ASSIGNMENT 2 (REPORT)

Question 1

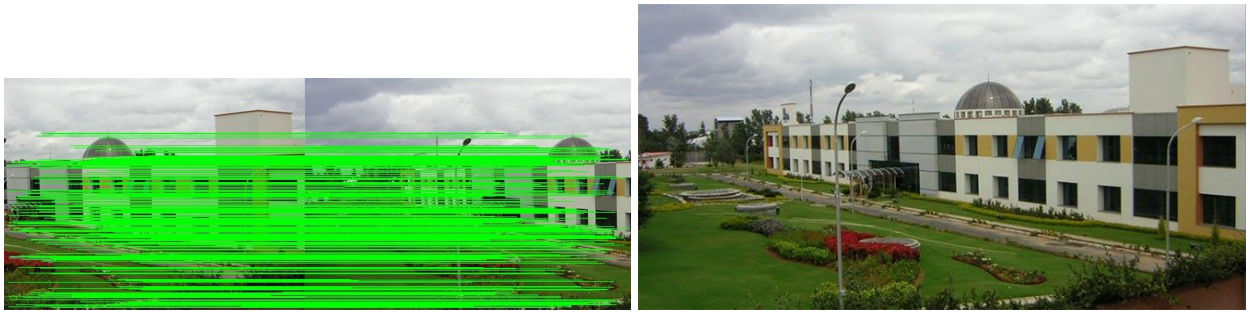
a) Play with Panorama: Use the code shown by Rahul (TA) and Stitch together a panorama of IIITB

b) Explain how SURF is different from SIFT (10 sentences)

c) Briefly explain the main principles of RANSAC/FLANN matching (5 sentences)

Approach

a) As the question clearly states to use the code form the material I first looked through the OpenCV_Tutorial2 slides provided by the TA's and the code mentioned in the class. This helped me to stitch together a panorama successfully. In the code, I make use of the SIFT functionalities to find the features of the image. Used a knn matcher. And then matched the key points. Finally stitching together the panorama of the resultant image.



b) SURF and SIFT are used in the detection and description of key points of the images. But SURF is different from SIFT.
The following are the differences between them :

SIFT - Scale-Invariant Feature Transform.

and

SURF - Speeded Up Robust Features.

As the name indicates SURF is faster than SIFT. Where SIFT uses Lowe approximated laplacian of gaussian with the difference of the Gaussian for finding the scale space SURF goes a bit further in its approximations and approximates LoG with a box filter. Using SURF the process can be done in parallel for different scales and rotation invariance is not required which helps in improving speed and robustness compared to SIFT. And another important improvement over SIFT is that SURF only compares the features if they have the same type of contrast this helps in faster matching without reducing the descriptors performance. This shows that SURF has added a lot of features to improve the speed at every step when compared with SIFT. Therefore SURF is 3 times faster than SIFT. SURF handles the images with rotation and blurring better than SIFT. But SIFT has the upper hand in handling viewpoint and illumination change. These are the primary difference between SIFT and SURF.

c) RANSAC - RAndom SAmple Consensus

RANSAC matching can estimate parameters of the model by random sampling of observed data. RANSAC finds the optimal matching keeping the in liners and outliners in mind.

It involves two steps :

1. Randomly select sample subset containing minimal data items so that a fitting model and its model parameters can be found using the elements in the subset.
2. Check for the elements in the dataset that are consistent with the generated model. It is an outlier if it does not fit the model within some error margin.

This results in the best fitting model or NULL is no good model is found.

FLANN - Fast Approximate Nearest Neighbor Search

It is a library that contains a collection of algorithms optimized for fast nearest neighbour search in the large datasets or higher dimensional features. It returns the nearest neighbour of a data item. It can improve the quality of matchings but it slows the algorithm.

RANSAC is used when matching the SIFT feature points have lots of mismatches. The RANSAC algorithm can be used to remove the mismatches by finding the transformation matrix of these feature points. And FLANN is used for keypoint matching because it is faster.

Question 2

Implement Bike vs Horse Classification

Dataset: available on LMS (notes folder)

Use Bag-of-visual words approach (SIFT/SURF + K-means + SVM/Logistic Regression/KNN)

Explain the procedure and your approach and observations

Reference paper: available on LMS (notes folder)

Extend to CIFAR 10, with 10 classes

Approach

For Bike vs Horse Classification:

I have read the reference paper and understood the concept involved in solving this problem.

First I split my data into test and train. I saved 25% of the data for testing. Using the split data function that I have written.

Then I loaded the images one category at a time and put them all in the images dictionary. The images have direct also has the labels to the images. I have loaded the labels into another variable for using it in building the model.

Generated keypoints and features for every category using

keypoints, descriptors = cv2.xfeatures2d.SIFT_create().detectAndCompute(image, None)

Then the descriptors needed to be clustered together optimally. I used Kmeans for this. Set the clusters as 100 and performed the clustering operation using the descriptors.

clusters = KMeans(n_clusters = 100)
fit = clusters.fit_predict(descriptors)

It took about 522.4269630908966 secs to form these clusters.

To generate a vocabulary I have created a Histogram that takes into account all the descriptors and which index they belong to using K-means. Now all the images get assigned to the index generated by their feature vector that is their descriptors by the Kmeans algorithm.

The Vocabulary generated is the Bag of visual words approach. This Vocabulary now contains all the training dataset images assigned to their labels generated by their features found through SIFT and clustered together using Kmeans.

Now, this Histogram is passed in SVM model [Using SVM model] along with the labels.

Now that the vocabulary and the model are been generated we go for the testing part of the algorithm.

Loaded the test images. These images are processed category by category and their features and keypoints are detected and passed into my previous kmeans model as it predicts which cluster does the descriptor belong to. Now, this info is used to create a histogram and passed into our model (SVM) for predicting which label it belongs to according to it. These predictions, therefore, become the predicted labels and I already know the ground truth as I have loaded the test images before along with their labels.

These are formatted in such a way that I can find the accuracy score for the SVM model that had been used.

I got an accuracy score of 0.8888888888888888 for SVM.

I also tried other models such as Logistic Regression, Ada Boost but none of them gave me a better average than SVM. Some models performed well for certain cases of randomized splitting.

LR - 0.9555555555555556

KNN - 0.4222222222222222

Ada Boost - 0.9333333333333333

The Bag-of-visual words method was followed as I have generated the features using SIFT and used the K-means clustering for predicting which cluster the feature belongs to. Then trained a model on this info and got the results.

Observations:

The dataset consists of diverse images of horses and bikes some of the images have a bit of occlusion and having different scales and alignment. The feature detection and clustering becomes more important for this kind of dataset.

So the clusters number was increased from 20 to 100 linearly to check if the model improves the result and fairly it does.

I also observed that the among all the models SVM gave a consistent performance through the parameter changing so even when Logistic regression has the highest score I prefer SVM over it and applied the same to the CIFAR 10 dataset.

CIFAR-10

There was some difficulty in understanding the batch file system in Cifar10 data set but I have gone through with it.

All the data was encoded in latin1 format and I have loaded and formatted the images into features and labels.

I loaded the data into test and train components respectively the train has all the 5 batch images. And the test images have the test batch as extracted from the cifar dataset respectively.

I have loaded and arranged them into folders using the load_CIFAR10 function. The five batches contained about 50000 images each batch consists of 10000 images.

Many of the images, when passed through the feature detection phase using SIFT, gave None as the output indicating that there were no features to extract. So I have ignored the features and the labels of the images and continued with the rest of the images.

The next step was clustering So this step I have used 200 clusters just to improve the accuracy, in the beginning, I have used the 100 clusters similar to the bike horse classification but that did not give a good accuracy so I have used 200 clusters in this case.

The Kmeans algorithm has run for about 2hours in my laptop and then the clusters were generated.

I have done a bit of scaling as there was a warning popping out and passed the arguments into a StandardScaler() to get them formatted for model generation.

I have used a similar approach in generating the vocabulary. I have created Histogram for the feature space to predict where the cluster belongs to using the Kmeans algorithm.

Then on I have used a SVM model as it was being consistent and productive. The SVM function took about half an hour to generate a model.

I then loaded the test images in a similar manner and passed them on to a prediction function which uses Kmeans predictions in where the feature belongs to and tells us where the point belongs to in the classification. After that, I have generated an similar histogram for its use in the model prediction and telling us the predicted label of the image based on the image.

I have then compared the test labels that i have genrated form the folder creation and compared them with the predicted labels to find the accuracy score of the model.

I got an accuracy score of 0.2979149959903769 for a SVM model.

I have loaded this model into pickle file so that it makes it easier for you to run the code.

I found the images very hard to look at as they were 32x32 so I thought the features were not being detected properly so the accuracy was a bit low. But for a multiclass prediction got to be tough assuming there was the probability of the image belonging to any class 0.1

so my model was better than a random guess because it has a accuracy of 30%. I thought I could improve it a bit more but that wasn't asked in the question. The question was just to extend the bikes vs horses classification to 10 classes and I did the same.

I am also attaching the code along with the jupyter notebook so that it would be easier to look at the itermidete steps and go through the code thoroughly.

References :

https://docs.opencv.org/3.1.0/

https://towardsdatascience.com/image-stitching-using-opencv-817779c86a83

https://www.pythonforbeginners.com/os/python-the-shutil-module

https://www.cs.toronto.edu/~kriz/cifar.html

https://stackoverflow.com/

https://docs.python.org/3/