroids are being updated.
   For each cluster j=1….K:
   - For each centroid you recompute its position
   - New centroid C_j =mean of all points E_i assigned to cluster j in previous step.
     $C_j(a)=1/n_j E_j \rightarrow C_j(\sum E_i(a))$ where a=1….d



6. Stop when none of the cluster assignment change. Here its done foe 20 iterations.

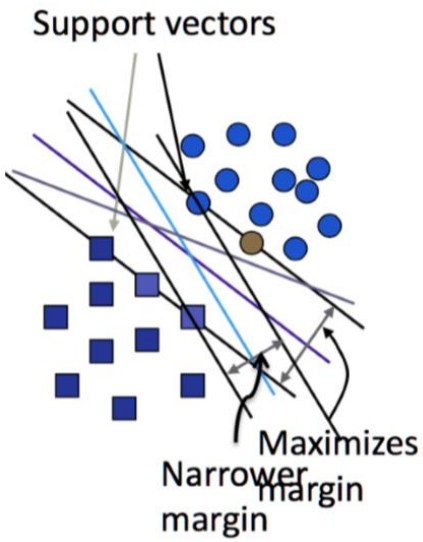


7. Complexity: (#iterations +#clusters+#instances +dimensions)

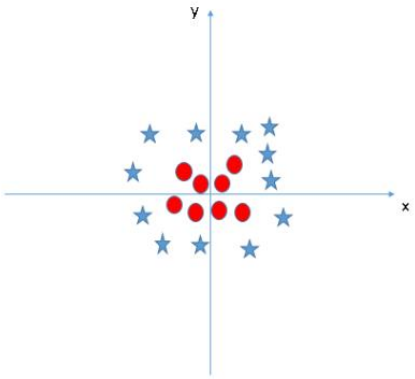
Support Vector Machine:

Goal: It maximizes the margin around the hyperplanes which separated the data points.

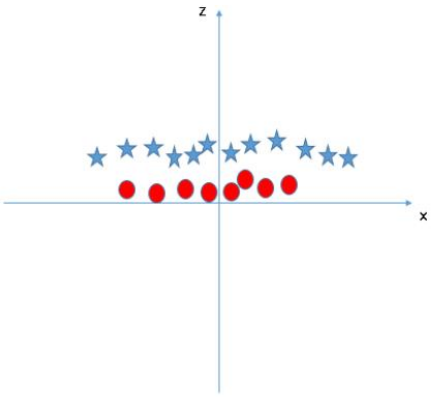
It is a type of a classifier that is based on classifying the data by separating a hyperplane. The output gives us the optimized hyperplane based on the training data (labeled) sets. Our goal is to find the hyperplane that is passing as far from all the points.



Support vectors are the subset of the training data which helps in determining the decision functions.



In case where linear hyperplanes cannot segregate the data points then we will add the new feature as a quadratic equation.



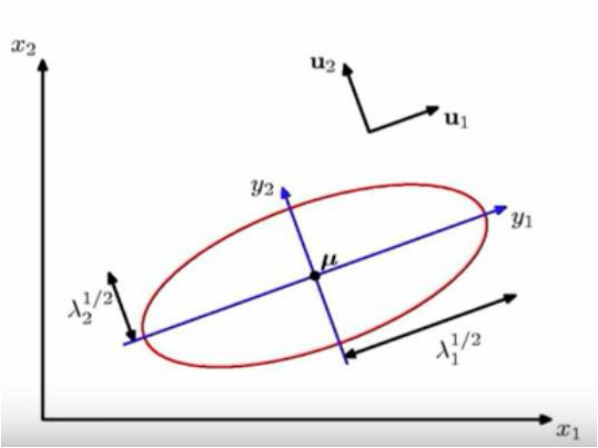
Principal Component Analysis:

We are given a set of input vectors which is used to compute the covariance matrix we will find the basis for covariance matrix. By applying PCA we can transform the vectors from the initial high dimension to the reduced dimension subspace which consists of few important eigenvectors which corresponds to the largest eigenvalue in the covariance matrix. Therefore, there are two things that are reduced:

1. Dimensionality of vector
2. Correlation between coordinates

Geometrically the eigen vectors determine the axes of an ellipse that surrounded the data and captured its variance and eigen values lamda determine the scale of each of the eigen vector (the scale of variance of the data in that direction). In

order to get direction of maximal variance we can simply take eigrn decomposition of covariance



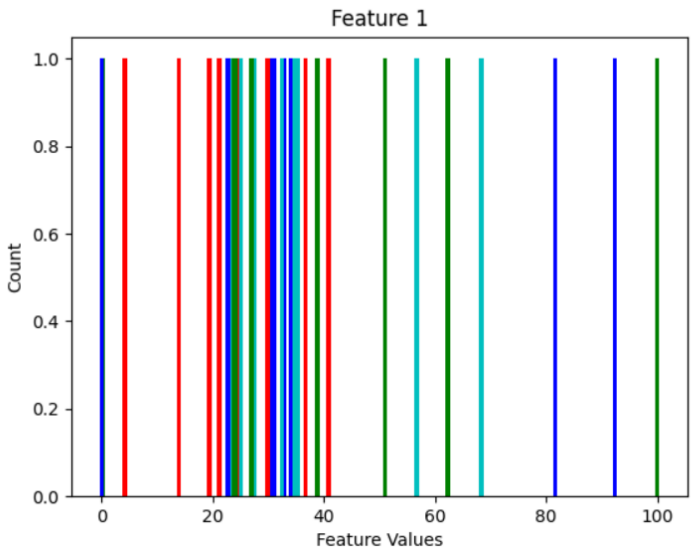
Algorithm: Computing PCA through SVD

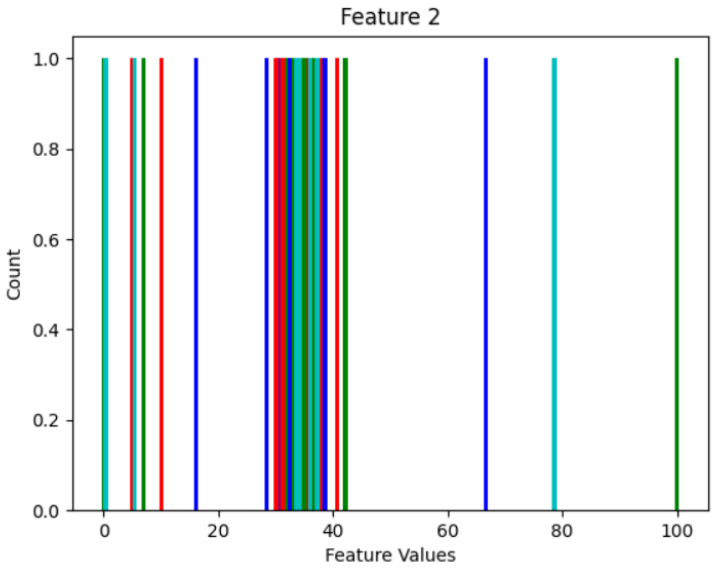
1. Consider a feature matrix X_{n*m} where n: number of samples per pixel and m denotes no. of features.
2. Computing the mean:

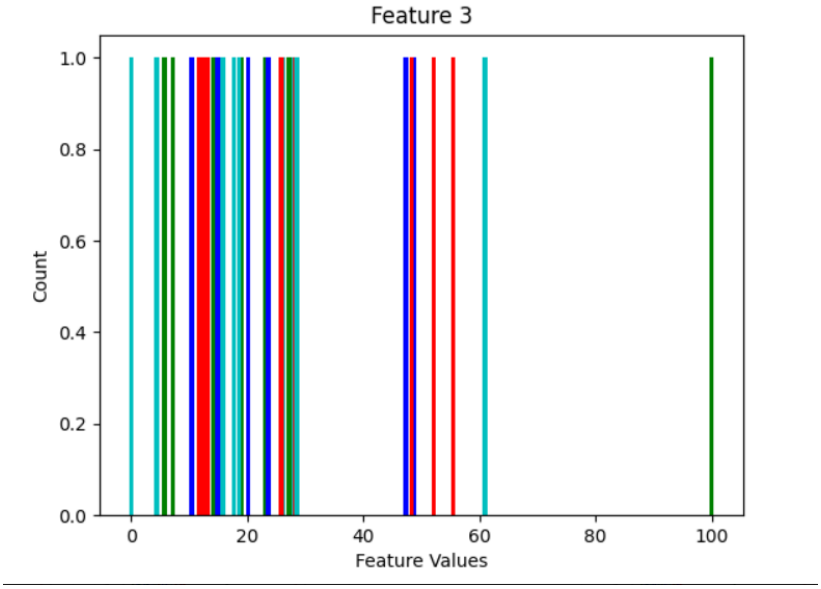
$$(m_X)_j = \sum_{i=1}^{n} \frac{X_{i,j}}{n}, j = 1 \cdots m$$

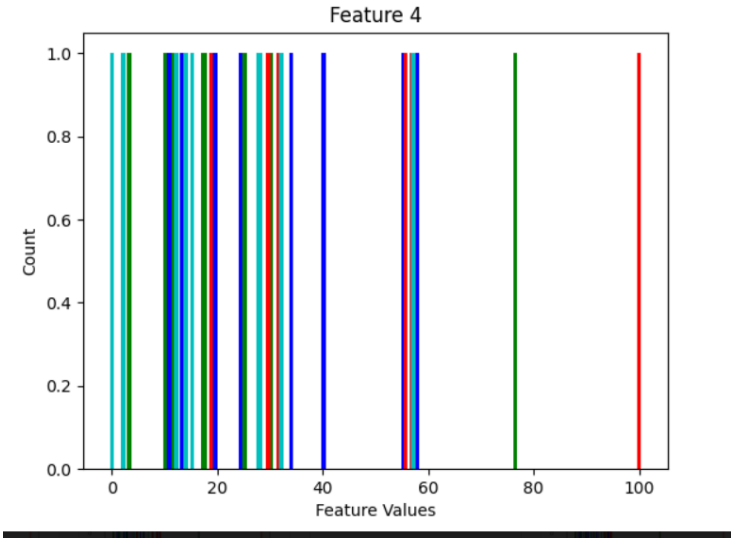
3. In order to get zero mean input data matrix Y_{n*m}, subtract the mean calculated above from each data point.
4. Computing SVD of Y_{n*m} :
   $Y=U*S*V^T$
5. In order to reduce the dimension, sort the eigen values in descending order.
6. Retain the first three top eigen values.
7. Now we have reduced dimension matrix as: $Y_{R=}X *V_R$.

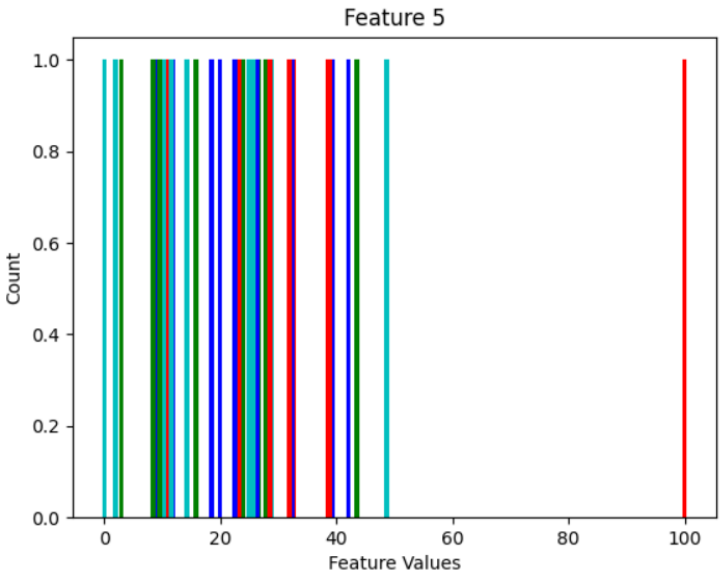
Discussion and Results:

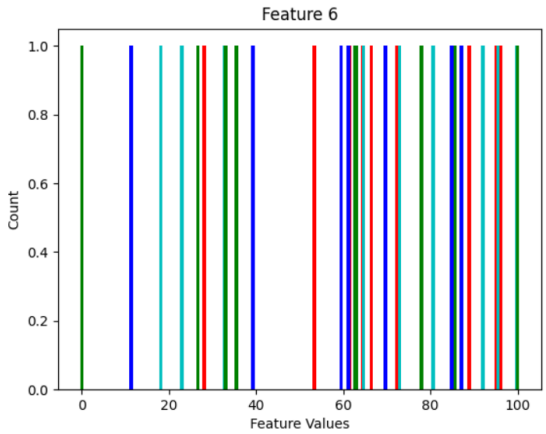

Feature 1
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

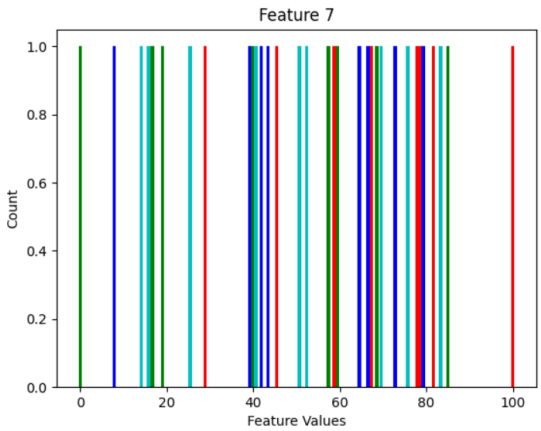

Feature 2
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

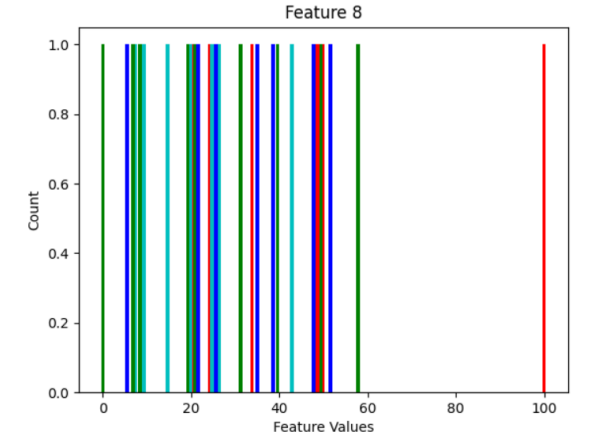

Feature 3
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

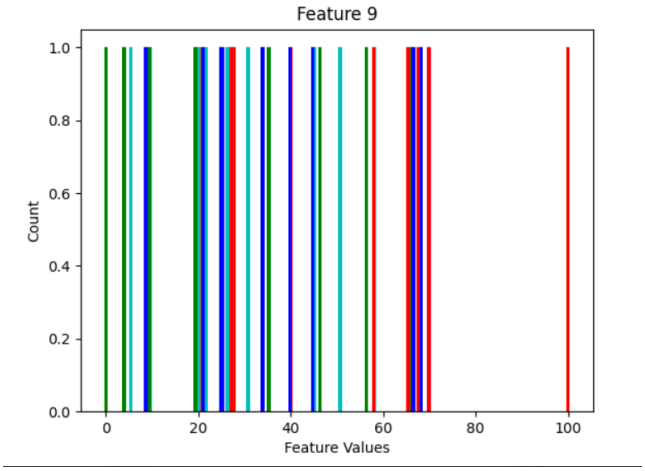

Feature 4
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

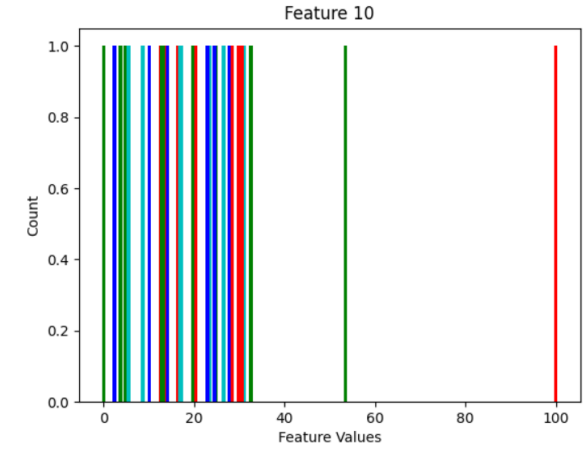

Feature 5
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

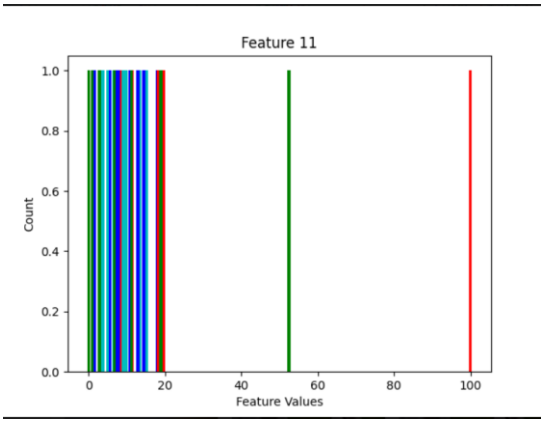

Feature 6
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

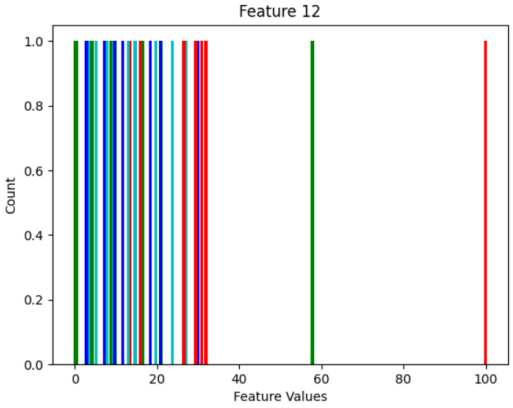

Feature 7
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

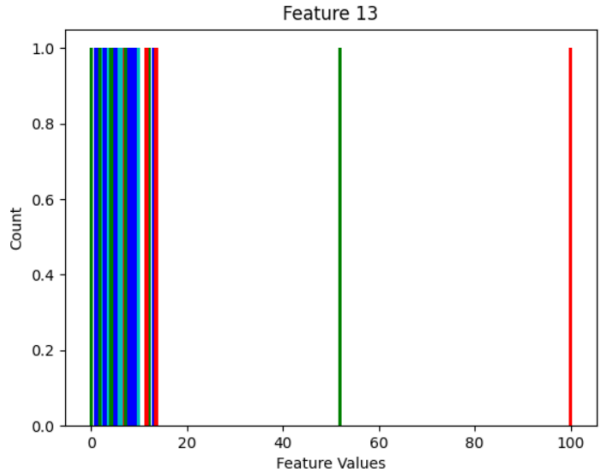

Feature 8
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

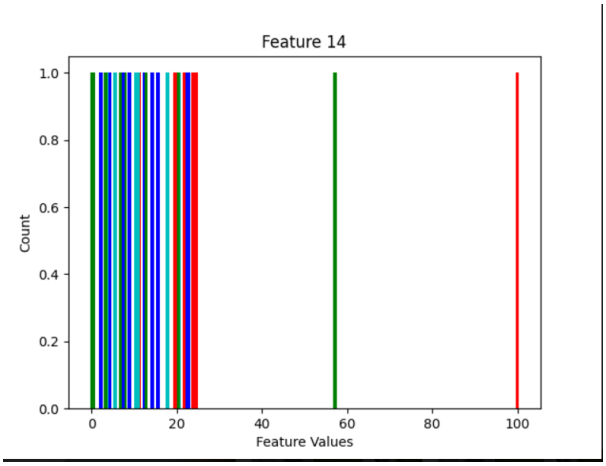

Feature 9
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100

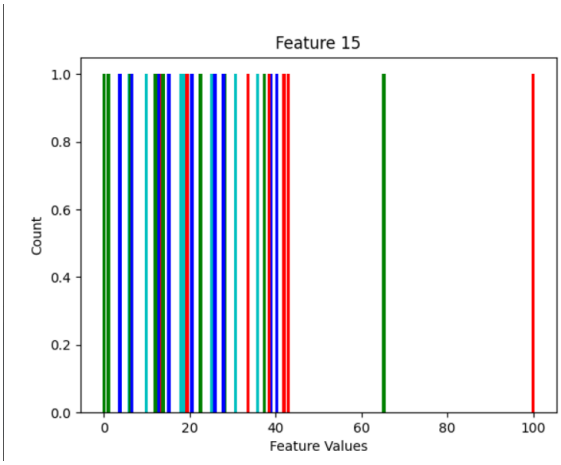

Feature 10
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Feature 11
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Feature 12
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100


Feature 13
Count
Feature Values
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100





Discriminant Power:

Results:

Histograms of 25 features is plotted. Each color is representing a ground truth labels. Features S5E5 gives good discriminant power whereas L5L5 gives least.

1. K means: Predicted classes_15D: 4 3 2 1 4 4 1 4 4 3 3 4
2. SVM: predicted classes: 2 2 1 0 1 2 0 1 1 3 0 2
3. RF: predicted classes: 2 1 0 0 3 2 0 3 3 0 0 2

Accuracy rate for K means: varies between 60-75 %

Accuracy rate for SVM: 64%

Accuracy rate for RF :83% : 2 misclassified labels

Applying PCA improves the result, although the accuracy is almost same but the computational time in k means is less in case of 3-dimensional feature. also, components are selected with high variance which in turn suppresses the correlated data thereby assigning better clusters to the data.

Texture Segmentation:

Motivation and Abstract:

In this problem we are exploiting features of a pixel in an image instead of the image. To perform texture segmentation, we will apply law filters for a given pixel. We will segment given image into various textures that is we will classify every pixel to a specific number of clusters. We are given the image which has 6 different textures and using K means and laws filters we will try to segment the features as best we can.

Procedure:

1. Subtract the global mean in order to ameliorate illumination effects.

2. Here 15 kernels are applied so we will get 15-dimensional feature vector at each pixel of an input image.

3. Calculate energy for each pixel with mask of size 15.

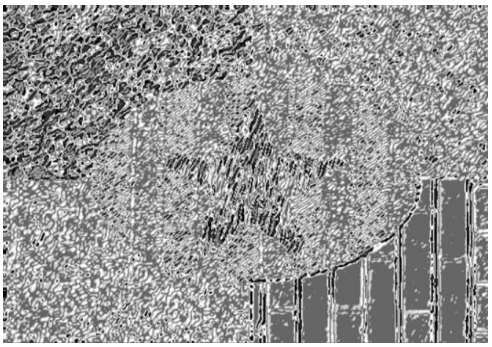
4. Now we have 15 values as features for every pixel.

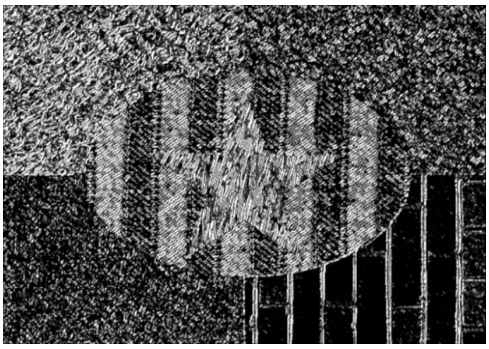
5. Obtain a feature matrix of dimension row*col*15

$$\text{Energy} = \sum_{x=1}^{Row} \sum_{y=1}^{Col} (\text{abs}(f(x,y)))\text{^}2$$

6. Perform K means clustering algorithm and give classes /label the images according to clustering. Here k=6.
7. Here we have 6 different patterns in an image. to denote them we have used 6 gray levels that is 0,51,102,153,204,255.

Results:





Improved Texture Segmentation:

In order to improve the k means algorithm performance we perform dimensionality reduction algorithm known as PCA.

Procedure:

Follow the same procedure as above. Perform PCA after extracting the image features and them apply k means clustering. PCA and k means is done using python inbuilt function.

Discussion and Results:

PCA will improve the result because it involves highly uncorrelated data variables (principal components) hence data by orthogonal transformation is transformed. As the data is having high covariance would get classified to correct clusters as compared to data with correlated points which would give more error rate.

Also, time complexity is less as computational time decreases for k means.

N_components : 2, 5

Mask_size: 7 and 15 are being used.

15 mask size gives better results

Above gives best results with well defined boundary and less distortion in regions. Visually, the designs in the image can be distinguished properly. Computational time is also less.

Problem 2:

1) according to the abstract SIFT is robust to:

- Image scaling and rotation
- Variation in illumination
- Addition of noise
- Change in 3D viewpoint

STAGES:

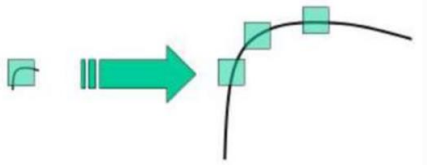
1. **Scale Invariant Feature Detection:**
    - Image is transformed into large number of feature vectors.

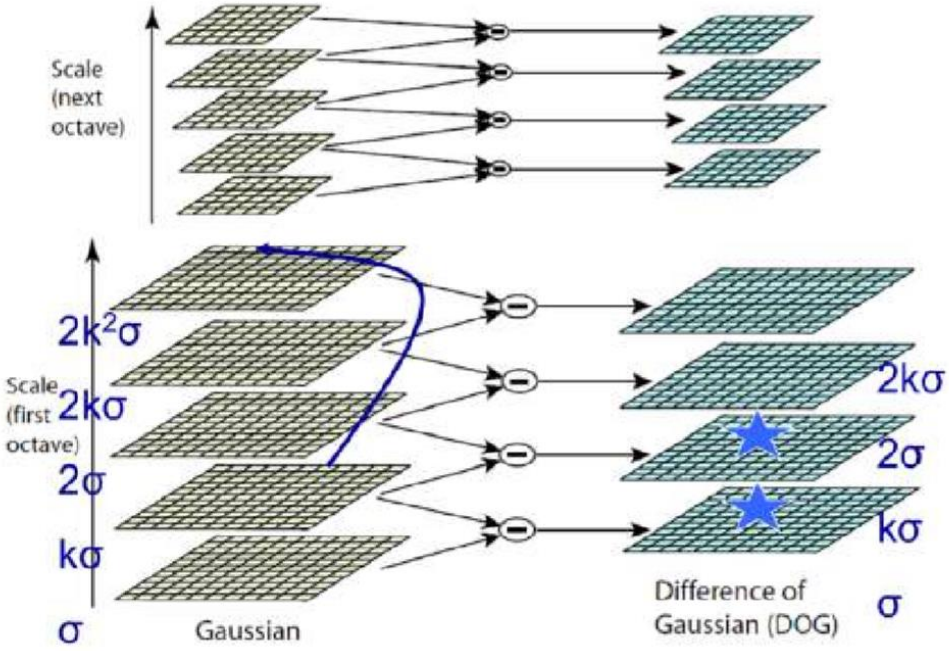
- We will smooth the images and resample them and then apply difference of Gaussian function.
- Key points are maxima and minima of the result of DoG.
- Edge response points and low contrast points are not considered.
- We obtain SIFT descriptors by looking the area around key location by considering the surrounding pixels.

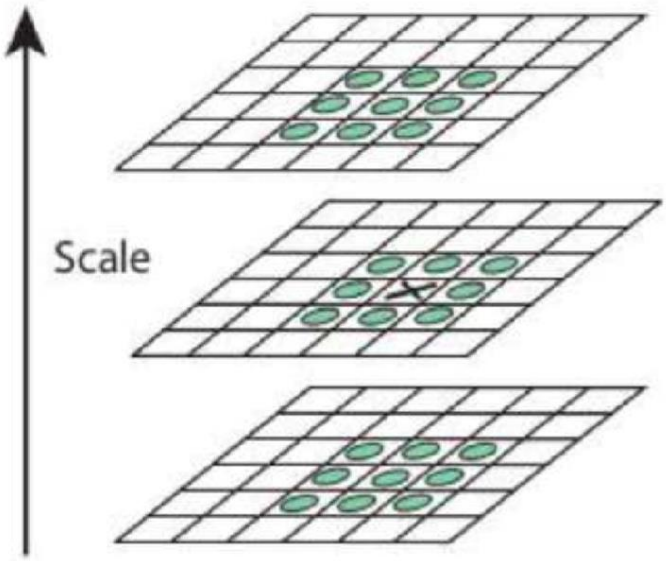
ALGORITHM:

1. **Scale space extrema detection:**

- When scaling on image is done in Harries Corner detection the corners are not detected properly and as a result SIFT involves Convolving the images with gaussian filters at various scales followed by difference of images which are gaussian (blurred).







Minima or maxima of DoG are taken as keypoints:

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

We compare every pixel in DoG images at the same scale (to its 8 neeigbors) and at neighboring scales (9 corresponding neighbors).

if pixel value is minimum or maximum among all, it is considered as keypoint.

Key Point Localization:

To fit unsatble keypoint , we have to fit those points to the nearby data in order to obtain accurate scaling ,location.

Orientaion Assignment:

Helps to obtain invariance to rotation by assigning one or more orientations to each keypoint based on gradient orientaions.

Keypoint Descriptors:

Now we have keypoints required scales and corresponing orienations. Here our goal is to find a decriptor which is distinctive vector for every keypoint.

How does it removes noise:

It eliminates the edges by computing principal curvatures at location of key point. DoG have a good response at edges

Assignment of Keypoint Orientaion:

Each keyppoint is assigned with orientaion in order to have invariance to rotaion.

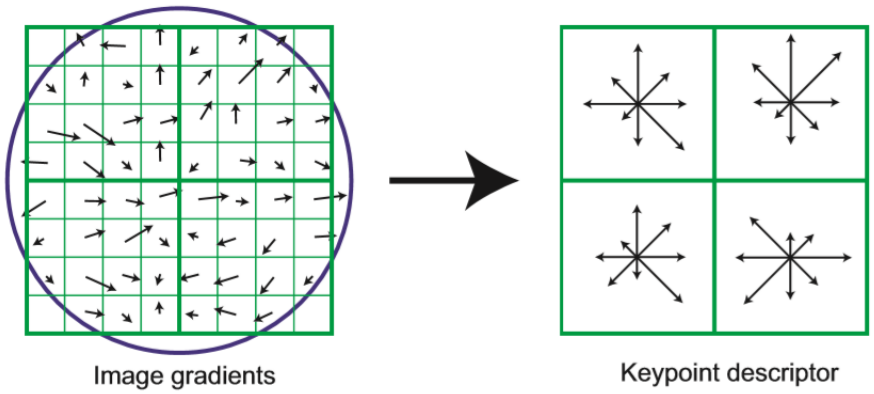
According to scale, neigborhood is taken around keypoint and is calculating the gradient direction and magnitude.

Orientation histogram is created comprising of 36 bins spanning 360 degree.

It is weighted by gaussian window having sigma =1.5* scale of keypoint and gradient magnitude.

Highest peak or peak >80% in histogram is used to calculate orienation.

3 Image Descriptor to reduce illumination Effects:



It uses the principle of comples neurons in terms of visual cortex. In this method gradient magnitude and orientation is calculated in order to fing the intensity of gaussian blurring around location where the keypoint is there using the scaling of keypoint. They are weighted by a gaussian mask as shown in figure. Rotaion of

descriptors and gradient orientaion is being made in order to achieve inavariance to rotation. In figure above small arrows represent the gradients for all levels.

Each sample is assigned with weight by weighting function. Less weights are given to the points which are far away from center. Above figure shows that for every histogram we have 8 direction.

Trilinear interpolation will help assigning the values of gradient to adjacent bins.

Result is we have keypoints with the same scale and location but different directions.

Consider 16*16 neigboring pixels around desired keypoint . we have 16 subblocks of 4*4 size

For each subblock , 8 bin orienation histogram is created (comprising of 16 samples).

In total there are 128 bins (8*16=128) to form keypoints descriptors represented by a vector. This vector has 3 dimensions x,y,theta. This feature vetcor is normalized in order to be illumination inavriant.

In case of image contrast where every pixel is multiplied by a constant and gradient also by the same . therefore using vector normalization it is **invariant to image contrast.**

The gradient values are computed using the pixel diiferences which in turn makes it **brightness invariant.**

This is how inavriance to illumination is obtained.

4) We chose DoG instead of Laplacian because of efficiency of DoG as every time we must smoothen the image in order to perform scale space feature description and using image subtraction, we can find D. Moreover, a close approximation is given by DoG function to scale normalized LoG.
5) 128 is the output vector size.

Feature Extraction in images:

Motivation and Abstract:

Many algorithms have been developed for corner detection in images such as Harris corner detecting technique. This technique was rotation invariant but still it wasn't scale invariant. in 2004, a new algorithm was proposed by Lowe from university of British Columbia named as Scale Invariant Feature Extraction Algorithm (SIFT). It helps in describing the local features. It generates key points.

Description of objects in an image are extracted from trained images and are useful in locating the object in test images.

Generated key points are invariant to scaling and rotation of images, orientation, changes in illumination.

Image Matching:

In order to match images, we have used Fast Library for Approximate Nearest Neighbor (FLANN). It performs well for nearest neighbor technique.

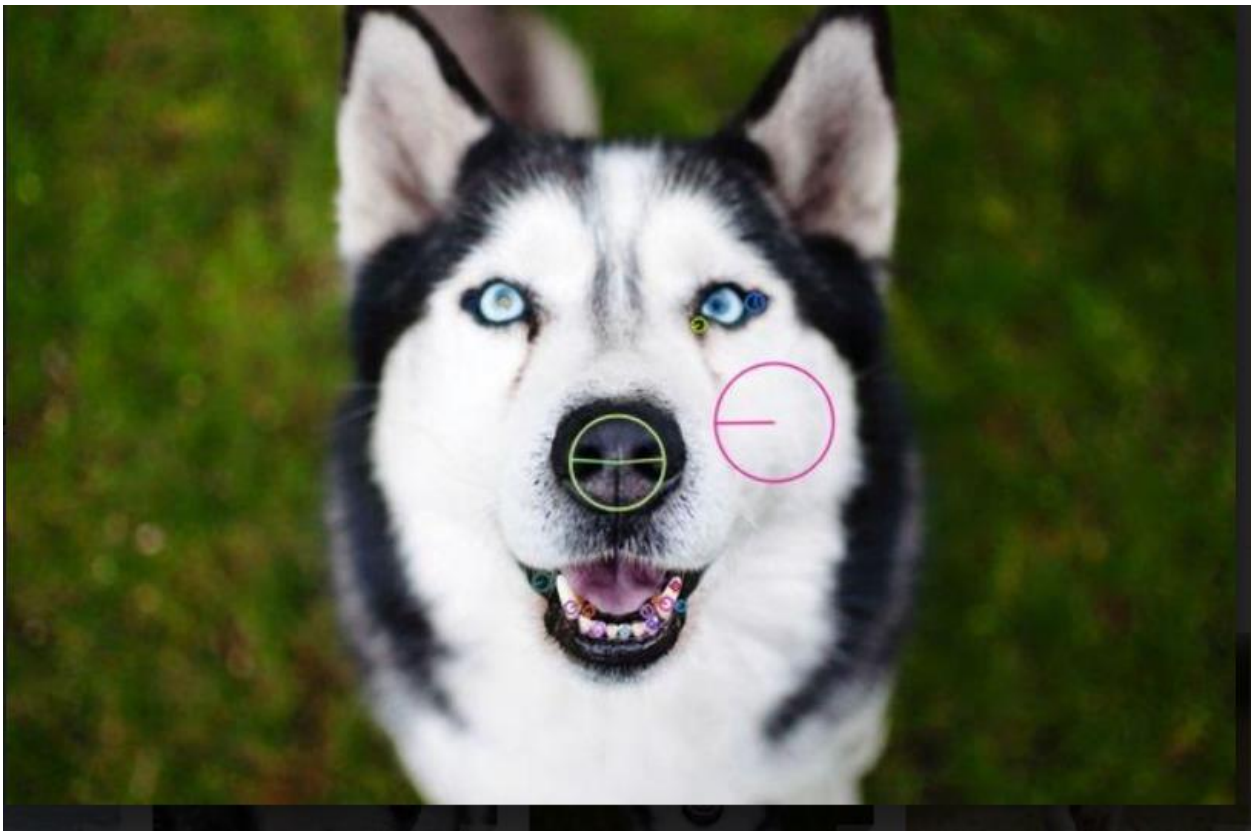
Procedure:

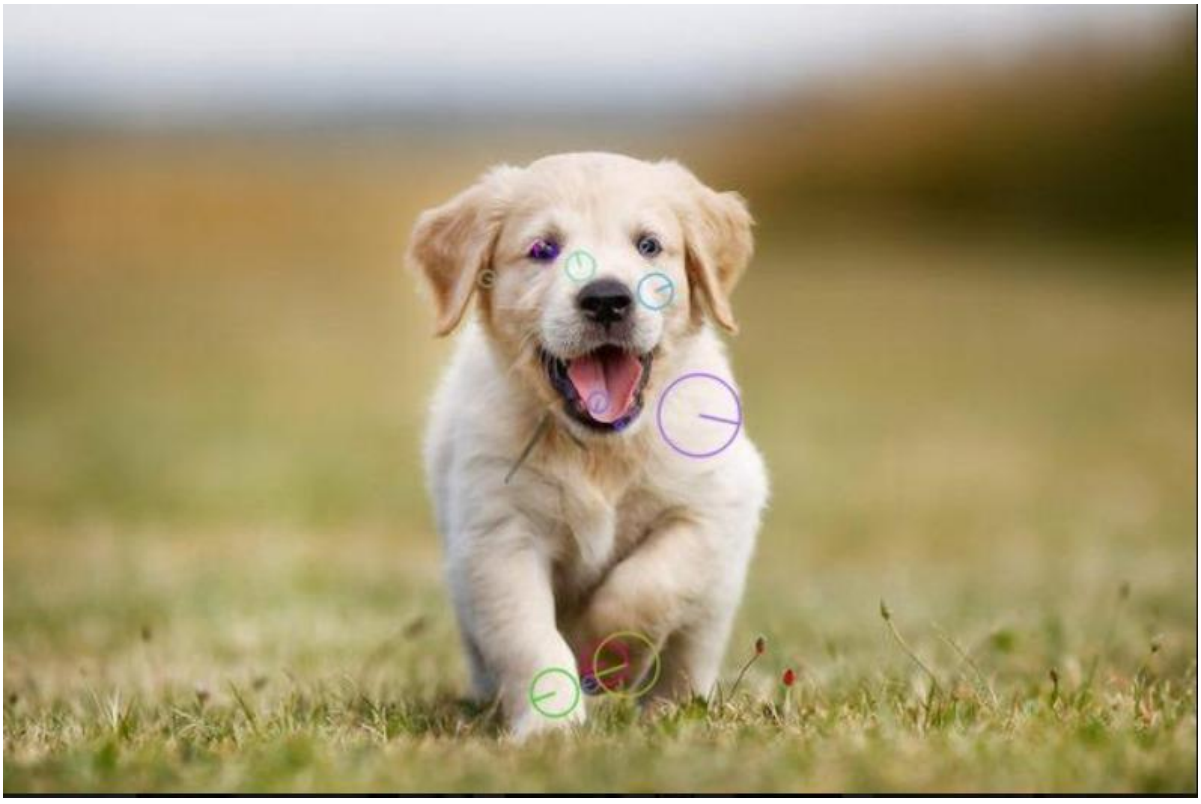
A. **Algorithm for K nearest neighbor (KNN):**
    a. Initialize value of k
    b. Find the distance of test image to each training image.
    c. Sort the calculated distance in step 2.
    d. Determine first k closest neighbor to the testing image.
    e. Class is assigned based on the majority class of the k neighbors' samples.
B. Using SIFT detector we will obtain the descriptors.
C. Apply FLANN based matcher to find matching points based on the descriptors calculated above.
D. Now our goal is to find the best matched key points.

E. Threshold is kept as 0.2 and based on this value and between the points look for min and max distance, a decision is being made for the points that is whether to be discarded or kept.
F. We will get the output image according to best points calculated in above step.
G. Try the above procedure for different mi Hessian value.

Results:

Following images show the orientation of the key points at different angles:





Following screenshot shows the key points and min and max distance is calculated

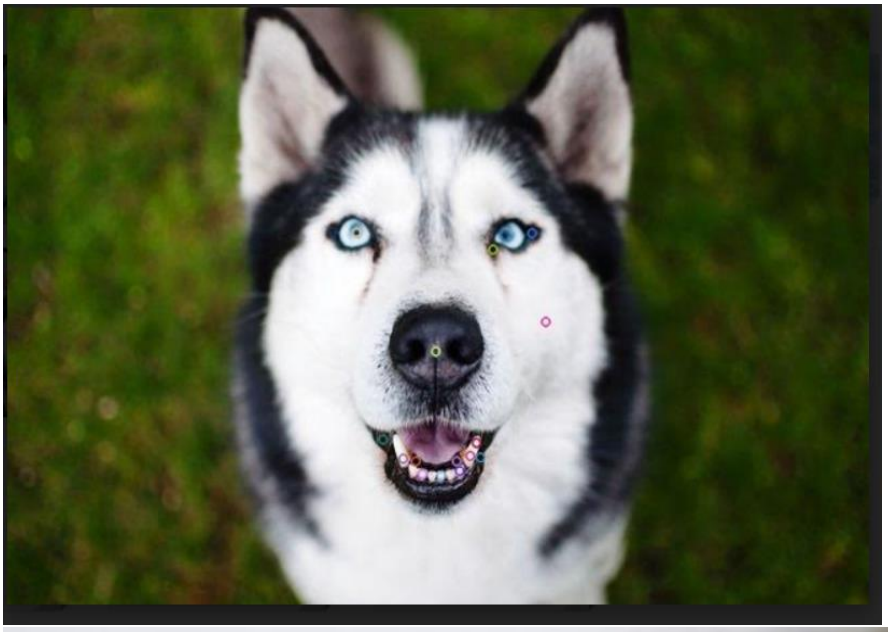
```
Max distance : 464.567535
Min distance : 100.000000
Best Match [0] Keypoint 1: 97  -- Keypoint 2: 281
Best Match [1] Keypoint 1: 117  -- Keypoint 2: 333
Best Match [2] Keypoint 1: 238  -- Keypoint 2: 288
Best Match [3] Keypoint 1: 289  -- Keypoint 2: 118
Best Match [4] Keypoint 1: 393  -- Keypoint 2: 333
Best Match [5] Keypoint 1: 462  -- Keypoint 2: 249
Best Match [6] Keypoint 1: 511  -- Keypoint 2: 117
Max distance : 473.968353
Min distance : 100.000000
Best Match [0] Keypoint 1: 14  -- Keypoint 2: 380
Best Match [1] Keypoint 1: 20  -- Keypoint 2: 371
Best Match [2] Keypoint 1: 74  -- Keypoint 2: 406
Best Match [3] Keypoint 1: 116  -- Keypoint 2: 227
Best Match [4] Keypoint 1: 118  -- Keypoint 2: 371
Best Match [5] Keypoint 1: 248  -- Keypoint 2: 313
Best Match [6] Keypoint 1: 249  -- Keypoint 2: 313
Best Match [7] Keypoint 1: 333  -- Keypoint 2: 333
Best Match [8] Keypoint 1: 355  -- Keypoint 2: 401
Best Match [9] Keypoint 1: 363  -- Keypoint 2: 303
Best Match [10] Keypoint 1: 369  -- Keypoint 2: 410
Max distance : 441.046478
Min distance : 100.000000
Best Match [0] Keypoint 1: 48  -- Keypoint 2: 313
Best Match [1] Keypoint 1: 55  -- Keypoint 2: 233
Best Match [2] Keypoint 1: 56  -- Keypoint 2: 410
Best Match [3] Keypoint 1: 60  -- Keypoint 2: 380
Best Match [4] Keypoint 1: 61  -- Keypoint 2: 240
Best Match [5] Keypoint 1: 71  -- Keypoint 2: 406
Best Match [6] Keypoint 1: 111  -- Keypoint 2: 323
Best Match [7] Keypoint 1: 129  -- Keypoint 2: 240
Best Match [8] Keypoint 1: 135  -- Keypoint 2: 260
Best Match [9] Keypoint 1: 205  -- Keypoint 2: 226
Best Match [10] Keypoint 1: 208  -- Keypoint 2: 406
Best Match [11] Keypoint 1: 267  -- Keypoint 2: 289
Best Match [12] Keypoint 1: 279  -- Keypoint 2: 194
Best Match [13] Keypoint 1: 321  -- Keypoint 2: 240
Best Match [14] Keypoint 1: 341  -- Keypoint 2: 406
Best Match [15] Keypoint 1: 344  -- Keypoint 2: 227
Best Match [16] Keypoint 1: 408  -- Keypoint 2: 323
Best Match [17] Keypoint 1: 439  -- Keypoint 2: 240
Best Match [18] Keypoint 1: 444  -- Keypoint 2: 338
Best Match [19] Keypoint 1: 456  -- Keypoint 2: 196
Best Match [20] Keypoint 1: 457  -- Keypoint 2: 242
```

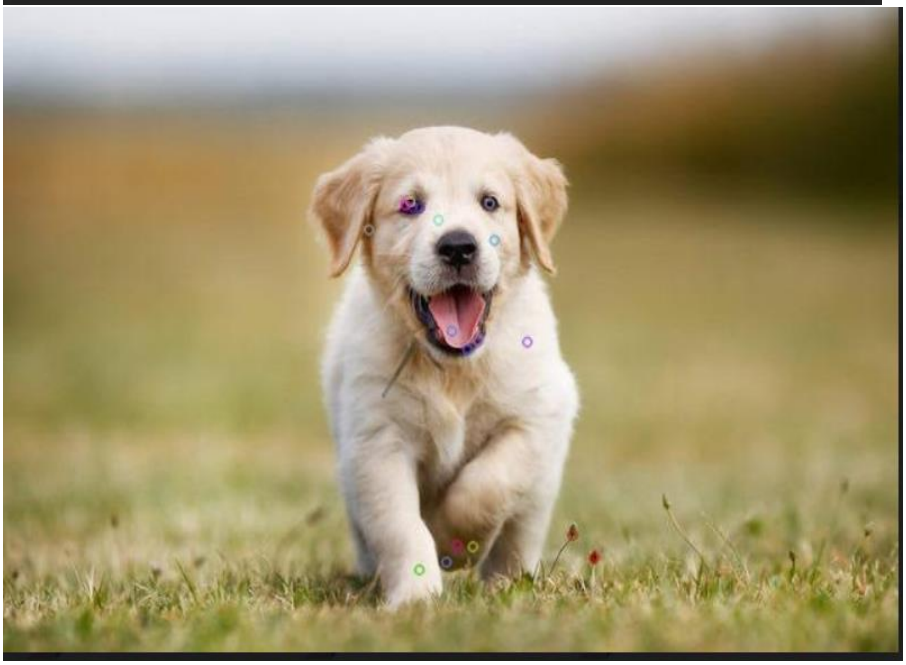
```
Best Match [19] Keypoint 1: 456  -- Keypoint 2: 196
Best Match [20] Keypoint 1: 457  -- Keypoint 2: 242
Max distance : 459.142670
Min distance : 80.715553
Best Match [0] Keypoint 1: 46  -- Keypoint 2: 227
Best Match [1] Keypoint 1: 48  -- Keypoint 2: 287
Best Match [2] Keypoint 1: 68  -- Keypoint 2: 20
Best Match [3] Keypoint 1: 111  -- Keypoint 2: 248
Best Match [4] Keypoint 1: 225  -- Keypoint 2: 304
Best Match [5] Keypoint 1: 274  -- Keypoint 2: 181
Best Match [6] Keypoint 1: 344  -- Keypoint 2: 341
Best Match [7] Keypoint 1: 408  -- Keypoint 2: 248
Best Match [8] Keypoint 1: 422  -- Keypoint 2: 287
Best Match [9] Keypoint 1: 433  -- Keypoint 2: 287
Best Match [10] Keypoint 1: 442  -- Keypoint 2: 290
Max distance : 453.411499
Min distance : 80.715553
Best Match [0] Keypoint 1: 19  -- Keypoint 2: 248
Best Match [1] Keypoint 1: 29  -- Keypoint 2: 290
Best Match [2] Keypoint 1: 37  -- Keypoint 2: 287
Best Match [3] Keypoint 1: 46  -- Keypoint 2: 227
Best Match [4] Keypoint 1: 48  -- Keypoint 2: 287
Best Match [5] Keypoint 1: 225  -- Keypoint 2: 304
Best Match [6] Keypoint 1: 256  -- Keypoint 2: 181
Best Match [7] Keypoint 1: 273  -- Keypoint 2: 248
Best Match [8] Keypoint 1: 278  -- Keypoint 2: 20
Best Match [9] Keypoint 1: 299  -- Keypoint 2: 287
Best Match [10] Keypoint 1: 338  -- Keypoint 2: 341
Best Match [11] Keypoint 1: 348  -- Keypoint 2: 248
```

Results:

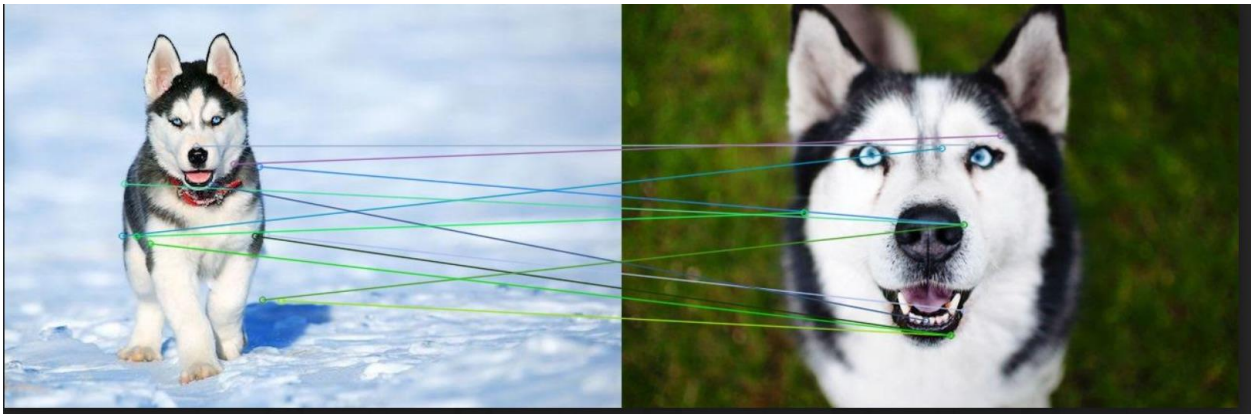
Minimum distance calculated is 80.711 for husky 1 and husky 3 by finding best matched keypoints.

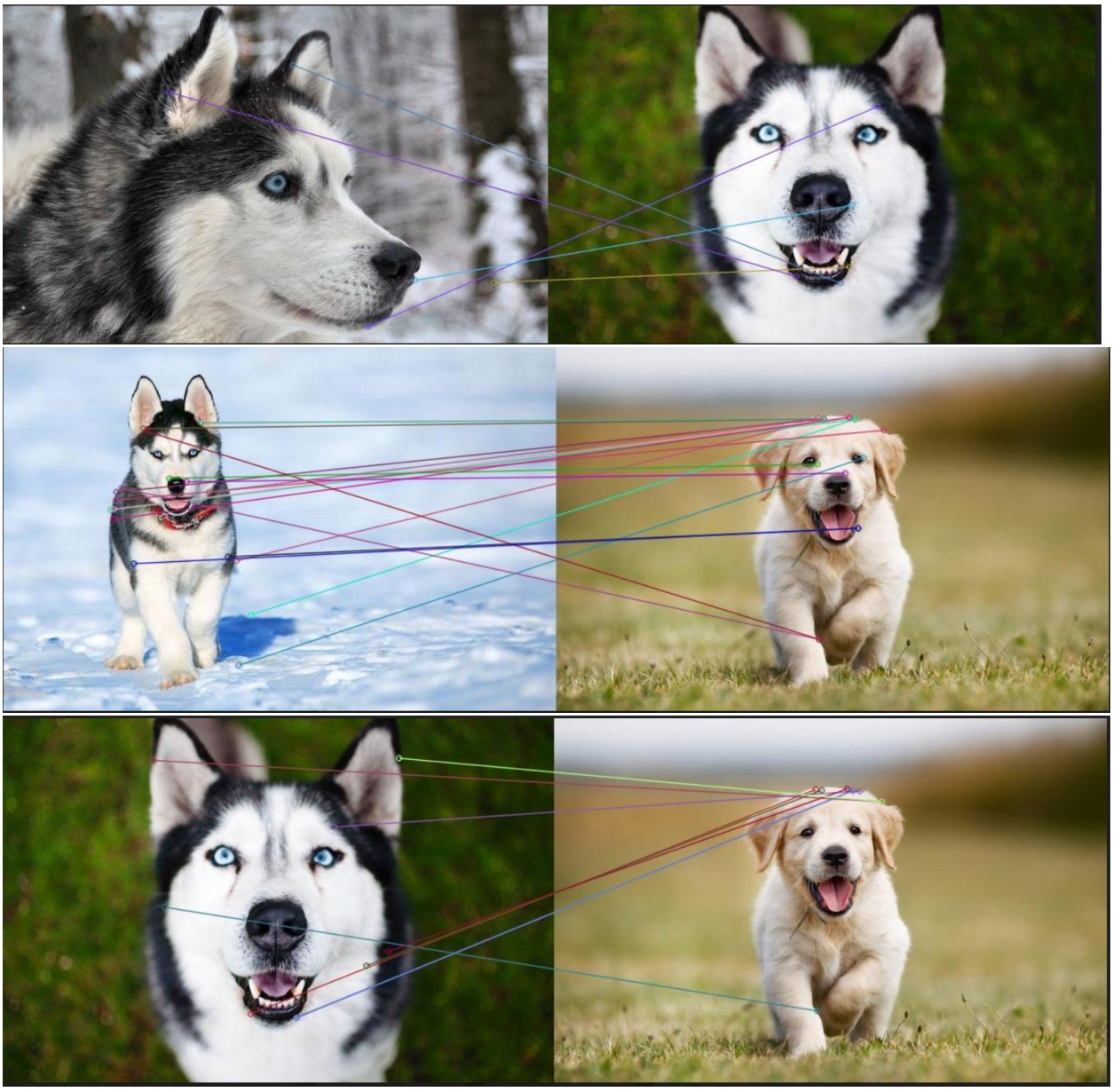
Following images shows key points in all images:





Images are matched shown in below figures:





Best matched key points are shown for min Hessian value 400.
Best match key points are obtained for husky 1 and husky 3, they have maximum number of key points as shown in screenshot .
Although we can see that there are lot of matches between husky and puppy images but since they both are dogs and they contain similar characteristics such as its face structure, ears etc that's why they are having some matching points.

Bags of Words:

Motivation and Abstract:

Here we are using concept of Bag of Words for image matching unlike above method where we used SIFT technique and used FLANN based matcher in order to match the images. In computer vision this method is proved to be more robust.

Approach:

Bag of Words is mainly used in Object Recognition in images. we will use SIFT technique in order to determine the descriptors to form words in the codebook.

Procedure:

Formation of Codebooks:

For a given image centroids are being decided and by using K means algorithm descriptors are clustered to the centroid whichever is the closest.

For every image a histogram is made which consist of the counting of descriptors (extracted by SIFT) which belongs to the code word or cluster is formed. This is being done for all the data points (features).

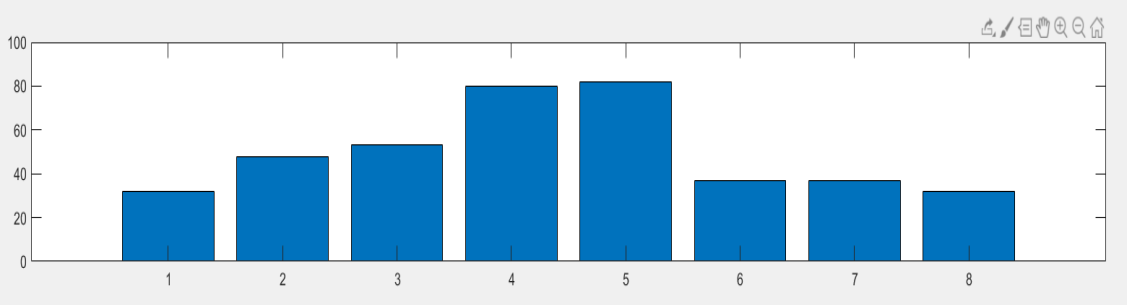
After this in order to match an image histogram matching is done.

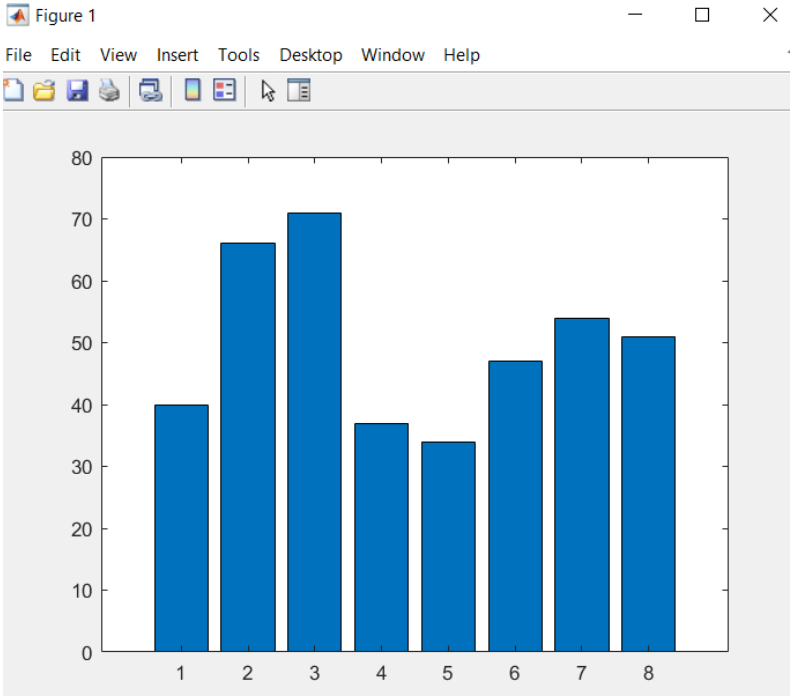
- For all the given 4 images we will obtain descriptors using SIFT.
- Obtain the codebook centroids by training them.
- Form the histogram of code words for every image and cluster every data point.
- Image matching is done with husky's 3 codewords with every other image.
- Look for the best matching histogram.

Histogram Matching:

To find the best matching we will calculate the histogram error as shown below and choose the one with minimum error.

$$\text{Hist Error} = \sum_{k=1}^{n} \text{abs}(\, \text{H}_{\text{test}}(\text{k}) - \text{H}_{\text{train}}(\text{k}))$$





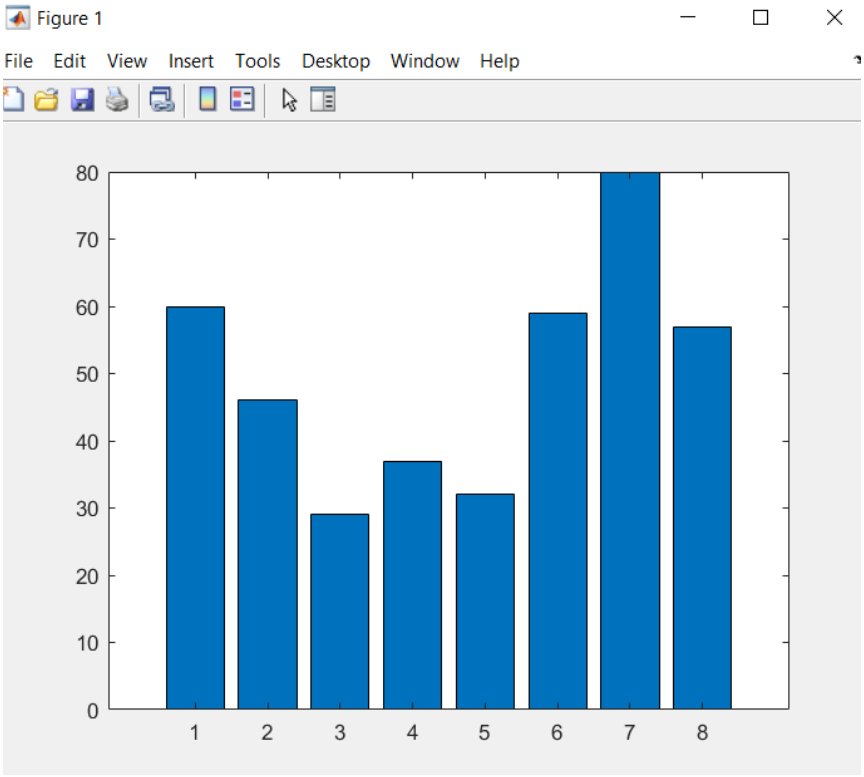


Figure 1
File Edit View Insert Tools Desktop Window Help
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8

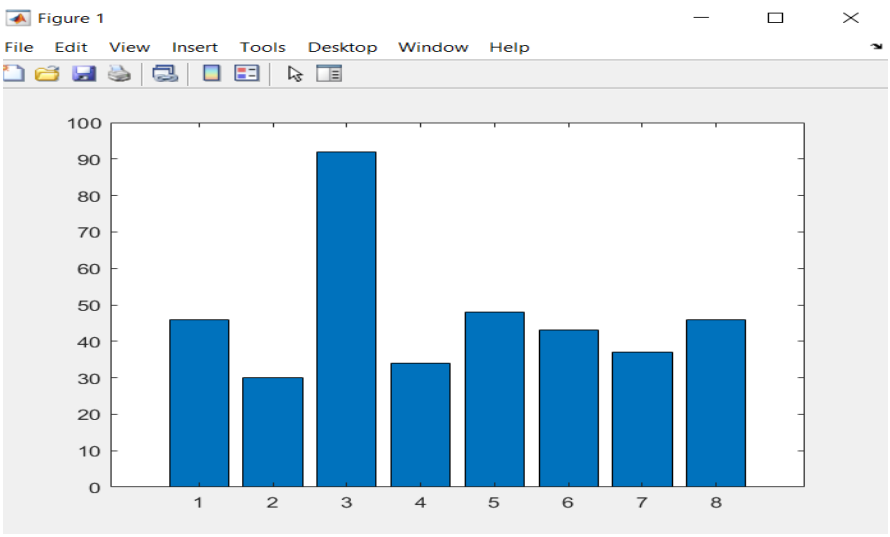


Figure 1
File Edit View Insert Tools Desktop Window Help
100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8

```
Des1 R 401
Des1 C: 128

Des2 R: 400
Des2  C: 128

Des3 R: 400
Des3 C: 128

Des4 R: 376
Des4 C: 128

Vocab R: 8
Vocab C: 128
-----------------
Error Matching with Image 1 - Husky_1: 21.375
Error Matching with Image 2: Husky_213.25
Error Matching with Image 3: Puppy_122.75
-----------------
-----------------
Best Match of Image 4 is with Image 2
```

Discussion:

Codebook is formed by using k means algorithm by extracting SIFT features. Each feature vector centroids represent centroid. Code words are created for all given images. histogram matching is done of husky3 is done with all other 3 images. We can infer from the above screenshot that husky 3 has a best match with husky 2. There is a least error that is 13.26 as compared to others.

Visually also, if we look the histogram of husky 3 husky 2 matches the most.

References:

1. https://sites.ualberta.ca/~chipuije/pca_cpp.html
2. https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
3. https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html
4. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html