EE 569 HOMEWORK 3

GUNJAN JHAWAR
gunjanjh@usc.edu

Contents

1.Problem 1

1.1Abstract and Motivation

In texture classification, the goal is to assign an unknown sample image to one of a set of known texture classes. Texture classification is one of the four problem domains in the field of texture analysis. Texture classification process involves two phases: the learning phase and recognition phase. In the learning phase, the target is to build a model for the texture content of each texture class present in the labels. These features can be scalar numbers or discrete histogram. These textures have a certain degree of randomness to it. It is important to study different textures present in the image as they provide details required for image segmentation [1].

Segmentation is the separation of one or more regions or objects in an image based on a discontinuity or similarity criterion. There are different kind of segmentation: Pixel based segmentation, region based segmentation and edge based segmentation. We attempt to differentiate texture K means clustering is performed. The main motivation behind studying texture classification lies behind improving image segmentation results.

In the problem, we create 5 × 5 laws filter to extract feature vector from each pixel in the image after performing boundary extension. After that we use K mean algorithm to cluster the images and classify them to four different textures. In the second part of the problem, we apply 3 × 3 law filter to get gray scale images. Then apply K mean to perform segmentation on the composite texture to classify different textures. Every texture is assigned a different intensity level.

1.2 Approach and Procedure

There are two parts in this section. The methodology applied for texture classification is described in Section A and texture segmentation is described in Section B.

1.2.1 Texture Classification

To implement texture classification the following process is used:

1) **Creating the filter bank:** We first generate 5×5 law filter from 1D Laws filter corresponding to the edge, spot and wave.

$$L = \frac{1}{6}[-1 \quad -2 \quad 0 \quad 2 \quad 1]$$

$$E = \frac{1}{4}[-1 \quad 0 \quad 2 \quad 0 \quad -1]$$

$$W = \frac{1}{6}[-1 \quad 2 \quad 0 \quad -2 \quad 1]$$

Each 1D filter is combined with another filter using tensor product. The 2D filter created will have a specific frequency to detect a kind of feature in the image and filter out the rest of the frequencies. All the 2D filter are stored in a filter bank which is a vector.

2) **Pre- Processing:** We read the input images and subtract the mean component from each image to remove the unnecessary high feature values.

3) **Feature Extraction:** We then perform the convolution of the images with the laws filter. Each training image would give 9 filtered images corresponding to each filter bank. To calculate the energy of each filter response we would apply the following formula:

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

This is applied to every filtered image and the Energy that we compute is saved in a vector called Feature Vector. So, the feature vector has 9 values corresponding to every image. After all energy calculations, we would have 12 vectors, one for each image which consists of 9 values. Then this information is used for classification.

4)**K mean for clustering the textures:** Our aim is to cluster the input images to 4 clusters {Rock, Grass, Weave, Sand} based on the same kind of features that they consist. We initialize random centroid for each cluster. Then, we compute the Euclidean distance between the feature vector and k mean vector. The k mean index which gives the smallest distance is considered as the bin index and the feature vector is stored in the bin. As we know, K mean is an iterative algorithm so it is run for many iterations. After every iteration, the k mean vector is updated according to the feature vector is has stored in it. It exits when the maximum number of iterations are reached.

1.2.2 Texture Segmentation

To implement texture segmentation the following process is used:
1) **Creating the filter bank:** We first generate 3×3 law filter from 1D Laws filter corresponding to the edge, spot and level.

$$E = \frac{1}{2}[-1 \quad 0 \quad 1]$$

$$S = \frac{1}{6}[-1 \quad 2 \quad -1] \qquad L = \frac{1}{6}[1 \quad 2 \quad 1]$$

Each 1D filter is combined with another filter using tensor product. The 2D filter created will have a specific frequency to detect a kind of feature in the image and filter out the rest of the frequencies. All the 2D filter are stored in a filter bank which is a vector.

2) **Pre- Processing:** We read the input images and subtract the mean component from each image to remove the unnecessary high feature values.

3) **Law Feature Extraction:** We then perform the convolution of the images with the laws filter. Each training image would give 9 filtered images corresponding to each filter bank. To calculate the energy of each filter response we would apply the following formula:

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

This is applied to every pixel of the filtered image and the energy that we compute is saved in a vector called Feature Vector. The feature vector for each image would be of size Image Width × Image Height.

4)**Normalization:** As we see that all kernels have zero mean except the L3 × L3 kernel. So, we need to divide all the feature vector by the feature vector obtained using the above-mentioned feature vector

.

5)**K mean for clustering the textures:** Our aim is to divide the single input images to 6 clusters based on the segmenting the pixel value if they are nearer to the k mean centroid. We initialize random centroid for each cluster. Then, we compute the Euclidean distance between the feature vector and k mean vector. The k mean index which gives the smallest distance is considered as the bin index and the feature vector is stored in the bin. As we know, K mean is an iterative algorithm so it is run for a number of iterations. After every iteration, the k mean vector is updated according to the feature vector is has stored in it. It exits when the maximum number of iterations are reached. We assign different intensity levels to different bin corresponding to the k mean vector. This makes it easier to visualize the number of different textures that an image has.

1.3Experimental Results

A.

Figure 1Output of Texture Classification where 1: Rock 2: Grass 3. Sand 4. Weave

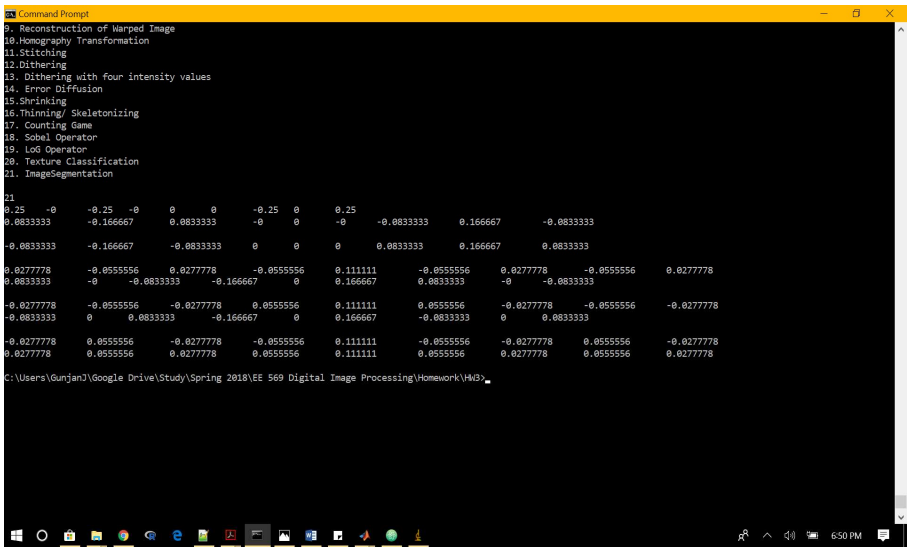
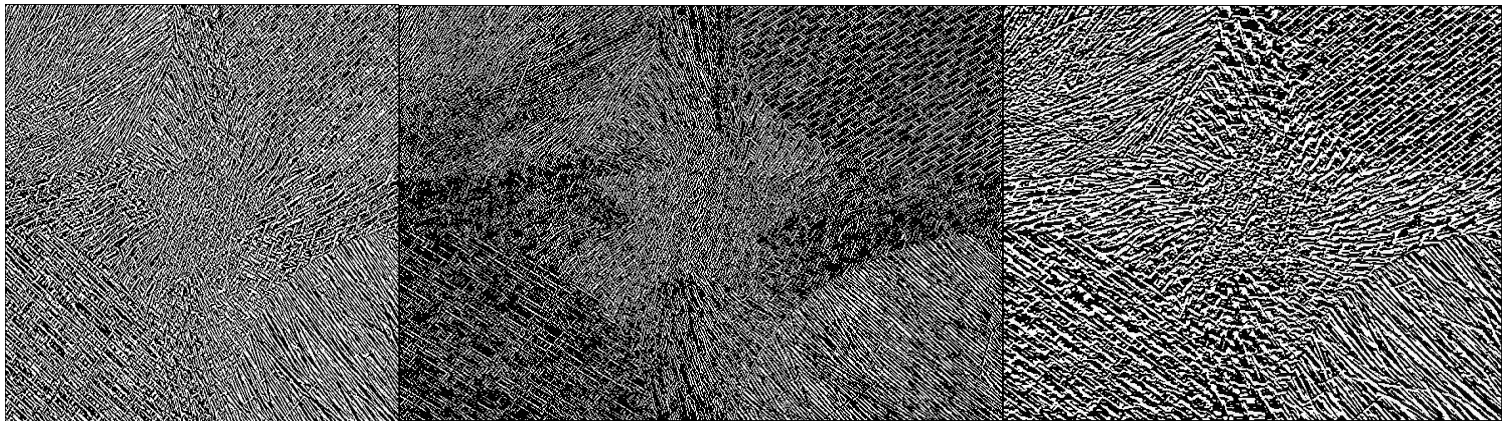
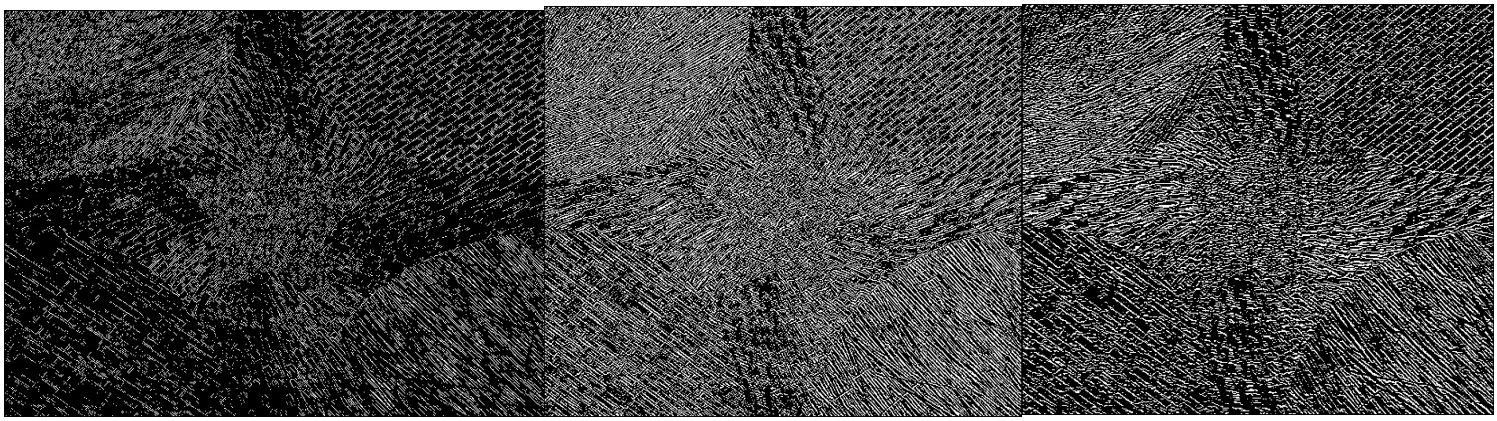


Figure 2Filter obtained for Image Segmentation





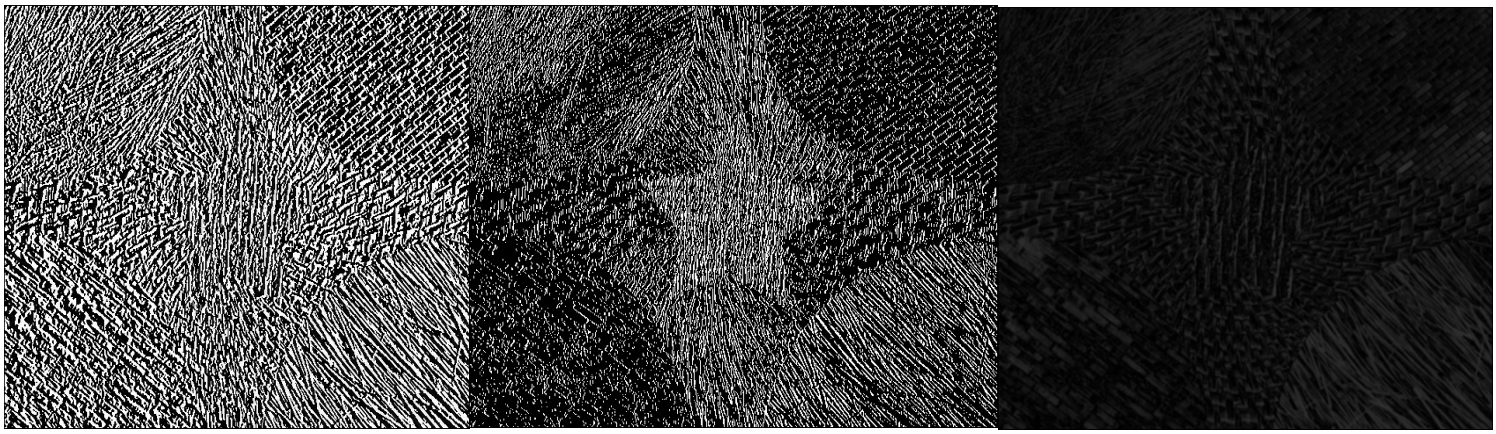


Figure 3 Image obtained after applying nine filter on comb.raw

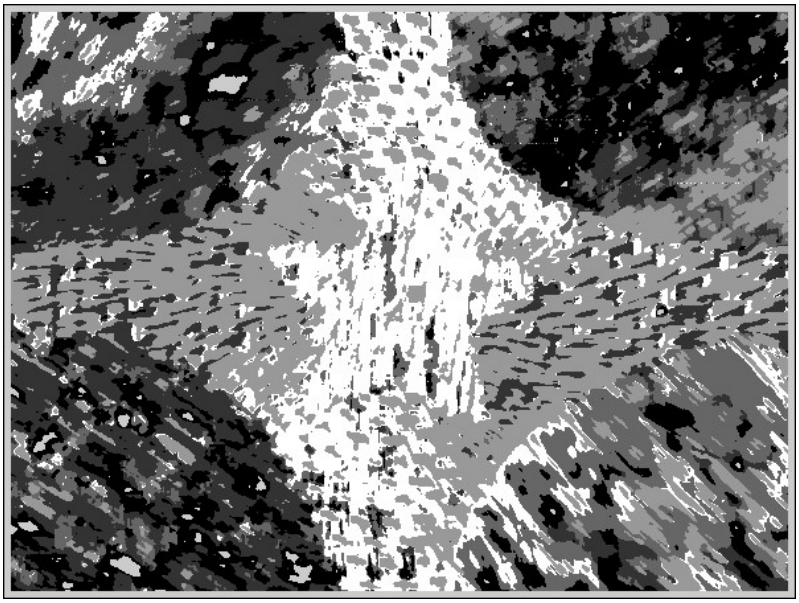


Figure 4 Output image after Image Segmentation

1.4 Discussion

A.

- The frequency response helps to extract the required the features of the image. We find that the E5E5 filter has the most discriminative power while the W5W5 has the weakest discriminative power. We can verify that by computing the feature vectors and taking out the difference between them, the one with the highest difference classifies the feature differently as in they do not have overlapping points while the one with the smallest difference would have the weakest discriminative power and overlapping points.

- If we verify the classification that we get by implementing the k mean we see that all the images are correctly classified except one which shows that we need more training data in order for k mean to give 100% accurate result.
- We can also deduce that k mean clustering depends highly on the quality of the features that are obtained and preprocessing of the image.
- The observation was made that running the program multiple times with random k mean initialization still gave the same accuracy.

B. Image segmentation is a challenging task which is highly dependent on the energy value extraction.

- When different window sizes are used less samples are available in case of smaller window to compute the quality of energy vector will not be adequate thus affecting the result. Larger window will give better result.
- Segmentation algorithm is also broken in edges as less pixel are available for computation.
- L3L3 filter was not taken after normalizing the feature vector which did not contribute to the distinctiveness of the texture energy.

2.Problem 2

2.1Abstract and Motivation

Edge detection aims at identifying points in a digital image where the brightness of the image changes drastically. The set of curved line segment are organized into edges. The detection of edge is a pressing problem but is used in various fields like object detection, object tracking, segmentation, registration and so on. It can be classifying as a viewpoint independent or viewpoint independent problem. The novel edge detectors that effect the flexibility of edge detection method by accommodation with other fields of science. Sobel operator is an algorithm where it creates an image emphasizing edges. It is a discrete differentiation operator computing an approximation of the gradient of image intensity function. In the problem, we will implement sobel operator and tune the threshold to obtain best edge map.

In this problem, we will implement the Laplacian of gaussian filter. We derive an algorithm which determines two knee locations of loG histogram. Then we partition the loG histogram to three intensity values to obtain the edge map.

The structured edge detector takes advantage of the structure present in the local image patches to learn both an accurate and computationally efficient edge detector. In this problem, we implement structured edge detector to extract edge segments from a color image using [2]. The algorithm is described with the description of multiple decision trees and their integration into one final probability function. The parameters that are used and how they are varied is also explained in the procedure part.

This problem, we compute the true positive, true negative, false positive and false negative to compute precision and recall of the edge map obtained from two images passed by the user. Precision and recall are metrics to define the output quality of the classifier.

2.2Approach and Procedures

There are three parts in this section. The methodology applied for sobel edge detector and zero crossing detector described in Section 2.2.1 in A and B. The algorithm and the implementation of structured edge detector is described in Section 2.2.2 in A, B and C respectively. The performance evaluation of the above implemented methods is described in Section 2.2.3.

2.2.1 Sobel Detector and Zero Crossing Detector

A. Sobel Detector

Sobel filter creates an image emphasizing edges. At each point in the image the result of Sobel operator is either corresponding the gradient vector of the norm of the vector. The sobel operator is based on convolving the image with a small separable and integer value filter in the horizontal and vertical directions and is inexpensive in terms of computation. On the other hand, the gradient approximation is relatively crude and particularity used for high variations in the image. [3]
[The horizontal and derivative approximations are stated below:

$$Gx = \begin{matrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{matrix} \qquad Gy = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$$

After the gradient map is implemented on the image, the magnitude of the image pixel is computed the formula below:

$$\sqrt{Gx^2 + Gy^2}$$

Then using the cumulative histogram of the image that we get after computing the magnitude of

Each pixel, we compute the threshold value. The threshold value is taken as the point where the top 90% for example as present.

B. Zero Crossing Detector

Zero crossing is a point where the sign of the mathematical function changes, it is represented by the intercept of the axis of the function. Laplacian of Gaussian filter uses that rapid change and zero crossing to the find the edge in the image. Since, these derivates are very sensitive to noise. It is common to smooth the image using Gaussian filter. These two processes is called Laplacian of the Gaussian operation.

A possible kernel for Laplacian operation is:

$$G = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

To include a smoothing gaussian filter, combine the Laplacian and gaussian function to obtain a single equation. When this is done, we get a filter like this:

$$LoG = \begin{matrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 1 & 2 & -16 & 2 & 1 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}$$

This LoG kernel is applied to the image. After applying, the LoG histogram will be in the form of a sharp symmetric spike. It detects all the zero crossing in the image and then the values are threshold and only the strong ones i.e. the ones that give large difference between negative minimum and positive maximum is kept. From those two values, the one that divide the histogram in almost the total pixel/4 and the other knee location should be closer to total pixel/2. So, now we have the two knee we can divide the whole histogram into three intensity values as -1/64 as the one side of the pixel value before the first knee location. The second pixels value as 128 for the pixel index that come in between the first knee and second knee location. And the third pixels value as 192 for the pixel index that come after the second knee location.

2.2.2 Structured Edge

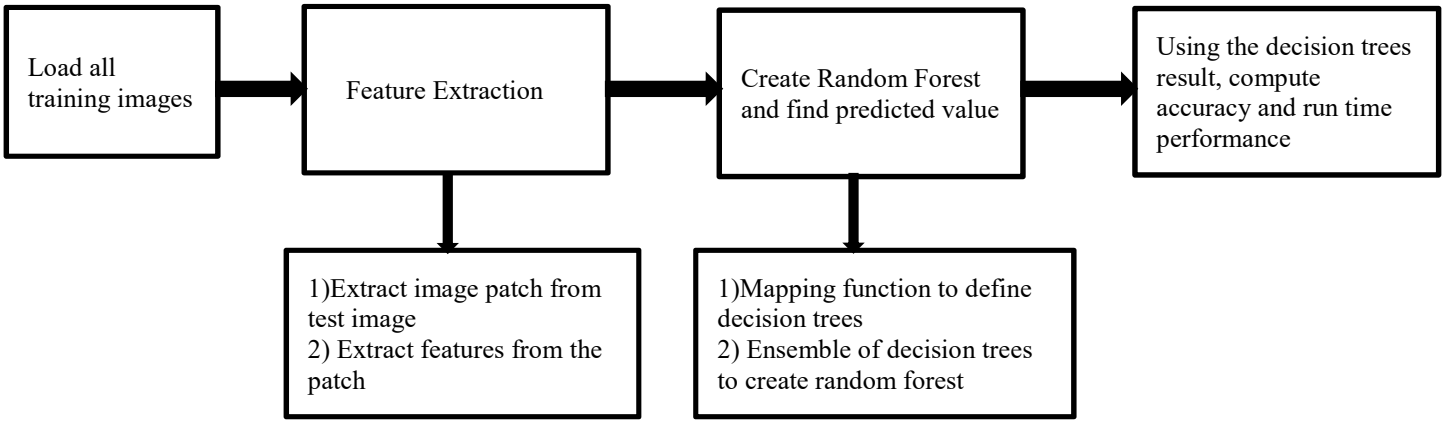
The novel approach to learning decision trees robustly maps the structured labels to a discrete space on which standard information gain measures may be evaluated. The result is an approach that obtains real time performance that is orders of magnitude faster than many competing state-of-the-art approaches, while also achieving state-of-the-art edge detection results on the BSDS500 Segmentation dataset and NYU Depth dataset. Finally, we show the potential of our approach as a general-purpose edge detector by showing our learned edge models generalize well across datasets. [4]

A. Implementation of Structured Edge Detection algorithm

1) **Load all training image:** The input to the algorithm is an image that can contain multiple channels like RGB/ RGBD.
2) **Feature extraction:** The learning approach predicts a 16 × 16 segmentation mask from 32 × 32 image patch. Each image patch is augmented with multiple additional channels of information and are stored in 32 × 32 ×K vector where K is number of channels. Two types of features are taken: Pixel lookups (x,y,z) and Pairwise Differences (x1,y,1z1) – (x2,y2,z2).
   Each gradient magnitude is spilt into 4 channels based on orientation. Then they are blurred with a triangle filter of radius 2 and down sampled by factor 2 for pixel lookup and radius 8 and down sample by 5 ×5. So, in the end, 3328 candidate features in pixel lookup and 7228 candidate features per patch in pairwise.
3) **Creating a mapping function and Ensemble Model for Random Forest:** To train the decision trees, we define a mapping that encodes y(j1) = y(j2) for every unique pair of indices where j1 is not equal to j2 and y(j) denotes j^{th} pixel of mask y.The ensemble approach combines weak learners to form a strong learner. Random forest

achieves robust results by combining the output of multiple decorrelated tress. The output is computed densely on the image with stride 2. Thus with 16 × 16 patches it gets $16^2T/4 = 64T$ predictions where T being the number of trees.

4) **Multiscale detection:** The edge detection is done on original, half and double resolution version of input image and the average of the three edge maps is taken.
5) **Parameters:** They include image and label patch size and decision trees parameters. All parameters are finalized using validation sets.



B. Decision Tree Construction and principle of RF classifier

The principle behind the RF classifier is that a group of weak learners can come together to form a strong learner. A decision tree classifies a sample by recursively branching left or right down the tree until the end node is reached.

1) For each node, it finds parameters of the split function which would result in a good split. The binary split function is computed for the input x and parameters if it is a zero, x is send to left otherwise to right node.
2) The parameters are chosen in a way that they maximize the information gain.
3) It is carried on till the depth of the tree is reached or the training set size falls below threshold.
4) The output of the tree on an input say x will be the prediction stored at the leaf node x reaches which can be an output value y or a distribution over the labels.

5) A random forest is an ensemble model of all independent decision trees. Choice of ensemble model is problem specific and depends on how the output is for the random forest. The common choices of the representation of the output can be majority voting for classification or averaging for regression.
6) The random forest should consist of diversity of trees which can be obtained by randomly subsampling the data which is used to train each tree or subsampling the feature and split to train each node.
7) As we are training the random forest with structured labels, it is more complex. We cannot define the information gain by the similarity of the output we get. We provide a mapping to the intermediate space to define whether a set of y belongs to a segment or not. It is calculated using Euclidean space in Z space. PCA is done to reduce the dimensionality of Z.
8) The z's are then clustered into k cluster using k mean to detect whether the cluster is an edge or not.

C. Application of SE Detector

The application of SE detector is done on Animal and House image using MATLAB. The available online code was used for performing structured edge detection as asked in the question from https://github.com/pdollar/edges. From the list of files, edgesDemo.m file was used to perform the Structured Edge detection algorithm. The logic used was:

1) Set the training parameters and model parameters for an image dataset BDS500.
2) Obtain the labels for these images.
3) Extract lot of P patches which act as training data.
4) Compute the gradient and color to extract features from the patches.
5) Train the structured random forest.
6) Perform edge detection by extracting patches from the image to be tested on.
7) Extract features from patches and applying classifiers.
8) Place the predicted value into the output image to get probability edge map.
9) Set the threshold to the probability edge map to obtain binary edge map.

2.2.3 Performance Evaluation

To evaluate the performance of different edge detectors, we have the used the online source code as mentioned above. Running the edgesEvalImg.m to obtain precision and recall. Using:

[thrs, cntR, sumR, cntP, sumP, V] = edgesEvalImg(Binary Map, <ground truth file>)

This is done because the edges detected can never be as perfect as humans. The one evaluated by humans are called ground truth. In comparison to that, how well the computer performs is done through these parameters.

We calculate the F measure using:

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{Precision + Recal}$$

F measures gives the final evaluation metrics.

2.3Experimental Results

A.

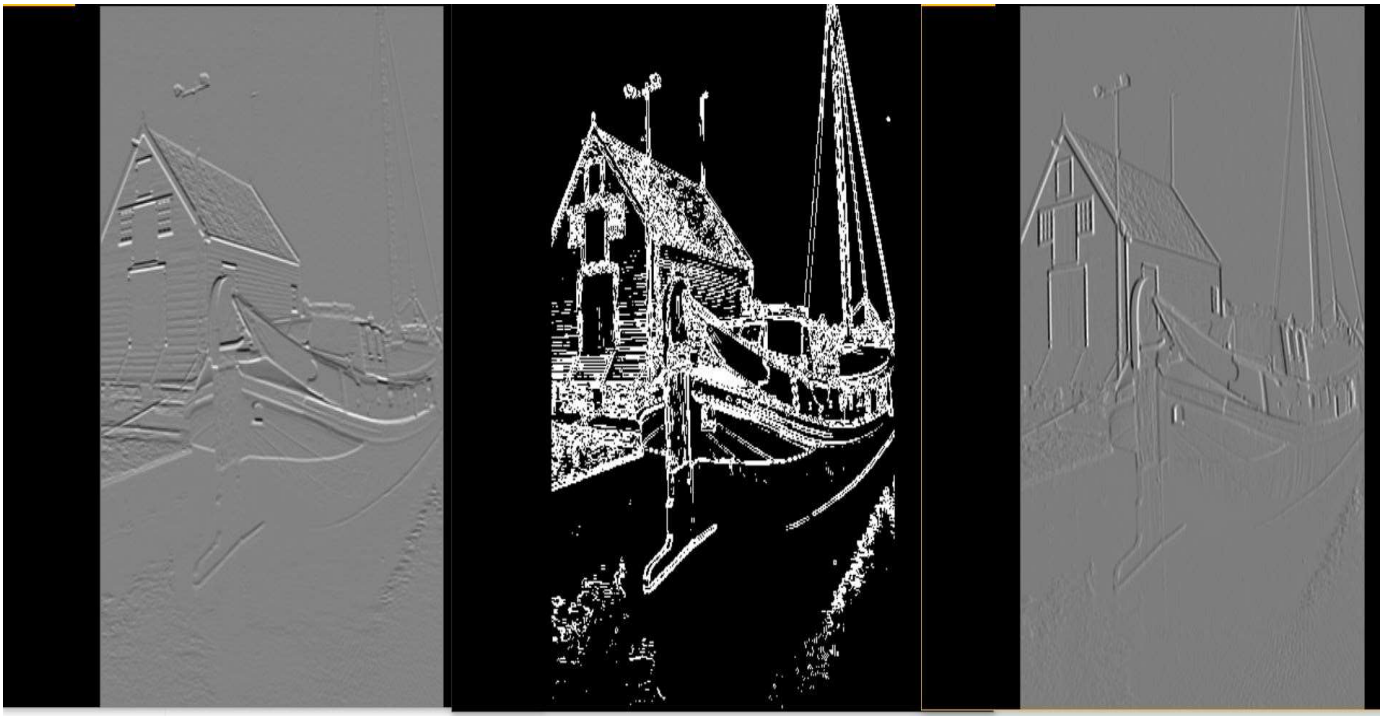


Figure 5Sobel operator with Gx, magnitude and Gy on boat.raw

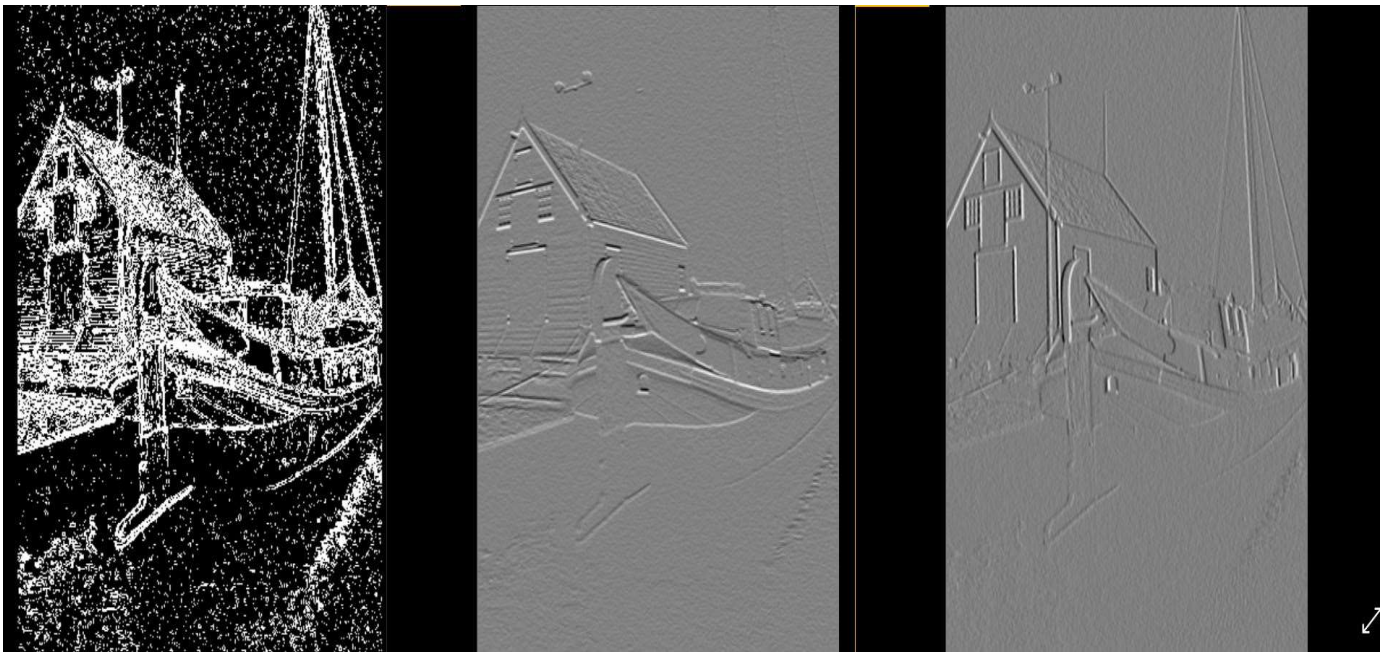


Figure 6 Sobel Operator on BoatNoisy with Magnitude, Gx and Gy.

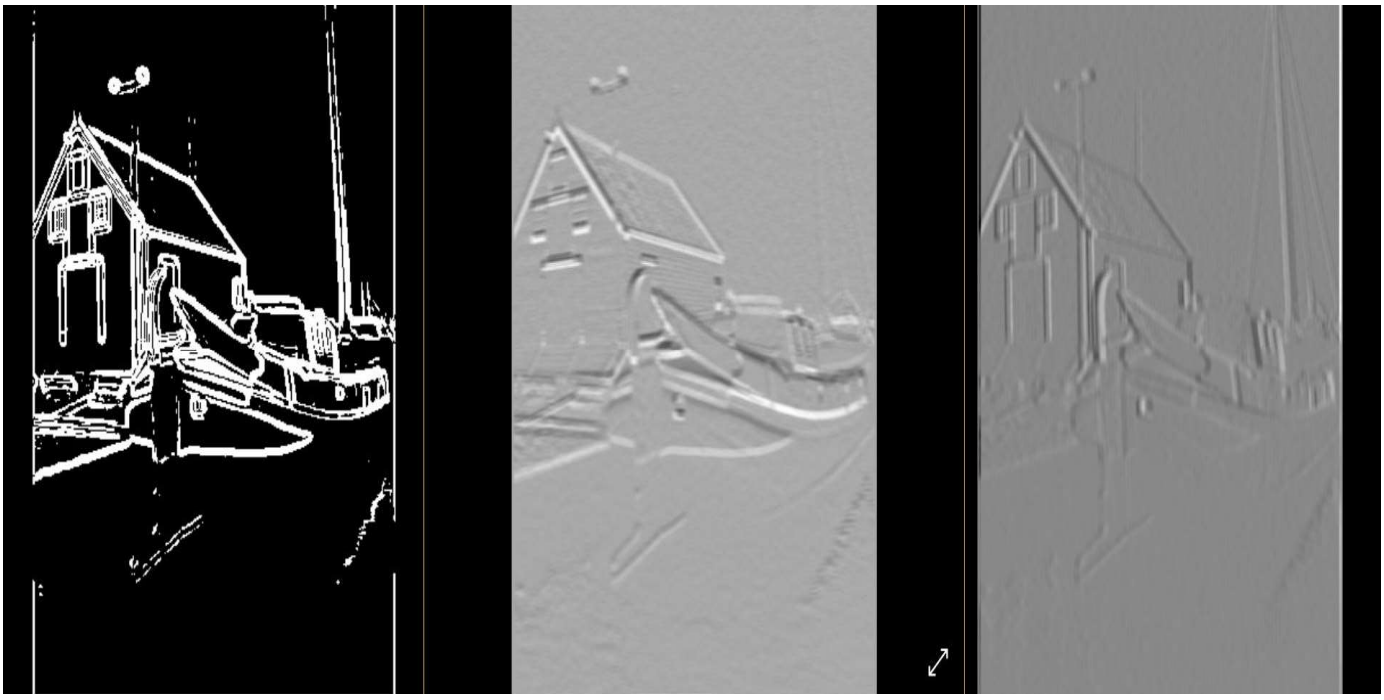


Figure 7 BoatNoisy is denoised and op is Magnitude, Gx and Gy

\

Figure 8 LOG with boat.raw and boat_noisy.raw

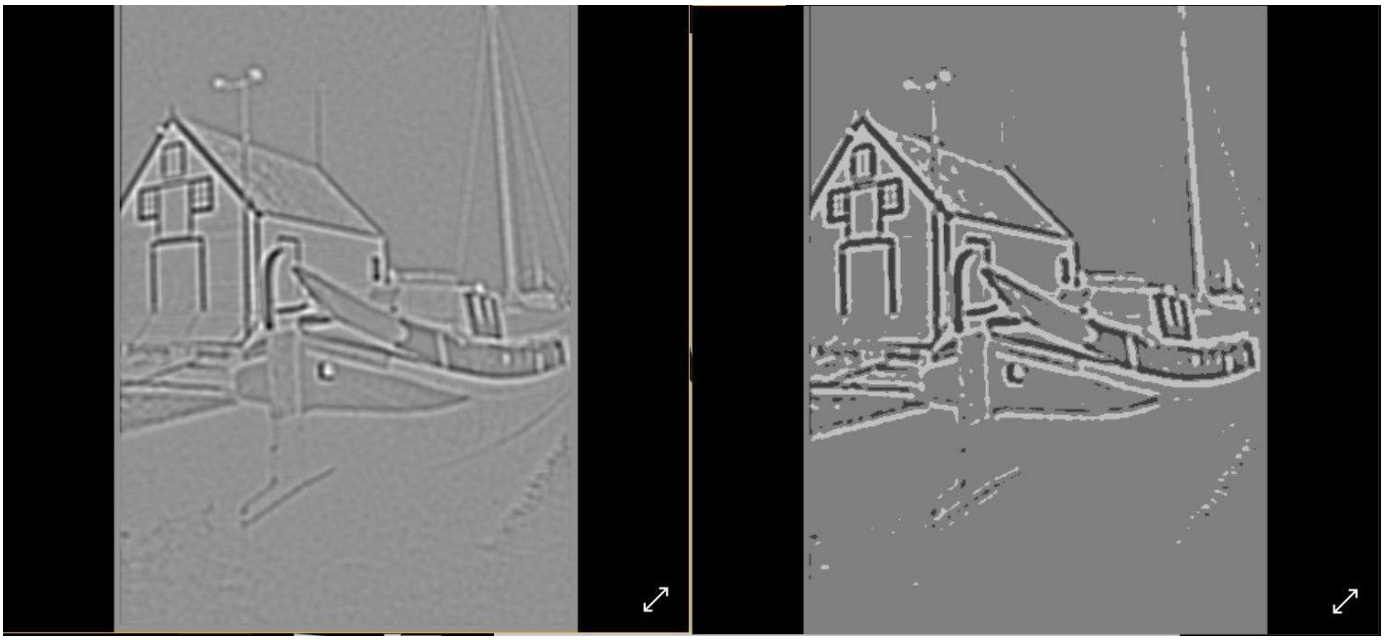


Figure 9 Knee Threshold on boat and boatnoisy

B.

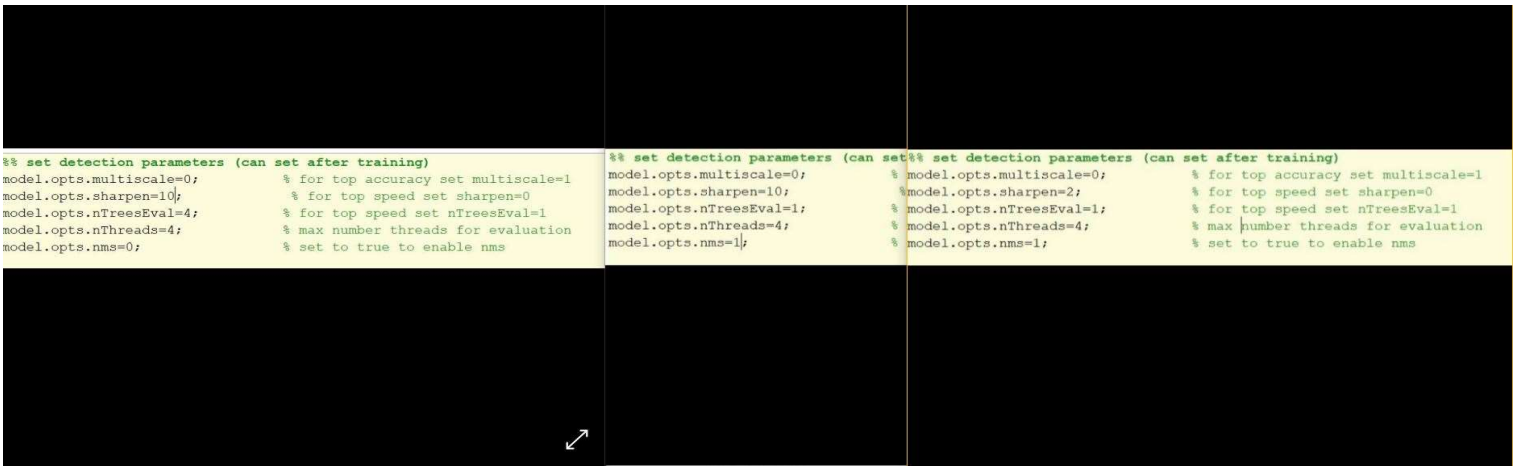


Figure 10Model Parameters

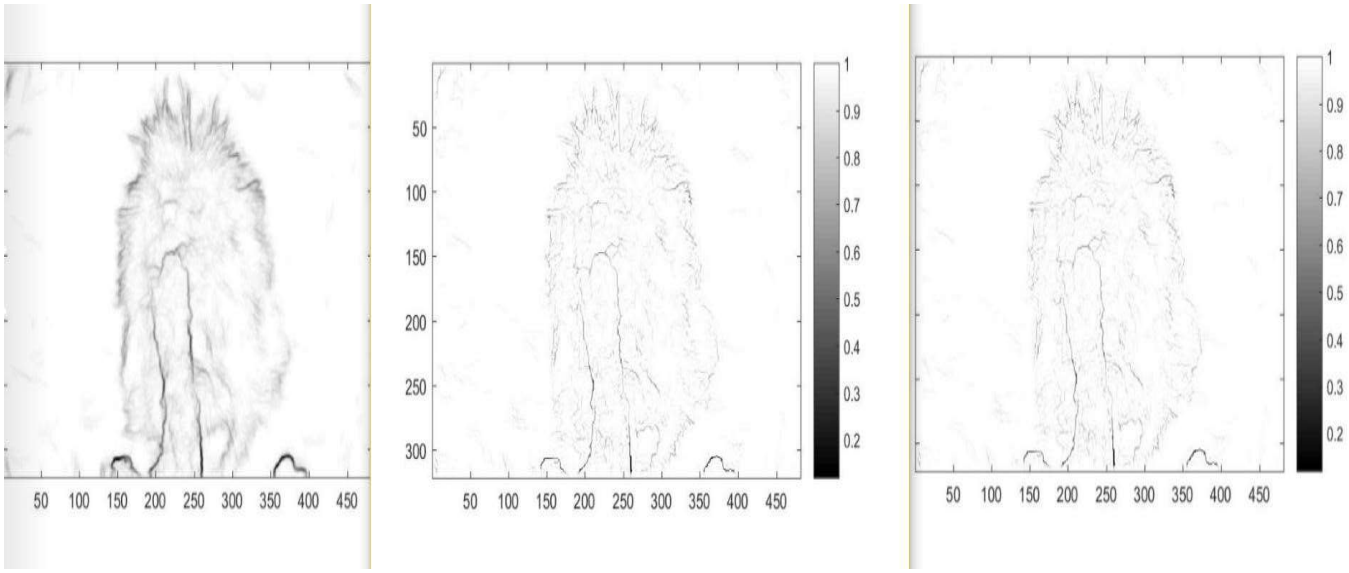


Figure 11Op for all 3 models

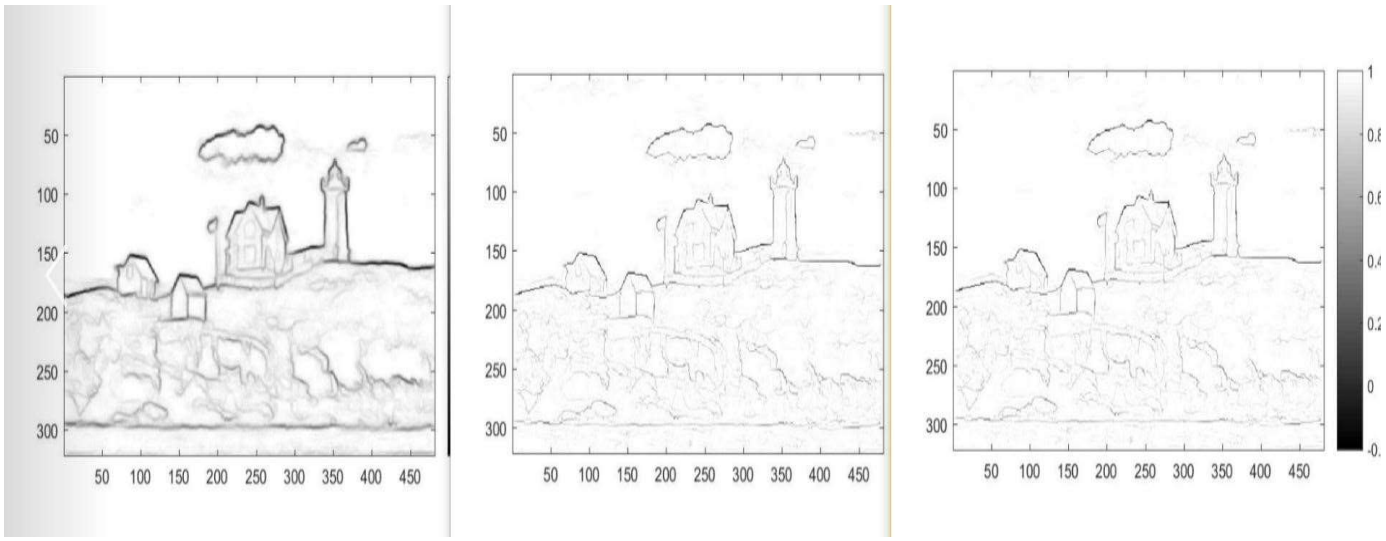


Figure 12 Op for corresponding model on House

```
Elapsed time is 0.715048 seconds.
Elapsed time is 0.076743 seconds.
>> edgesDemo
Elapsed time is 0.640597 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.070874 seconds.
>> edgesDemo
Elapsed time is 0.589677 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.066089 seconds.
>> edgesDemo
Elapsed time is 0.563216 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.065079 seconds.
>> edgesDemo
Elapsed time is 0.627408 seconds.
Elapsed time is 0.037984 seconds.
```

```
>> edgesDemo
Elapsed time is 0.575250 seconds.
Elapsed time is 0.080322 seconds.
>> edgesDemo
Elapsed time is 0.677401 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.081362 seconds.
>> edgesDemo
Elapsed time is 0.664643 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.079243 seconds.
>> edgesDemo
Elapsed time is 0.563800 seconds.
Model supports sharpening of at most 2 pixels!
Elapsed time is 0.041666 seconds.
>> edgesDemo
Elapsed time is 0.558405 seconds.
Elapsed time is 0.039037 seconds.
>>
```

Figure 13 Time for Model 1 and 2 respectively

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	Image	Threshold	Precision1	Recall1	Precision2	Recall2	Precision3	Recall3	Precision4	Recall4	Precision5	Recall5	Avg Precisi	Avg Recall	F Measure	
	Animal	0.01	0.162285	0.51041	0.175903	0.474006	0.107055	0.407194	0.191413	0.539158	0.156232	0.457618	0.1585	0.4776	0.238	
	House	0.01	0.087525	0.3497	0.177	0.4353	0.158331	0.4452	0.130852	0.376	0.1176	0.35321	0.1341	0.39188	0.19982	

Figure 14 Performance evaluation

2.4 Discussion

A. Sobel detector provides good approximation of gradient map and orientation of edge. One major disadvantage is its sensitivity to noise. Sobel operator is unable to predict edge pixels which are thin in nature.

While LoG is 2D isotropic measure of 2D spatial derivative of an image. It has gaussian filter applied to it which helps to sustain noise and second derivative aid in rapid intensity changes. If comparing both of them, sobel is robust in efficient in extracting change in gradient while LoG is more robusr in case of complex situations.

B. Structured Edge detection ignores unimportant edges though they have high gradient change. This method analyses and trains on the textures and edges. The most prominent thing to notice is the performance difference between sobel, structure edge and Zero crossing edge detector is because structured edge uses machine learning in some way and train the images and predicts the output. We can see that we get better output when we use non max suppression, we get more crisp edges. Also, when we have more trees we get more runtime.

C. The Animal image is easier to get high F measure for. The Animal Image has more F measure as compared to the House image because of less background cluttering in the image. In the house image there are a lot of background that is not letting it perform well.

   The F measure is a test of accuracy in statistics. Its value id affected by both precision and recall. If one metric is low it results in bad F measure. There is an inverse relation between precision and recall. If recall increases, precision decreases and vice versa. F measure is maximum when precision and recall are maximum.

3.Problem 3

3.1Abstract and Motivation

Salient point extraction is an important task in computer vision specially used in areas of object recognition, video tracking, nevigation and gesture recognition. SIFT and SURF are patented local feature detector and descriptor. The extracted features are required to be scale and rotation invariant. The feature extracted from SIFT and SURF are scale invariant and rotation invariant.

In this problem, we will implement SIFT and SURF algorithm to extract salient points in the image. Then compare the performance and efficiency of both the algorithms.

In this problem, we will utilize the features obtained by both the SIFT and SURF algorithm to perform object matching in two different image using Brute Force matching technique.

In this problem, we will implement the bag of words algorithm which is used for image classification of small system where user can set the number of clusters.

3.2Approach and Procedures

There are three parts in this section. The methodology extracting and description of salient points is applied in part A, image matching in part B, bag of words in part C are described.

3.2.1 Extraction and Description of Salient Points

SIFT was proposed by David G Lowe. SIFT stands for Scale Invariant Feature Transform

The main idea behind getting a good feature is to get good classification accuracy because if the features themselves are not good, the classifier won't be able to find good output. To implement this algorithm, we have used OpenCV that contains SIFT detector and extractors. Following steps are followed to extract SIFT feature:

1) **Scale Space Extrema Detection:** We consider the input image of different scale to make it scale invariant. This is done using Gaussian smoothing 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. Additionally, each image is smoothened by a sigma value that is multiple of k, k being square root of 2 heuristically. Similarly, the value of sigma was assigned as 1.6. After applying Gaussian smoothening, the difference of each value is taken. It is done to approximate the LoG function as the earlier one would take a lot of time to compute. The DoG provides an edge map. Detection of point of interests are taken. It is iterated through all possible points and select the points of interest lies in the local maxima and minima at a given scale.

2) **Key point localization:** In this step, points of interest computed weak and noisy points due to low contrast. To avoid such outliers, we compute the Taylor series expansion of DoG and remove the points that lie of below a threshold of 0.3. We compute the hessian matrix and compare its eigen values. If the ratio of first and second eigen values are greater than 10 that point of interest is an edge point. And we discard them.

3) **Orientation Assignment:** To achieve rotational invariance, we assign an orientation value to each point. It is done by first taking a neighborhood around point of interest. The neighborhood size depends on the scale at which point was detected and a histogram of 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 value is selected. All the orientation that lie within 80 percent of the maximum increases the robustness.

4) **Key point descriptors:** It is important to us to create feature values for a point of interest. The author suggests the histogram of the neighborhood orientation as features as they provide variation against illumination effects, varying angle difference etc. To do this, we first consider a 16 ×16 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.

Like SIFT method, we implemented SURF method on OpenCV. SURF stands for speeded up robust features. This provides a faster way to obtain the feature points.

3.2.2 Image Matching

In the first part, we extract important features from the image. Using these features we can compute the relevant matches among images. The above steps are performed as above along with one more additional step that is:

1) **Key point matching:** After extracting SIFT features from two images, nearest neighbor seach is performed to find matching points. It might happen that these two neighbors are close to the test point. In case like that we take the ration of the first best and second distance and then threshold. If the above ratio is 0.8 or above, we reject the matching. This is a good way to reduce a lot of irrelevant matches.

After SIFT or SURF feature extraction using OpenCV, we perform Brute Force matcher and FLANN based matchers.

3.2.3 Bag of Words

We aim to perform classification of unknown test image. part, we extract important features from the image. Using these features, we can compute the relevant matches among images. It is the vector of occurrence of count of words and is expressed as a histogram of each word. Training the bag of words is done to cluster the training images to clusters. The calculated features of test images are then compared to the one stored in our dictionary.

The following steps are the implementation of bag of words:

1) Firstly, we extract the local features of the image using SIFT. Then place all the features into a single data set.
2) Over the feature vector, we find the centroid co-ordinates and assign the centroid code names. K means clustering algorithm is computed over the feature vectors.
3) Finally, a feature vector is obtained based on global identities. Then we classify them based on K nearest neighbor.

3.3Experimental Results

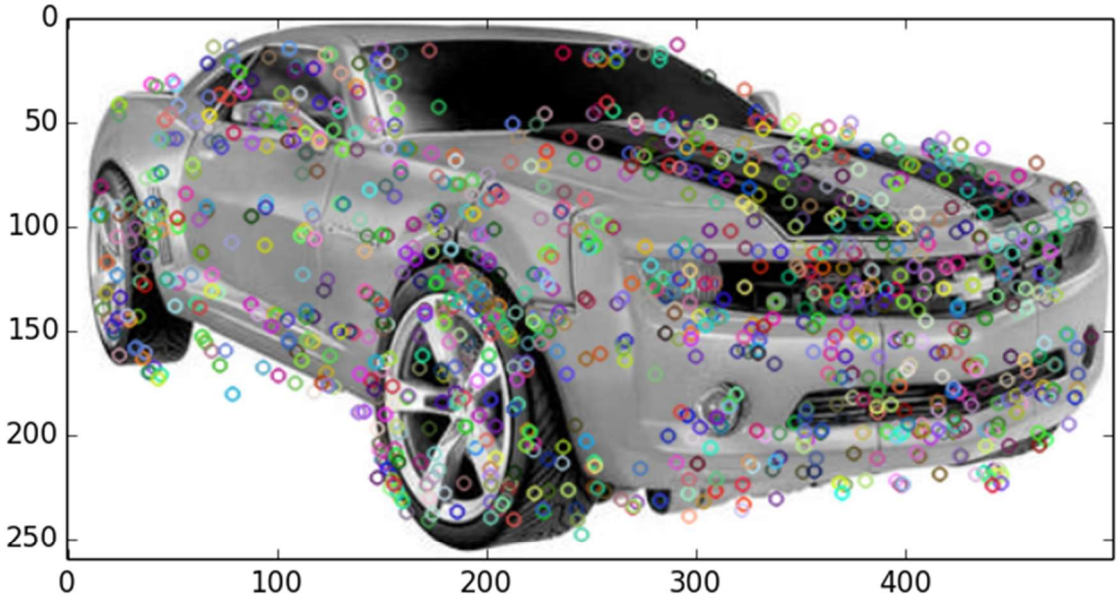


Figure 15 SURF on Bumblebee.raw

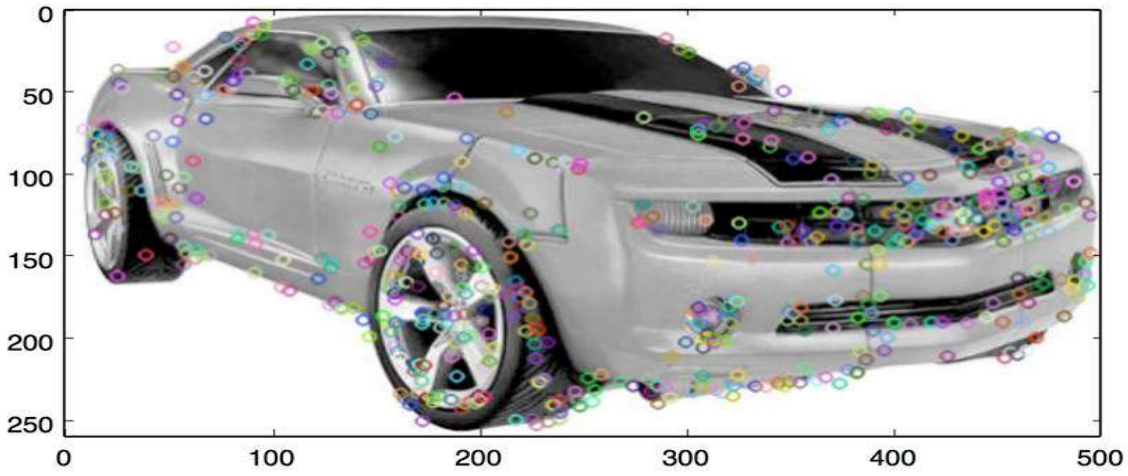


Figure 16 SIFT on Bumblebee.raw

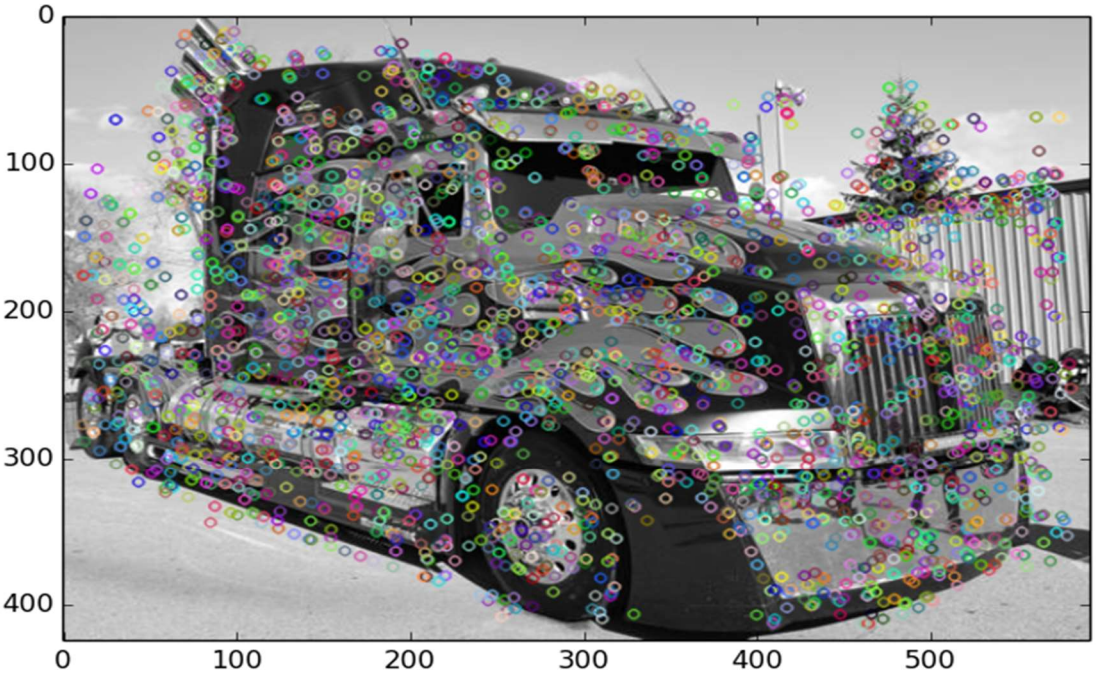


Figure 17 SURF on Optimus Prime.raw

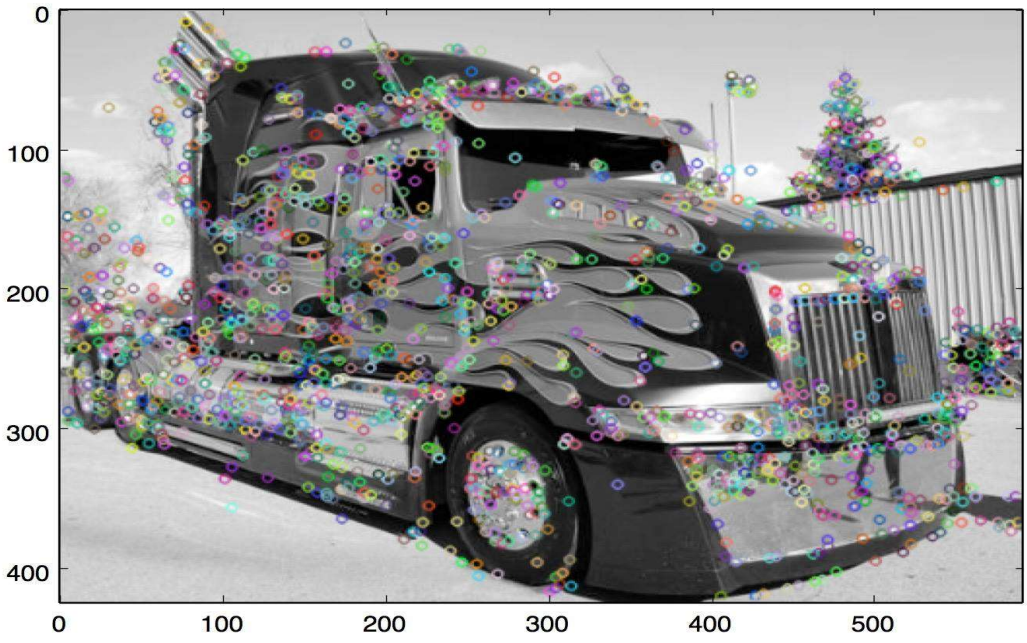


Figure 18 SIFT on OptimusPrime.raw

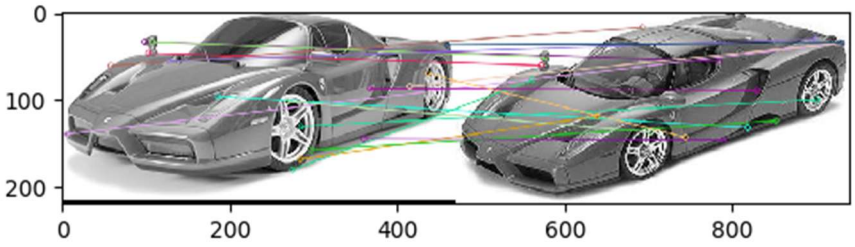


Figure 19Image Matching between Ferrari 1 and Ferrari2 using SIFT

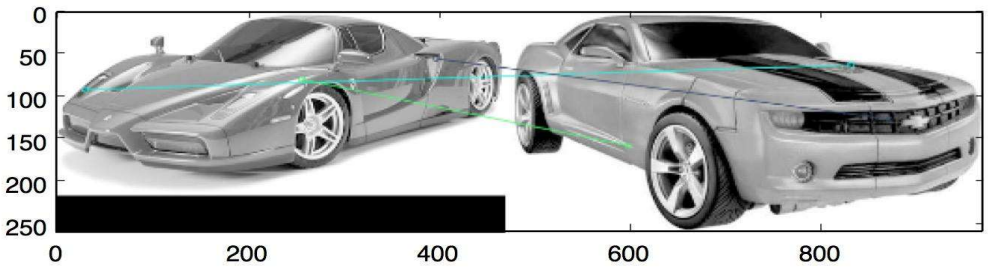


Figure 20 Image Matching between Ferrari1 and Bumblebee using SIFT

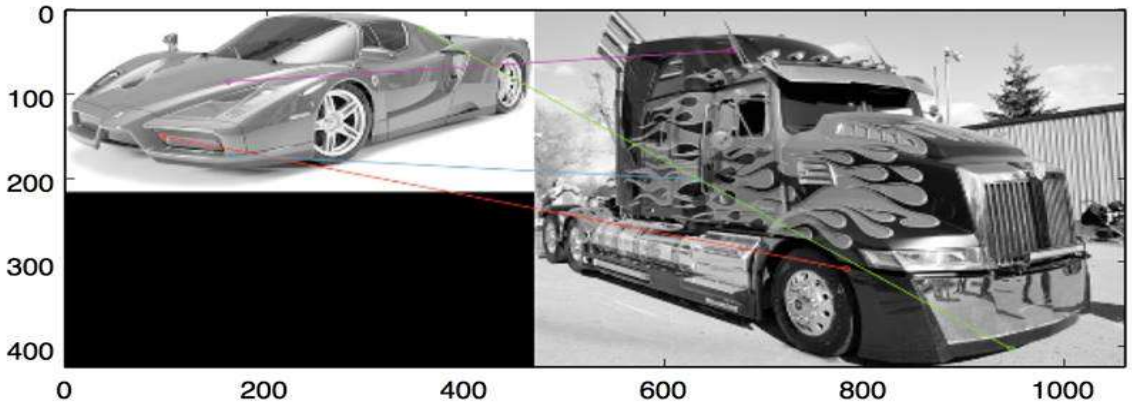


Figure 21 Image Matching between Ferrari1 and Optimus Prime using SIFT

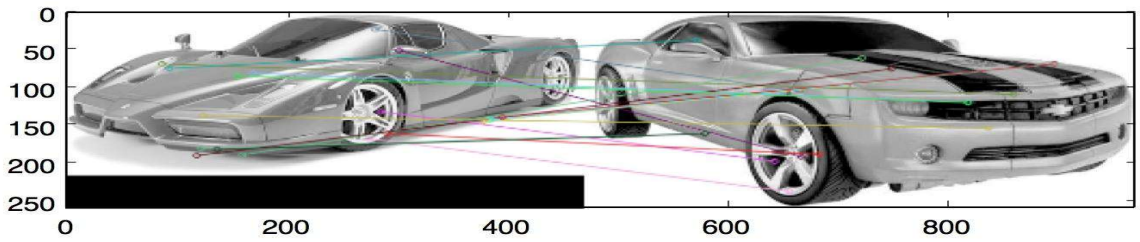


Figure 22 Image Matching between Ferrari1 and Bumblebee using SURF

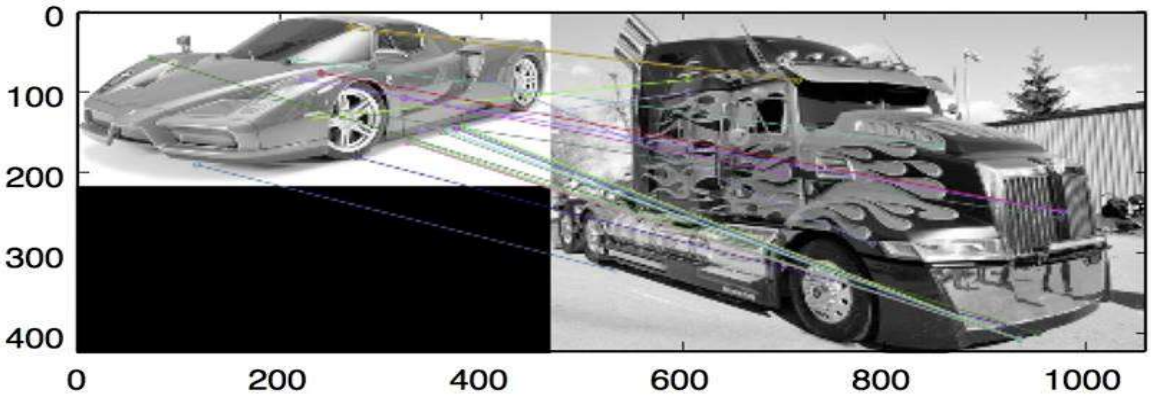


Figure 23 Image Matching using Ferrari1 and Optimus using SURF

Figure 24 BOG output predicting Ferrari2

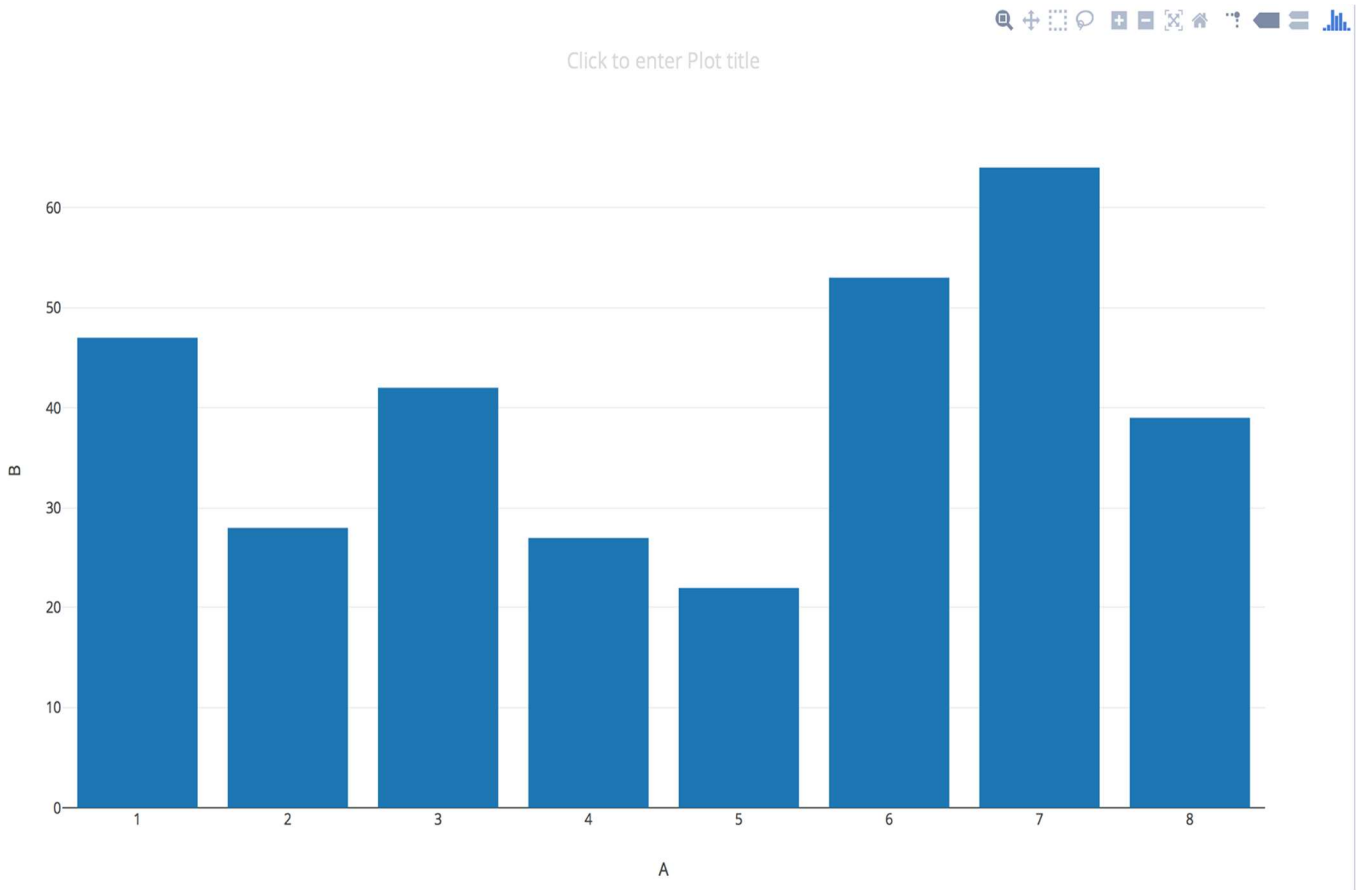


Figure 25 Histogram of Ferrari2 in terms of codewords

3.4 Discussion

A. SIFT and SURF features are compared in terms of their efficiency and performance.
   - SURF is faster in computation as compared to SIFT.
   - The data set is small so there is not much difference in the computation time.
   - Biggest advantage of SIFT is that it is invariant to scaling, rotation and change in illumination.
   - SIFT provide more features than SURF in our image.
   - SIFT uses non- maximum suppression for key point detection eliminating outliners using Hessian matrix. On the other hand, SURF uses hessian matrix to determine points of interest.
   - SIFT involves weighted orientation of surrounding pixels for orientation calculation while SURF is a sliding orientation.

B. We see that image matching does not work well when the images are different like Ferrari and bumblebee while it works well when we take Ferrari 1 and Ferrari 2 image. It is fails due to the following reasons:
   - The dimensions of the objects being matched may be different due to which the key points as they are quite apart which lead to incorrect matching. One way to aid that is to use bilinear interpolation.
   - The orientation of the images that are matched may be different.
   - Brute force matcher is also a big issue. As it computes a match for every point noisy matches that could be filtered out. In order to do that, we calculate the maximum and minimum values of distance between each match. Then we consider the matches which are less than the twice the minimum value. It might give error so we would use FLANN based matcher as it uses nearest neighbor technique.

C. Using the bag of words, quality of the image is a major driving factor. The algorithm has correctly classified the test image.

4 References

[1] http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OJALA1/texclas.htm

[2] Dollár, Piotr, and C. Lawrence Zitnick. "Structured forests for fast edge detection." Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013.

[3] https://en.wikipedia.org/wiki/Sobel_operator

[4] https://www.microsoft.com/en-us/research/publication/structured-forests-for-fast-edge-detection/

[5] https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm