VISUAL RECOGNITION

ASSIGNMENT-2

Srujan Swaroop Gajula

IMT2016033

QUESTION-1:

Panorama Output:

Institute1

Institute2

Panorama formed by combining Institute1 and Institute1

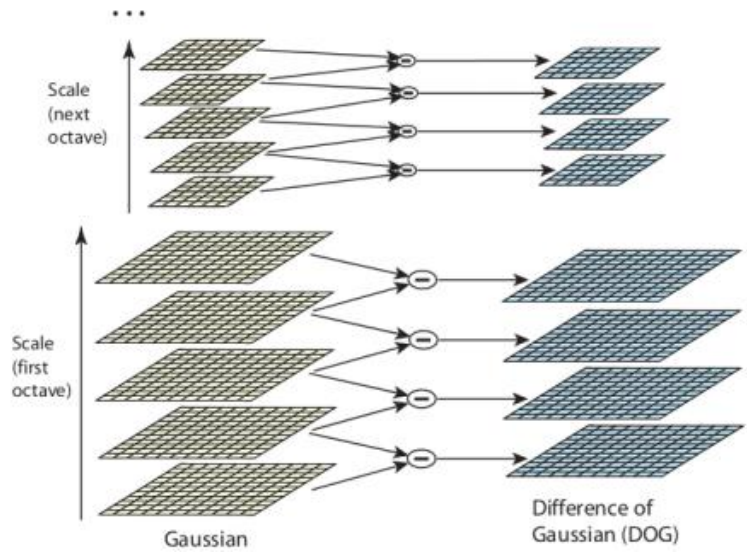
SecondPic1

SecondPic2

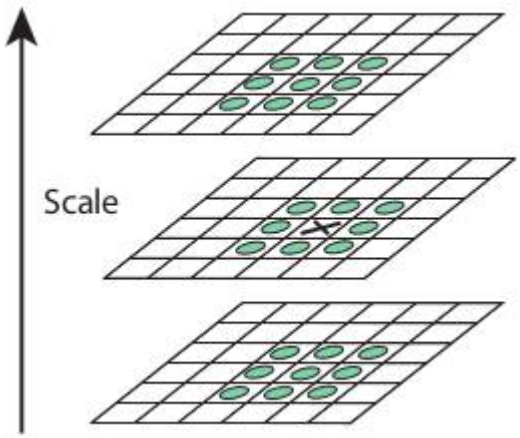
Panorama formed by combining SecondPic1 and SecondPic2

Differences between SIFT and SURF:

- **In SIFT**, Lowe approximated Laplacian of Gaussian with **Difference of Gaussian** for finding scale-space. Figure below shows how it is done.



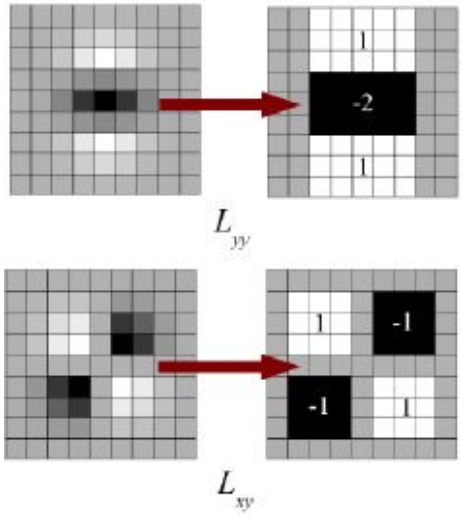
- Once this **DoG** is found, images are searched for local extrema over scale and space.



Here crossmark(x) is compared with its 8 neighbours and 9 pixels in above and below scale.If it is a local extrema then it is a keypoint.

- SIFT was comparatively slow and people needed a more speeded-up version. Then SURF(Speeded Up Robust Features) came in.

- **SURF** approximates Laplacian of Gaussian with **Box Filter**. Below image shows a demonstration of such an approximation.



- One big advantage of this approximation is, convolution with box filter can be easily calculated with the help of integral images. And it can be done in parallel for different scales. Also, the SURF rely on the determinant of **Hessian matrix** for both scale and location.

- **In SIFT** orientation is assigned to each keypoint to achieve invariance to image rotation. A neighbourhood is taken around the keypoint location depending on the scale, and the gradient magnitude and direction is calculated in that region. An orientation histogram with 36 bins covering 360 degrees is created. The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation. It creates keypoints with same location and scale, but different directions.

- For orientation assignment, **SURF** uses wavelet responses in horizontal and vertical direction for a neighbourhood of size 6s. Adequate gaussian weights are also applied to it. Then they are plotted in space as given in below image. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window of angle 60 degrees.

- For **SIFT** 16x16 neighbourhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor.

- For **SURF** neighbourhood of size 20x20s is taken around the keypoint where s is the size. It is divided into 4x4 subregions. For each subregion, horizontal and vertical wavelet responses are taken and a vector is formed which is of 64 dimensions.

MAIN PRINCIPLES OF RANSAC:

- RANSAC: RANdom SAmple Consensus
- Using **SIFT** we find features for the two images. We get a list of features.
- Then find matches that are corresponding interest points in two images using Flann or Knn match.

- But, all the resulting matches may not be correct, some may match one interest point to another.
- RANSAC is used to eliminate these mismatches for better output.

RANSAC method:

- **STEP1:** Select four feature pairs at random.
- **STEP2:** Compute Homography(H) that is transformation matrix so first images will be aligned with the second image.
- **STEP3:** Now, we can rotate using the above matrix and find **inliers** that are points which are closer.We call them inliers when corresponding points are less than some threshold (||p1,h*p2)|| < epsilon).
- If we do not get the minimum number of inliers we then start the process again from step1
- **STEP4:** We repeat the above process and estimate inliers again can we keep the value for which we get the largest set of inliers. So by this method, we get the best Homography matrix for which we get the largest set of inliers.

Question-2:

Here for classification of images we use Bag of Words technique which is quite famous in classification context.
In this method we create a vocabulary that can best describe image in terms of extrapolable features.

Bag of Words:

Briefly our method is to find descriptors for each image then divide them into clusters and assign corresponding cluster label to each

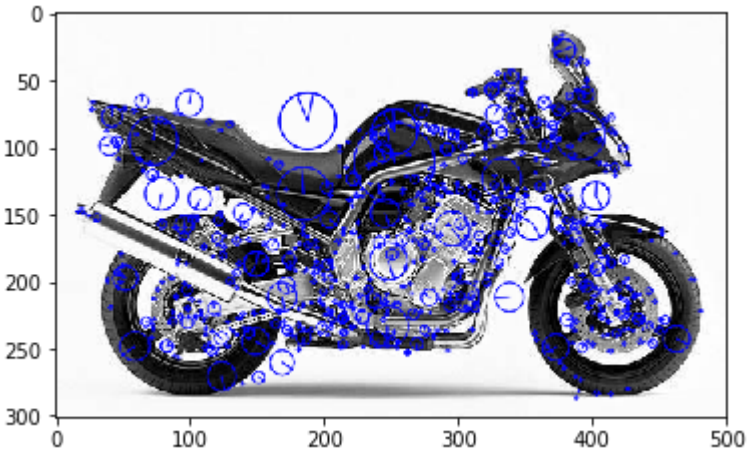
descriptor and calculate frequency of each cluster in the image and train model using this frequencies.Later predict on test images by sending their frequencies to model.

Approach:

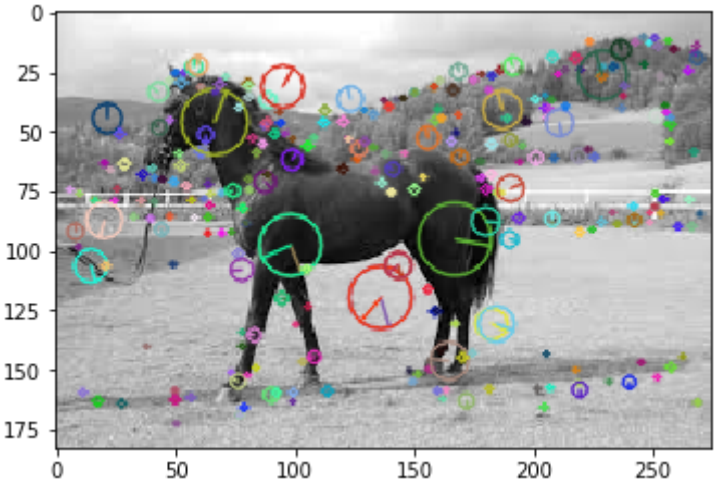
1. ***Feature Extraction:***

First we have to find interest points in images.Because similar images mostly have same interest points like nose,eyes,hands etc in human. To find interest points I used SIFT, it gives interest points by giving Keypoints and descriptors of the interest points.

These are the interest points for one bike image



For horse image keypoints are



2. ***Clustering:***

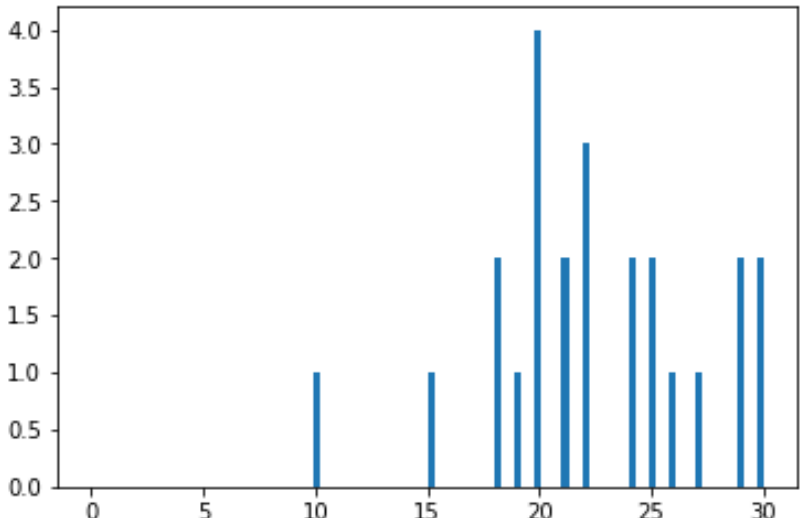
For example we get nose descriptors of all human images.So we have a lot of same features.When we apply clustering all noses go into one cluster, all eyes go into another cluster etc.For this I used K-means clustering to group them.

How many clusters should we have?

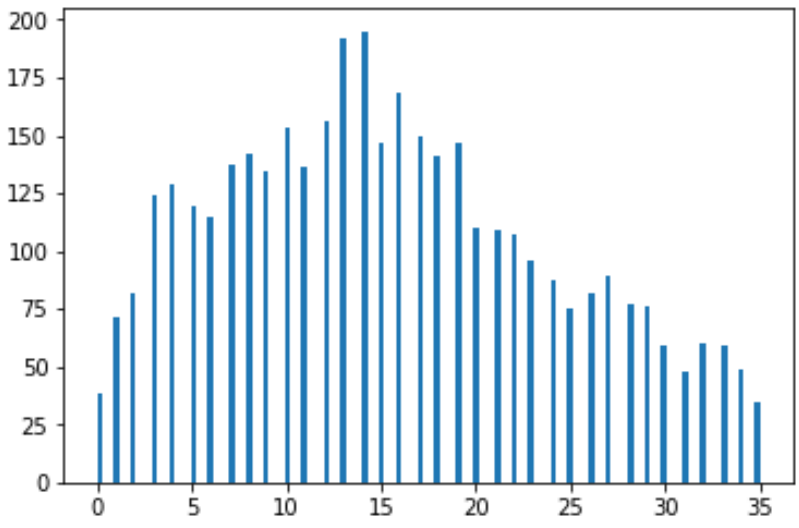
It is trail and error.I tried it for different values and finalized the value for which I got highest accuracy.

3. ***Histograms:***

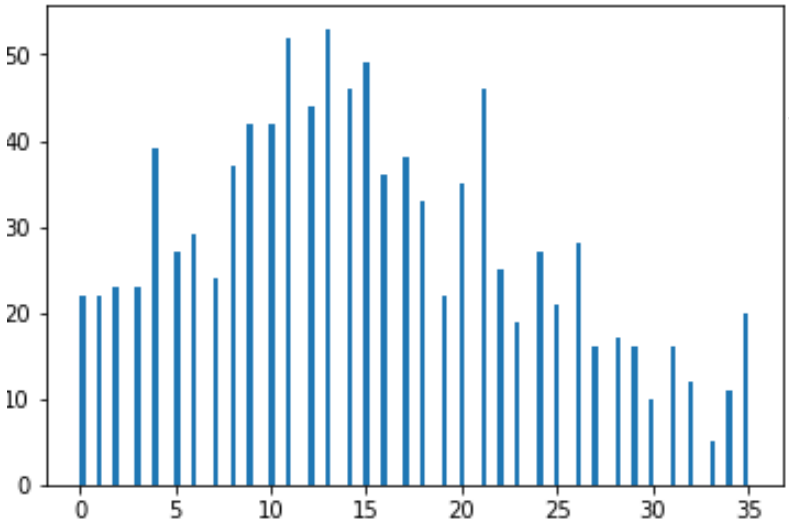
Now, each descriptor got assigned to a cluster.We can plot a histogram such that frequencies of each cluster in an image are plotted.



This is the count of descriptors with respect to clusters of the first bike image.



This is frequency count of all train image descriptors.So, here we know how many descriptors of train image have 0 cluster , 1cluster so on.



This is frequency count of all test image descriptors.So, here we know how many descriptors of test image have 0 cluster ,1cluster so on.

Then using this histogram frequencies we train that like if an image has this much count for some labels then it comes under this class. So we train models on histograms and predict our output using this model.

Observations:

So,I used LinearSVC for training of images using 30 clusters.I divided the date into 80% train data and 20% test data.I got 88.888% accuracy.

```
clf = LinearSVC()
clf.fit(train_hist, np.array(y_train))
print (" .....Accuracy........=  ",clf.score(test_hist,y_test))
```

```
 .....Accuracy........=   0.8888888888888888
```

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(train_hist, np.array(y_train))
print (" .....Accuracy........=  ",neigh.score(test_hist,y_test))
```

```
 .....Accuracy........=   0.8888888888888888
```

So,I used Kneighbours Classifier for training of images using 30 clusters.I divided the date into 80% train data and 20% test data.I got 88.888% accuracy.

So,I used Logistic Regression for training of images using 30 clusters.I divided the date into 80% train data and 20% test data.I got 91.66% accuracy.

```
log = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial').fit(train_hist, np.array(y_train
print (" .....Accuracy........=  ",log.score(test_hist,y_test))
```

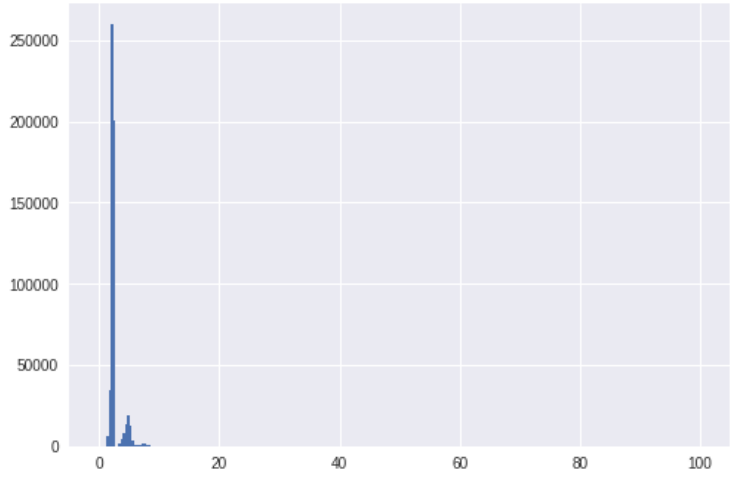
```
.....Accuracy........=   0.9166666666666666
```

For Ciphar10 dataset,

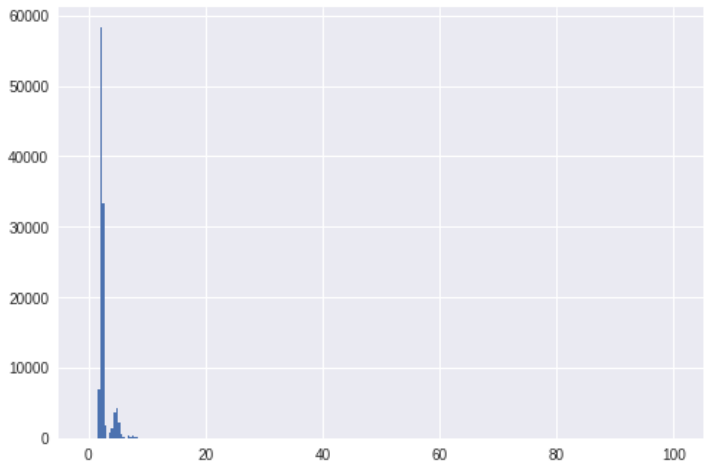
first while finding descriptor for train images some of them were None because resolution of images were very less.Their size was 32x32. So, I discarded the images for which descriptors were None as they are no help for training the model.

For test images which got None descriptors I appended numpy array of 128 dimension with all zeros.

Train images cluster frequencies:



Test images cluster frequencies:



Then I trained this on LinearSVC using 100 clusters and predicted using test data set.
I got accuracy of 26.7% which is very less because the images have low resolution and so number of descriptors are very less and model cannot train properly using this. So, accuracy was very less.

```
clf = LinearSVC()
clf.fit(train_hist, np.array(train_labels))
print (" .....Accuracy........=  ",clf.score(test_hist,test_labels))
```

```
 .....Accuracy........=   0.267
```

Then I trained this on Logistic Regressionusing 100 clusters and predicted using test data set.I got accuracy of 26.71%.

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression(random_state=0, solver='lbfgs',multi_class='multinomial').fit(train_hist, np.array(train_labels))
print (" .....Accuracy........=  ",log.score(test_hist,test_labels))
```

```
 .....Accuracy........=   0.2761
```

Sources:

https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html
https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html
https://www.learnopencv.com/homography-examples-using-opencv-python-c/