MACHINE PERCEPTION

Assignment 2

By: Shachi Bhavsar (MT2016036)
Meet Patel (MT2016102)

Que: 2A) Segmentation

Pick 5 images from Berkeley Segmentation dataset

Run Kmeans and Meanshift

Show results along with the ground-truth

K-Mean: K-Means algorithm requires that the number of clusters to be pre-determined. In practise, it is often difficult to find the right cluster number, so that we often just pick a sufficiently large cluster number. So sometimes the output may end up having too few clusters or too many clusters to be useful.

cv2.kmeans() function is used for data clustering in OpenCV.

It first read image using **cv2.imread()** and then convert it into grayscale image using cv2.cvtColor().

In k-mean, each feature should be put in a single column and result will be a 1-D array. This can be done using image.reshape(-1 , 3). Where, we use -1 on an axis to automatically infer shape. It should be of **np.float32** data type.

After that define iteration termination **criteria** as:

(cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, max_itr, epsilon)

When any of the below condition is satisfied, algorithm iteration stops.

cv2.TERM_CRITERIA_EPS - stop the algorithm iteration if specified accuracy, *epsilon*, is reached.

cv2.TERM_CRITERIA_MAX_ITER - stop the algorithm after the specified number of iterations.

Where, **max_itr** represents max no. of iterations and **epsilon** represents required accuracy.

Now convert back into uint8. While separating data we used **label.flatten().** It is because 'label' returned by the OpenCV is a column vector. But we needed a plain array.

We need to reshape it back to the shape of original image by img.reshape(). After that plot original and segmented image using subplot() of matplotlib**.**

MeanShift: Mean shift is a non-parametric **feature-space** analysis technique.

Given a set of datapoints, the algorithm iteratively assign each datapoint towards the closest cluster centroid. The direction to the closest cluster centroid is determined by where most of the points nearby are at. So each iteration each data point will move closer to where the most points are at, which is or will lead to the cluster center. When the algorithm stops, each point is assigned to a cluster.

Unlike K-means, meanshift does not require specifying the number of clusters in advance. The number of clusters is determined by the algorithm with respect to the data.

It first read image using **cv2.imread()** and then convert it into grayscale image using cv2.cvtColor().

Define Termination Criteria for Meanshift. Here criteria is same as K-mean only difference is in maximum number of iterations.

Apply MeanShift to color_image with sp=10, cp=10, macLevel=7 and predefined criteria.

Syntax: cv2.pyrMeanShiftFiltering(src, sp, sr[, dst[, maxLevel[, termcrit]]]) → dst

where,

src – The source 8-bit, 3-channel image.

dst – The destination image of the same format and the same size as the source.

sp – The spatial window radius.

sr – The color window radius.

maxLevel – Maximum level of the pyramid for the segmentation.

termcrit – Termination criteria: when to stop meanshift iterations.

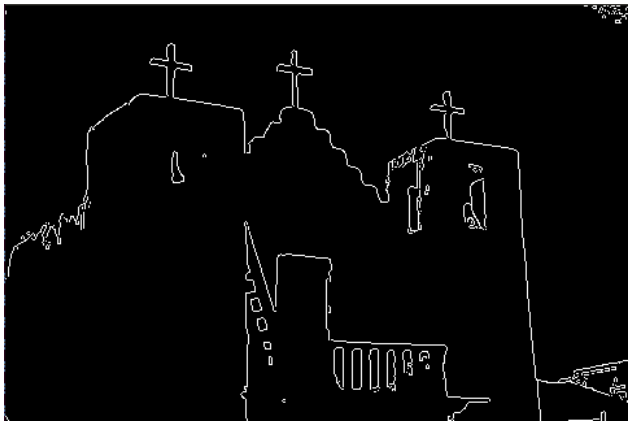
Now, plot original and segmented image using subplot() of matplotlib.

Output : Image-1:

Original Image

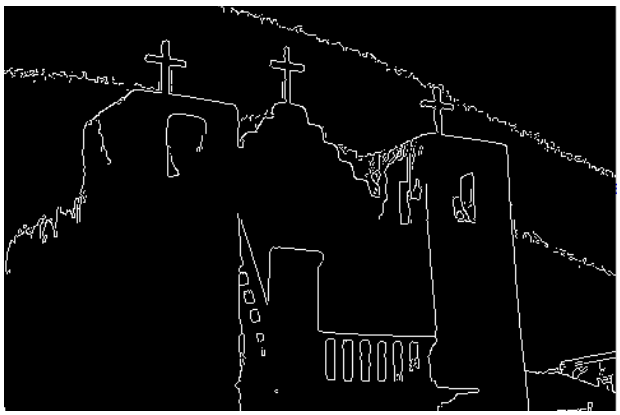
Ground Truth

K-means with K=3



Edges from Segmented Image

K-means with K=6

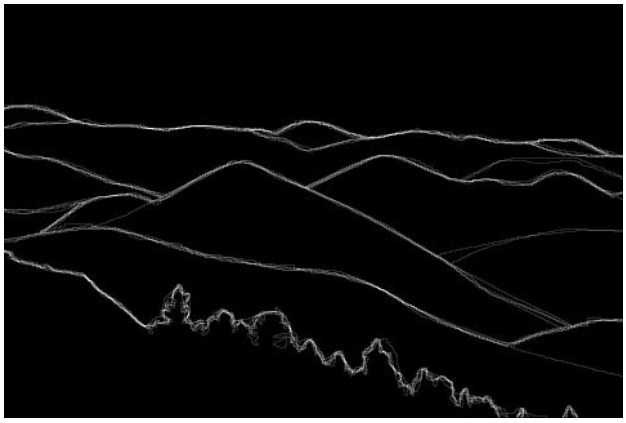


Edges from Segmented Image

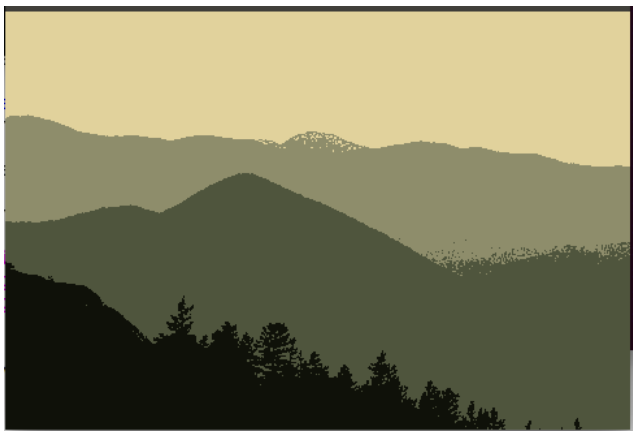
Segmentation using MeanShift

Image-2 :

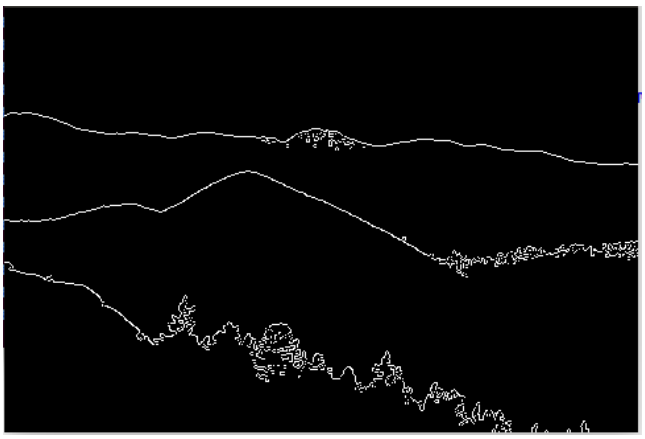
Original Image



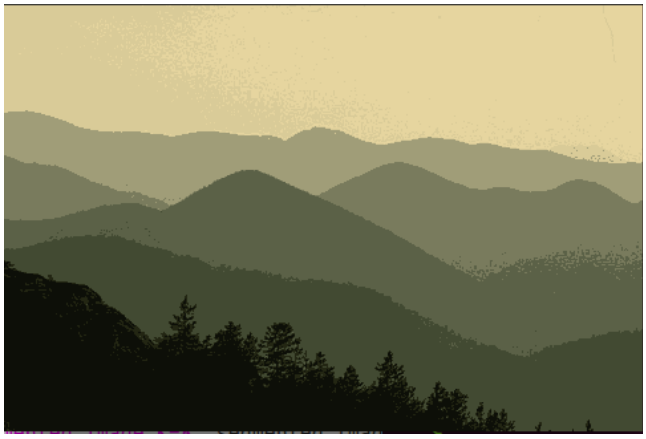
Ground Truth



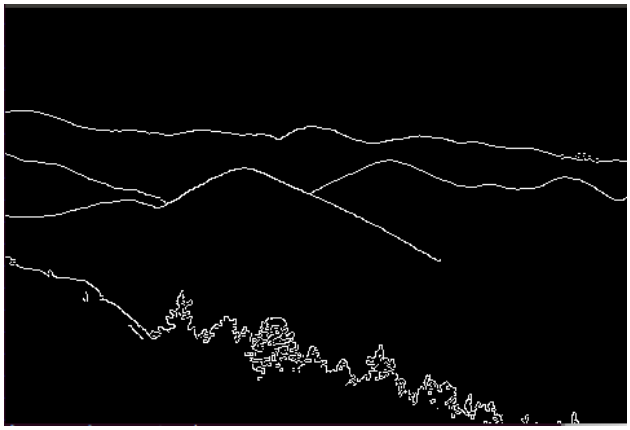
K-means with K=4



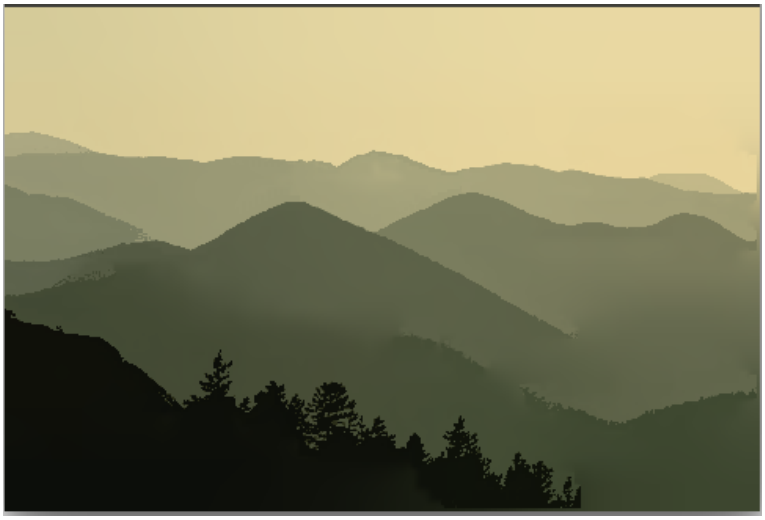
Edges from Segmented Image



K-means with K=8

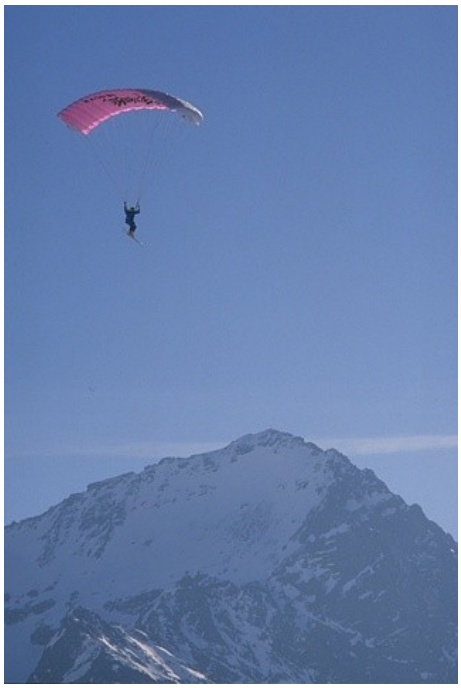


Edges from Segmented Image

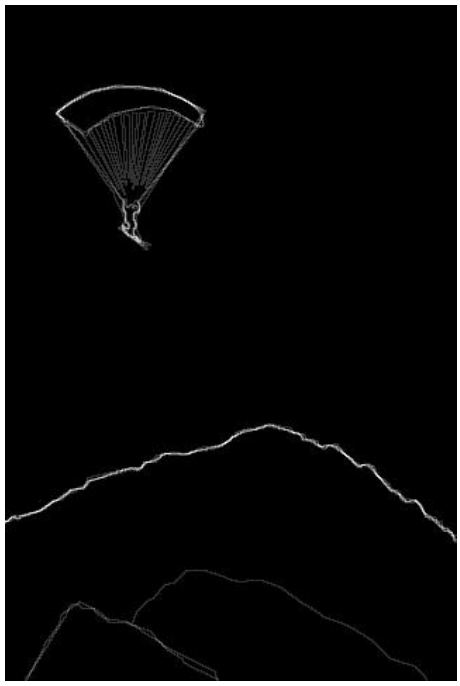


Segmentation using MeanShift

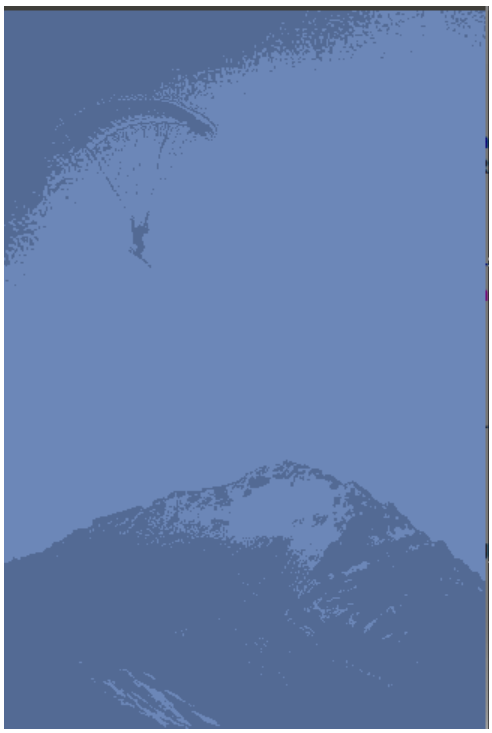
Image-3 :



Original Image

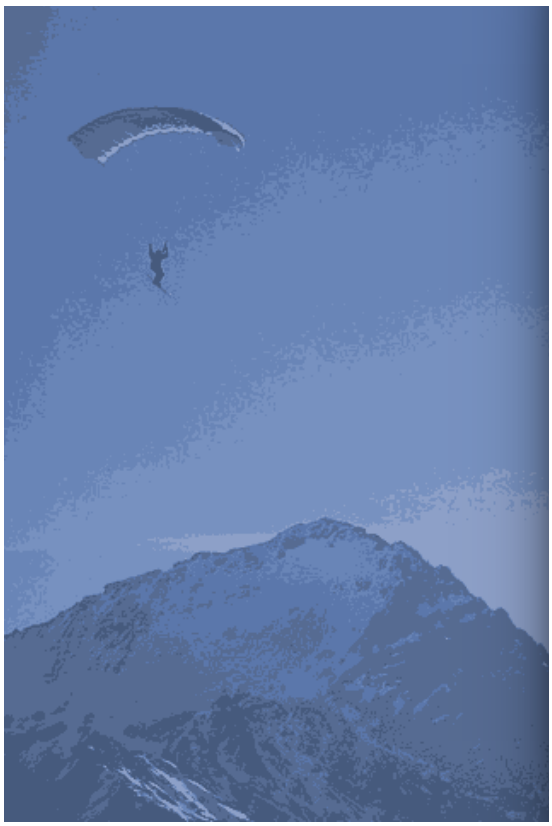


Ground Truth

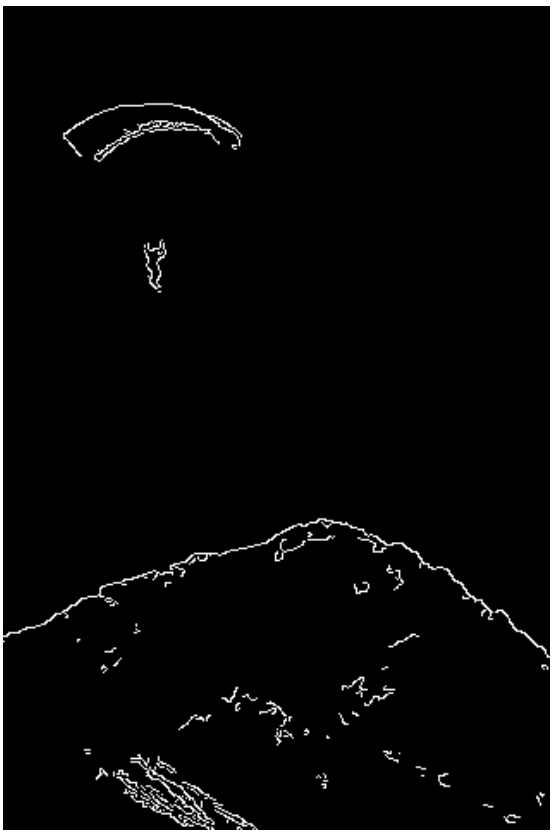


K-means with K=2

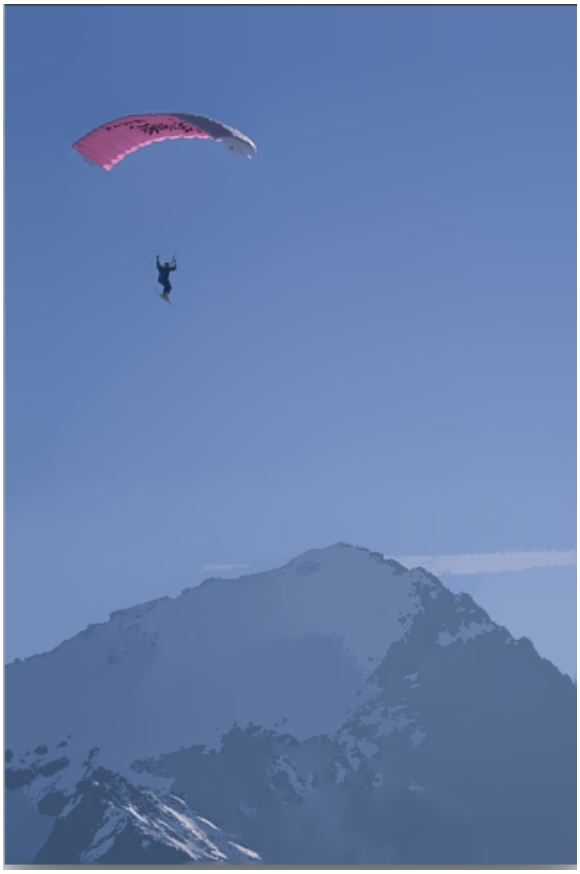
Edges from Segmented Image



K-means with K=6



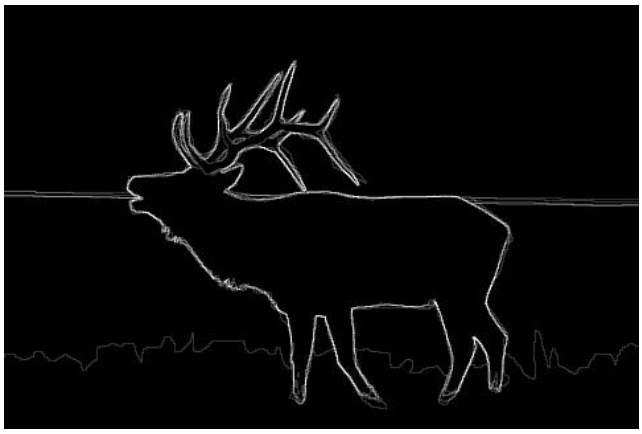
Edges from Segmented Image



Segmentation using MeanShift

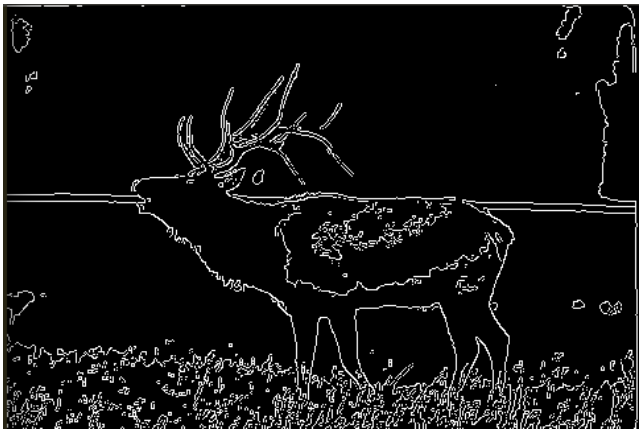
Image-4 :

Original Image



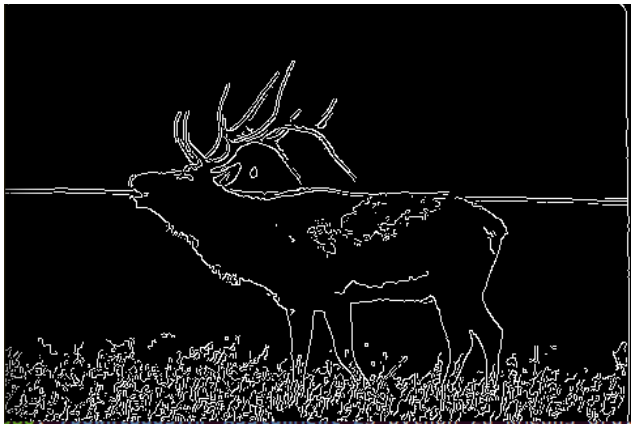
Ground Truth

K-means with K=4



Edges from Segmented Image

K-means with K=8

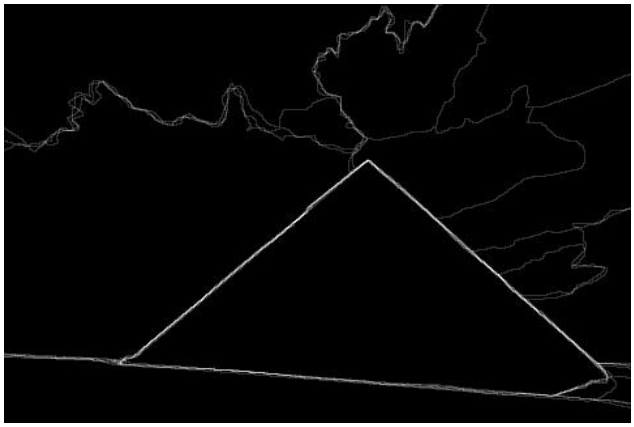


Edges from Segmented Image

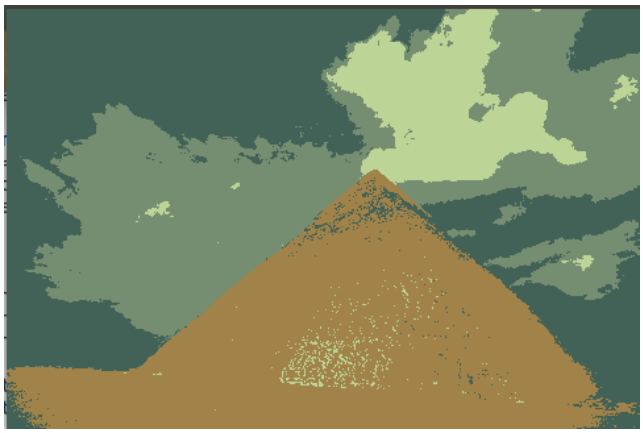
Segmentation using MeanShift

Image-5 :

Original Image

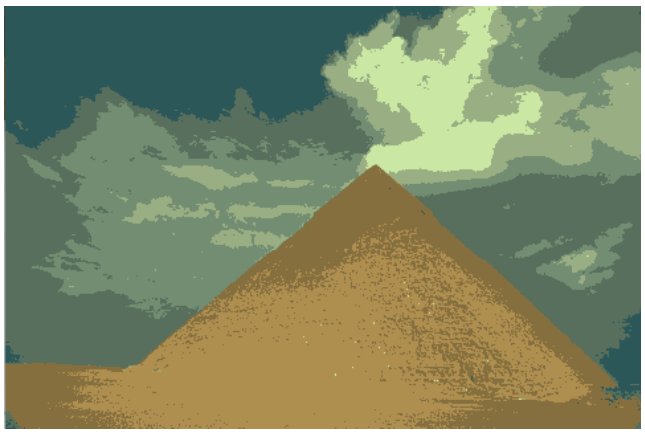


Ground Truth

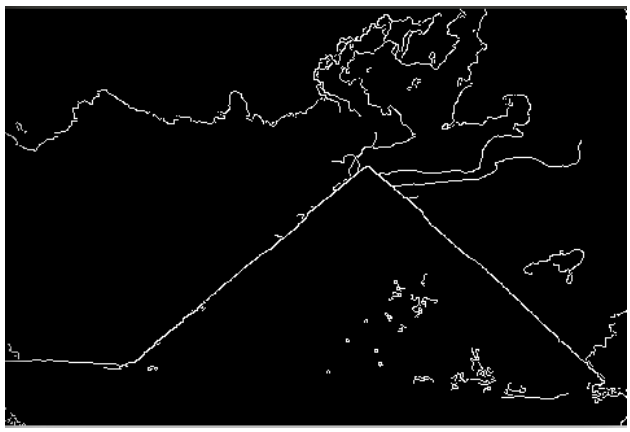


K-means with K=4

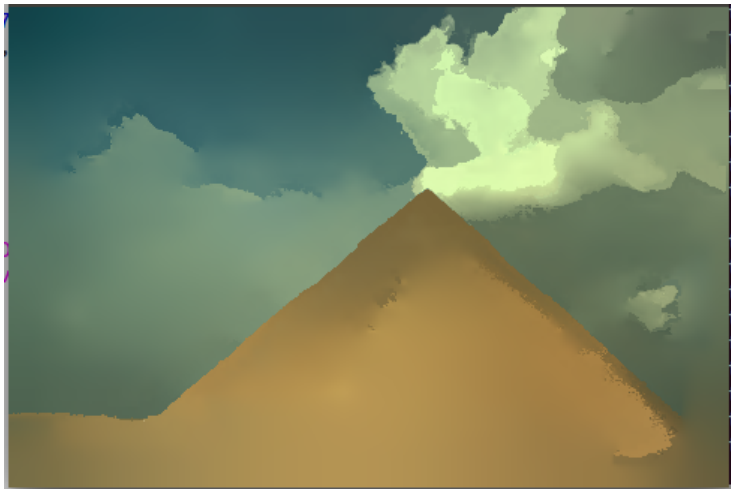
Edges from Segmented Image



K-means with K=7



Edges from Segmented Image



Segmentation using MeanShift

Here In output, for each image we have shown original image, its ground truth taken from berkeley dataset, then output of K-means clustering for two different values of k and its corresponding edges of segmentated image and output from MeanShift segmentation.

Que 2-B) Explain how SURF is different from SIFT

- SIFT uses a descriptor of length 128. SURF has several descriptor types of varying length like 64 or 128 in length.

- SIFT extracts more key features from an image than SURF. In an 70 x 70 average photo, SIFT extracts nearly 40-50 features, however SURF extracts around 8-10.

- SURF is faster than SIFT. Benefits will change based on application. For example in face recognition, some of features may be missed in SURF so SIFT is more beneficial and accurate because it extracts more features.

- SURF can use the same matching scheme as SIFT, but has one additional improvement. It allows a quick distinction between bright features on a dark background and dark features on a bright background.

- SIFT is mathematically complicated and computationally heavy. SIFT is based on the Histogram of Gradients. The gradients of each Pixel in the patch need to be computed. So the computational cost for SIFT is much greater than SURF. Because of that SIFT is not the best choice for real-time applications while SURF can be used for real-time applications.

- Shift consists of four essential stages: scale - space extrema detection, key points localization, orientation assignment,and generating keypoint descriptor.

  In the first stage, the key points are extracted based on their strength that are invariant to orientation and scale using difference in gaussian. In the second stage, the wrong points are removed. Then in next stage, one or more orientations are assigned to each keypoint. In the final stage, a vector descriptor is made for each keypoint.

  SURF speeds up its computations by fast approximation of Hessian matrix and descriptor using “integral images”. Haar wavelets are used during description stage.

- For Keypoint Detection, In scale space, SIFT uses different-scale images convoluted with a Gaussian function. While SURF uses different-scale box filter convoluting with an original image.

- For keypoint selection, SIFT Detect extrema in DoG space; do non-maxima suppression while SURF uses a Hessian matrix to determine candidate keypoints; do non-maxima suppression.

- SIFT calculates a gradient amplitude of a square area; regard the direction with the maximum gradient strength as the main direction. While SURF calculate a Haar wavelet response in x and y directions of each sector in a circular area; regard the direction with maximum norm as the main direction.

- For feature extraction, SIFT divides a 16×16 region into 4×4 sub-regions; create a gradient histogram for each sub-region and SURF divides a 20×20s region into 4×4 sub-regions; calculate a Haar wavelet response.

- SIFT is robust to the change in intensity, scaling and rotation. It is based on the difference of Gaussians. While SURF is not that much good in change of intensity and scaling and rotation.

- The main advantages of the SIFT and SURF descriptors are that they are quite robust to noise and error detection. Due to the zero mean of the white noise, the histogram in SIFT and the summation in SURF, can suppress noise to a low level. The statistics of the gradient over an angle of 45 degree makes SIFT robust to localization error, while the summation of gradient over the 4 × 4 block makes SURF robust to localization error.

Explain main principles of FLANN matching:

FLANN stands for Fast Library for Approximate Nearest Neighbors. It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. It works more faster than BFMatcher for large datasets.

When we do matching, we take one feature from set S1 and look for nearest neighbor in the set S2. FLANN is just fast way to find (approximate) nearest neighbor.

It can provide automatic selection of index tree and parameter based on the user's optimization preference on a particular dataset. They are chosen according to the user's preferences on the importance between build time, search time and memory footprint of the index trees.

FLANN uses the Hierarchical K-means Tree for generic feature matching and to improve efficiency. In hierarchical K-means tree, K-means clustering is used to classify the data subset at each node. Kmeans tree is a tree where every inner node is split in k-ways. Nearest neighbors are discovered by choosing to examine the branch-not-taken nodes along the way.

FLANN uses a priority-queue (Best-Bin-First) to do ANN from Hierarchical K-Means Tree.

FLANN has Locality Service Hashing and the hierarchical clustering trees implementation, which makes it possible to use automatic algorithm configuration. It takes care of memory constraints as well as computation time to select the optimal algorithm for a specific dataset.

The feature points are detected with SURF detectors and represented in SURF-128 descriptors. The program by default set up 4 Randomized KDTrees and search for 2-nearest-neighbors. The matched key-point is only counted when the second of the nearest-neighbors is doubly farther than the first.

Steps of code used for stitch together panaroma:

1) Finding Surf descriptors in both images.
2) Matching the surf descriptors between two images using Flann Matcher.
3) Using RANSAC to estimate the homography matrix using the matched surf descriptors.
4) Warping the images based on the homography matrix.

Output:

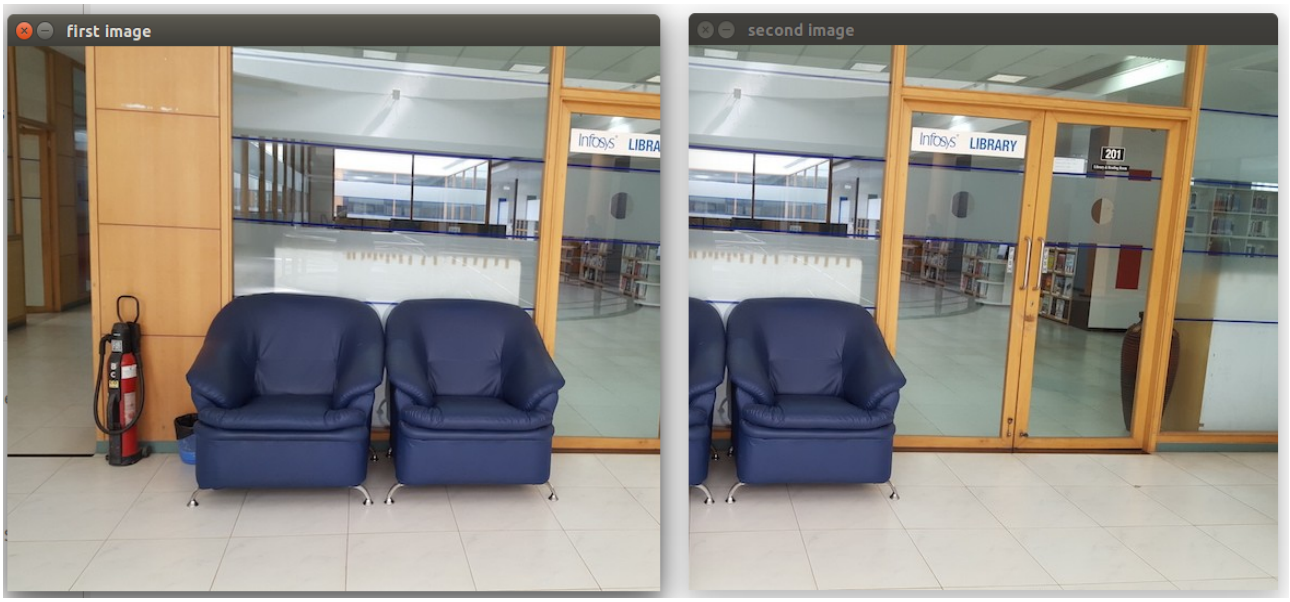


Fig. Input image1 for stiching

Fig. Stiched image

Fig. Image 2

Fig. Image 3

Que 2C) Implement Bike vs Horse Classification using Bag-of-visual words approach.

Here we want to classify image into bike or horse. We are using Bag-of-words approach. In this approch, we extract words(In our case features of image – descriptors) from given data(all training images) and put it in a single bag, then apply clustering to get similar words and then after computing feature histogram we will apply some classification algorithm.

Here we have training and testing images. Directory structure is – one directory named ‘dataset’, inside it, two class directory named ‘Bikes’ and ‘Horses’, each class directory contains two directory named ‘train’ and ‘test’ , which contains training and testing images. We have 100 images of each class, out of which 70 images are used for training and 30 are used for testing.

We have used OpenCV 3.1.0 , Python 2.4 and other required packages are numpy, scipy and scikit-learn.

Procedure Steps for Training:

1) Get image path of training images and store it into list along with its class label(0/1). We have used listdir() method of ‘os’ module to get all the file/directory names.

2) Apply SIFT for feature extraction. SIFT will give keypoints and descriptors for one image. We will apply SIFT to all the images and store all the descriptors to a list. This two steps is done in single script named – Training_Step-1_ReadImages_GetFeatures. At the end we are storing descriptor list to file and in next step we are reading the file to get it back. We have done like this because for many images we need to compute SIFT and it is taking some time and if we want to test for different parameters of K-means clustering, we don’t have to wait for descriptor computation every time.

3) Till now, we have seperated descriptors of each image in a list. Now we will combine all the descriptors and get bag of words(features). We need to iterate through every image and apply numpy.vstack() to stack descriptors in sequence.

4) Apply K-means to descriptor(bag of words) with appropriate value of K and iterations. As we change the value of K and iterations, we will get different accuracy. kmeans() takes descriptor matrix as argument, which is MxN matrix where, M is number of descriptors and N is size of a descriptor(64 or 128). It returns codebook -- kxN matrix of centroids where k is number of clusters and distortion – it is distortion between observations(descriptors) and centroids.

After appling K-means we have to compute Histogram of features for all images. feature_histogram is the total_no_of_images by k matrix. Each row contain histogram of one image. For each image descriptor, we used scipy.vq() function to compute histogram. It returns code which has codebook index for each observation and distance which is distance between observation and nearest code.

5) Classification Algorithm – Logistic Regression. Before appling Logistic Regression, we scale the feature histogram. It is the preprocessing step for Logistic Regression. We used sklearn.preprocessing.StandardScaler() to scale, then we used fit() method to generate mean and standard deviation of feature histogram matrix and transform() method to apply transformation accordingly mean and standard deviation on feature histogram matrix.

Now, apply Logistic Regression and use sklearn.linear_model.LogisticRegression() to get modal. First define LogisticRegression and then use fit() to fit the model according to the feature histogram matrix and predefined lables(class id). So finally, we got Model of Logistic Regression and now we can use this for testing. We store Logistic Regression Model, scaler, number of clusters and codebook into file and at the time of testing we will read back from file.

Procedure Steps for Testing:

1) Get image path of testing images and store it to list along with its known class lable(0/1).

2) Get keypoints and descriptors(features) of all the testing images using SIFT and store it into list.

3) Compute Histogram of features for each image using codebook(centroids) computed from training images.

4) Scale the test_features as preprocessing step of Logistic Regression. Now Apply Logistic Regression Inference algorithm to all the images and predict its class and store it to list.

5) Compare predicted class with actual class and compute accuracy.

We can change value of K(Number of segments – unique features) and iterations of K-means, different values gives different accuracy.

We can also use SURF instead of SIFT. Here, we have tried with both. Syntex of both is given in code with proper comment. SURF is taking less time than SIFT. For relatively small value of K, like K=10 and iteration=5, SURF works better than SIFT – here accuracy with SURF is 88% while accuracy with SIFT is 78%, because SURF generates less descriptors than SIFT, so for k-means with less k and iteration, it is good with less descriptors.

For K=20 and iteration=10, accuracy with SIFT is 90% and accuracy with SURF is 91.7%. For K=50 and iteration=20, accuracy with SIFT and SURF both are 91.7%.

Output :

Actual : Horse
Predicted : Bike

Actual : Horse
Predicted : Bike

Actual : Horse
Predicted : Horse

Actual : Horse
Predicted : Horse

Actual : Bike
Predicted : Horse

Actual : Bike
Predicted : Horse

Actual : Bike
Predicted : Bike

Actual : Bike
Predicted : Bike