EE569 Digital Image Processing: Homework #4

Name: Nazim N Shaikh
USC Id: 8711456229
Email Id: nshaikh@usc.edu
Submission Date: 03/19/2019

Problem 1: Texture Analysis

I. Abstract and Motivation

- Textures are set of elements occurring in image in some repeating pattern. Image texture is a feature used in partitioning images into regions of interest and further helps in classifying those regions. It also helps to quantify various image qualities such as roughness, smoothness, bumps, etc. by providing information about spatial arrangement of intensity values.
- Texture analysis is widely used in many applications such as image segmentation of ground images obtained through satellite, remote sensing, biomedical, Image Classification, etc.

II. Approach

- There are 2 main approach to texture analysis – Texture Classification and Texture Segmentation.

a) Texture Classification: In this approach, we try to assign labels to textures in a sample image from a given set of various texture images. Here, we make use of unsupervised classification algorithm called K-means clustering where each image is grouped into a cluster having similar properties based on feature vectors. The feature vectors are formed by applying a set of laws filters on the images and finding localized energy values for each image.

The algorithm is implemented as follows:

1. We are given a set of 12 input texture images. For each texture image, subtract image mean from input image. It is also known as DC component removal. This is done to remove any high feature values which might obscure relevant features. This is preprocessing step.

2. Convolve each preprocessed image with a set of 2D laws filters. (boundary extension was done by pixel replication). The 5x5 laws filter are generated from a given set of five 1D filters (shown in table below). Each 1D filter is combined with another by taking a tensor product to obtain 25 5x5 filters.

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

Table 1: 1D Kernel for 5x5 Laws Filters

The convolution results in 25 filtered images for each of 12 input texture images.

3. Compute average energy values for each filtered image of each input image using below formula:

$$E = \frac{\sum_{i=0}^{N} \sum_{j=0}^{M} |\, I(i,j) \,|}{N * M}$$

Where N and M are image height and width respectively.
This gives us 25D feature vector for each input image, resulting in final feature vector of size 12x25.

4. The feature space obtained is of 25 dimensions, all of which may not be important for classification. So, we perform dimensionality reduction that will give us a feature set that has high discriminant power (or more variation in dataset). This is achieved using Principal Component Analysis (PCA) which reduces feature space from higher to some lower dimension (In our, 3). PCA calculates eigen values and eigen vectors for higher dimension vector space and reduces it to n dimensions whose basis are eigen vectors corresponding to n largest eigen values. Here, PCA has been implemented using OpenCV's PCACompute function

5. We now apply K-means algorithm to cluster input images into K (here K = 4) distinct clusters. It is an iterative algorithm, described as below:
   i. Start by initializing K centroids for K clusters
   ii. Find the Euclidean distance between each image and each of K centroids. Assign image to a particular cluster for which Euclidean distance is minimum
   iii. Recalculate the centroids by considering images in respective clusters
   iv. Repeat till there is no change in the K centroids.

b) Texture Segmentation: It is a process of determining boundaries between different texture regions in an image based on texture properties. It involves first classifying the image into texture regions and then recognizing boundaries for each region. Here, we again make use of k-means clustering for classification and feature extraction is done using 5x5 Laws filters (as explained before) but at pixel level rather than for each image.

   Texture segmentation is implemented as follows:

1. Here, Preprocessing is done by calculating local mean for each pixel using a window approach. The mean value is then subtracted from respective pixel to reduce illumination effects.
2. Convolve input image with a set of 2D laws filters. The 5x5 laws filter are generated from a given set of five 1D filters (shown in table below). Each 1D filter is combined with another by taking a tensor product to obtain 25 5x5 filters
3. Compute average energy values for each pixel of each filtered image by applying windows of various sizes over each pixel. We use below formula:

$$E = \frac{\sum_{i=0}^{h} \sum_{j=0}^{w} | I(i,j) |}{h * w}$$

   Where w and h are window width and height respectively.

   This gives us 25D feature vector for each pixel.

4. Energy Normalization: From the obtained 2D law filters, since all the filters but L5TL5 have zero mean, we use the energy obtained by filter L5TL5 to normalize all other features at each pixel.

5. We now use k-means algorithm to segment the image into clusters.
   i. Start by initializing K centroids for K clusters
   ii. Find the Euclidean distance between each image and each of K centroids. Assign image to a particular cluster for which Euclidean distance is minimum
   iii. Recalculate the centroids by considering images in respective clusters
   iv. Repeat till there is no change in the K centroids.
6. After segmentation, denote segmented regions, by assigning gray value to each texture. As there are 7 types of textures in the input comb image, we use 7 gray levels $(0,42,84,126,168,210,255)$

III. Experimental Results

a) Texture Classification

Reduced 3-D Feature Plot:

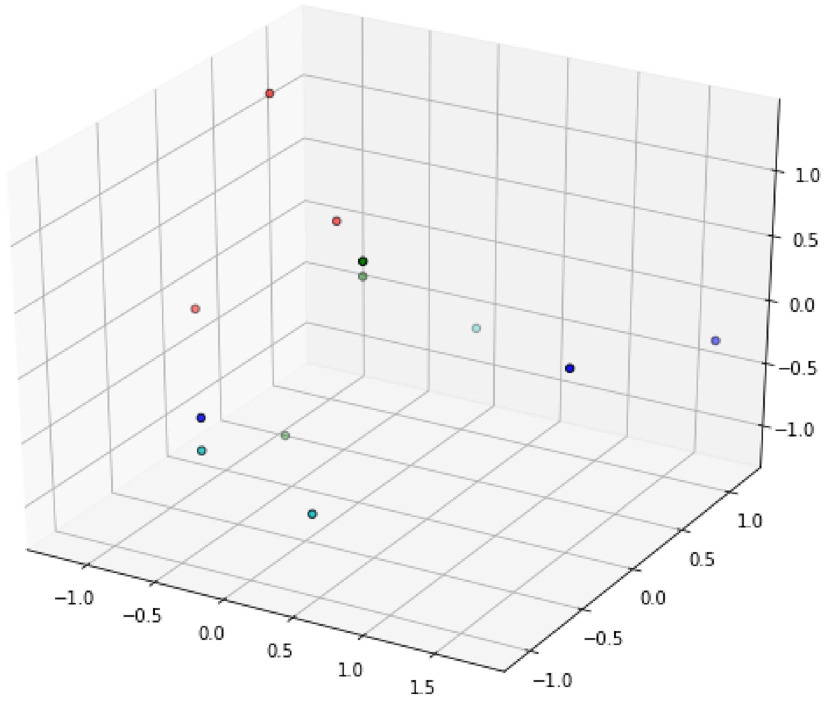


Figure 1 PCA plot

Classification Result:

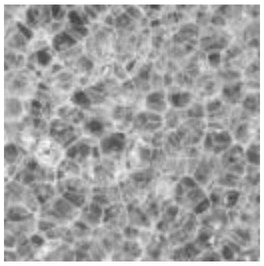
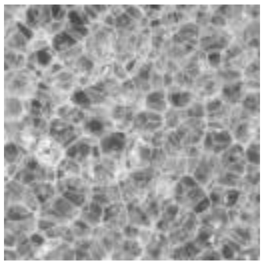
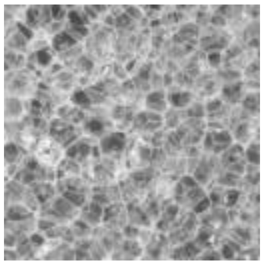
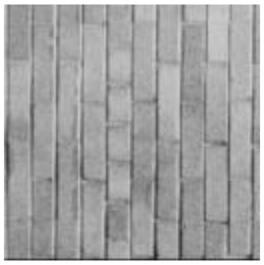
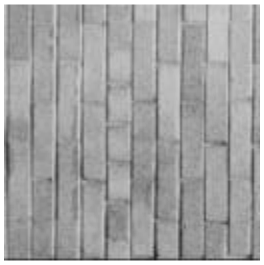
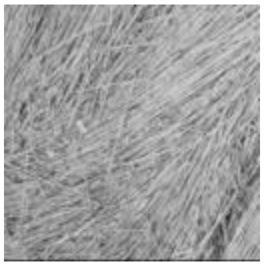
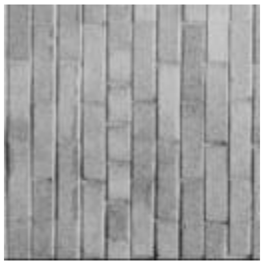
Class 0	  
Class 1	   
Class 2	
Class 3	

Table 2 Texture Classification output

Max Variance – 3942309.36 (Filter L5'L5)

Min Variance – 2128.1 (Filter S5'S5)

b) Texture Segmentation

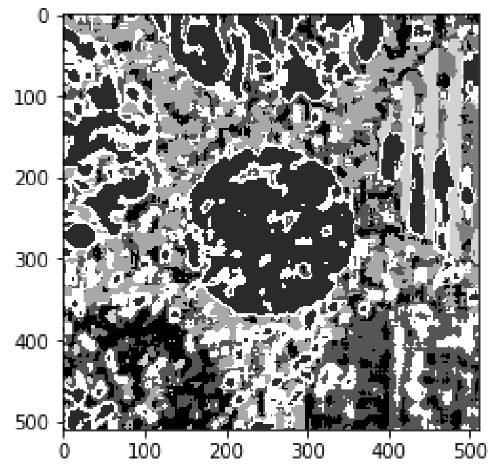


Figure 2 Segmentation with Window Size = 13

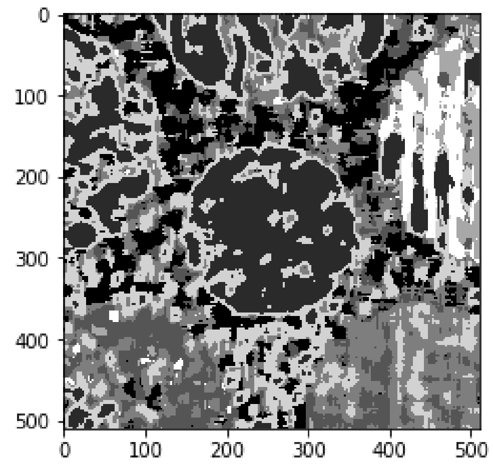


Figure 3 Segmentation with window size = 15

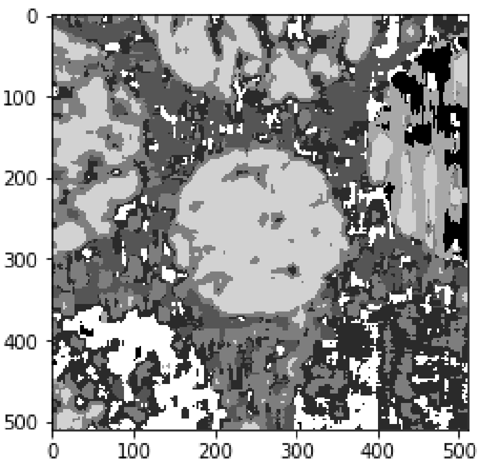


Figure 4 Segmentation with window size = 17

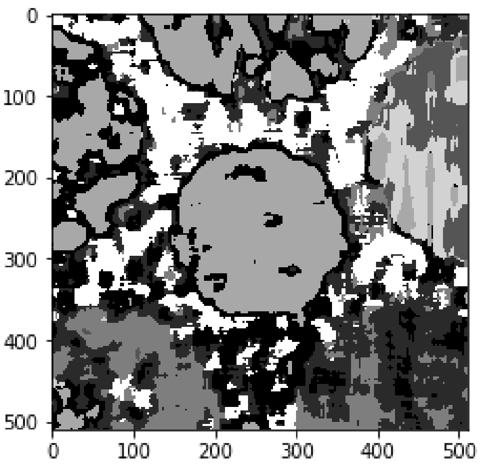
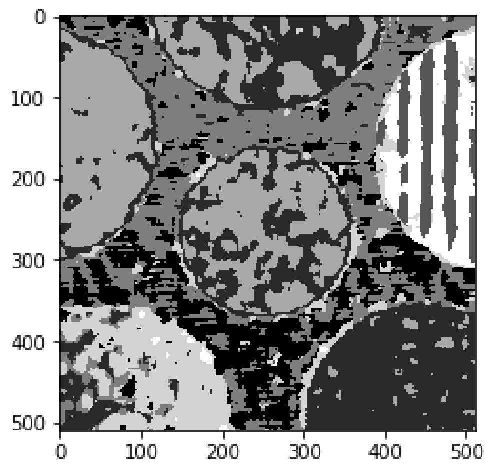


Figure 5 Segmentation with window size = 21

c) Further Improvements (Segmentation using PCA with std normalization and hole merging)



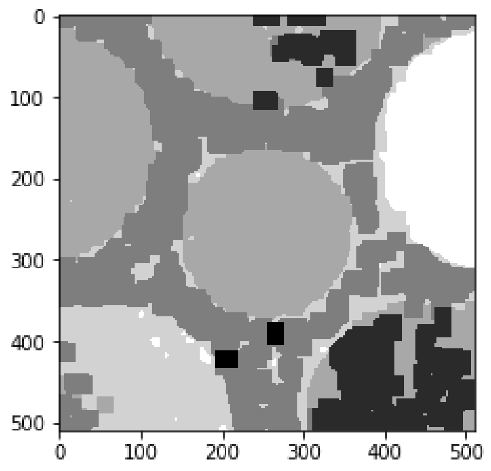


Figure 6 Segmentation for window size = 13

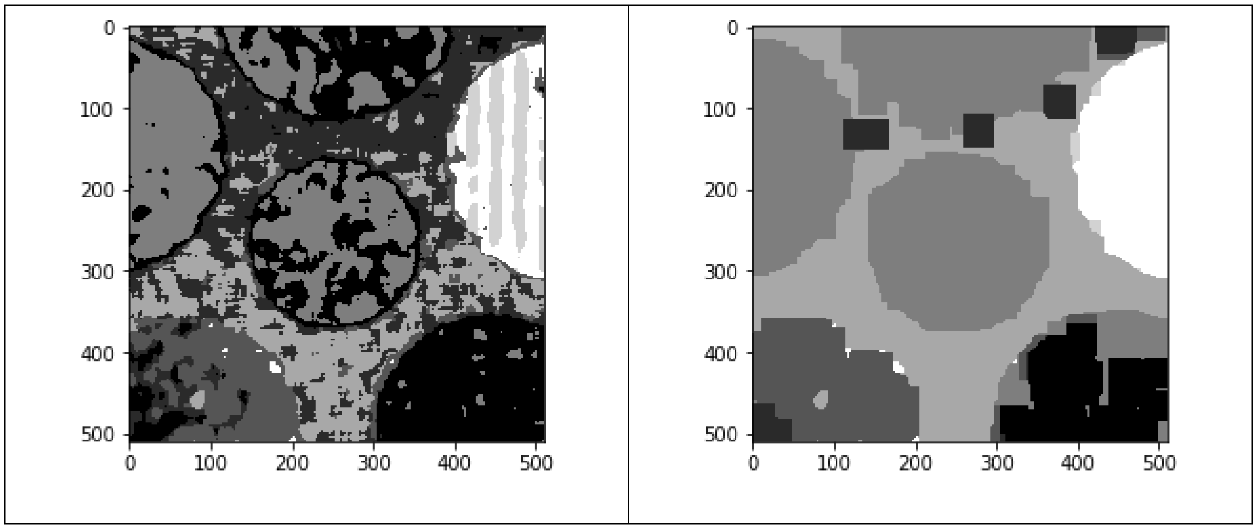


Figure 7 Segmentation using PCA with window size = 15

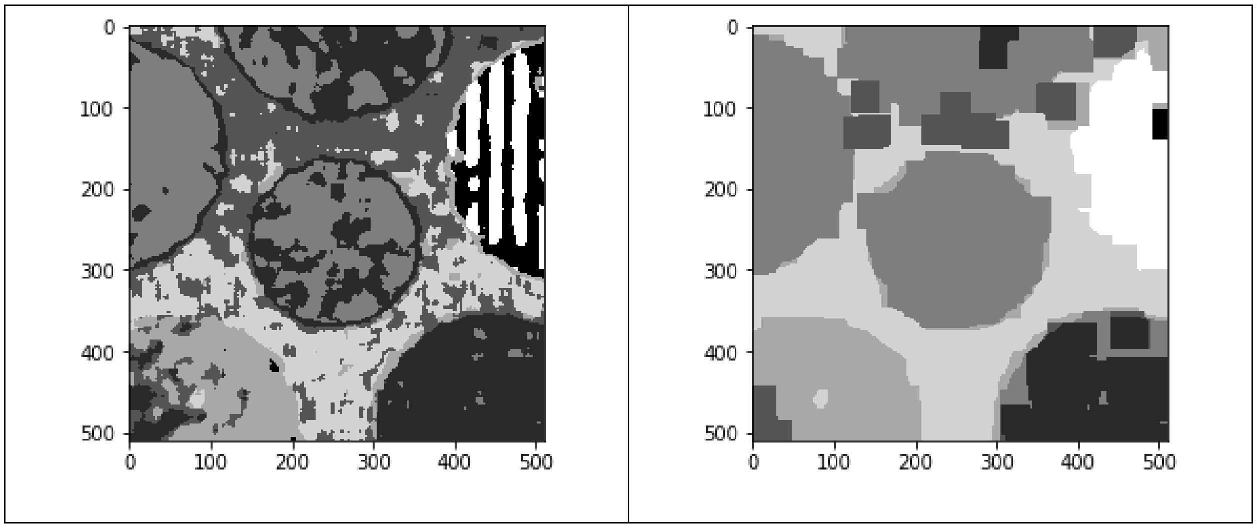


Figure 8 Segmentation using PCA with window size = 17

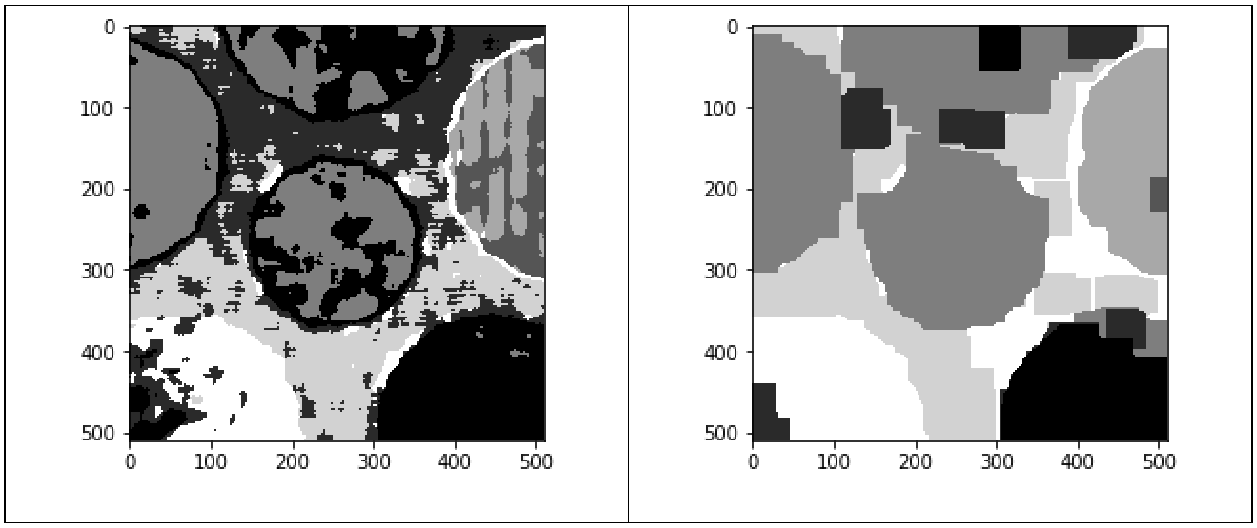


Figure 9 Segmentation using PCA with window size = 21

IV. Discussion

a) Texture Classification

- Boundary extension was done using pixel replication while applying Law's Filters. A padding of 2 was applied on all sides as the filters were of size 5x5 covering extra 2 pixels on all centering.
- Discriminant power of the features can be studied multiple ways
    - Here, Variance of feature vectors was calculated. Feature vectors that have more variance will result in less discriminant power. $L5^T L5$ filter has lowest discriminant power since it gave highest variance and similarly $S5^T S5$ filter has highest discriminant power and will contribute more for better clustering.
    - Second method is to find number of overlapping class data points for each feature after applying filter on the images. Features having large number of overlapping points won't be able to classify data well and hence, will have less discriminant power.
- The output for texture classification using PCA is shown table 2. We see that there is only one misclassification wherein class 3 (Grass) has been misclassified as class 1 (Brick).
- Classification was also tried using 15 filters where few filters were averaged, however the classification was not so well. Hence entire 25 filters were considered.
- Reduced 3D feature plot is shown in figure 1, where each point is one of 12 textures represented in lower dimension. Same color points indicate data belonging to same texture class. This helped in getting only 3 best features for classification of textures. This helped in faster convergence of K-means algorithm.
- Texture classification was also done without PCA; however, no difference was found in the output.
- In both cases (with and without PCA), there was only one same misclassification. I suspect this may be due to small number of training samples. With more training data, k-means will give better accuracy.

- Also, Kmeans algorithm was initialized using k++ as well as random initialization. However, no difference was found in the output. This again may be due to less data sample. This shows that k-means clustering is well dependent on data size.

b) Texture Segmentation
- Output for texture segmentation is shown above section. Similar to texture classification, laws filtering was performed for feature extraction and also energy feature normalization was done with respect to L5'L5 filter.
- Segmentation was performed for different window sizes (figures 2 - 5). As seen from the output, larger the window size, better is the segmentation. This is because higher window will result in large number of relevant samples being included in the feature vector. However, Segmentation does not change much once an appropriate window size is reached.

c) Further Improvement
- For further improving segmentation, PCA with standard normalization was performed and feature L5'L5 was not considered as seen from previous section, it does not contribute to good clustering.
- As seen from the output (figures 6 - 10), for window size 13, PCA output gives more clear segmentation compared to non-PCA output. As window size is increased further, output gets better.
- Another advantage here is, due to reduced dimension, K-means converges faster.
- Also, hole filling was been implemented post PCA based clustering. This resulted in cleaner output

- All the implementation was done in python. Appropriate inbuilt libraries were used wherever needed

Problem 2: Image Feature Extractor

I. Abstract and Motivation

- Feature extraction is core step of image processing. Features are the brain of an image. They provide with meaningful information which helps in describing the image in various applications such as object recognition, gesture recognition, etc. For accurate image description, these features need to be rotation as well as scale invariant. One way to achieve this is using a feature extraction technique called Scale Invariant Feature Transform (SIFT).
- In this problem, we implement SIFT and utilize extracted features to perform image matching.
- We also implement, something called Bag of words model. Bag of words model is used to solve the problems in object Detection, Image Retrieval, etc. arising due to factors such as different camera positions, illumination differences, object variation, etc. Here, we combine features extracted through SIFT and bag of words to classify an unknown image.

II. Approach

a) SIFT: It is one of the many feature extraction techniques which is robust to geometrical modifications. SIFT was proposed by David Lowe in 2004 paper called 'Object recognition from Local Scale-Invariant Features'. It makes use of key points and descriptors for extracting features using a staged filtering approach.

   The overview of SIFT algorithm is as follows:

   1. Scale-Space Keypoint Localization: Here, we identify key locations in scale space by obtaining minima or maxima of difference of gaussian function. The scale space consists of n octaves with ~4 images in each octave. This is computed by building an image pyramid by resampling input image at different levels (shown in figure below). This is done to achieve scale invariant. The input image is first convolved with the Gaussian function using $\sigma = \sqrt{2}$ to give an image A. This is then repeated a second time with a to provide further smoothing of $\sigma = \sqrt{2}$ giving new image, B. The difference of Gaussian function is obtained by subtracting image B from A. To generate next pyramid level, image B is resampled using bilinear interpolation. Once, we have pyramid, the images are then searched for local maxima over scale and space. Maxima is determined by comparing each pixel in

pyramid to its 8 neighbors as well as pixels in next scale and previous scale. If it is a local maximum, it is a potential keypoint.

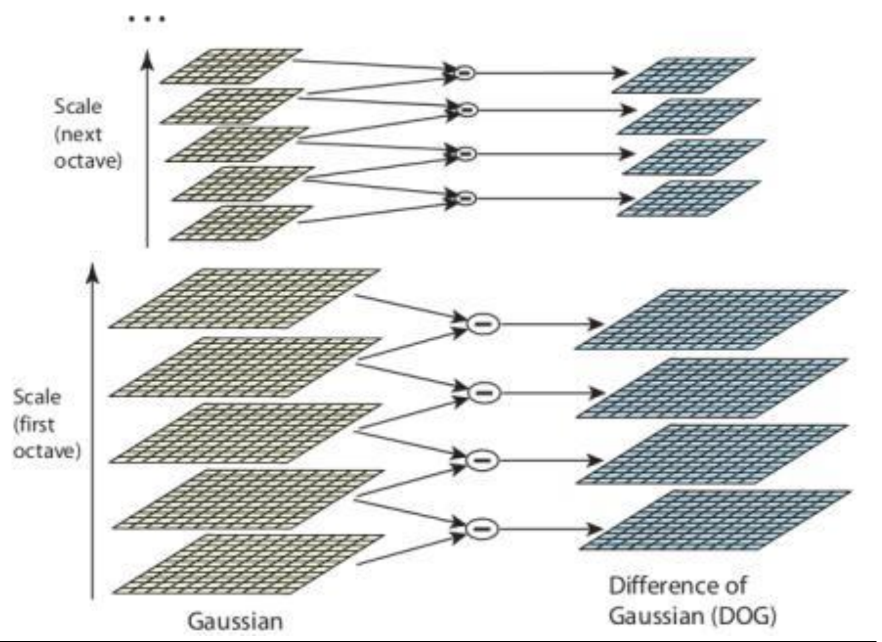


Table 3 Gaussian Image Pyramid (source: opencv sift tutorial)

2. Keypoint Fine Tuning: Once we have keypoints, we need to refine them as few of the keypoints may be weak/noisy due to either low contrast or they are part of an edge. Taylor series expansion of scale space is used along with thresholding to get more accurate key location.
   To avoid low contrast keypoints, we compare the intensity at extrema location with '**contrastThreshold**'. If it's less than that, the keypoint is then rejected.
   To avoid edge keypoints, we compute hessian matrix and compare the ratio of its eigen values with '**edgeThreshold**'. If the ratio is greater than threshold, that keypoint is discarded.
3. Orientation Assignment: Now, each keypoint is assigned a canonical orientation to achieve rotation invariance. We consider a small region around keypoint location and calculate gradient magnitude and direction in that region. Then, orientation histogram is created using gaussian-weighted circular window with σ equal to 1.5 times the scale of keypoint. It is weighted by gradient magnitude. Histogram has 36 bins covering 360-degree rotation range. To get final orientation, from the histogram, choose the highest peak along

with any peak which is at top 20%. This results in keypoints with same location and scale, but with different directions.

4. Keypoint Descriptor: We now consider a 16x16 neighborhood around each keypoints, which is further divided into 16 4x4 sub blocks (shown in figure below). For each sub-block, 8 bin orientation histograms are created resulting in 128 bins for every 16x16 neighborhood. This represents 128 SIFT key vector for each keypoint forming a keypoint descriptor.

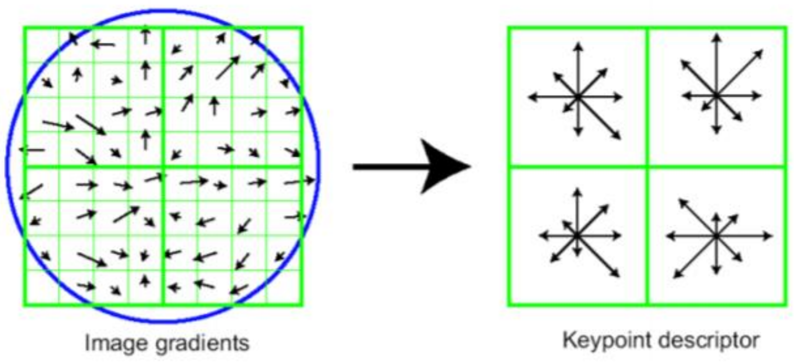


Figure 10 Keypoint neighboring (source: class notes)

b) Image Matching: The SIFT features obtained above can be further used for finding relevant matches among images. In addition to above steps, we perform one final step as below:

   1. Keypoint Matching: After computing SIFT features for 2 images, we identify nearest neighbors to find matching points. However, there may be cases, where first-closest match is very near to the second. In that case, we take ratio of first-closest distance to second-closest distance and apply thresholding. If the ratio is greater than 0.8, we reject the match. This significantly helps in reducing lot of false matches. Matching can be done either using brute force match or FLANN based matcher, both of which uses nearest neighbor to find the best match.

c) Bag of Words: It is an algorithm, originally developed for text data in which frequency of occurrence of each word is used to know the keywords of the document and get a frequency histogram from it. Same concept is extended to image, wherein we represent image as a frequency histogram of image features. In this, the frequency of occurrence of training image features are stored in the form of visual vocabulary, which are used for comparing with frequency of testing images to determine which classes they belong to.

Bag of words classification is implemented as follows:

1. For each training images, extract SIFT features and build visual dictionary
2. Cluster the features into 2 groups using k-means algorithm to build visual vocabulary for train images of each class
3. For each class of train images, plot frequency histogram of 2 bins from vocabularies and frequency of vocabularies in the image, giving us bag of visual words. Each feature in 2 bins is a vector of size 128 indicating best possible centroids.
4. Repeat above process for test image
5. Compare the histogram of test image with training images by finding intersection ratio. The histograms which has high ratio, test image would be classified to that one.

III. Experimental Results

a) Image Matching:

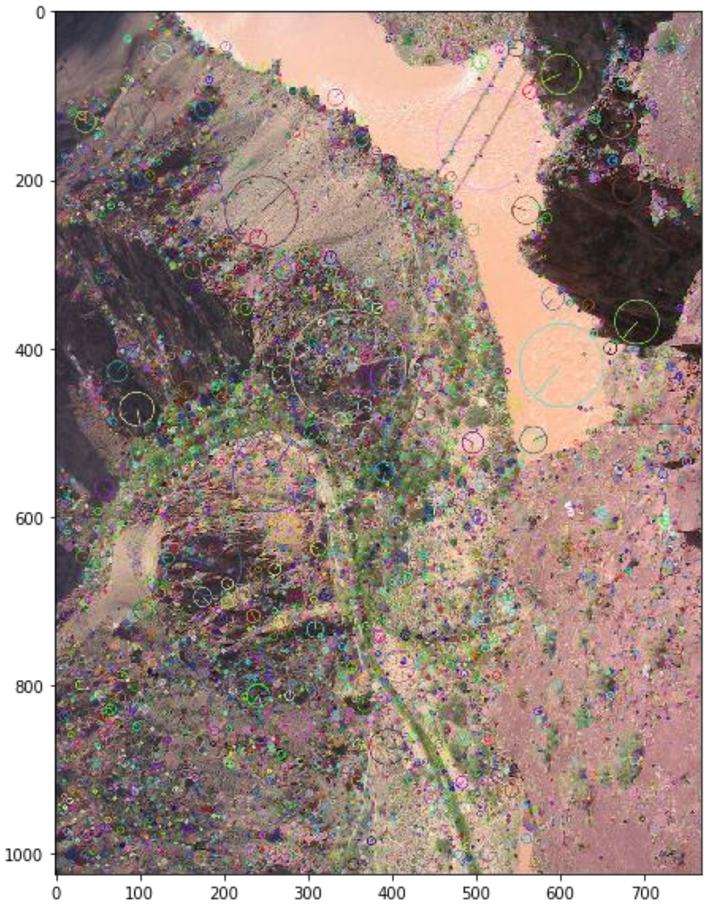


Figure 11 SIFT features for river 1 image

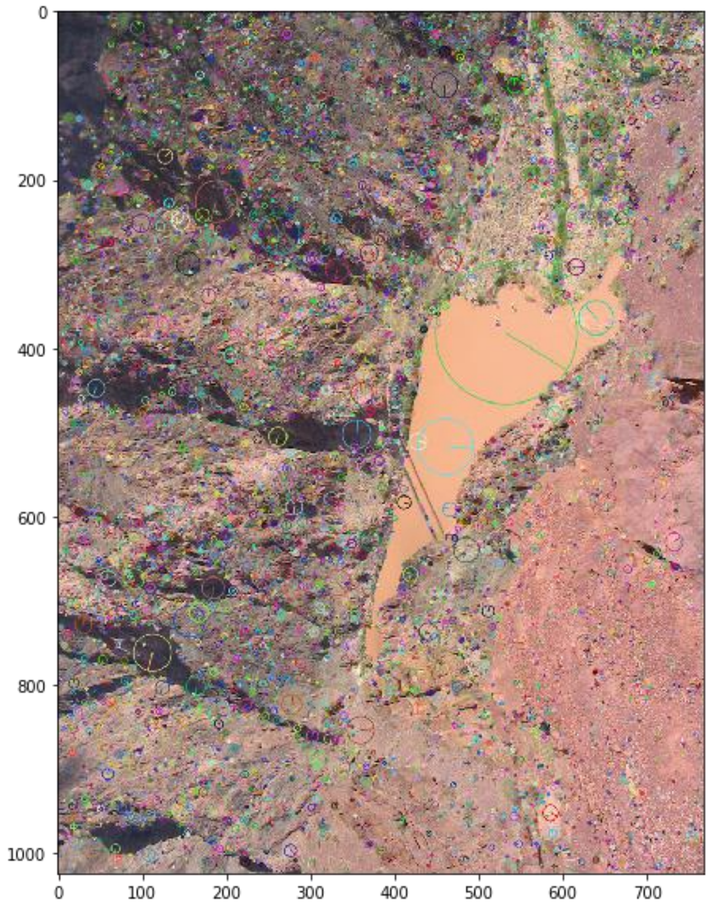


Figure 12 SIFT features for river image 2

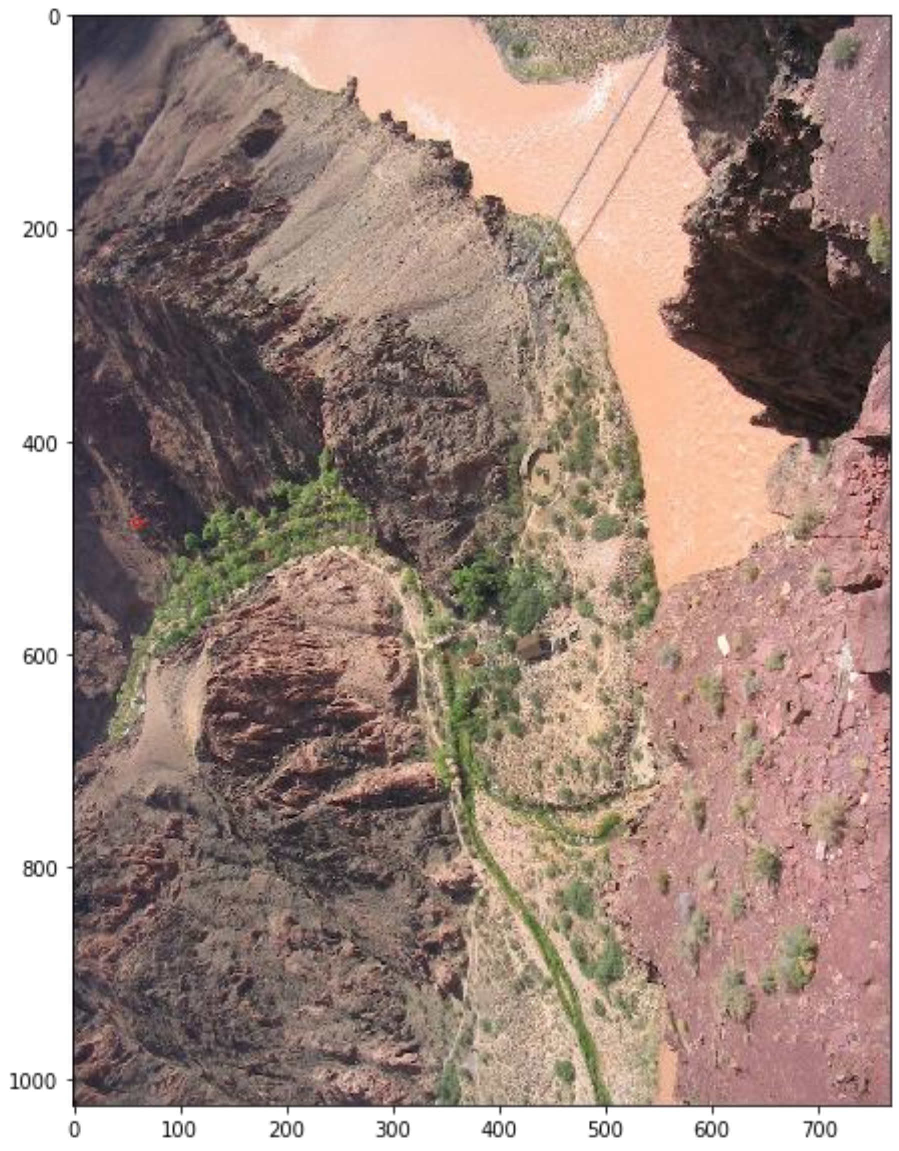


Figure 13 Keypoint with largest scale in river 1 Image

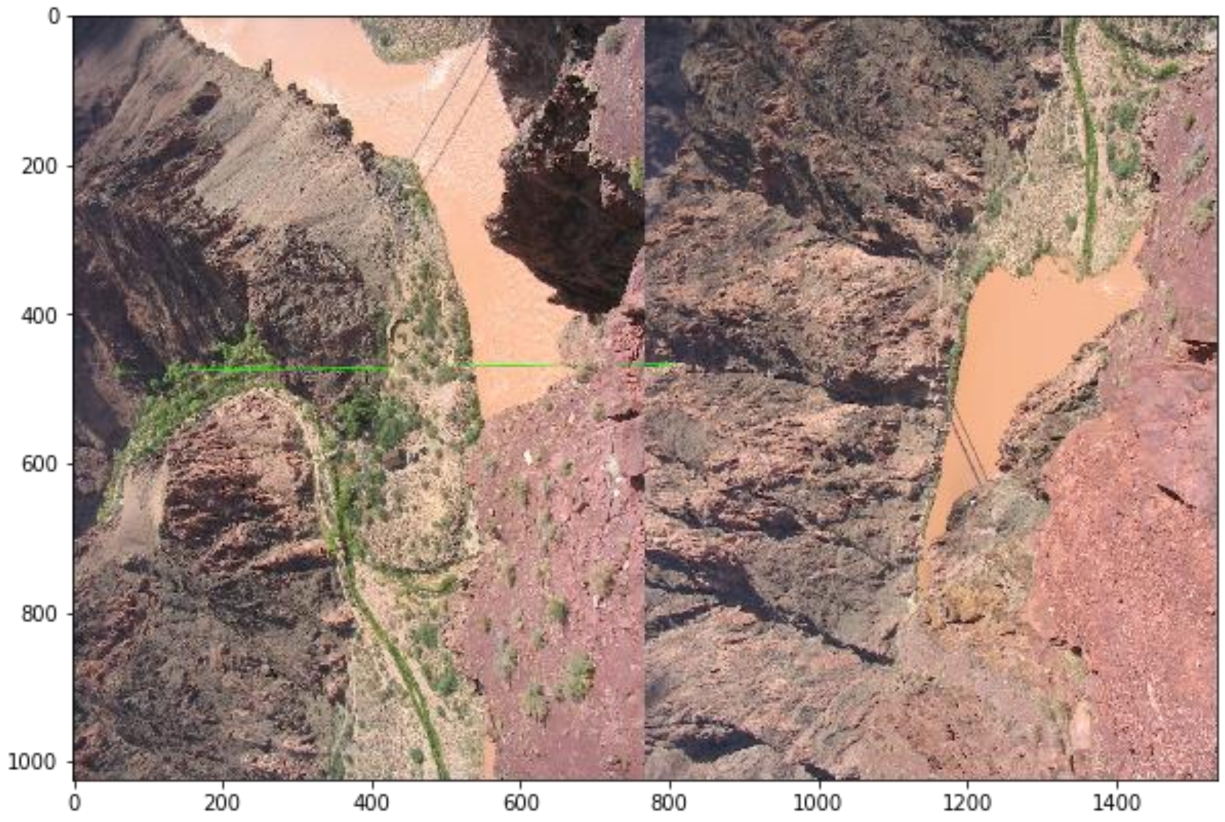


Figure 14 Matched Key Point with river 2 image

Orientation of key point in river image1: 206.179
Orientation of matching key point in river image2: 129.88

b) Bag of Words:

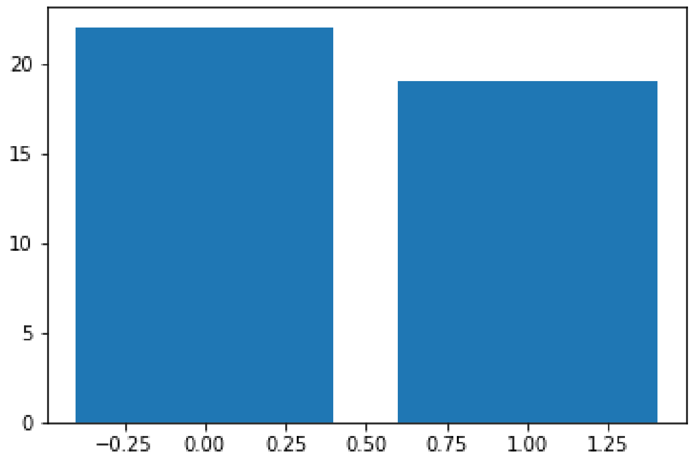


Figure 15 Bag of Words Histogram for zero's image

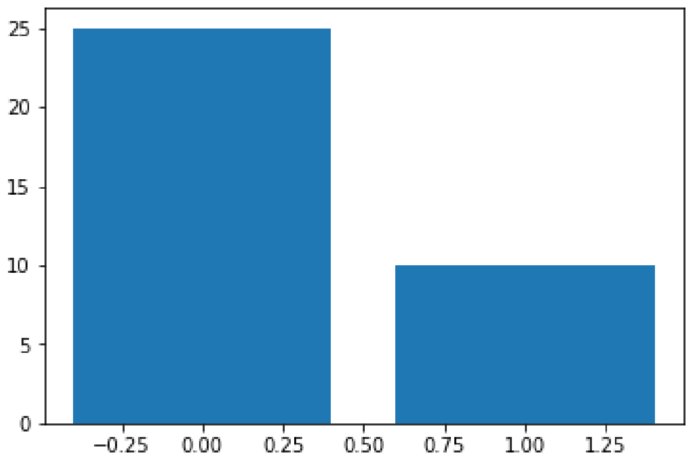


Figure 16 Bag of words Histogram for one's Image

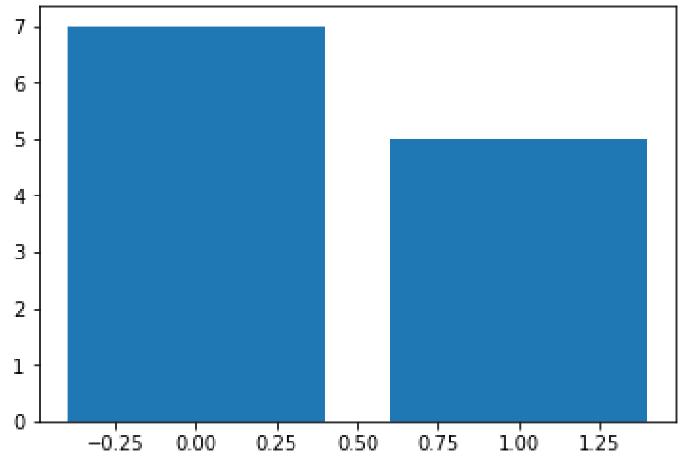


Figure 17 Bag of Words Histogram for image of eight

Intersection with zero's histogram: 0.58
Intersection with one's histogram: 1.2

IV. Discussion

a) SIFT:

- SIFT is robust to translation, scaling, rotation and also partially invariant to illumination changes and affine or 3D projection.
- The scale invariant features are identified using staged filtering approach where we identify keypoints in scale space by taking maxima or minima of difference of gaussian function.

- Rotation and translation invariant are achieved by assigning a canonical orientation to each keypoint, determined by peak in histogram of local image gradient orientation
- To reduce Illumination change, SIFT descriptor is first normalized to unit length and further enhanced by thresholding the gradient magnitudes at a value of 0.1 times the maximum possible gradient value
- Difference of Gaussian is used as it is easier to compute and is computationally faster that Laplacian of Gaussian. Additionally, since Laplacian of gaussian filters are sensitive to noise resulting in noisy image, difference of gaussian helps in smoothening the image.
- As per paper, 8 orientation planes were used, each sampled over a 4 x 4 region. For sampling the image at a larger scale, the same process is repeated for a second level of the pyramid one octave higher. However, this time a 2 x 2 rather than a 4 x 4 sample region is used. Therefore, total size of output SIFT key vector is 8x4x4 + 8x2x2 = 160 elements for each keypoint

b) Image Matching:
- The keypoints for river images is shown in fig 1 and 2.
- The keypoint with largest scale was obtained by finding largest l2 norm of descriptor. It is highlighted in fig 3
- Also, keypoint orientation was calculated for matching keypoints in both images. We see that, even though orientations are different, keypoints were successfully matched. This is because features extracted through are rotation invariant.
- Image matching was done using Brute Force Matcher, output of which is shown in fig 4
  - In this method, descriptor of each feature in first image is matched with descriptors of all the features in the second image using knn match to get k best matches. However, this results in large number of matching points. To reduce the number of matching points, ratio test can be used (as per Lowe’s paper [3]), where we calculate distance values of matching points and kept only those values where the ratio is less than some set threshold. This results in optimum matching points.

        - In our case, the descriptor of keypoint with largest l2 norm was matched with all the features in second image to get closest match (using k = 2 and then applying ratio test).
    - Another method is to use FLANN based matcher. It also uses nearest neighbor approach; however, it is ideally used when the dataset is large and for searching high dimensional features. It is computationally faster than Brute Force match.

  c) Bag of Words:
    - The bag of words histogram for images of zeros, ones and eight are plotted as shown in figures 14 -16. Histogram intersection method was used for image classification. Intersection of histogram of eight with zeros histogram as well as with one's histogram was calculated. As, per the result, intersection value is more with one's histogram. So, we conclude that image of eight classifies as one
    - Also, from visual inspection of histogram, we see that histogram of eight is almost similar to one with few differences in frequency values of each bin. This may be due to slight difference in the orientation between 2 images.
    - Few disadvantages to this method:
        - Visual words are not discriminant, meaning features cannot be separated.
        - There is no spatial relationship between visual words.

- All the implementation was done in python. Appropriate inbuilt libraries were used wherever needed

References

[1] Class reading materials
[2] https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect12.pdf
[3] Distinctive Image Features from Scale-Invariant Keypoints - https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
[4] Object Recognition from Local Scale-Invariant Features - https://www.cs.ubc.ca/~lowe/papers/iccv99.pdf
[5] https://docs.opencv.org/3.4/da/df5/tutorial_py_sift_intro.html
[6] https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[7] https://towardsdatascience.com/bag-of-visual-words-in-a-nutshell-9ceea97ce0fb