Aakash Shanbhag

adshanbh@usc.edu

3205699915

AAKASH DHANANJAY SHANBHAG

Contents

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Problem 1: Texture Analysis and Segmentation.

1.1 Motivation and Abstract.

Texture classification is an easy task to the human eye but seem to be difficult problem for computers. Texture can be considered as a pattern which repeats periodically and can be classified and segmented according to the requirement. It clearly provides details of the spatial arrangement of the intensity values. There are two tasks in this problem which deal with segmentation and classification of the various textures in the images.

Classification in this problem is done by the K means algorithm which has been a key classification algorithm for over 50 years. The fundamental task includes that of extracting key feature values from the images using the Laws filters and representing this feature space. For both the problems we first apply 5x5 laws filters using the tensor product to produce 25D feature set. Clustering algorithm is then applied on this convolved feature set on the image and its performance is evaluated. Feature reduction is also carried out by Principal Component Analysis to reduce the feature vector set to a lower dimension.

In case of the second portion of the problem Segmentation of an image is carried out this seems to be a continuation of classification with different energy parameter calculation and on different window sizes. Again PCA is performed along with K means clustering to obtain detailed segmented portions of the image which are defined by the different colors. The effect of different window sizes on the segmentation quality is also discussed in brief. The last portion of the problem discusses the further improvement that can be done to this task of segmentation by different methods.

1.2 Approach and Procedure.

There are 3 parts to this problem and each part is dealt with in a different manner as discussed below through examples and figures with detailed discussion on the implementation of the algorithm for each of the cases. Textures generally obey statistically patterns which can be traced and fruitfully utilized.

1.2 (a) Texture Classification

The steps involved in this case of texture classification are mentioned below and there are two parts to the algorithm which are followed in this step. Unsupervised learning by K means is carried out on the 12 input textures for which the ground truth is completely unknown and hence since we do not have any prior training data we classify it and obtain a model which can be used as a training data.

These 12 images serve as training data which can be then used to classify the unknown 6 textures and hence by the nearest neighbor approach help in suitably classify the images specific to the

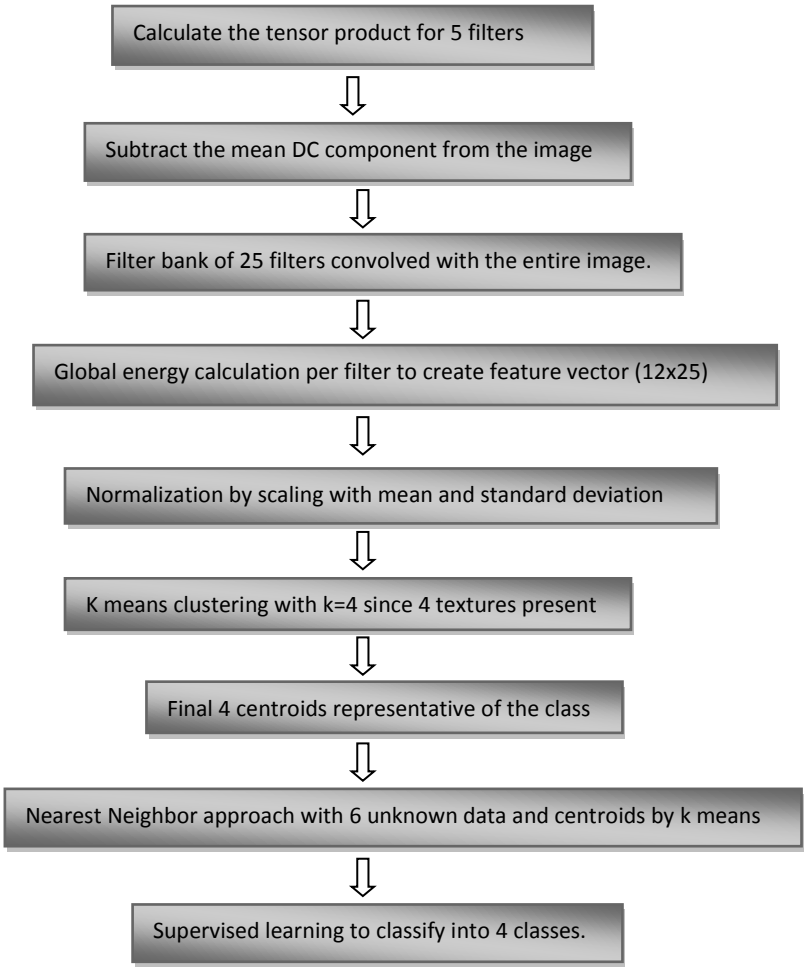
Aakash Shanbhag

adshanbh@usc.edu

3205699915

clusters. Since in this step we are using the 12 images as ground truth it falls under the Supervised learning approach.

The steps involved in carrying out the classification are described in the flow diagram below:



According to the steps mentioned above we firstly carry out some pre-processing steps which involve the subtraction of DC component in the image from the entire image as it hardly provides any significant information and such redundancy can be easily removed with only AC components that contribute to the feature vector calculation.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Tensor product calculation is carried out for the 5 filters which are mentioned below 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 tensor product obtained is as follows:

L5L5

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

L5E5

-1	-2	0	2	1
-4	-8	0	8	4
-6	-12	0	12	6
-4	-8	0	8	4
-1	-2	0	2	1

L5W5

-1	2	0	-2	1
-4	8	0	-8	4
-6	12	0	-12	6
-4	8	0	-8	4
-1	2	0	-2	1

L5R5

1	-4	6	-4	1
4	-16	24	-16	4
6	-24	36	-24	6
4	-16	24	-16	4
1	-4	6	-4	1

L5S5

-1	0	2	0	-1
-4	0	8	0	-4
-6	0	12	0	-6
-4	0	8	0	-4
-1	0	2	0	-1

E5L5

-1	-4	-6	-4	-1
-2	-8	-12	-8	-2
0	0	0	0	0
2	8	12	8	2
1	4	6	4	1

E5E5

1	2	0	-2	-1
2	4	0	-4	-2
0	0	0	0	0
-2	-4	0	4	2
-1	-2	0	2	1

E5W5

1	-2	0	2	-1
2	-4	0	4	-2
0	0	0	0	0
-2	4	0	-4	2
-1	2	0	-2	1

There are 25 such combinations that can be obtained similar to the above tensor products.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

These 25 filters are convolved with the image with adequate image boundary extension by image replication and thus create a 25 D feature set. The global energy of each of these 25 stacks of 12 images is calculated using the formula:

$$Energy\ per\ image = \frac{1}{Width * Height} * \sum_{i=0}^{height} \sum_{j=0}^{width} I(i,j) * I(i,j)$$

Hence this energy of each filter is calculated which produces 25 different energies for a single image. Similarly since we are dealing with 12 images we generate a 12x25 feature vector set as shown below:

```
feature vectors calculated successfully

1.04516e+008    4.70662e+006    548891  1.26133e+006    907009  2.58991e+006
169526  34733.9 92011.5 45929.6 383101  32614.1 8876.54 25182.7 10745.5 845340
74420.8 23254.1 76906.4 26036.4 571488  43941.4 10875   1.26133e+006    13590.6


8.96661e+007    4.84849e+006    849879  2.05655e+006    1.20882e+006    2.5051e+
006     181576  50906.8 152040  59135.7 479417  44787.6 14742   46117.3 16203.3
1.22772e+006    120760  42843.2 151543  44571.6 629876  53112.6 16419.8 2.05655e
+006    18394.1

5.88683e+007    3.14759e+006    674157  1.70969e+006    883826  2.00612e+006
147340  40739.5 124639  46974.7 403694  38327.2 11786.4 36234.1 13533.3 1.04539e
+006    98009.7 33054.2 115220  35340.3 513215  45242.7 13447   1.70969e+006
15411.1

1.21071e+007    1.01506e+006    303839  877871  346663  1.05525e+006    93816.7
27782.8 78812.1 32409.2 326909  30487.2 8475.46 23776   10315.3 810328  78830.5
24556.1 78924   27851.1 376568  34301.5 9730.6  877871  11724.3

1.29677e+007    1.0749e+006     332654  962282  377520  1.09332e+006    100363
31218   90749   35169   334176  32833.1 9893.04 29086.8 11487.3 831676  84518.6
28136.2 92837.4 31108   385867  36594.5 11138.5 962282  12750.8

1.72412e+007    1.27877e+006    313852  889382  390125  1.23877e+006    102291
28221.3 84796.8 33363.5 278488  27047.5 8197.48 24961.9 9618.39 658565  66104.7
22211.5 77352.3 24263.9 361194  32807.5 9712.35 889382  11388.5

1.48021e+008    6.63238e+006    675933  674212  1.41431e+006    3.76852e+006
296377  41322   45648.7 77839   273815  25794   4615.39 6308.91 7605.21 250887
24441.3 5099.88 10724.9 7466.71 646272  58854.7 9572.48 674212  16750.9

1.48164e+008    7.16991e+006    635606  610151  1.4042e+006     4.56622e+006
367437  46129   49428.9 89169.1 318324  32170.8 5994.67 8340.73 9887.17 322455
33025.7 7380.83 16170.6 10683   734837  70751.6 11514.8 610151  20140.1

1.37544e+008    7.34845e+006    723040  655242  1.62846e+006    4.38535e+006
309363  42061.9 44777.8 79301.7 432215  34223.9 5420.09 6886.13 9376.42 415842
35656   6293.85 10779.2 10324.3 922020  68273.4 10337.7 655242  18198.1

2.88365e+007    1.54726e+006    230459  410421  372445  636226  56968   14337
29129.6 18672.5 63494.4 6505.54 2285.47 5975.38 2529.65 107189  11016.3 4560.57
15554.5 4469.22 113885  11521.4 3677.45 410421  4320.31

2.64332e+007    1.55175e+006    235897  437643  376901  596706  54971.1 13925.3
30596.6 17902   64974.6 6570.2  2363.05 6864.39 2505.53 108404  11305   4768.56
17229.8 4557.51 113768  11353.5 3638.52 437643  4137.76

2.92929e+007    1.45443e+006    243259  468896  364103  659377  60852.9 16000.6
33465.2 20485.1 69752.9 7400.92 2656.76 6815.69 2939.05 116806  12724   5453.15
17500.1 5393.1  123117  12761   4151.36 468896  4842.79
```

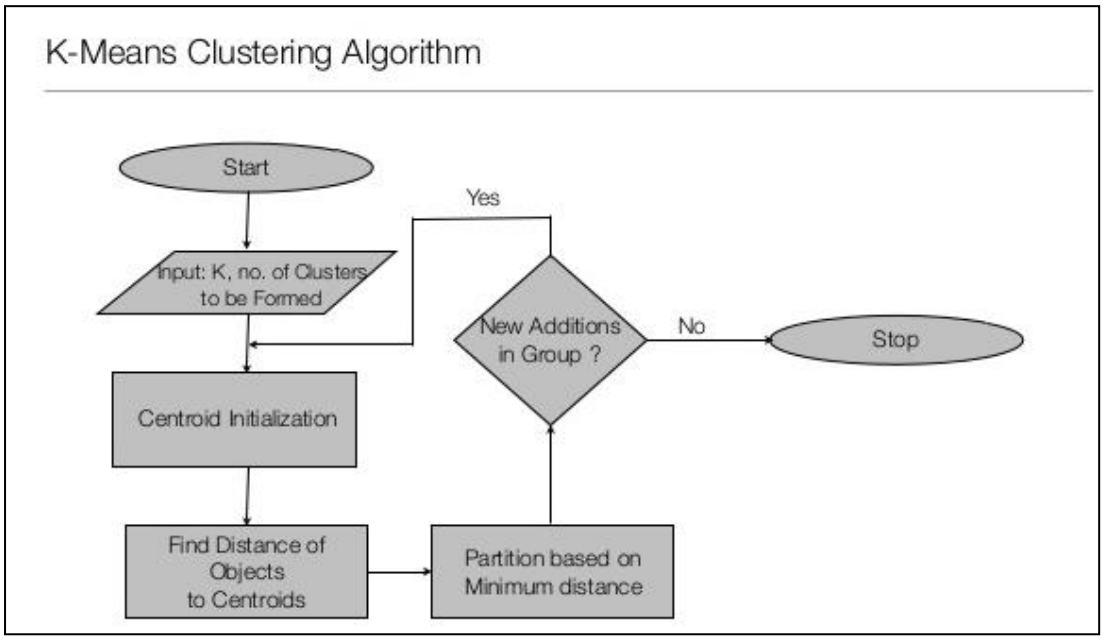
Aakash Shanbhag

adshanbh@usc.edu

3205699915

These feature vectors need to be normalized in order to eliminate inaccurate distance calculations. Hence rescaling with the help of mean and standard deviation is carried out. This is called as feature extraction.

K means clustering process is now carried out with k=4 as 4 initial centroids are used and hence with random initial guesses we obtain the final centroids after it converges to represent 4 final centroids. The K means algorithm is discussed briefly below:



Once this process is carried out each of the points in the feature space are allocated a cluster and centroid is the representative of these clusters. In this manner we can cluster the 12 images for unsupervised k means clustering.

These 12 images act like ground truths and the centroids are the representatives. Hence for the unknown 6 texture images the nearest neighbor approach can be utilized after the calculation of the feature vectors of the 6 unknown textures.

According to the least Euclidean distance to the respective classes we can easily obtain the class the respective images belong to in this method.

Principal Component Analysis (PCA) can also be utilized in order to reduce the 25 D feature space to 3 D feature space and hence it is important to consider only the most important Eigen values by the Singular Value Decomposition technique of decomposition. PCA calculates the Eigen values and Eigen vectors of higher dimensional data space and reduces it to n dimensions whose basis are the Eigen vectors

Aakash Shanbhag

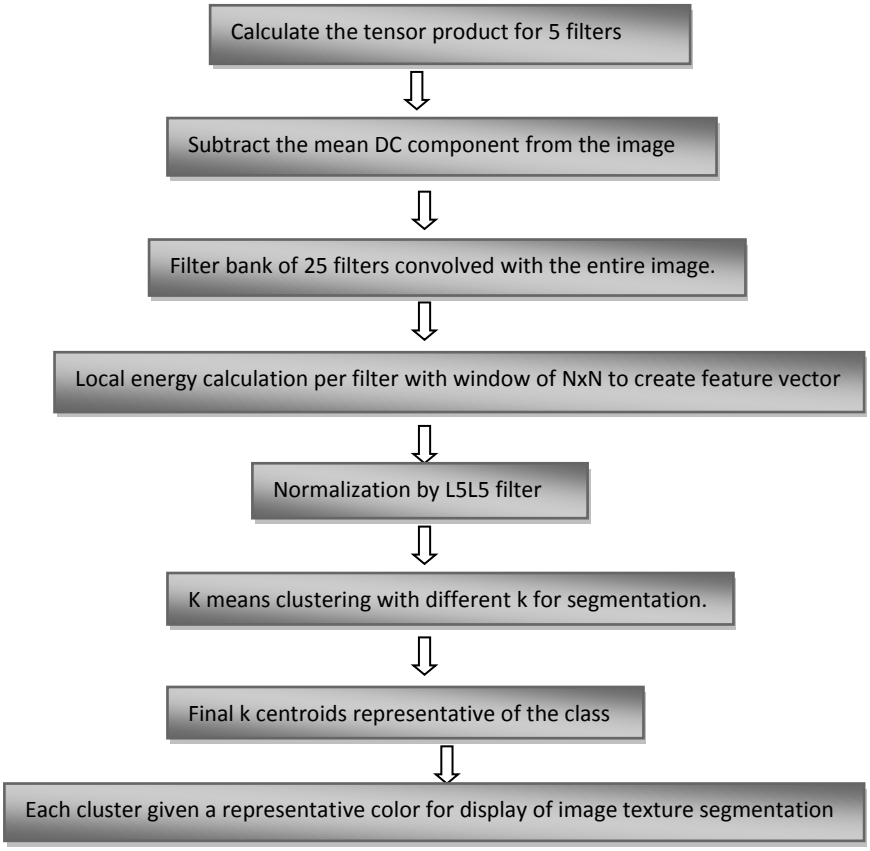
adshanbh@usc.edu

3205699915

corresponding to 3 largest Eigen values. To implement this, we utilized OpenCV's PCA function to calculate the principal components and project the high dimensional data onto a 3D plane.

1.2 (b) Texture Segmentation

The process that is carried out for texture segmentation is pretty much similar to that if the classification process with the only difference being that instead of calculation of the global means of the entire image we just calculate it over a smaller window. Since texture is generally a local property rather a global one we obtain the energy per pixel and hence generate a feature space for a single image of dimension which is equal to the width*height*25. In this example we have a single kitten image which needs to be segmented with different values of k. Again k means is applied for clustering in this example.



As seen above after the 25 filters are convolved on the image while calculating the energy a window is considered of dimension NxN which is necessary for calculation of energy per pixel. Again boundary extension is needed for this purpose. The formula needed for calculation of energy per pixel is as follows:

Aakash Shanbhag

adshanbh@usc.edu

3205699915

$$E(i,j) = \frac{1}{N*N} * \sum_{h=-(N-1)/2}^{(N-1)/2} \sum_{h=-(N-1)/2}^{(N-1)/2} I(i+h, j+h) * I(i+h, j+h)$$

Further K means algorithm is used to cluster this feature vector of size= (width*height)*25 into k clusters with each representative cluster in the image given a specific color. Hence the different textures in the image are given k different colors. The choice of color is arbitrary and depends only on the number of clusters we want to create. It is important to normalize the feature vector by L5L5 to remove the effect of high feature values. Different window sizes are considered and the outputs are shown below.

1.2 (c) Texture Segmentation Improvements.

Principal Component Analysis (PCA) is carried out for this process for the best window size of 15x15, and the process followed for PCA are mentioned above. One thing to keep in mind is that all features are not important to classify the textures. This gives us a chance to reduce to dimension of the feature vector.

Find principal components (directions) so that the variance of the data is maximized. Now, project the entire samples into a subspace different from initial one. Computation of Covariance matrix and then compute eigenvectors and Eigen values via SVD (Singular Value Decomposition).Now, we perform the sorting algorithm to sort the Eigen values we computed to descending order. Select the largest k Eigen values. Projection matrix is now set according to the data of the Eigen values.

It is important to calculate this PCA on a window size that is large enough so those local regions of textures are taken into consideration.

Aakash Shanbhag

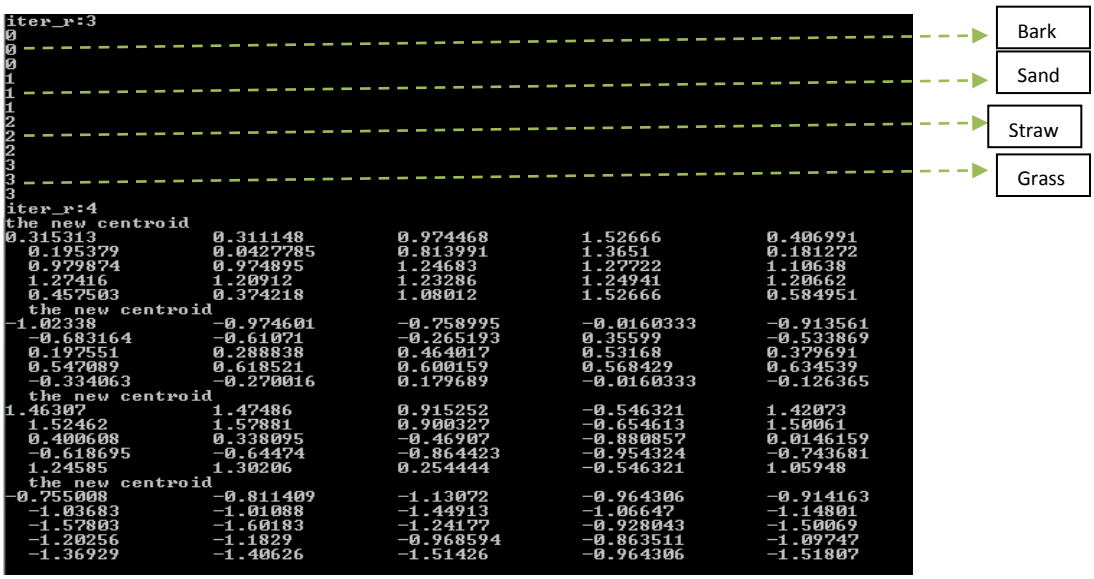
adshanbh@usc.edu

3205699915

1.3 Experimental Results.

1.3 (a) Texture Classification

After application of K means we obtain the following output as seen below



The Index above indicates each individual type as a part of unsupervised learning or training for supervised learning portion. The final centroids are also shown along with the index and hence we can state the following about the unsupervised output:

Texture1	**Bark**
Texture2	**Bark**
Texture3	**Bark**
Texture4	**Sand**
Texture5	**Sand**
Texture6	**Sand**
Texture7	**Straw**
Texture8	**Straw**
Texture9	**Straw**
Texture10	**Grass**
Texture11	**Grass**
Texture12	**Grass**

Aakash Shanbhag

adshanbh@usc.edu

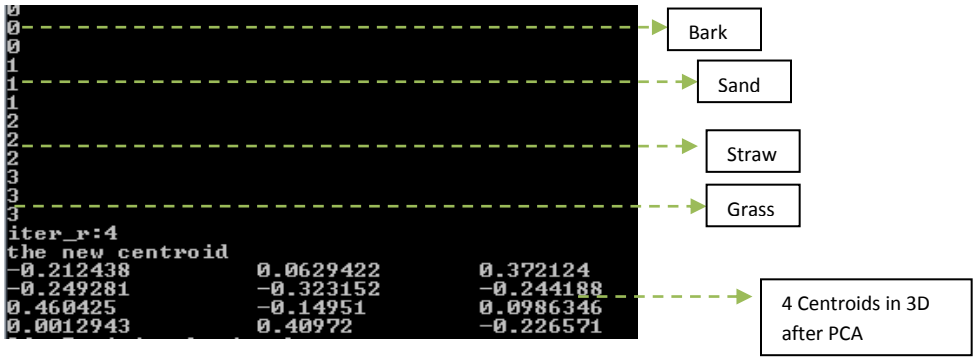
3205699915

Now in case of the Supervised learning portion the centroid obtained above acts like the training data and the output for the 6 unknown textures is as follows:

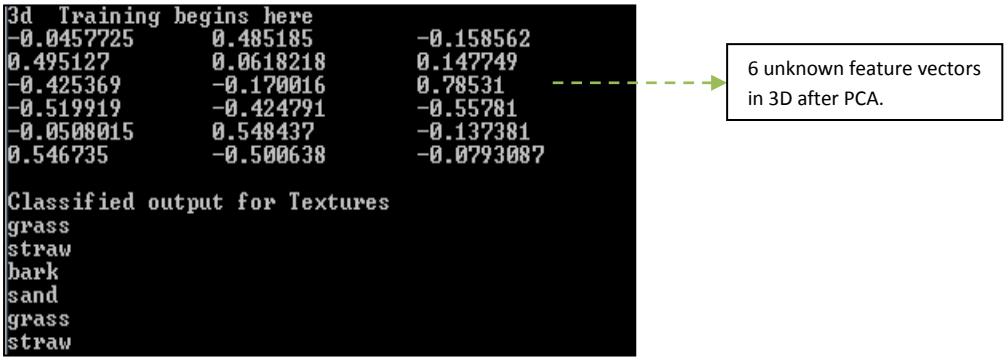
```
Classified output for Textures
grass
straw
bark
sand
grass
straw
```

TextureA	**Grass**
TextureB	**Straw**
TextureC	**Bark**
TextureD	**Sand**
TextureE	**Grass**
TextureF	**Straw**

After application of PCA we convert from 25 D to 3 D feature space and the output of the PCA



Similar to above example we obtain the same outputs as previous ones without PCA.



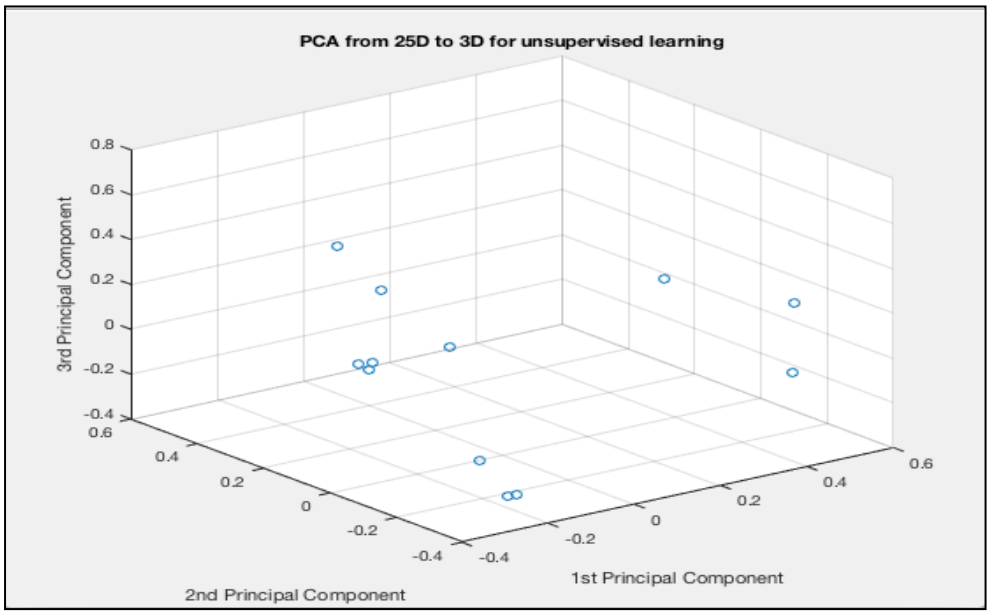
The Supervised learning algorithm works faster with 3D and produces the same results.

Aakash Shanbhag

adshanbh@usc.edu

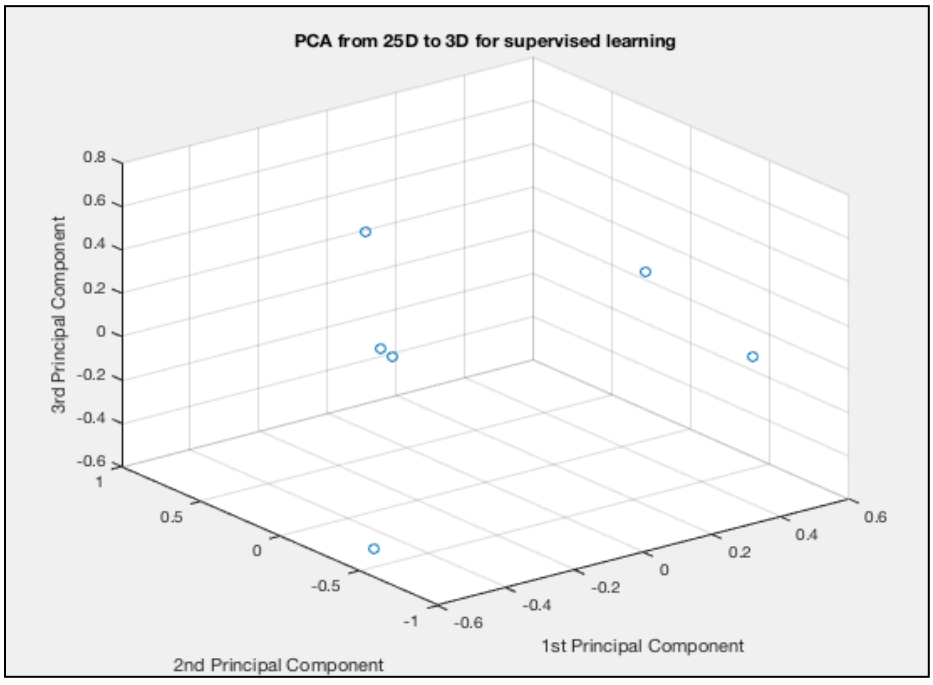
3205699915

Output of the PCA conversion from 25 D to 3D for unsupervised 12 feature points



.

Output of the PCA conversion from 25 D to 3D for supervised 6 feature points for unknown textures.



Aakash Shanbhag

adshanbh@usc.edu

3205699915

1.3 (b) Texture Segmentation

The input image used in this case was this case was the kitten.raw image which is first converted to gray scale which is shown below. Two different k values were tested for this image with window sizes of 13, 15 and 21 respectively and the outputs are compared.

Fig 1.1: RGB kitten.raw image.

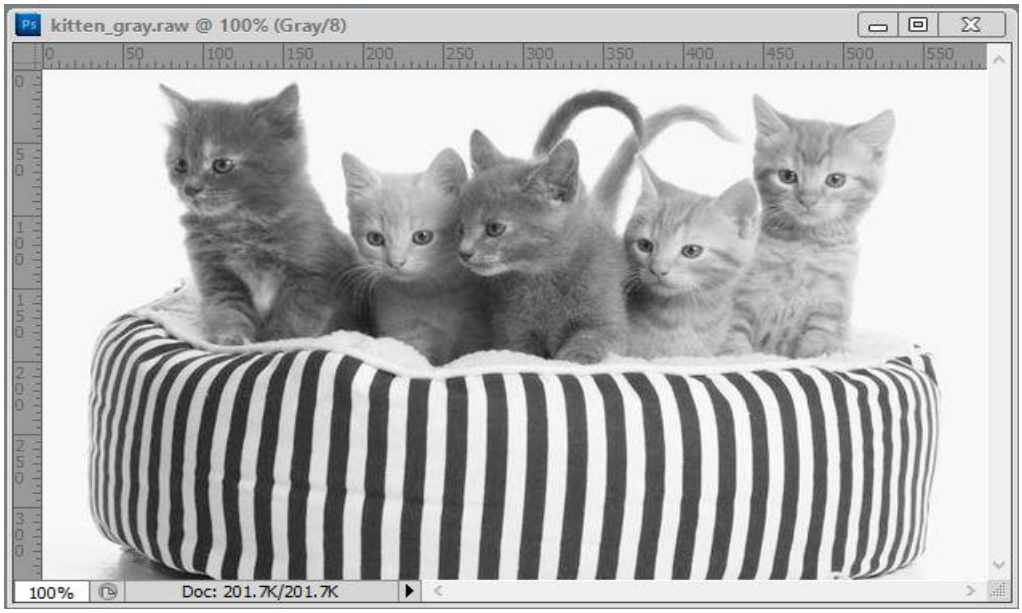


Fig 1.2: Gray kitten.raw image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Consider the case when K=3 we will have only 3 color representatives.

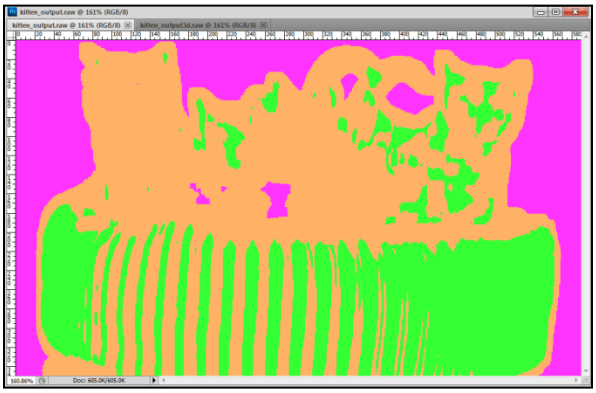


Fig 1.3: K=3, Window size=13

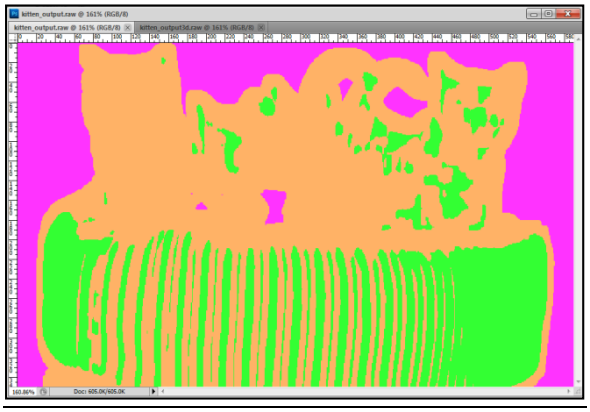


Fig 1.4: K=3, Window size=15

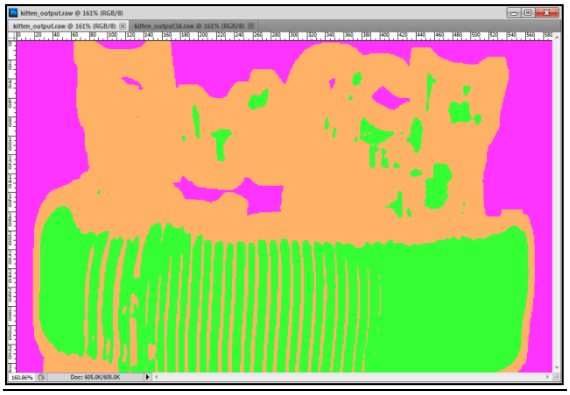


Fig 1.5: K=3, Window size=21

Aakash Shanbhag

adshanbh@usc.edu

3205699915

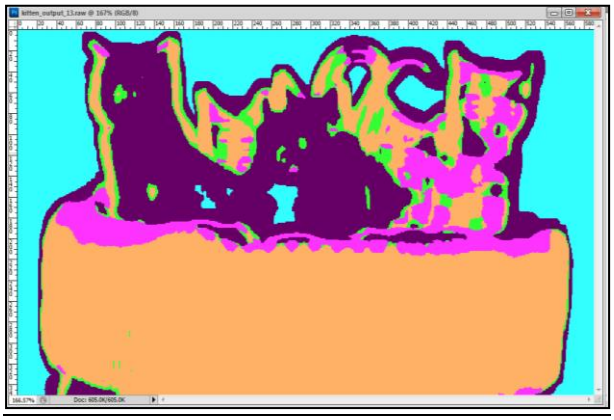


Fig 1.6: K=5, Window size=13

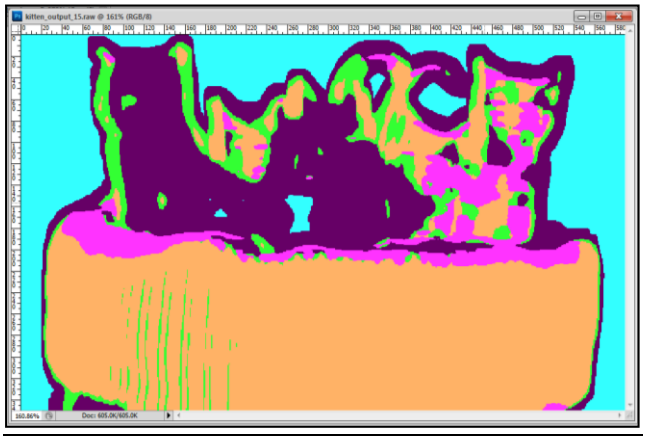


Fig 1.7: K=5, Window size=15

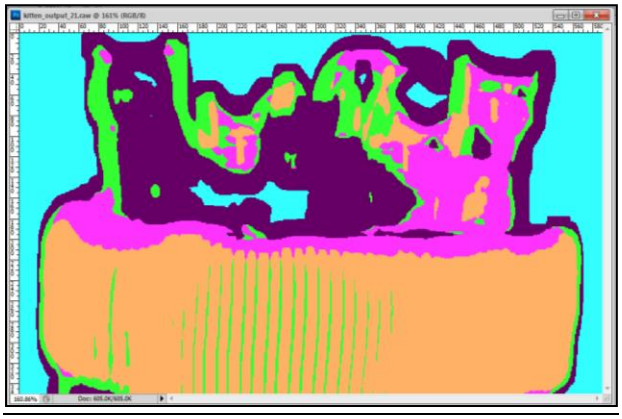


Fig 1.8: K=5, Window size=21

Aakash Shanbhag

adshanbh@usc.edu

3205699915

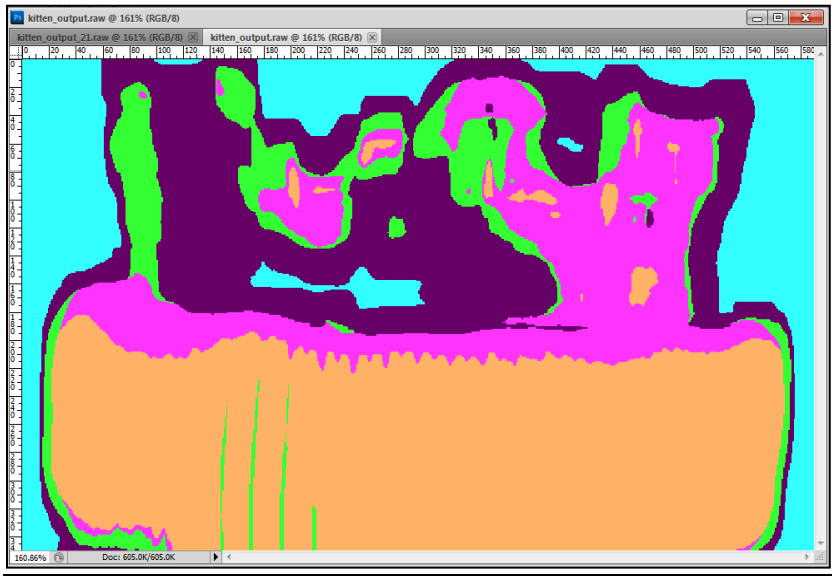


Fig 1.9: K=5, Window size=31

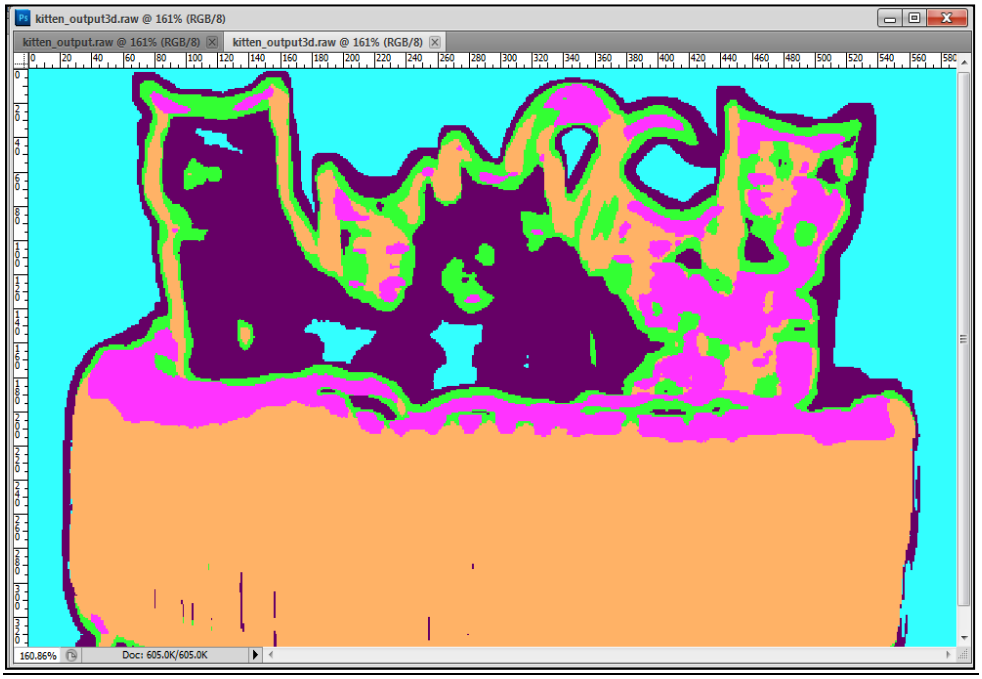


Fig 1.10: K=5, Window size=15 with PCA

Aakash Shanbhag

adshanbh@usc.edu

3205699915

1.4 Discussion and Conclusion.

According to the requirements all the texture classification and segmentation was carried out successfully. The detailed procedure and method is discussed above and some important conclusions for each of the portions are as follows:

1 (a) Texture Classification

Supervised and unsupervised learning was carried out successfully with classification of each of the 12 textures into each of the 4 textures and the unknown textures were found out on that basis. For the discriminant power I consider the Texture 1 and 7 which was classified as Bark and Straw and 25 filters

Filter	Texture 1	Texture 4	Difference
L5L5	0.699628	-1.061461	1.761089
E5L5	0.353547	-0.735586	1.089133
S5L5	0.457348	-0.325925	0.783273
W5L5	0.699869	0.296119	0.40375
R5L5	0.756398	0.663023	0.093375
L5E5	0.506361	-1.01917	1.525531
E5E5	0.07557	-0.659311	0.734881
S5E5	0.199401	-0.283354	0.482755
W5E5	0.491035	0.318428	0.172607
R5E5	0.561151	0.683733	0.122582
L5S5	0.21183	-0.965616	1.177446
E5S5	-0.018178	-0.58585	0.567672
S5S5	0.176378	-0.168967	0.345345
W5S5	0.445095	0.341583	0.103512
R5S5	0.50581	0.642857	0.137047
L5W5	0.316257	-0.818954	1.135211
E5W5	0.202763	-0.371951	0.574714
S5W5	0.360782	0.056534	0.304248
W5W5	0.469655	0.36309	0.106565
R5W5	0.466004	0.567925	0.101921
L5R5	0.691841	-0.080427	0.772268
E5R5	0.547309	0.197814	0.349495
S5R5	0.473014	0.294162	0.178852
W5R5	0.474551	0.368651	0.1059
R5R5	0.435956	0.479545	0.043589

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Now if texture 1 and 4 was considered we saw L5L5 was showing the strongest discriminant power whereas the R5R5 showed lowest change in discriminant power. Similarly if we compare any two different sets we will have different filters that will have the largest and the smallest discriminant powers, so it is completely dependent on the two textures which we take an input. Similarly if texture 1 and texture 7 are considered W5L5 has the largest discriminant power and E5E5 has lowest.

PCA is carried out in this case which helps in reducing the feature space and hardly has a vast difference in the output. Only difference is that using PCA we immensely reduce the time required for computation and k means requires lesser iterations and converges quickly. Hence PCA helps in immensely reducing the time for computation. Hence feature reduction can affect the computation speeds.

1 (b) Texture Segmentation

While taking into consideration texture segmentation for the kitten image, different window sizes were taken into consideration along with different k values. We can infer that if we keep k=3, it is very difficult to cluster the entire image into more textures and the kittens cannot be separated as different textures. Hence k=5 was chosen which produced the desired output and helped in distinguish the different kittens, couch and background.

With different window sizes we saw that as we increase the size of the window from 13 to 15 to 21 and 31 it can be stated that the segmentation gets better as the size increases but as the size increase beyond a point like in case of window size=31, we get a blocky images and hence generally a optimally large value of window size is chosen.

1 (c) Further Improvement

We can observe that we could not get optimal segmentation for the input kitten image following the earlier procedure. So, we choose a different method –advanced texture segmentation technique. It includes considering two spirals for coarse and fine tuned segmentation- My output for this technique was inaccurate since if we use the coarse segmentation with parameter of K=3 we will be able to distinguish the couch background and kittens but since we are not able to completely extract the exact features of the textures, applying a fine tuned segmentation of k=5 over this does not serve the purpose. Hole filling and dialation are possible solutions for this process but again inaccuracy is observed as dialation and hole filling do not seem to maintain the edges and blocky image is obtained in due course of action.

I applied PCA for further improving my result which was fruitful once the optimal size of the window was chosen which is seen in Fig 10 which shows the results after PCA for k=5 with window

Aakash Shanbhag

adshanbh@usc.edu

3205699915

size=15 which can be compared with the normal 25 D output of Fig 8. Vast differences in output are not expected using PCA but a better and computational efficient answer is obtained.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Problem2: Edge and Contour Detection.

2.1 Motivation and Abstract.

Edge detection and contour detection are one of the fundamental issues of computer vision and image processing algorithms. Edges are locations where there is a sudden change in the intensity values and hence in frequency domain can be considered as presence of higher frequency. Different high pass filtering approaches are necessary to be carried out in order to reduce this effect. It is used in image processing pipelines to achieve tasks such as object detection, scene detection, object tracking and so on. In this problem we will apply Canny edge detector and Structure Forests for fast edge detections and carry out the performance analysis on the images provided. Finally performance evaluation of the structured edge random forest classifier is also carried out.

Canny edge detection developed by John Canny is basically an extension to Sobel edge detection. This algorithm adds a layer of intelligence over the Sobel edge detector as it performs double hysteresis thresholding to remove weak unconnected edges from the image. This method can be used in real time applications and it can be parallelizable. It takes into consideration only the edges that are a strong enough to pass the non maximal suppression stage. Initial noise is also eliminated in this process by application of Gaussian filter.

Structure Forests method developed by Piotr and Zitnik for edge detection takes the advantage of the structure present in the local image patch to train a random forest classifier. Structured edge provides good edge detection performance with very less computation time compared to other algorithms. This work was done with the motivation to apply effective edge detection in real time. In this problem, we will explore the working and quality of result of this method.

2.2 Approach and Procedure.

There are two methods which need to be carried out in this process. The first section describes the Canny edge method which includes the non semantic edge detection and the second approach in which semantic content based contour detection is possible with ground truth evaluation to be followed. OpenCv's inbuilt function of Canny edge detection was used for this purpose and Matlab was used for Structured Edge Detection by downloading the toolbox from the online source cited below: https://github.com/pdollar/toolbox.

2.2 (a) Canny Edge Detection

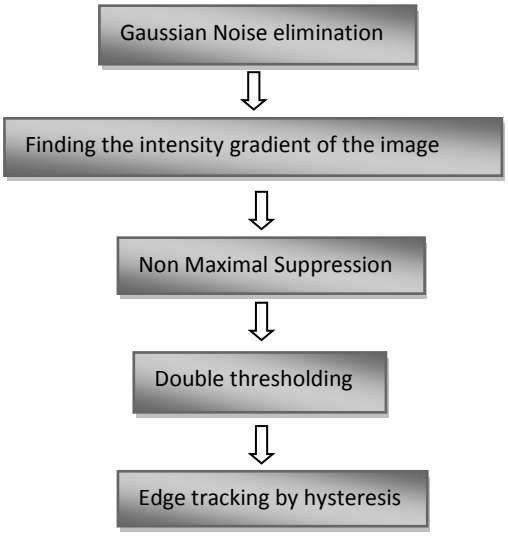
The process carried out for Canny edge detection is mentioned briefly below by the flow chart which takes into consideration the 5 basic steps needed for Canny edge detection. The first step includes the elimination of noise in the image to remove incorrect edge detection chances. It is also necessary that a Gaussian with a significant sigma is chosen for this purpose in order to eliminate any noise and infuse a certain degree of blur in the image. High pass filtering is analogous to taking derivative functions under most cases and hence linear approximation of the derivative which is the linear difference between pixel intensities is carried out.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Flow Chart for Canny Edge Detector



The above process includes the calculation of the intensity gradient which is important as it considers the linear difference approximation among pixels similar to be Sobel operator. Difference of Gaussian is first taken also termed as D.O.G. Non maximal suppression (NMS) Suppress all the gradient values to 0 except the local maximal (sharpest change). Double thresholding filters out the edge pixel with the weak gradient value and preserves the edge with the high gradient value. The thresholding takes place according to the following method:

Gradient > high threshold: Strong edge pixel

Low threshold<Gradient < high threshold: Weak edge pixel

Gradient < low threshold: It will be suppressed

Edge tracking by Hysteresis first connects the strongest edge points and based on the outputs of the 8 neighborhood the connectivity of the rest of the points is taken into consideration. Canny edge detection was performed using OpenCv's Canny operator. The image was smoothened by a Gaussian filter of size 3x3 to remove unwanted noise. The value of sigma controls the thresholding difference between the lower and upper threshold. A higher value of sigma indicates that the threshold is wide whereas a lower value of sigma indicates tighter threshold. The range of sigma which produces decent results lies between 0.25 to 0.4.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

2.2 (b) Structured Edge Contour detection.

Edge detection is a critical component of many vision systems, including object detectors and image segmentation algorithms. Structured forest is a state of the art algorithm that is used to compute the edges on a given image. This edge detector considers a patch around the pixel and determines the likelihood of the pixel being an edge point or not. Patches of edges exhibit well-known forms of local structure, such as straight lines or T-junctions. Edge patches are classified into sketch tokens using random forest classifiers. Structured learning is utilized to address problems associated with training a mapping function where the input or output space may be complex. The most important features of structured edge random forest classification are as follows:

- Structured Random Forest
- Multi-Scale Detection
- Edge Sharpening

Initially the random forest is trained by taking a patch from the input and ground truth images and forming a structured output of whether a certain combination should be classified as an edge or not. After the random forest is trained by an appreciable amount of samples, it can be tested. In the testing phase, only the test image is given for segmentation and due to the speed of random forests, it computes the segmented output.

A random forest is an ensemble approach. They run on the divide and conquer paradigm that helps to improve their performance. The main idea of ensemble approach is to put a set of weak learners to form a strong learner. The basic element of a random forest is a decision tree. The decision trees are weak learners that are put together to form a random forest.

The input or output space can be complex in nature. For example, we can have a histogram as the output. In a decision tree, an input is entered at the root node. The input then traverses down the tree in a recursive manner till it reaches the leaf node. Each node in the decision tree has a binary split function with some parameters. This split function decides whether the test sample should progress towards the right child node or the left child node. Often, the split function is complex. A set of such trees are trained independently to find the parameters in the split function that result in a good split of data. Splitting parameters are chosen to maximize the information gain. The flow chart of this process is mentioned below which clearly delineates the process that is explained above and gives an algorithmic explanation.

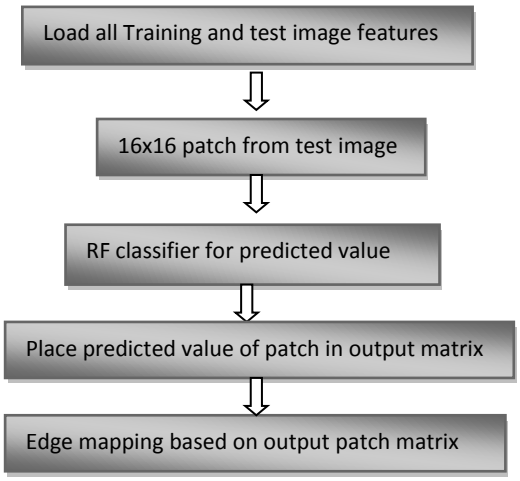
Matlab was used for this process and all the Mex files were compiled from the toolbox as mentioned above. It is important to note that once the Edge maps are produced by the files, binarisation of the image needs to be carried out in order to calculate all the precision, recall values of the individual man labeled ground truths for Performance Evaluation purposes.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Flow Chart for Structured Edge detector.



2.2 (c) Performance Evaluation.

Performance evaluation is a key component in checking whether the estimates generated by the edge detection modules actually have similarity with the ground truths labeled by humans. Matlab was used for calculation of these parameters for different thresholds. In effect we need to calculate the parameters which help us ascertain how good an edge detection algorithm works and hence some of the parameters necessary for evaluation of the performance are as follows:

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

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

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

True Positive-Edge pixels in the edge map coincide with the ground truth.

False Positive- Pixels in the edge map coincide with the non edge pixels ground truth.

True Negative- Non edge pixels in the edge map coincide with the non edge pixels ground truth.

False Negative- Non edge pixels in the edge map coincide with the edge pixels ground truth.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

2.3 Experimental Results.

The outputs of both the algorithms are mentioned below and performance evaluation is carried out for the Structured Edge Random Forest Classifier with different thresholds mentioned below:

2.3 (a) Canny Edge Detector

Different threshold conditions are mentioned below for the Canny edge detector and its performance for various thresholds is observed with fixed sigma=0.33 for the Gaussian filter.

For the Castle image

Fig 2.1: Castle input image.

Fig 2.2: Castle image with thresholds [0,255]

Fig 2.3: Castle image with thresholds [85,163]

Aakash Shanbhag

adshanbh@usc.edu

3205699915

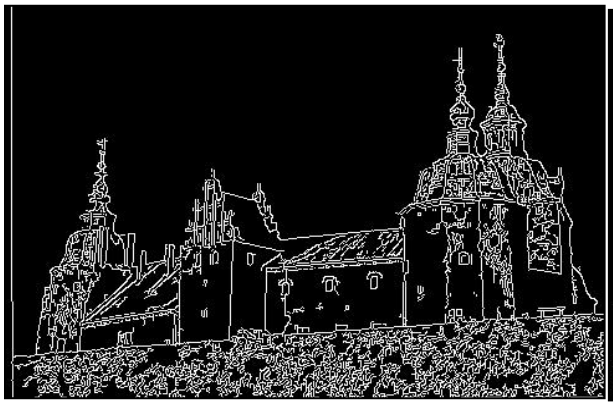


Fig 2.4: Castle image with thresholds [31,212]

Fig 2.5: Castle image with thresholds [55, 70]

For the Boat image

Fig 2.6: Boat input image.

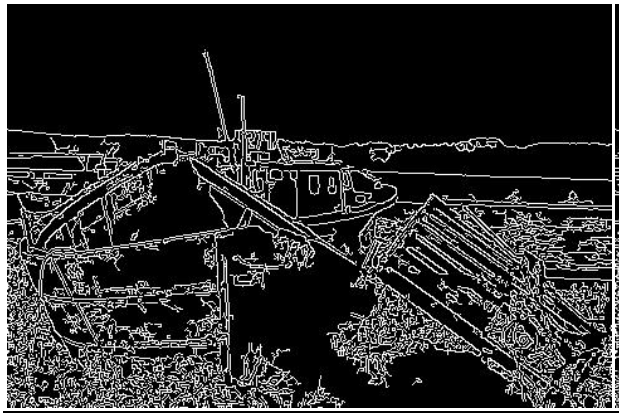


Fig 2.7: Boat image with thresholds [0,255]

Fig 2.8: Boat image with thresholds [85,163]

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Fig 2.9: Boat image with thresholds [31,212]

Fig 2.10: Boat image with thresholds [50, 70]

2.3 (b) Structured Edge Contour detection.

The structured edge outputs for both the castle and the boat image are displayed at a threshold of 0.2 and its edge map along with the binary map are observed.

For the Castle image

Fig 2.11: Castle input image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

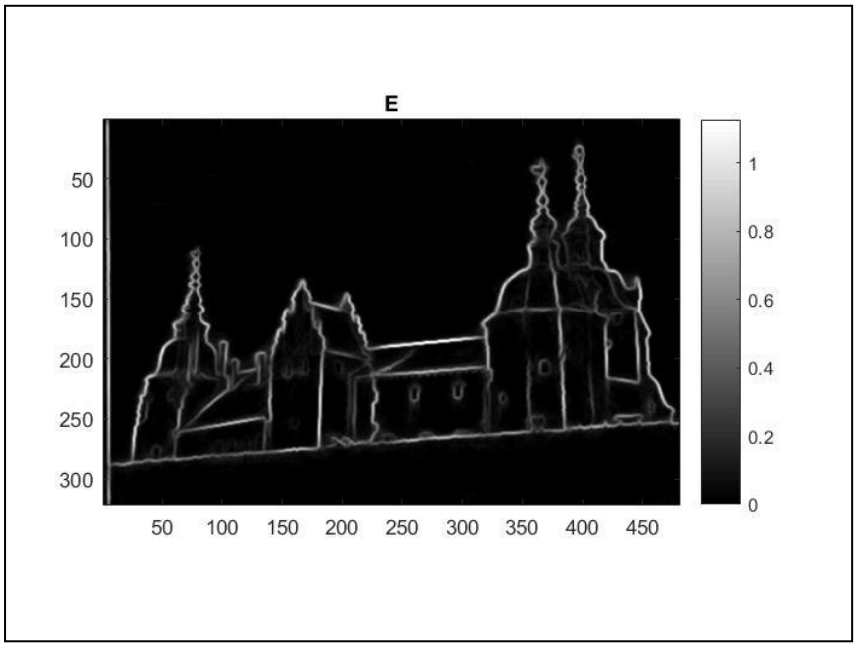


Fig 2.12: Castle Edge Map image.

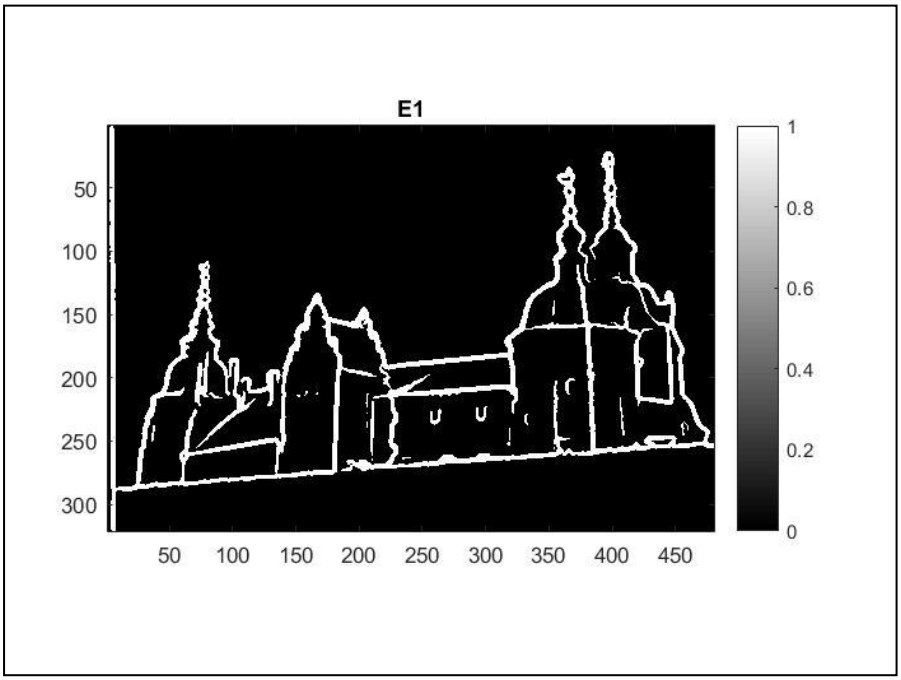


Fig 2.13: Castle Binary Map image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

For the Boat image

Fig 2.14: Boat input image.

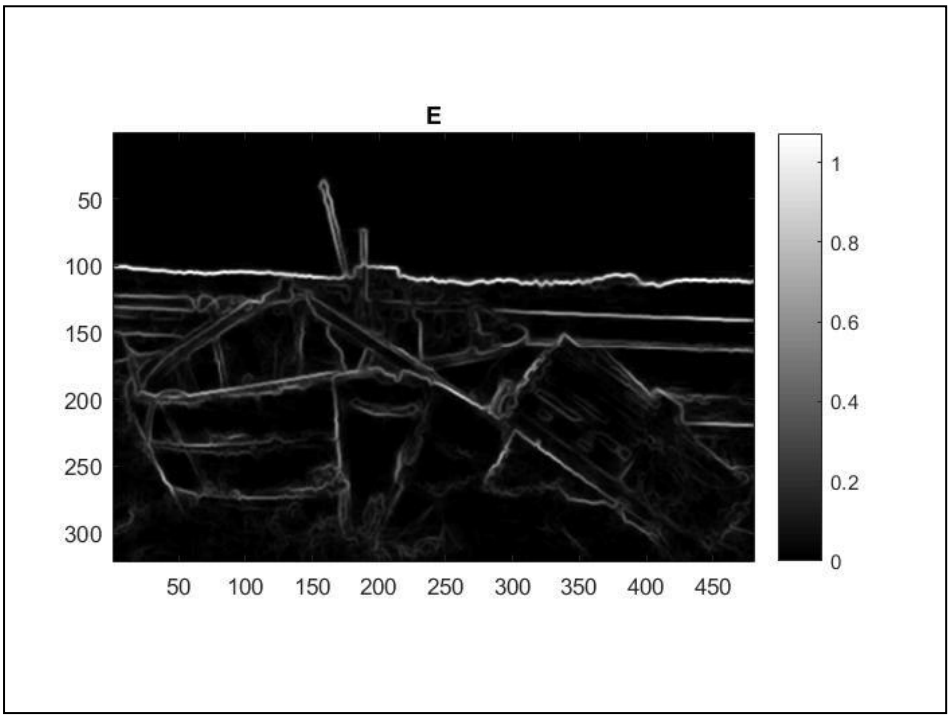


Fig 2.12: Boat Edge Map image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

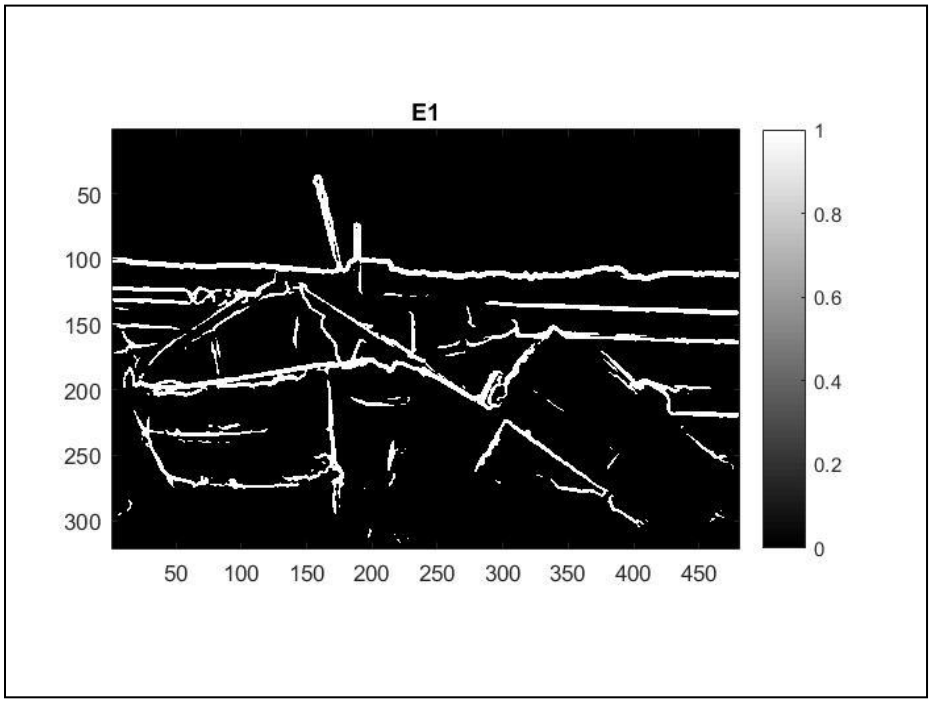


Fig 2.12: Boat Binary Map image.

2.3 (c) Performance Evaluation.

According to the edge maps obtained from above and the binary data that we finally obtain, comparison with respect to the ground truth is carried out and precision, recall and the f measure is calculated. In this case the precision, recall and f measure are calculated for thresholds 0.2 and 0.3 for both the castle and the boat image.

For the Castle image with Threshold=0.2

Ground Truths	Precision	Recall	F measure
Castle_gt1	0.4827	0.9524	0.6407
Castle_gt2	0.4801	0.9607	0.6410
Castle_gt3	0.6848	0.9015	0.7783
Castle_gt4	0.7915	0.8543	0.8217
Castle_gt5	0.6709	0.9105	0.7725
Castle_gt6	0.7535	0.8740	0.8084
Mean	0.6439	0.9085	0.7437

Aakash Shanbhag

adshanbh@usc.edu

3205699915

For the Castle image with Threshold=0.3

Ground Truths	Precision	Recall	F measure
Castle_gt1	0.5445	0.9075	0.6806
Castle_gt2	0.5445	0.9186	0.6837
Castle_gt3	0.7122	0.7920	0.7500
Castle_gt4	0.8231	0.7504	0.7851
Castle_gt5	0.6994	0.8017	0.7471
Castle_gt6	0.7729	0.7555	0.7641
Mean	0.6827	0.8209	0.7351

For the Boat image with Threshold=0.2

Ground Truths	Precision	Recall	F measure
Boat_gt1	0.5634	0.4472	0.4986
Boat_gt2	0.5187	0.6780	0.5877
Boat_gt3	0.5771	0.4457	0.5029
Boat_gt4	0.5894	0.6475	0.6171
Boat_gt5	0.4495	0.6652	0.5365
Boat_gt6	0.5858	0.6320	0.6080
Mean	0.5473	0.5859	0.5595

For the Boat image with Threshold=0.3

Ground Truths	Precision	Recall	F measure
Boat_gt1	0.6534	0.3277	0.4362
Boat _gt2	0.6225	0.5149	0.5636
Boat _gt3	0.6597	0.3224	0.4331
Boat _gt4	0.6979	0.4851	0.5723
Boat _gt5	0.5249	0.4915	0.5077
Boat _gt6	0.6801	0.4642	0.5518
Mean	0.6397	0.4343	0.5107

Aakash Shanbhag

adshanbh@usc.edu

3205699915

For the Boat image with Threshold=50 and 70 for Canny Edge Detector

Ground Truths	Precision	Recall	F measure
Boat_gt1	0.0736	0.2691	0.1156
Boat _gt2	0.0316	0.2180	0.0551
Boat _gt3	0.0670	0.2444	0.1051
Boat _gt4	0.0431	0.2410	0.0731
Boat _gt5	0.0197	0.1579	0.0351
Boat _gt6	0.0254	0.1408	0.0431
Mean	0.0434	0.2119	0.0712

For the Boat image with Threshold=31 and 212 for Canny Edge Detector

Ground Truths	Precision	Recall	F measure
Boat_gt1	0.1066	0.1450	0.1229
Boat _gt2	0.0286	0.0799	0.0421
Boat _gt3	0.1036	0.1385	0.1185
Boat _gt4	0.0681	0.1373	0.0910
Boat _gt5	0.0231	0.0716	0.0350
Boat _gt6	0.0283	0.0596	0.0384
Mean	0.0597	0.1053	0.0746

For the Castle image with Threshold=50 and 70 for Canny Edge Detector

Ground Truths	Precision	Recall	F measure
Castle_gt1	0.0429	0.2721	0.0742
Castle _gt2	0.0338	0.2356	0.0591
Castle _gt3	0.0466	0.1677	0.0729
Castle _gt4	0.0856	0.2470	0.1271
Castle _gt5	0.0582	0.2386	0.0935
Castle _gt6	0.0851	0.2734	0.1297
Mean	0.0587	0.2391	0.0928

For the Castle image with Threshold=31 and 212 for Canny Edge Detector

Ground Truths	Precision	Recall	F measure
Castle_gt1	0.1055	0.1381	0.1196
Castle _gt2	0.0698	0.0951	0.0806
Castle _gt3	0.0832	0.0648	0.0729
Castle _gt4	0.1022	0.0700	0.0831
Castle _gt5	0.1080	0.0923	0.0996
Castle _gt6	0.1447	0.0995	0.1179
Mean	0.1022	0.0933	0.0956

Aakash Shanbhag

adshanbh@usc.edu

3205699915

2.4 Discussion and Conclusion.

The outputs were generated successfully for the different edge detectors and the f measure was calculated for the different edge detectors and their performance can be easily evaluated.

(a) Canny Edge Detector

The outputs for different thresholding values have been displayed from fig 2.1 to 2.10 and we can clearly see that the Castle image performs much better than the boat image in terms of canny edge detection for a particular set of thresholds.

We cannot explicitly tell the objects location only on the basis of the edge map in case of the boat image but can still make out specific locations in the Castle image. Location is completely dependent on the image in consideration and the characteristic difference between the object and the foreground in that particular image.

We cannot fine tune directly to show the edges of the object since there is no method of semantic labeling in canny edge detector. It just runs on the principle of obtaining all the strongest edges and the edges it maps without any consideration of the content of the image or any degree of intelligence. We can change the sigma values and the window sizes in the Gaussian but they do not directly help in object edge detection. If we increase the value of sigma from 0.33 to a larger value we may be able to differentiate to a certain extent the background and object but this cannot be assured for all images which have closely camouflaged backgrounds and foregrounds. Hence a layer of intelligence for semantic labeling is crucial.

The relationship of the threshold values and the image content with the case of double thresholding purely depends on the fact if:

Gradient > high threshold: Strong edge pixel

Low threshold<Gradient < high threshold: Weak edge pixel

Gradient < low threshold: It will be suppressed

As already mentioned above, we can change the different values of threshold to obtain and eliminate particular set of edges but no ways will be able to make it on basis of content of the image purely. Only gradient assumptions work in case of Canny for edge detection.

(b) Contour Detection with Structured Edge

Aakash Shanbhag

adshanbh@usc.edu

3205699915

The SE algorithm is explained above along with the flow chart which explains the need of finding the correct patch for multi scale detection and edge handling.

The Structured edge was carried out using Matlab and the toolbox presented by the author of the paper Piotr Dollar; these parameters were responsible for obtaining the outputs shown in Fig 2.12 and 2.14 respectively.

```
%% set detection parameters (can set after training)
model.opts.multiscale=0;          % for top accuracy set multiscale=1
model.opts.sharpen=2;             % for top speed set sharpen=0
model.opts.nTreesEval=4;          % for top speed set nTreesEval=1
model.opts.nThreads=4;            % max number threads for evaluation
model.opts.nms=0;                 % set to true to enable nms
```

Structured Edge performs really when in edge detection tasks even on the visual front. It has an added layer of intelligence that enables it to disregard unimportant edges and label the image even though it has a very high gradient change.

This is because Canny edge detector is not intelligent to detect textures and only depends on the gradient values and the thresholds set. Figure 2.13 clearly shows the object Castle as a contour and eliminates the background carefully which is not seen in the canny edge detection portion which takes the other features in the image as well.

(c) Performance Evaluation

The difference between the f measures between the Canny edge detector and the Structured edge are clearly seen the section 3.3(c). As discussed before, Canny doesn't have the ability to discard noisy edges introduced due to texture.

By Intuition, it is easier to get a higher F score on Castle image. Visually, this is due to the fact that this image is way simpler to segment than the Boat image. Since the boat is camouflaged into its background, it is difficult to segment it with high precision. The F measures of Canny edge detector are very low as compared to the Structured edge as it clearly outperforms Canny because of the training stage.

The F measure is a measure of test accuracy in statistics. It value is directly affected by both precision and recall. If any one of these metrics is low, then it results in a bad F1 score. Additionally, it is interesting to note that there exists an inverse relation between precision and recall. If recall increases, then precision gets low and vice versa. To maximize the product of the two which follows an inverse relation, we need to have them to be equal. This can also be explained by mathematically. We know that the sum of Precision and Recall is constant in theory. Maximum value of F could be 1 in theory.

Hence Recall=constant-Precision, if we substitute this in the f measure equation mentioned above we would obtain F=2/(constant)*Recall*(constant-Recall) which maximizes when differentiated and set to 0 to Recall=Precision.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Problem 3: Salient Point Descriptors and Image Matching

3.1 Motivation and Abstract.

Meaningful features are the requirement for any kind of object tracking, segmentation, video tracking and many such applications. The robustness of features leads to a successful salient description of an image in terms of its attributes. Extraction of salient features in computer vision and machine learning is essential as it helps in scene analysis. SIFT and SURF are such algorithms that extract important local features in an image that are scale and rotation invariant and provide robust outputs.

SIFT (Scale Invariant Feature Transform) was developed by David Lowe and SURF (Speeded Up Robust Features) was developed by Herbert Bay. In this problem, we implement SIFT and SURF algorithms to compare their performance and efficiency with respect to finding the key points and image matching characteristics. It is followed by the bag of words concept which uses these descriptors to differentiate images of similar objects with different orientation, illumination differences and camera angles.

The underlying idea of this model is to create and learn a visual vocabulary from the training set and classify unknown images based on the vocabulary. This model coupled with robust features such as SIFT or SURF can provide surprisingly good results. We will explore this problem in detail in the following sections using the K means algorithm to further cluster such features.

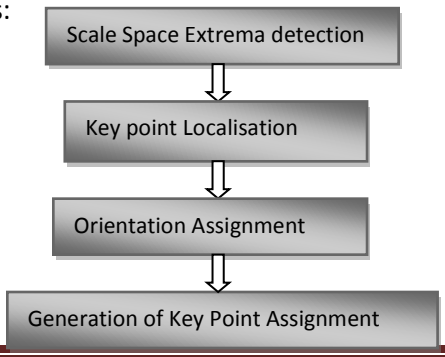
3.2 Approach and Procedure.

The algorithms used for characterizing the images into specific robust features are described in brief in the two sections followed by a novel way of creation of codebook in the bag of words which follows later.

3.2 (a) Extraction and Description of Salient Points

3.2 (a) 1. SIFT algorithm

SIFT stands for Scale Invariant Feature Transform. It is an approach of extracting feature descriptors, which are invariant to scaling, rotation, and change in illumination, noise and small changes in the point of view. To implement this, we have used OpenCV that contains SIFT detectors and extractors. The steps included in this transform are as follows:



Aakash Shanbhag

adshanbh@usc.edu

3205699915

Most of the heuristics obtained are from the paper by David Lowe.

Scale Space Extrema Detection:

To make the key points scale invariant, we consider the input image on different scales. This is done by applying Gaussian smoothening with different values of sigma. The scale space consists of n octaves with 3 images in each octave. Each octave is down sampled by a factor of 2 following the first octave. Additionally, each image in an octave is smoothened by a sigma value that is a multiple of k. The author heuristically found the value of k to be 0.707. Similarly, the value of sigma was found out to be 1.6. After applying the Gaussian smoothening, the difference of each Gaussian is taken. This is to approximate the LoG function as the latter takes a lot of time to compute. The DoG gives us an edge map. To detect the point of interest from this edge maps, we iterate through all the possible points and select the point of interest which lies in the local maxima and minima at a given scale. This is done by comparing a given point at different scales. The Fig 3.1 below gives a brief overview of how these scales are considered and the output is obtained.

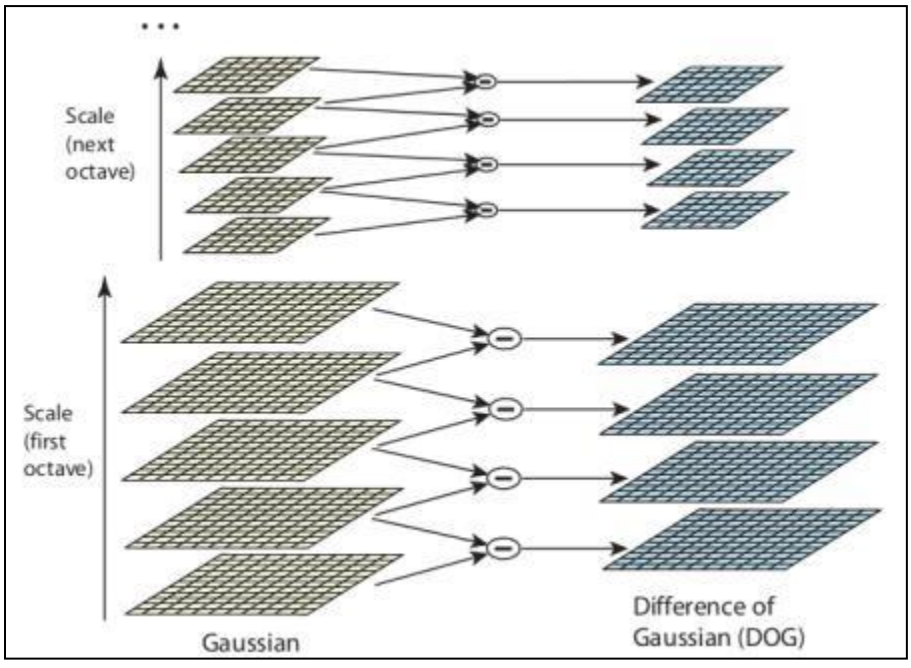


Fig 3.1: Different scales of the same image and the Laplacian of Gaussian calculated for the same.

The Laplacian of Gaussian or LoG is calculated in accordance of the heat equation as mentioned below which explains the basic process of creating points which can be faithfully observed as descriptors:

Aakash Shanbhag

adshanbh@usc.edu

3205699915

$$\frac{\partial G}{\partial \sigma} = \sigma \Delta^2 G \qquad \text{Heat Equation}$$

$$\sigma \Delta^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta^2 G$$

$$\text{Typical values}: \sigma = 1.6;\ k = \sqrt{2}$$

Key point Localization:

In the previous step, we found out the interest points. These points of interest may contain weak/noisy points due to low contrast or they may be part of an edge. To avoid such outliers, we compute the Taylor series expansion of DoG and remove the points that lie below a threshold of 0.3. We also compute the hessian matrix and compare its Eigen values. If the ratio of first and second Eigen values are greater than 10 that means that the point of interest is an edge point. So we discard it. The Hessian Matrix is the most important factor in deciding the number of points that will be eliminated. It takes a ratio of the Trace over the determinant and according to heuristic computation decides the max values.

Orientation Assignment:

To achieve rotational invariance, we assign an orientation value to each key point. This is done by first taking a neighborhood around the point of interest. The neighborhood size depends on the scale at which the point was detected. Then histograms of 36 bins are created that represents 360 degrees. The orientation of each pixel in the window is weighted by its gradient magnitude and a Gaussian weighted circular window with sigma that is 1.5 times the scaling factor. The weighted orientation is plotted on the histogram and the maximum histogram value is selected. All the orientations that lie within 80 percent of the maximum are considered to increase the robustness. Such a consideration is again obtained heuristically and has to be ascertained in order to have robust descriptors. The figure below defines how the strongest angle gradients are only considered and the rest seem to be neglected or suppressed.

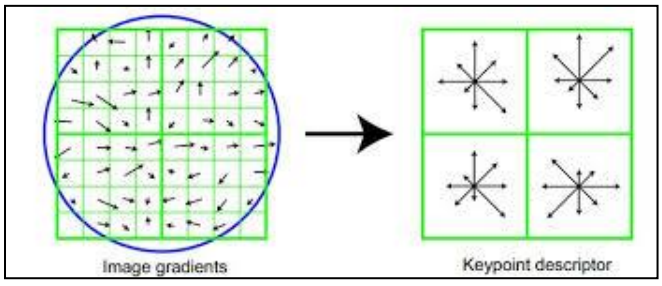


Fig 3.2: Depending on the gradient magnitude and the angles present, maximum angle orientation is set for the key point descriptors.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Key point Descriptors:

Now it is important for us to create feature values for a point of interest. The author suggested taking the histogram of the neighborhood orientation as features as they provide variations against illumination effects, viewing angle difference etc. To do this, we first consider a 16x16 neighborhood from the interest point. This neighborhood is divided into 4x4 units from which histogram of the pixels in each of the unit are calculated. The resolution of histogram was taken to be 8. This in turn forms 128 histogram bins in the 16x16 neighborhood. The 128 bins are taken as a feature vector for a key point. This vector is normalized to a unit vector to cancel the illumination effects. It is hence produces robust key point descriptors.

3.2 (a) 2. SURF algorithm

SURF stands for Speeded Up Robust Features which can be used for tasks such as object recognition, image registration, classification or 3D reconstruction. It is partly inspired by the scale-invariant feature transform (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT.

It includes the integration of image for speeding up filtering after which it uses a blob detector to find the interest points based on the Hessian matrix similar to the SIFT method. Integral images in SURF are created since the mask convolutions are independent of the size of the filter. Integral image is of the same size of the image that needs to be analyzed and the value of the integral image at any location can be determined as a sum of the intensity values lesser than or equal to where we evaluate the integral of the image. Firstly we find the interest points in the image by computation of the determinant of the Hessian Matrix. We use LoG (second order derivative filter) as the derivative filter in the Hessian matrix.

Using the integral image, we obtain the sum of intensities for squares present in Hessian box filters – multiplied by the weight factor and then addition of net resultant Sum. Evaluation of filter size is done using a threshold, wherein we control the number of interest points which are to be detected. 3 X 3 Non-maximal suppression below and above the scale space of each octave. Correct scale needs to be maintained in this case hence interpolation of interest points is carried out.

Feature vector direction and angle is obtained by the Haar transform in this case. 64 dimensional feature vectors are obtained after we have already suppressed the non maximal angles for 16 sub regions which are the two dimensions of the Haar wavelet.

SURF is also applied using the OpenCv library. It is computationally efficient and can handle large Hessian values robustly.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

3.2 (b) Image Matching

After the extraction of the key point descriptors and exactors we can carry out the 2nd part of the problem which deals with Image matching. The first step to this process includes calculation of the descriptors and the extractors in the similar manner using SIFT and SURF. The reason why we extract features from an image is to find relevant matches among images. After extracting SIFT features from two images, we carry out the Brute Force Matching algorithm. The Brute Force matcher computes a match for every point and may contain noisy matches. The result is displayed in section 3.3(b). In order to increase the efficiency there are other techniques of matching such as Flann based matching which uses the nearest neighbor approach and takes into the consideration the ratio of the first and second best guess and if the ratio is greater 0.8 omits that matching which eliminates the redundancy decreases the inaccurate matches.

3.2(c) Bag of Words

Bag of Words algorithm is based on the idea of document classification. It can be used for image classification in small systems wherein user defines the number of clusters. Bag of words is the vector of occurrence of count of words and is expressed as a histogram of each word. Training of Bag of Words is done to cluster the training images into several clusters. The calculated features of test images can be compared to words in a dictionary to get histogram. The steps followed in this process are as follows:

1. Calculate the SIFT descriptors of all the training and testing images.
2. Club the SIFT descriptors of the training images together in a bag and choose K=8 from this bag of descriptors of the training images.
3. Now the K means algorithm is applied after choosing the 8 feature descriptors from the set or bag and each of the image descriptor is then classified on the basis of these 8 descriptors.
4. The algorithm runs until convergence of the k means is reached in this case.
5. It is important to then consider the nearest neighbor algorithm for the descriptors of the test image descriptor.
6. Once this is carried out, each of the image descriptors are classified into 8 bins on the basis of the codeword generated and the histogram of each of the 4 image descriptors is calculated.
7. Once all the histograms are obtained, the histogram which has the largest similarity with that of the test image descriptor histogram or the lowest error margin in between the histograms is considered as the best match to the test image. Hence through this histogram matching principle we can state which image is closely associated with the test image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

3.3 Experimental Results.

3.3 (a) Extraction and Description of Salient Points

This part of the problem deals with extraction of key points using SIFT and SURF algorithms for the SUV and the Truck image. The time required for the obtaining the key points of the SURF features is lower than that of the SIFT feature key points.

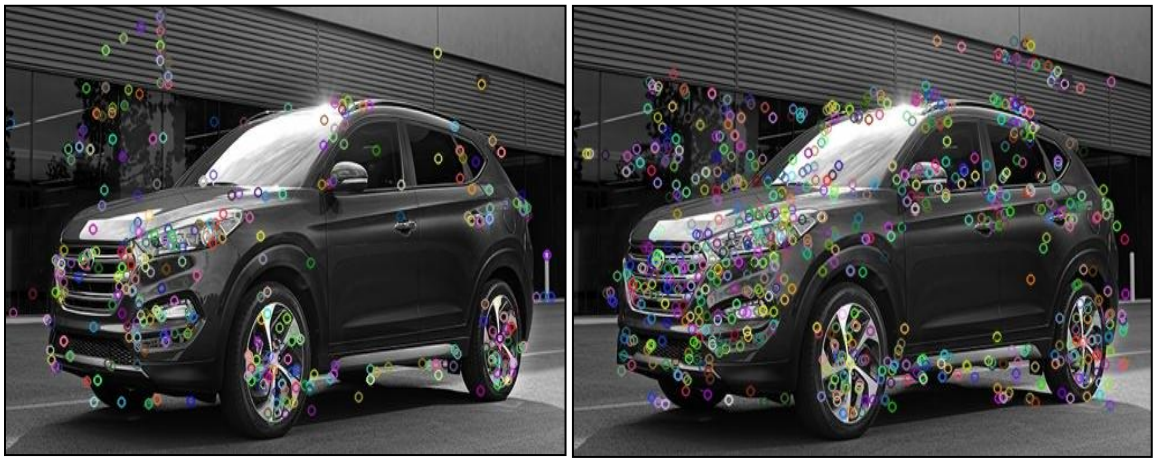


Fig 3.3: SIFT (left) and SURF (right) key points on the SUV image.

Fig 3.4: SIFT (left) and SURF (right) key points on the Truck image.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

3.3 (b) Image Matching.

Image matching is shown for images below which display the dependence of obtaining the correct key points and correspondingly achieving the correct match. Viewing angle is also responsible for such issues.

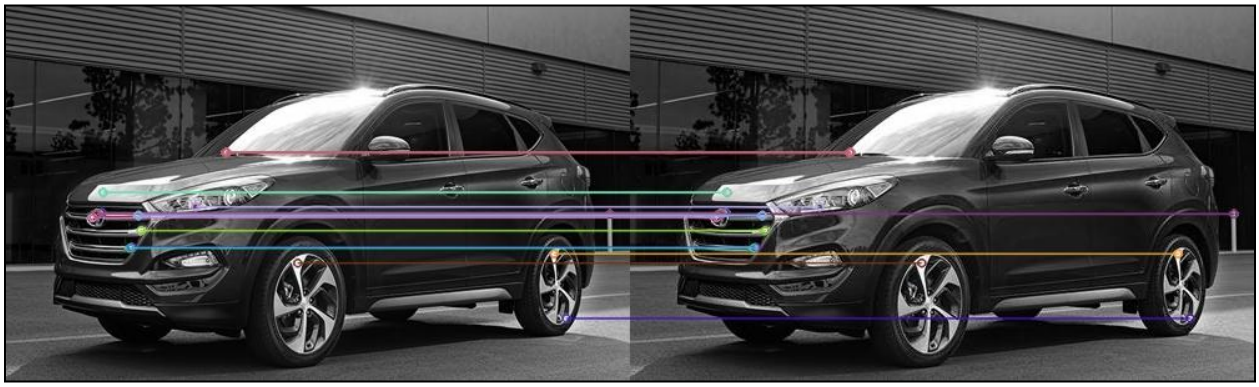


Fig 3.5: SIFT matching for the SUV with itself.

Fig 3.6: SURF matching for the SUV with itself.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Fig 3.7: SIFT matching for the Truck with itself.

Fig 3.8: SURF matching for the Truck with itself.

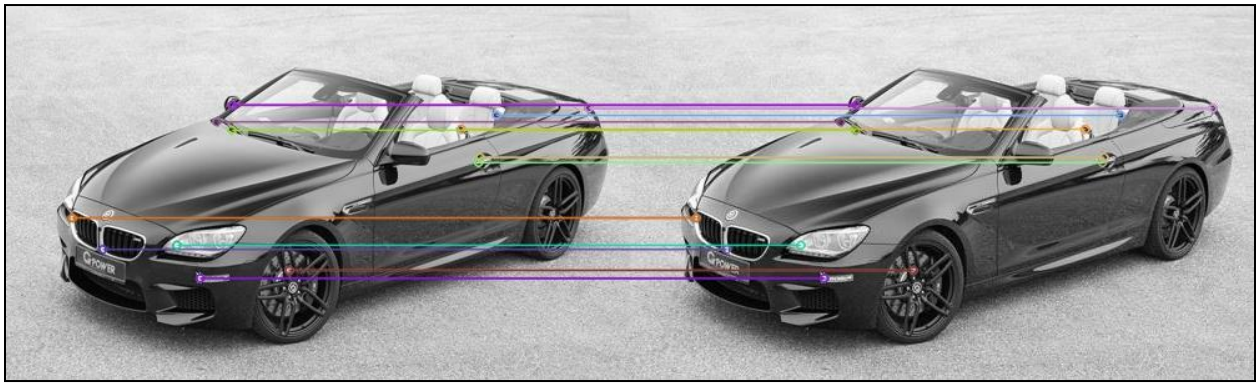


Fig 3.9: SIFT matching for the Convertible_1 with itself.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

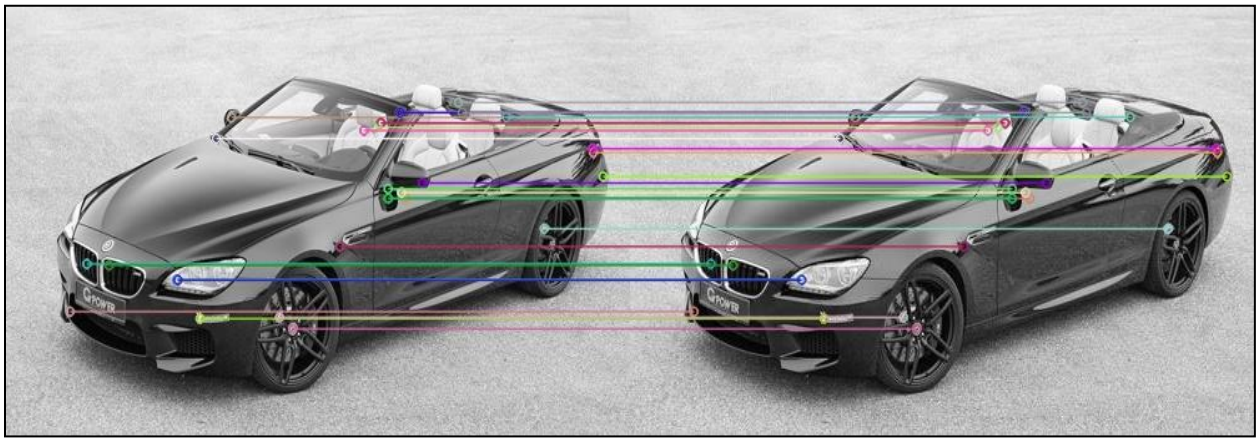


Fig 3.10: SURF matching for the Convertible_1 with itself.

Fig 3.11: SIFT matching for the Convertible_2 with itself.

Fig 3.12: SURF matching for the Convertible_2 with itself.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

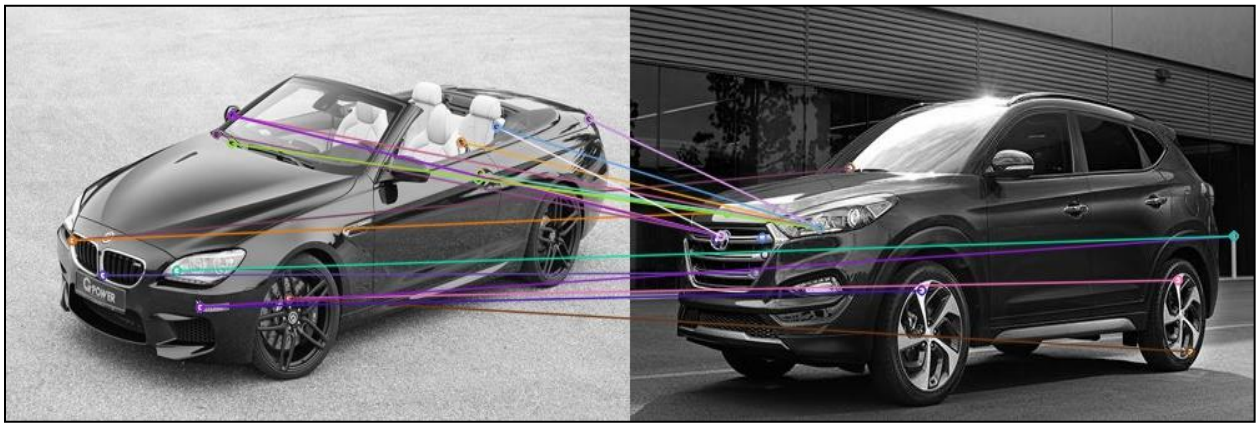


Fig 3.13: SIFT matching for the Convertible_1 with SUV.

Fig 3.14: SURF matching for the Convertible_1 with SUV.

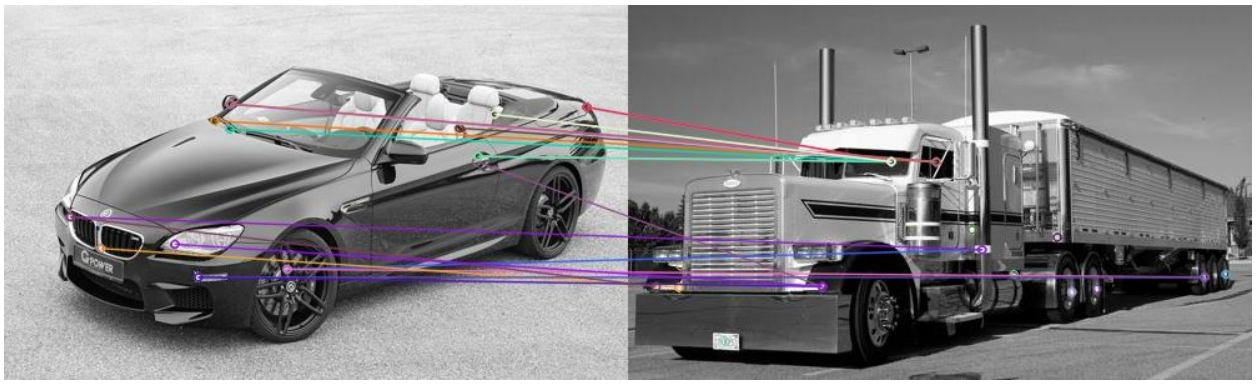


Fig 3.15: SIFT matching for the Convertible_1 with Truck.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

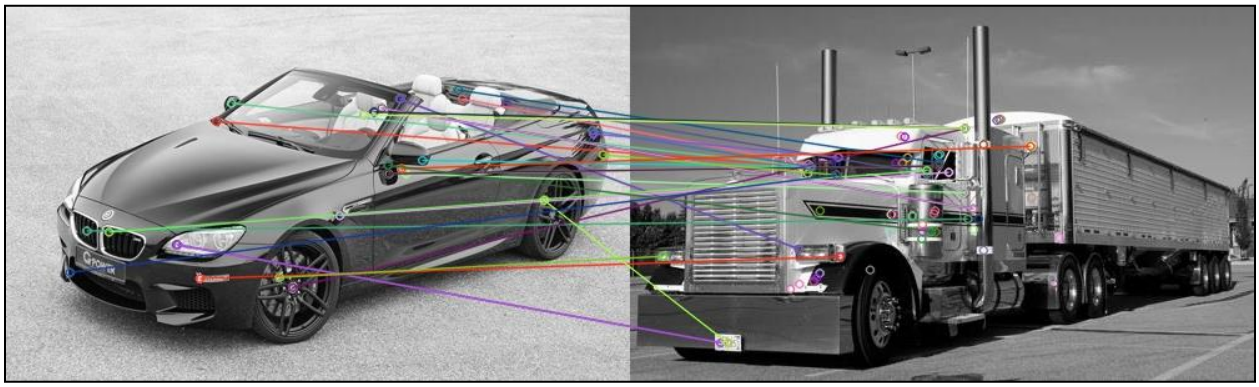


Fig 3.16: SURF matching for the Convertible_1 with Truck.

3.3 (c) Bag of Words.

The final histograms that are obtained for different values of Hessian min value are shown below and the closest histogram to the convertible 2 image is discussed in each image. The error is calculated by considering the mean of the absolute differences in the histograms.

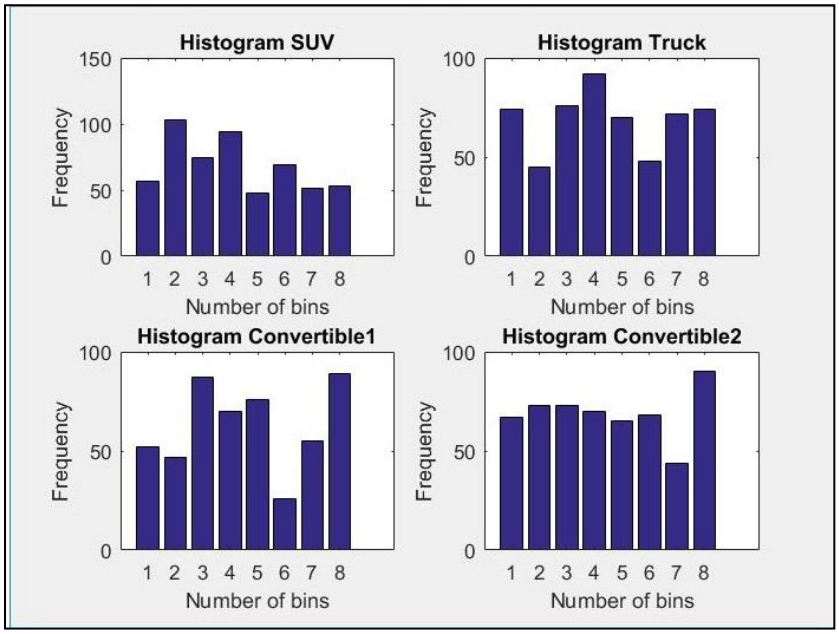


Fig 3.17: Hessian min value=550. Convertible 2 histogram matches with Convertible 1 histogram.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

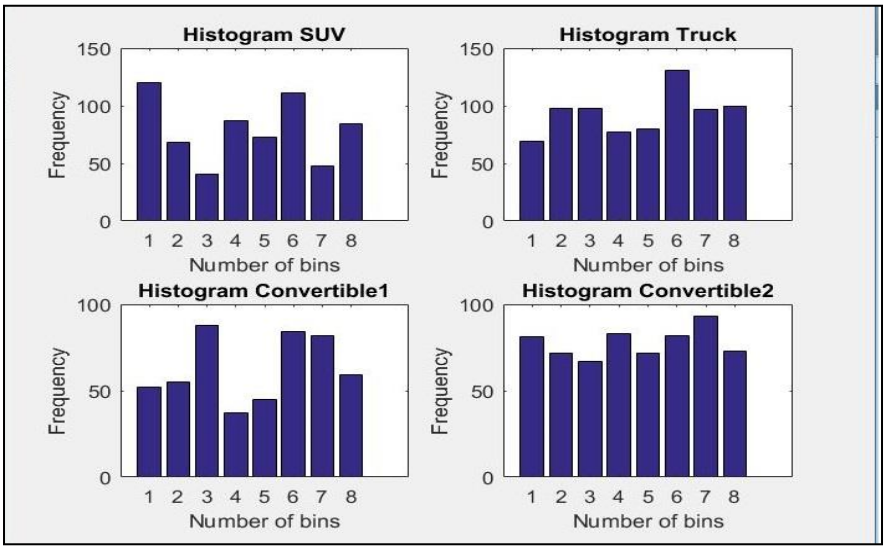


Fig 3.18: Hessian min value=750. Convertible 2 histogram matches with SUV histogram.

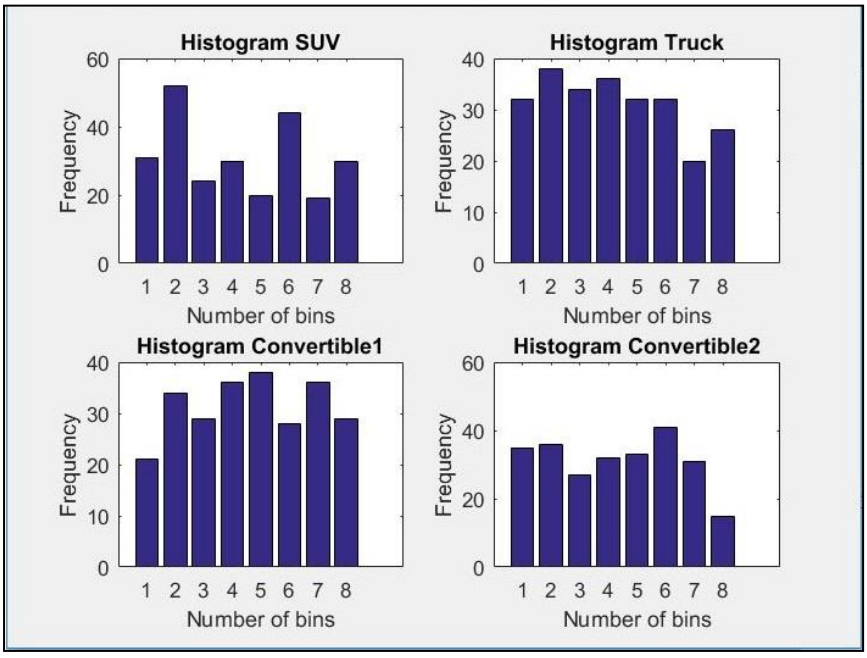


Fig 3.19: Hessian min value=250. Convertible 2 histogram matches with Truck histogram.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

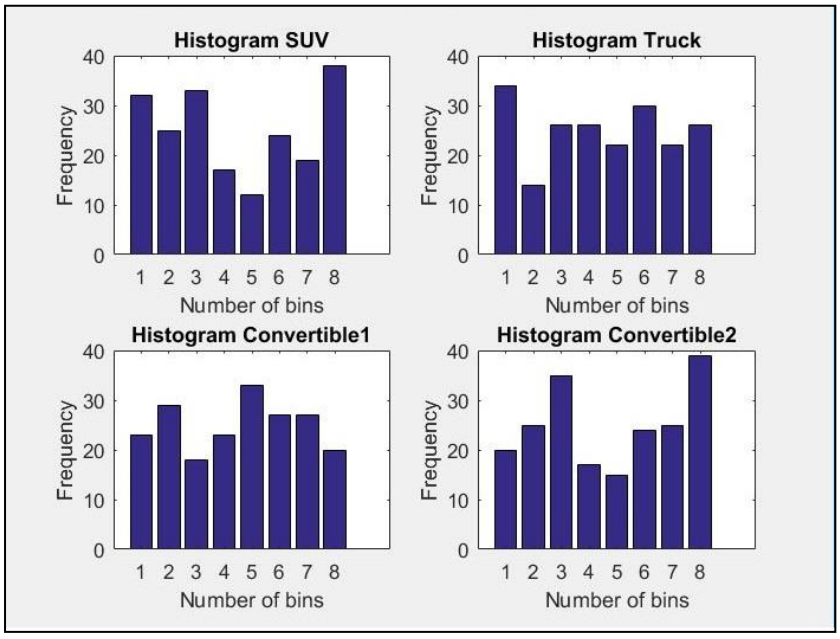


Fig 3.20: Hessian min value=20. Convertible 2 histogram matches with SUV histogram.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

3.4 Discussion and Conclusion.

The outputs for SIFT and SURF algorithms for image key point generation and image matching along with the bag of words were shown in the section 3.3 above which clearly showed that both the SIFT and SURF have specific pros and cons. Their differences are discussed below along with discussion on the bag of words.

(a) Performance, efficiency, weakness and strengths of SIFT and SURF.

As seen from the images in fig 3.3 and fig 3.4, we can clearly see that the SURF has more key points than the SIFT and the differences are mentioned below

SIFT	SURF
Uses DoG convolved with Gaussian as Scale space	Uses Integral image as scale space
Non Maximal Suppression to remove outliers from the Hessian matrix for key point detection	Hessian matrix is used for key point detection
Orientation on the basis of the magnitude of gradient along with weights of surrounding pixels	Orientation by Gaussian weighted Haar transform
16x16 window used for key point descriptors	Quadratic grid of 4x4 sub regions along with wavelet responses taken into consideration.
Computational heavy and slower than SURF	Computational better than SIFT with more key points.
Descriptors calculated are 128 D gradient histograms	Descriptors record Haar wavelet response with 64 D.
Since pyramidal structures of DoG are used may falter in cases with similar backgrounds & blurring	Performs better in illuminance change and blurring since integral image is used.

(b) Image Matching

I applied the OpenCv's Brute Force Matcher to find the match between the specific key points. But since it uses the brute force approach of matching all the key points, some noisy and redundant key points are also selected.

In OpenCv's image matching algorithm Flann based image matching is carried out which takes into consideration the ratio of the 2 nearest neighbors and if the their ratio is greater than 0.8 then match will be eliminated. Flann based image matching takes into consideration the nearest neighbor approach to calculate the points. From the figures 3.5 to 3.12, when two similar images are considered both the SIFT and SURF perform well and do not cause any issue. More key points can be assigned depending on the parameters that have been set on the Hessian matrix. Hence Flan can be used as substitute to Brute Force matcher to have better image matching.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

Consider the figures 3.13 an 3.14 where the convertible 1 is matched with the SUV using SIFT and SURF. In this case some errors in matching with the SUV occur with the SIFT as well as the SURF. The top portions of the SUV are not matched at all in this case which creates these errors. Some of the reasons for this could include the viewpoint or orientation associated with the convertible and the SUV. Since it is invariant to scale and rotation there may be many unwanted points in the convertible 1 image which are not mapped to the points in the SUV image. Different backgrounds also may cause in increase of key points that are not accurately mapped.

Figures 3.15 and figure 3.16 show that match between the convertible 1 and the truck are better than that of the convertible and the SUV since most points match and they have very little background points that are associated. Outputs will be a lot different if all the images have similar backgrounds which helps in maintaining the orientation as well as increases the matching portions in key points of both the images.

(c) Bag of words.

In this case the bag of words was use to test the convertible 2 image using the SIFT descriptors. Nearest neighbor approach was used to calculate the histograms of each of the images. By visual inspection we can clearly make out that Convertible 1 and convertible should have similar demarcation.

But once different values of minimum Hessian matrix are used we begin to obtain different answers on the basis of the histograms plotted from fig 3.16 to 3.20 at different Hessian values. When the Hessian value is taken to be 550, we are able to have strongest resemblance of Convertible 2 with Convertible 1 which is expected, but if the Hessian value is kept to 250 it shows resemblance to Truck. At Hessian values of 200 and 750 it shows resemblance to SUV. The error is calculated by considering the mean of the absolute differences in the histograms. Chi square and different histogram matching techniques can also be used for this purpose which also produces similar results.

We can hence conclude that the Bag of words works well for different descriptors provided that the Hessian values of each of the descriptor have maximum match. Since k means is used, initial centroid guess is also crucial as it sets a hard bound in classification. Hence when the Hessian value is set to 550 we get that the Convertible 2 has strongest resemblance with Convertible 1.

Aakash Shanbhag

adshanbh@usc.edu

3205699915

References:

[1] EE569 Discussion Notes and Discussions.
[2] Image Processing –William Pratt textbook
[3] http://docs.opencv.org/3.2/doc/tutorials/tutorials.html
[5] https://github.com/pdollar/edges
[6] Ali Ghodsi, “Dimensionality Reduction A short tutorial”, Department of Stats, University of Waterloo, Canada, 2006.
[7] David G. Lowe. “Distinctive Image features from Scale Invariant Key points”, University of British Columbia, Vancouver, Canada.
[8] P M Panchal, S R Panchal, S K Shah, “A comparison of SIFT and SURF”, IJIRCCE, Vol 1, Issue 2, April 2013.