EE-569

INTRODUCTION TO DIGITAL IMAGE PROCESSING

HOMEWORK #3

DIXITH REDDY GOMARI
USC ID-3098766483
gomari@usc.edu

Problem 1: Texture Analysis and Segmentation

1.1 Motivation:

- Texture classification and segmentation is an important technique in computer vision applications.
- These techniques are used after preprocessing of the image (filtering, denoising etc.,) for image analysis.
- They are mainly performed for object recognition in images.
- Basic approach in these techniques is extracting the energy information in the pixels of the images to classify under various classes.
- Once the training is done we use the feature vectors of the training images to classify the test images.

a) Texture Classification:

1.2 Approach and Procedure:

- We are given 5 1D kernels which we need to use to generate 25 5X5 kernels by tensor product of each other. The 5 1D kernels are as follows:

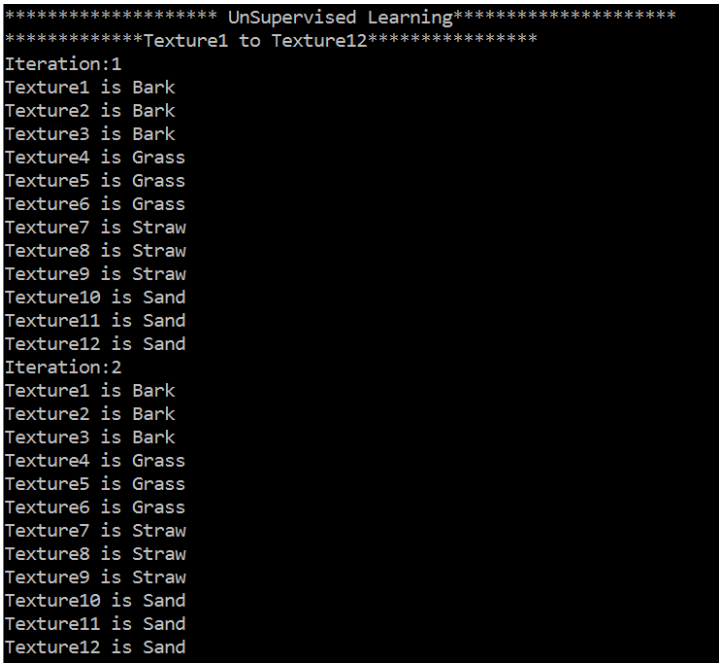
Name	Kernel
L5 (Level)	[1 4 6 4 1]
E5 (Edge)	[-1 -2 0 2 1]
S5 (Spot)	[-1 0 2 0 -1]
W5(Wave)	[-1 2 0 -2 1]
R5 (Ripple)	[1 -4 6 -4 1]

- Once we get the 25 5X5 kernels we need to use each filter to extract the energies in the given texture images.
- Convert the image format to double and find the global mean of the intensities of the pixels present in the image.
- Subtract the global mean from each intensity in the image.
- Appropriate boundary extension to the image needs to be done to apply the 5X5 filters.
- Perform image convolution with each of the 5X5 kernels and modify the center pixel of the image.
- Once we get the convoluted image, find the energy by adding up all the squares of the values of the convoluted image and finally dividing it by the dimensions of the image.

$$E = \frac{1}{N^2}\sum feature^2$$

- Since we have 12 training images we get 12X25 energy feature vector, which should be used for training.
- Here we are using supervised and unsupervised learning.
- For Unsupervised learning, we must randomly pick 4 energy feature vectors and form a codebook.
- Perform k means clustering on the data assuming the 4 random points as our centroids of the clusters.
- For the points with minimum Euclidean distances to the point in the codebook, that point needs to be classified into that respective class.
- Once we get all the points classified, calculate the mean of those points in those respective classes and update your codebook with those values.
- Iterate this process until the values in the codebook converges to some values.
- For Supervised Learning, we know that our training images belong to a certain class, using this information codebook is formed by just the mean of all those points in the certain class.
- Once the codebook is obtained, this needs to be used to classify our test images.
- This can be done by calculating the Euclidean distance between the codebook and normalized the energy vector of the test images, in this case a 6X25 vector.
- As k-means classifies based on minimum distance to mean, the 6 test images get classified based on the codebook.
- We were asked to reduce the feature dimension from 25 to 3 using PCA.
- Principal component analysis reduces the dimensions by only selecting the important features of the lot.
- Its basic principle is to find the components so that the variance of the data is maximized.
- Once it is done perform the k-means clustering on the images. Results are shown below.

1.3 Results:



```
************TextureA to TextureF***************
TextureA is Sand
TextureB is Straw
TextureC is Bark
TextureD is Grass
TextureE is Sand
TextureF is Straw
```

```
******************Supervised Learning************************
************TextureA to TextureF***************
TextureA is Sand
TextureB is Straw
TextureC is Bark
TextureD is Grass
TextureE is Sand
TextureF is Straw
```

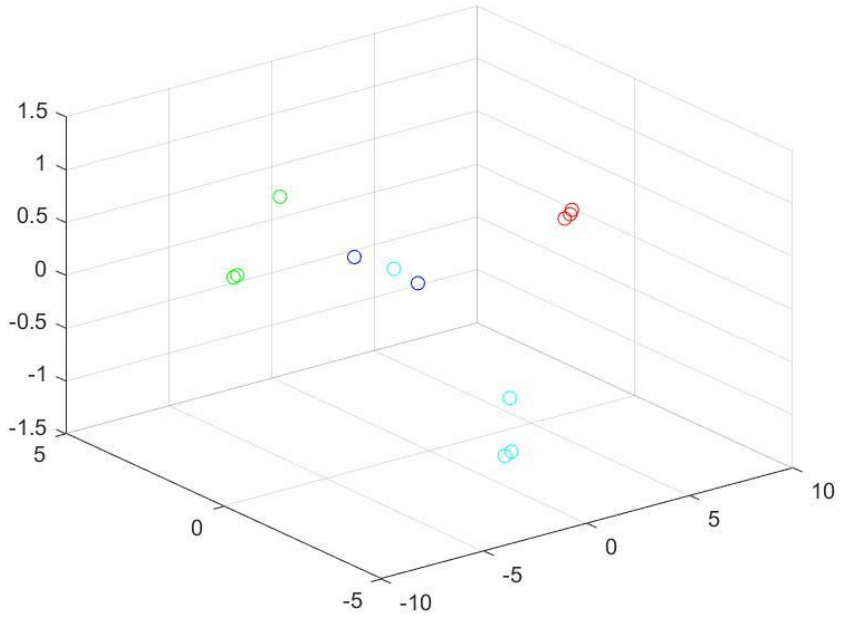


Figure 1 Clustering of the feature vector after PCA

```
idx_3features =

     4
     2
     2
     4
     4
     4
     3
     3
     3
     1
     1
     1
```

```
idx_25features =

     1
     1
     1
     4
     4
     4
     2
     2
     2
     3
     3
     3
```

1.4 Discussion

- We have used PCA to reduce the dimensions from 25 to 3.
- As we can see in the results, reduction of dimensions wrongly classifies one image.
- This happens because, there might be a case where we need more dimensions to define a class.
- Increasing the dimensions will increase the accuracy, but at certain point, there will be overfitting with too many dimensions.
- Selecting the number of features to define a sample is a very important factor in any classification problem.

b) Texture Segmentation:

1.5 Approach and Procedure

- Similar to that of the texture classification, we use the 25 5X5 kernels obtained by the tensor product of the 5 1D kernels.
- We get 25 corresponding gray scale images for each of the 25 kernels.
- In the next step, we need to calculate the energy feature vector in which we use a window method to get the energy matrices.
- I have taken 13X13 window which is centered at the pixel we want to get the energy for.
- Apply the kernels one-by-one on the image within the search windows, which gives 25 matrices which have energy produced by each kernel of every pixel.
- So, we have obtained a 25-dimensional vector to define each pixel.
- We can notice that all the energy vectors of all the kernels after normalization has a 0 mean except for that of $L5^{T}L5$. So, the feature obtained by $L5^{T}L5$ is not useful.
- To compensate for this, we use this feature vector to normalize the features of every other kernel.
- Final step in the texture segmentation is K-means clustering of the data obtained in the 25-dimensional energy vector.

- I have divided the total image into 5 classes. Each color distribution over image represents the similar pixels in the image.
- The results obtained by applying this algorithm on the kitten's image is as shown below.

1.6 Results

Figure 2 Segmentation for Window Size 13

Figure 3 Segmentation for Window Size 15

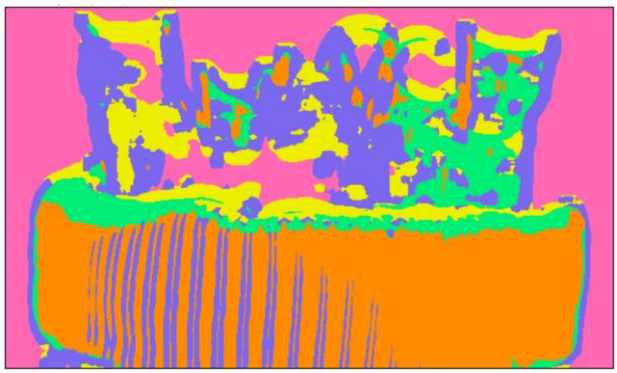


Figure 4 Segmentation for Window Size 21

1.7 Further Improvement

- I have used PCA for further improvement of classification of similar objects in the images.
- PCA reduces the feature dimension from 25 to 3 dimensions via selecting the values with maximum variance in the distribution.
- We can use various other methods to classify the objects properly with the blobs.
- The effective result of the PCA for a window size of 21 is given below

Figure 5 Image Segmentation after PCA

1.8 Discussion

- Image segmentation is done by the analysis of the feature vectors and matching the similar features based on their energy.
- We can see different outputs for different window sizes.
- We can clearly see that as the window size increases texture segmentation converges to a proper blob of an image.
- With further improvement, due to PCA, the blobs are even more visible in this case.
- Feature identification increases the classification accuracy, but further reduction also decreases the accuracy.
- Increase in the feature dimension increases the accuracy, but only till a certain point, so there is a trade-off between selecting the dimensionality of the feature vector.

Problem 2: Edge and Contour Detection

2.1 Motivation:

- Edge detection is an important objective in computer vision applications.
- It is mainly used in applications such as object recognition, object detection, scene detection, object tracking etc.
- Edges can be defined as a sudden transition in the image pixel intensity.
- Canny edge detector is an extension of the Sobel edge detector.
- Instead of single threshold in Sobel, Canny uses double hysteresis threshold to identify the edges.
- Canny edge detector is relatively very fast when compared with Sobel operator.
- Coming to the structured edge detection, it uses patches of the image instead of single pixels to identify the edges.
- It uses random forest approach to identify the structure of the edge which is more effective in identifying the edges.

2.2 Approach and Procedure:

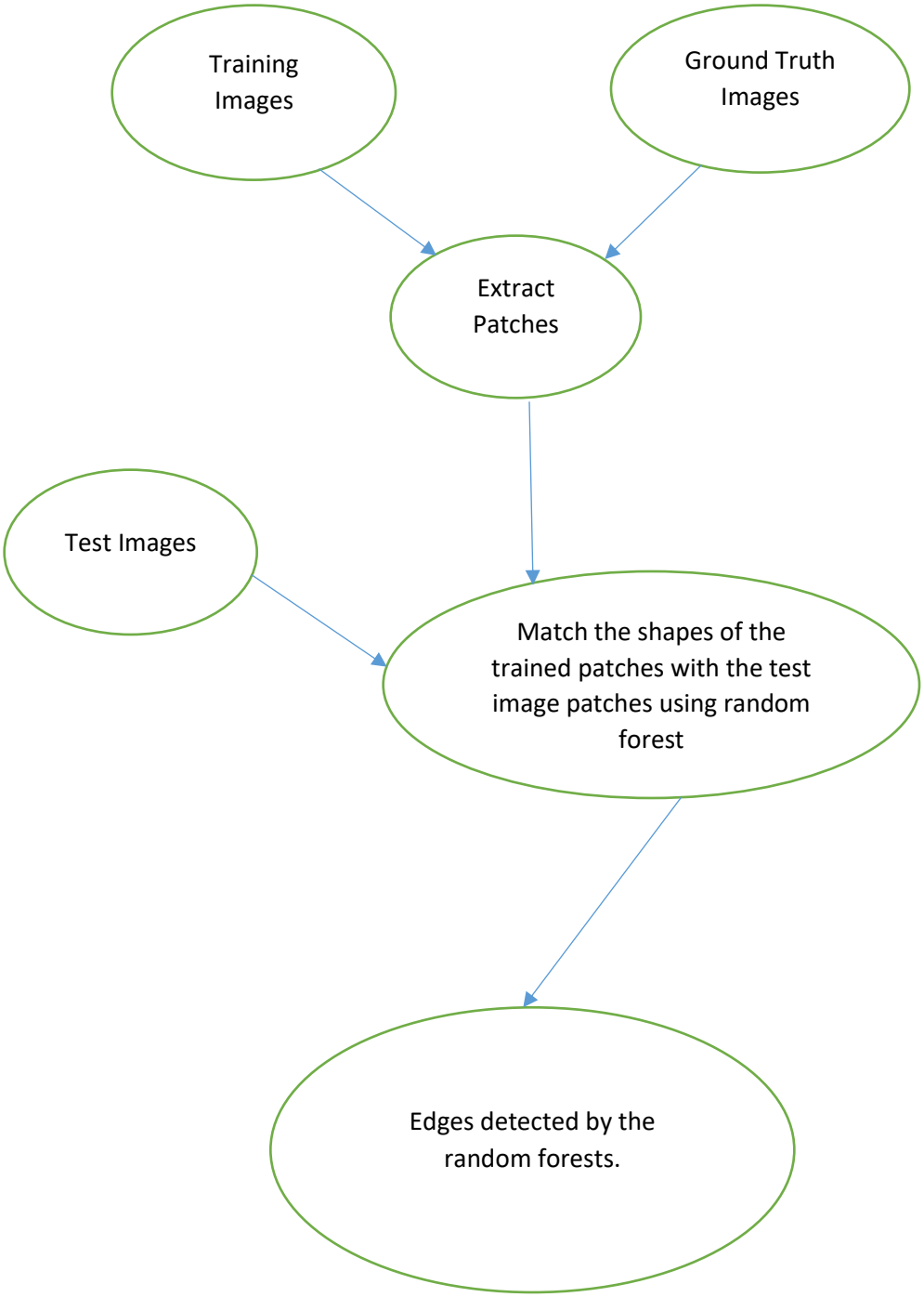
a) Canny Edge Detector:

- Canny edge detection uses hysteresis thresholding to identify the edges in the images.
- We need 2 thresholds to identify the edges in the canny edge detector.
- A pixel is said to be an edge if it is above the upper threshold.
- If the pixel intensity is in between bother the threshold values, it checks for the neighbors' threshold, if it stays above the lower threshold, it classifies it as an edge.
- Basically, it checks for the continuity in the edges if it is above the lower threshold.
- I have tried the canny edge detection with different threshold values.

b) Structured Edge Detection:

- Structured edge detection used random forest technique to identify the edges in an image.
- This detector assumes a patch around the pixel and identifies weather it is an edge or not.
- Initially we have sketch tokens which are used to build random forests to identify the edge patches.
- A sketch token defines different shapes possible for the patch to be an edge.
- Initially we train the random forest by taking the patch from the input and ground truth images, which form a structural output to classify a patch as an edge or not.
- Random forest basically has several trees, so when a test patch is passed through various decision tress, each tree will classify it as some sort of an edge or not.
- Robustness increases when we use random forest, instead of a single decision tree.
- It reduces the probability of classifying the patches wrongly.
- When the patch is passed through a decision tree, it classifies as an edge path which is stored at the end node of the decision tree.

The flowchart for structured edge is as given below:



c) Performance Evaluation:

- First we get the binary image obtained by the edge detectors.
- We obtain a probabilistic output from the canny and structured edge detectors, therefore it needs to be converted to binary before processing.
- I have tried various thresholds for the structured edge and canny edge detectors and found the precision, recall and F measure values respectively for both boat and castle images, using the following formulas:

$$\text{Precision}: P = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Positive}}$$

$$\text{Recall}: R = \frac{\#\text{True Positive}}{\#\text{True Positive} + \#\text{False Negative}}$$

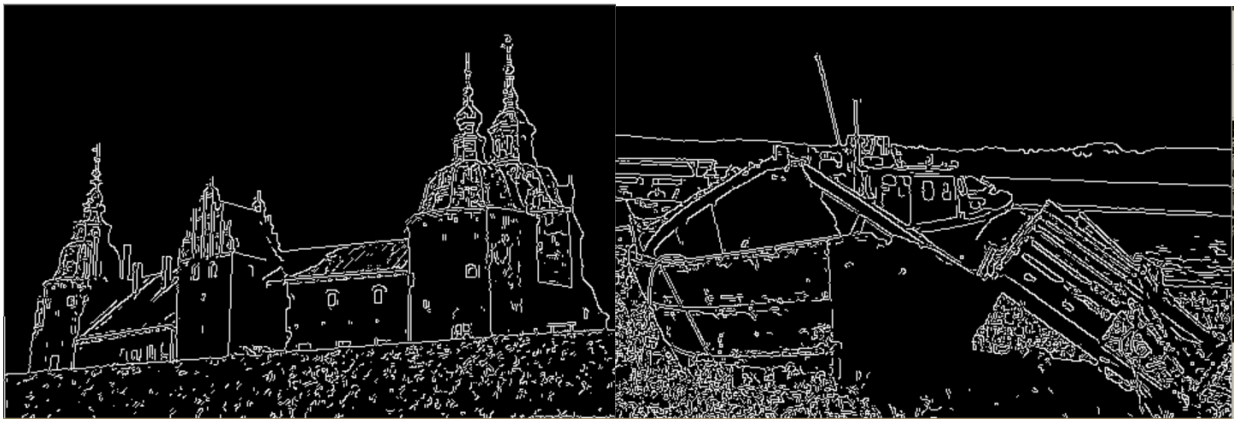
$$F = 2 \cdot \frac{P \cdot R}{P + R}$$

2.3 Results

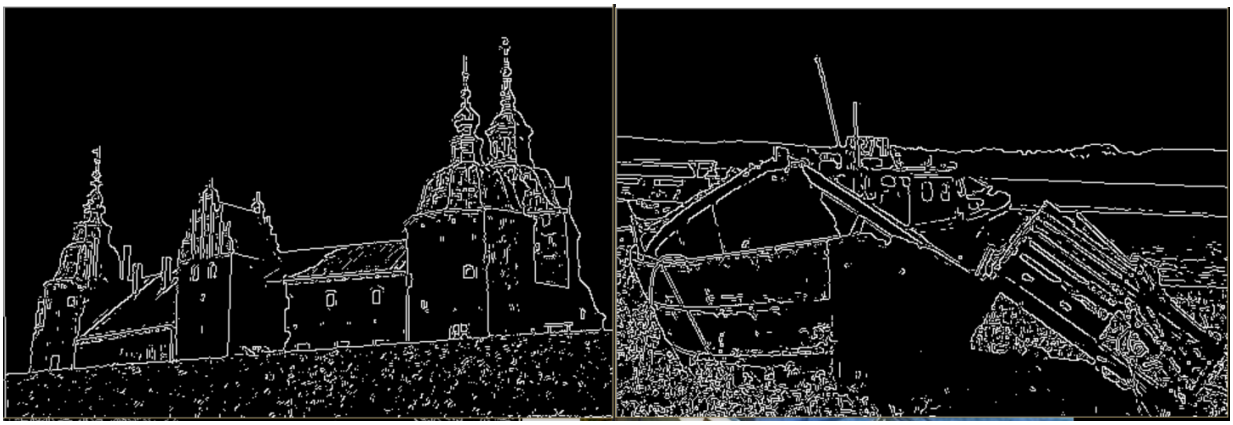
Figure 1 Original Castle and Boat Images

For upper threshold=240, lower threshold=150

For upper threshold=210, lower threshold=150



For upper threshold=200, lower threshold=190



For upper threshold=240, lower threshold=230

Structured Edge outputs

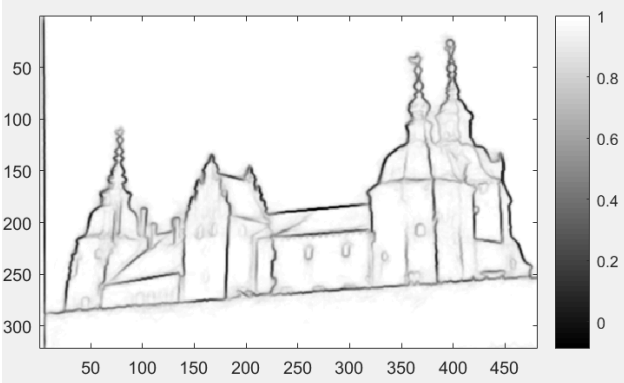


Figure 2 Probabilistic image of Castle

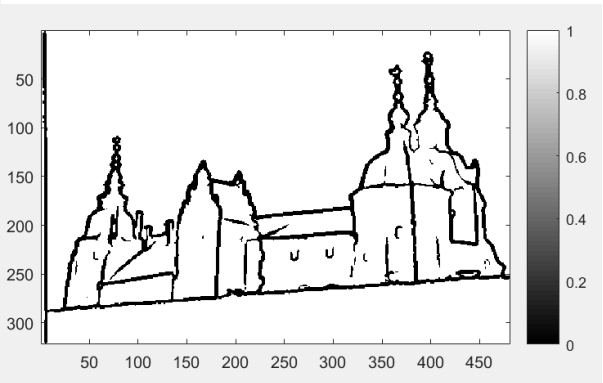


Figure 3 Binary Image of Castle

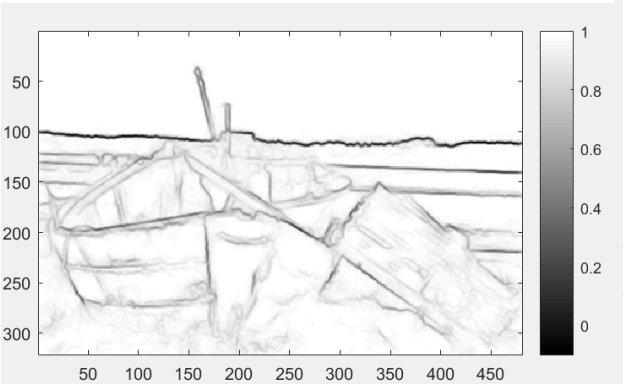


Figure 4 Probabilistic Image of Boat

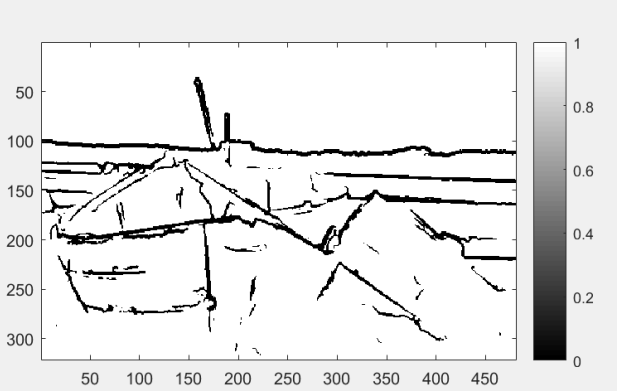


Figure 5 Binary Image of Boat

Precision, F measure and recall for Structured Edge Image for a Threshold of 0.27

Boat Image Threshold=0.27				Castle Image Threshold =0.27			
Precision	F Measure	Recall		Precision	F Measure	Recall	
0.6216	0.4569	0.3612		0.5218	0.6665	0.9223	
0.592	0.579	0.5665		0.5218	0.6694	0.9336	
0.6332	0.4574	0.358		0.7125	0.7711	0.8401	
0.6665	0.5942	0.536		0.8126	0.8024	0.789	
0.4953	0.5151	0.5365		0.7044	0.773	0.8563	
0.6553	0.5783	0.5175		0.7741	0.788	0.8024	
0.61065	0.53015	0.47928		0.67453	0.74507	0.85728	Mean Values

PFR values for Boat image for UT=60 and LT=40

Image	Precision	Recall	F_Score
Boat_gt1	0.08992	0.31107	0.13951
Boat_gt2	0.03759	0.065095	0.06509
Boat_gt3	0.08288	0.28004	0.12791
Boat_gt4	0.05552	0.28222	0.09279
Boat_gt5	0.02585	0.18551	0.04538
Boat_gt6	0.03586	0.17938	0.05977
Mean	0.0546	0.2468	0.0884

PFR values for Boat image for UT=30 and LT=250

Image	Precision	Recall	F_Score
Boat_gt1	0.03103	0.01628	0.02136
Boat_gt2	0.03131	0.02837	0.02977
Boat_gt3	0.05766	0.02832	0.03799
Boat_gt4	0.04846	0.03407	0.04001
Boat_gt5	0.04523	0.04138	0.04322
Boat_gt6	0.02343	0.01690	0.01963
Mean	0.0395	0.0276	0.0320

PFR values for Castle image for UT=60 and LT=40

Image	Precision	Recall	F_Score
Castle_gt1	0.0820	0.1827	0.1132
Castle_gt2	0.0799	0.1484	0.1038
Castle_gt3	0.0895	0.0952	0.0923
Castle_gt4	0.0879	0.0797	0.0836
Castle_gt5	0.1002	0.1187	0.1087
Castle_gt6	0.1628	0.1482	0.1552
Mean	0.1004	0.1289	0.1095

2.4 Discussion

a) Canny Edge Detection

- Let us consider the case of the canny edge detection, where we can see different edge maps of the boat and castle images.

- As we can see from the results, the objects could be easily identified for different threshold values. But for lower values of threshold we get various unwanted edges which results in failure of the object recognition.
- Similarly, for higher threshold value some of the weaker edges maybe lost which again may result in the failure of the object recognition.
- We cannot tune the parameter to show only the edges of the object, as the threshold values at either extremal will result in either fainting of the edges or overcrowding of the edge map.
- Image content is greatly affected by the threshold values of the canny edge detector.
- Coming to the texture of the image, we cannot correctly preserve the texture of the image at either extremal values, there should be trade off value where we can clearly see the texture of the images.
- Threshold of the canny edge detector affects the smoothness of the image. With lower threshold, we can see unwanted edges which decreases the smoothness of the image. While for higher threshold edges decrease increasing the smoothness.
- Weak boundary regions can be identified when the threshold is lower. As the threshold value increases the weak boundary regions get eliminated. In order to identify we need to have lower threshold values.

b) Structured Edge

- We can clearly see that SE method gives much better outputs than that of the outputs obtained from canny.
- It gets an excellent output because if the random forest approach which reduces the probability of error in assignment of the edge.

c) Performance Evaluation

- We can see the F measure values generated for all ground truth images for both castle and boat for both structured edge and canny edge detectors.
- F measure of the boat is very less when compared to that of the castle image. Since there are no other objects covering the castle image, and there are few objects which shadow the boat, we can intuitively say that F measure for boat is maximum.
- F measure is a measure of accuracy. Both precision and recall affect the F measure. Since P is inversely proportional to R we need to find a trade-off between P and R value to have a high F measure.

P+ R = C, then P = C -R

F = 2. (P.R) / (P + R)

F = 2. (C- R) *(R) / C

Take derivative, and F'= 0. Then, 2C – 4R = 0. R = C/2. So, when R = P, F- Measure is maximum.

Problem 3: Salient Point Descriptors and Image Matching

3.1 Motivation:

- SIFT and SURF are used in applications such as object recognition, motion tracking and navigation etc.
- Extracting salient features is a very trending topic in machine learning and computer vision applications.
- In this problem, we use SIFT and SURF techniques to identify the salient features in the given images.
- We also do the object matching using the key point descriptors between several images.
- We build the bag of words to find how close the images are related.

3.2 Approach and Procedure:

a) Extraction and Description of Salient Points

SIFT Algorithm

- SIFT is used to detect the scale invariant descriptors in an image as the name suggests.
- SIFT is invariant to scaling, rotation and illumination etc.
- It mainly consists of four parts:
  1. Scale Space Extrema Detection
  2. Key Point Localization
  3. Orientation Assignment
  4. Generation of Key Point Descriptors

Scale Space Extrema Detection

- To make the image key points scale independent, extraction of key features is done on various scales of Gaussian parameters.
- Let's say we consider 5 octaves, and 3 images in each octave. For every octave there is a down sampling by 2.
- Sigma for these filters is considered as 1.6 and each octave has a sigma with a multiple of this sigma of k, which is given as 1.414.
- We can even use Laplacian of Gaussian (LoG) to extract the scale space detection, but it takes a lot of time to compute.
- Therefore, we use Difference of Gaussians which is approximate measure of LoG, in which we find the difference between the output produced by the Gaussian filters within the same octave.
- DoG gives us an edge map, in which we iterate multiple times to find the points of interest we must take the local minimum and maximum at a given scale into consideration.

Key Point Localization

- In key point localization, we use the points that are of interest from the first step of the algorithm.

- We compare these interest points with hessian threshold, which eliminates noisy and weak points.

Orientation Assignment:

- Initially we assign an orientation value to the key points that are obtained from the second part based on the neighborhood of the points of interest.
- Size of the neighborhood depends on the octave at which the key point is detected.
- Then we consider 36 bins, with each bin having a difference of $10^{o,}$ which represents 360^{o}.
- Orientation is then obtained by the gradient magnitude of the pixel that we are calculating.
- Out of all the orientations assigned, we consider the maximum orientation value.

Key point descriptors:

- To get the key point descriptors we need to consider 16X16 neighborhood from the point of interest which is divided into 4X4 units from which histograms of the pixels is calculated.
- Resolution of histogram is taken as 8.
- This results in 128 bins in the 16X16 neighborhood, which are taken as the feature vectors for a key point.

SURF Algorithm

- SURF is similar to that of SIFT, but it is faster and robust against image transformations.
- There will be speeding up of filtering due to the image integration and then it uses a blob detector in finding the interest points based on hessian matrix.
- We get faster filtering as the mask convolutions is independent of the size of the filter.
- The process is similar to that of the sift until you find the determinant of the hessian matrix.
- Next step is to find the major interest points in which we use 3X3 non-maximal suppression below and above the scale of each octave.
- We need to interpolate the interest points as the scale space is large and irregular shaped to obtain correct scale.
- In order to find the feature direction, we need to use the haar transform and find the max weight to obtain the feature direction.
- Eventually we need to generate 64-dimensional feature vectors for 16 sub regions which is generated by the square description window.

Bag of Words Algorithm:

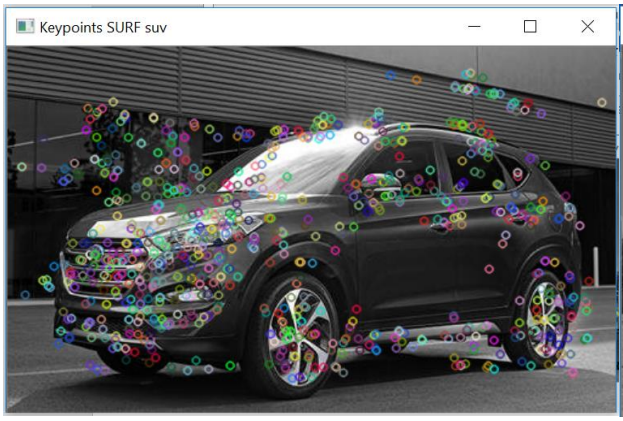
- We are given 4 images to perform bag of words. We were asked to form 8 clusters for which we have extracted the SIFT key points and descriptors.
- All these were added to the bag of words trainer.
- Clustering is done via k-means clustering algorithm to build the dictionary which describes any image.
- Once the dictionary is obtained, we get the histograms of the images which has 8 bins.
- Now we take the test image and follow the above steps to obtain the histogram of it.

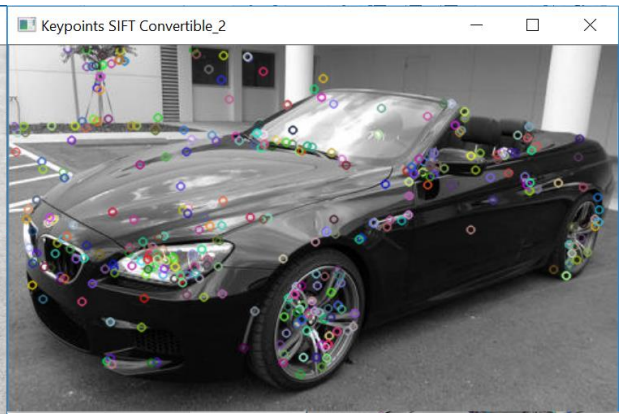
- Now we compare the histograms of the test image with the histogram of the training images. The value closest to that of the training images is what it belongs to.

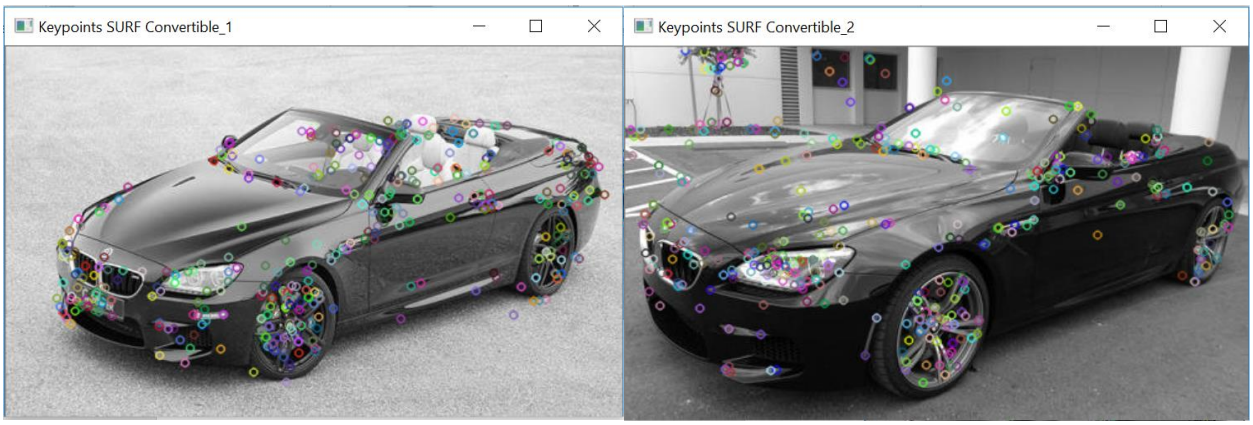
Distance can be calculated with the following formula.

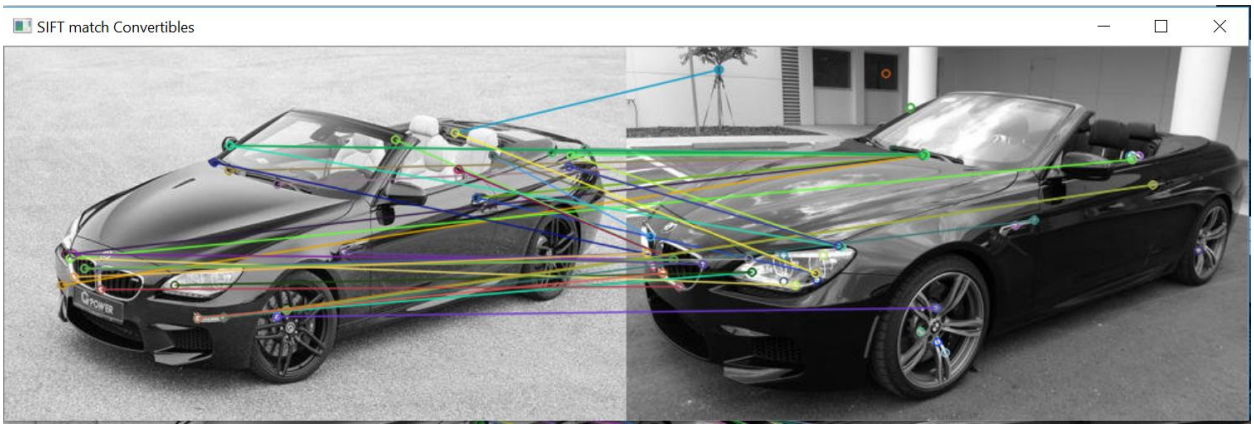
$$D = \frac{1}{8} * \sum_{i=0}^{7} xi - yi$$

3.3 Results

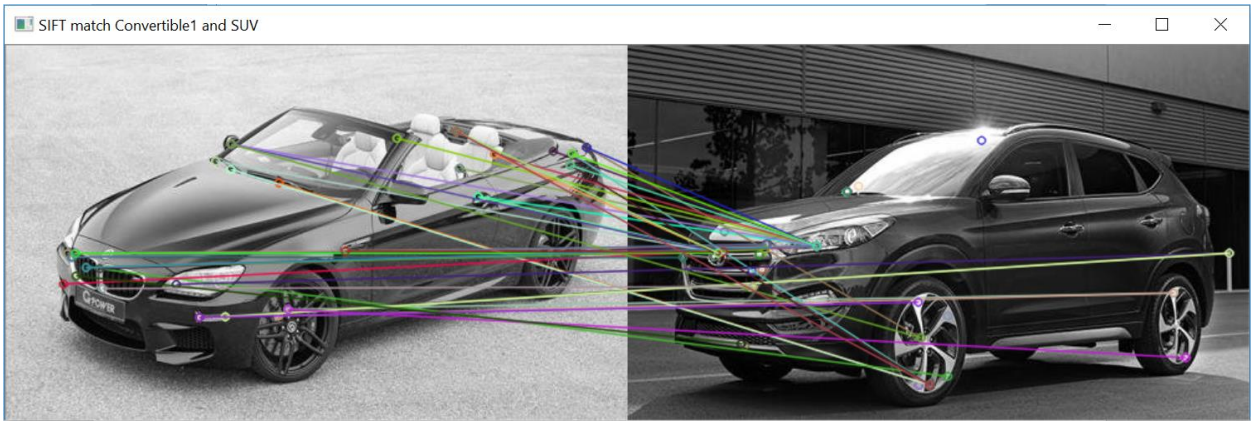


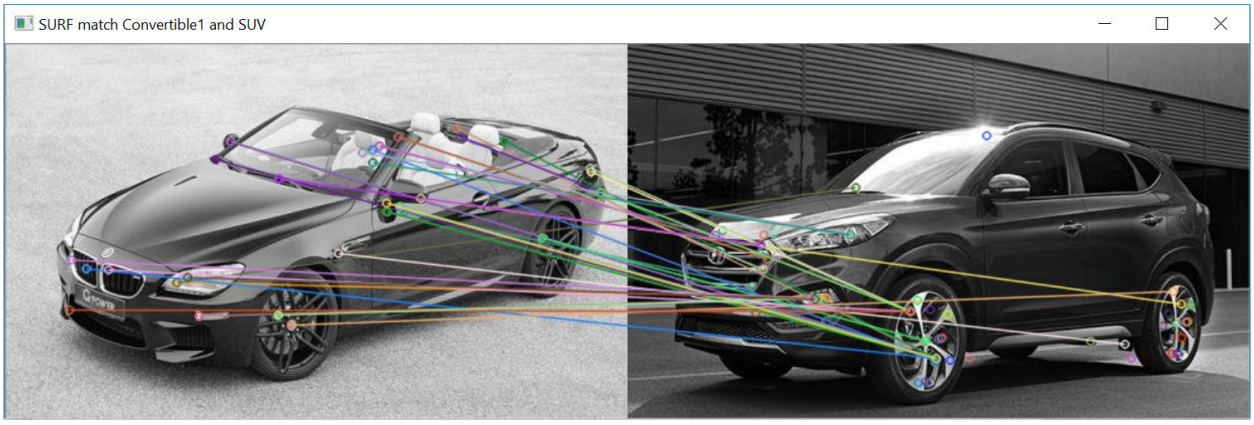


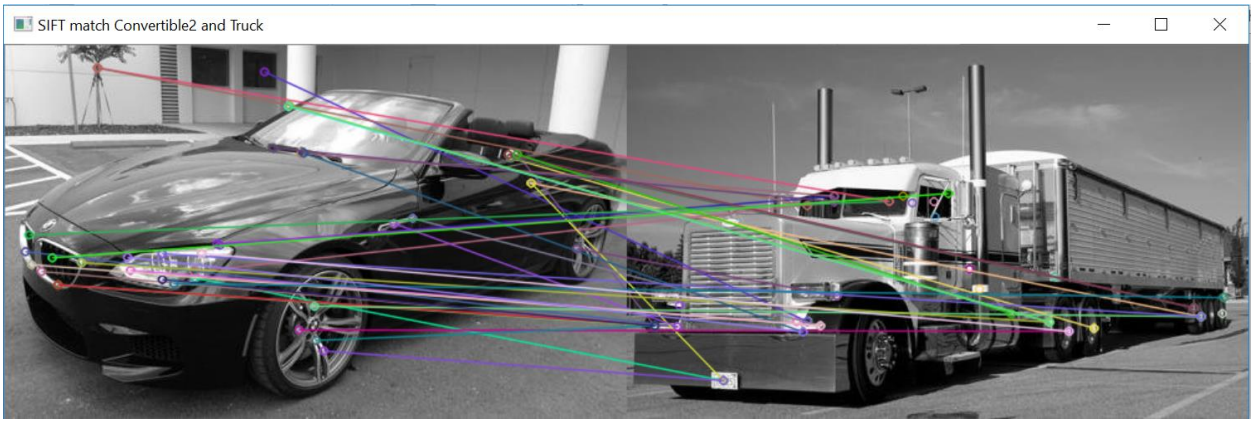

Keypoints SURF Convertible_1
Keypoints SURF Convertible_2

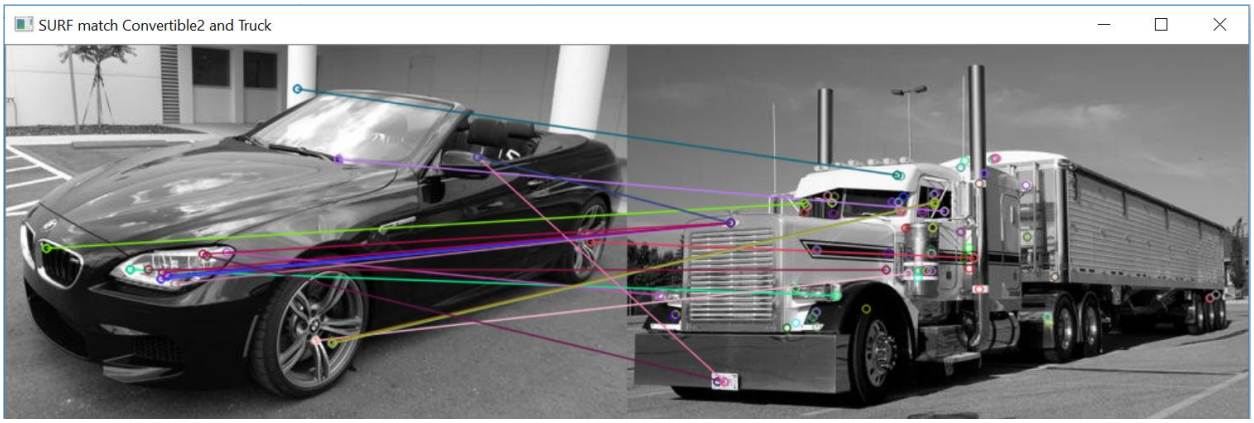

SIFT match Convertibles

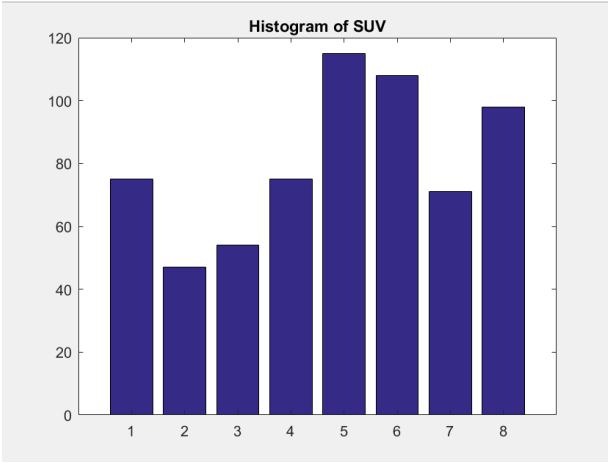

SURF match Convertibles

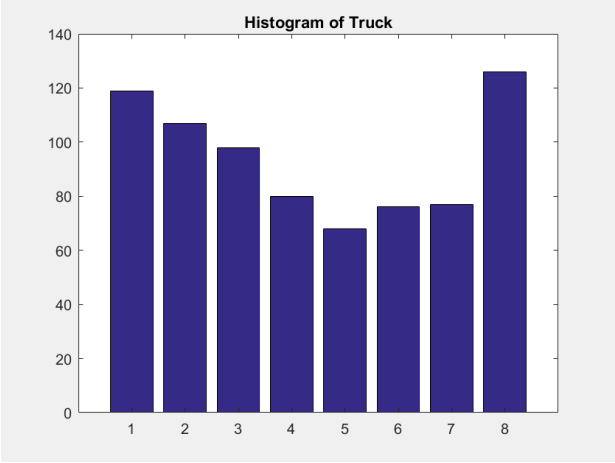

SIFT match Convertible1 and SUV

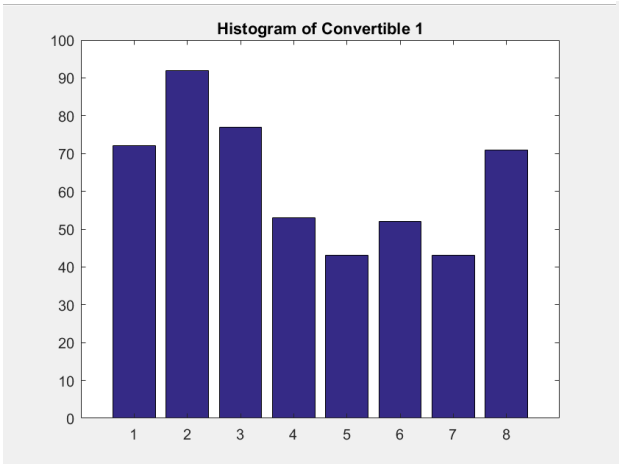

SURF match Convertible1 and SUV

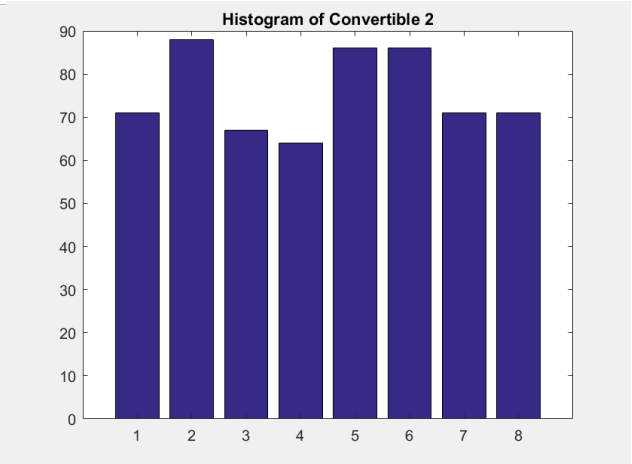

SIFT match Convertible2 and Truck












```
18.375
25.375
16.375
Convertible 2 belongs to Convertible 1
```

3.4 Discussion

- When we compare the performance of SIFT and SURF, the latter outclasses SIFT.
- SURF takes very little time when compared to the computational time of that of SIFT.
- The time complexity advantage arises because of SURF uses integral images to generate the scale space.
- As we can see in the final figure, the three numbers are the error measure of the histogram of convertible 2 to the other three histograms.
- Lower the error measure similar are both the objects.
- Hence the convertible 2 is correctly classified as it belongs to convertible 1.