Karthik Uppuluri Karthik.Uppuluri@rice.edu

ELEC 547: Assignment 5

Image Classification

1.2 Technical Write-up

a. Method:

Goal: Image Classification:
Dataset: 3-class (Airplane, Butterfly and Buddha)
Features: SIFT (scale-invariant feature transform).
Classification Model: Bag-of-words

Implementation of Algorithm in detail:

Step1: Feature matrix generation (Training data) (VL_sift)

Given the training data set, SIFT features are obtained for each Image in each class, and all the features of the entire training data set are appended to generate a '**Training feature matrix**'.

To generate SIFT features I used VLFeat's SIFT implementation [1]. Typical sift features and descriptors for the given classes are shown below

Airplane:

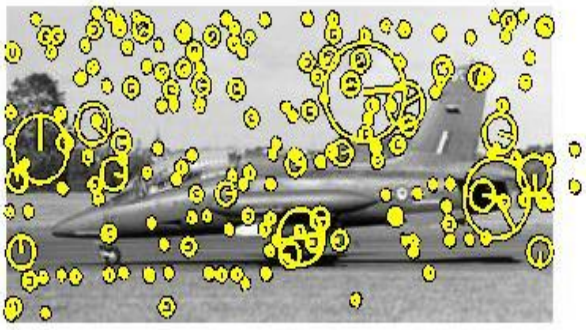

Figure1.a SIFT features

Figure1.b SIFT feature descriptors

Butterfly:

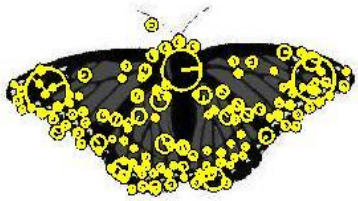

Figure2.a SIFT Features

Figure2.b SIFT Feature descriptors

Buddha:

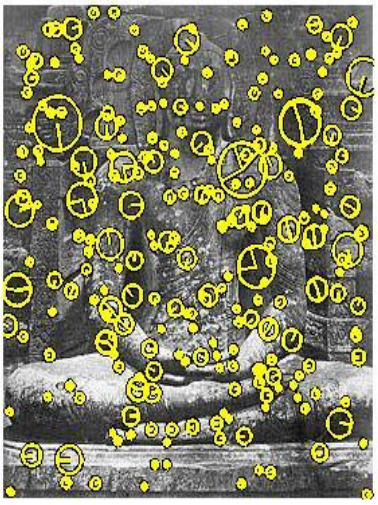

Figure3.a SIFT Features

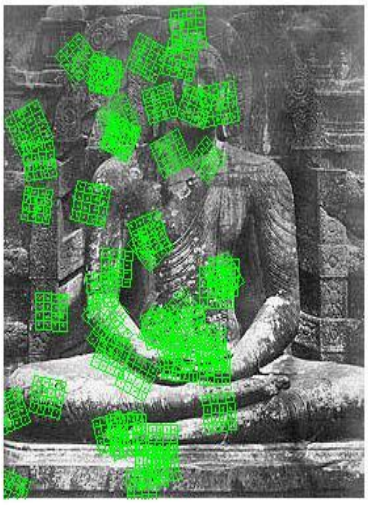

Figure3.b SIFT Feature descriptors

Step2: Feature clustering and code book generation (Kmeans++ and knnsearch)

Once all the feature descriptors of all the images in all the classes are obtained and grouped, the feature vectors are clustered in to **1000** clusters using **kmeans++**. The centroid of each cluster represents a distinct visual word. So, in this case I have 1000 visual words or a dictionary generated with 1000 unique words.

For clustering purpose I used VLFeat's implementation of "kmeans ++". The major disadvantage with ordinary kmeans is that it is very slow and the solution is not always consistent and optimal. Kmeans ++ address both the problems.

Step3: Representative histogram generation for each class (knnsearch)

Once visual dictionary/code book is generated representative histogram for each class is created by counting the occurrences of these distinct words in their respective class datasets.

Once histograms are generated for each class, they are normalized by dividing the bin count with total number of features in each class.

I used Correlation distance metric as it gave better accuracy.

Step4: Testing (knnsearch)

Feature matrix for each test image is generated and it is clustered with the code book generated during the training. Histogram for each test Image is generated by counting the number of occurrences of distinct visual words in test image. And the histogram is normalized by dividing with number of features in each test image under consideration. Once the normalized histograms are generated for each test image, they are associated with one of the representative histograms created earlier. And the classification accuracy in each case is calculated by constructing a confusion matrix.

I used **Euclidean distance** to measure the nearest histogram to the representative histogram.

b. Points of possible improvement

- The feature matrix was really huge (around 16209152 points).
  - **a. Better features which are equivalently effective and intrinsically lesser in dimension**
  - b. **Usage of dimensionality reduction techniques before clustering**.
  - c. **Using integral images to make computation faster, likewise in SURF.**
- **Kmeans ++** is comparatively faster than ordinary kmeans, but I have experimented with some other kmeans implementations like **litekmeans** [2] which was un-believably faster but the results obtained are never consistent. So a blend of kmeans++ and litekmeans would have given an optimal solution with a best speed.

c. Confusion matrix

Classes	Airplane	Butterfly	Buddha
Airplane	68.75%	0	31.25%
Butterfly	0	80%	20%
Buddha	7.14%	14.29%	78.57%

Butterfly class got the highest accuracy. The reason I could speculate is that the number of features for butterfly is extremely larger than the other classes. The validity of the confusion matrix presented can be re-checked by just running the code as written in the readme file without computing clustering again.

2.2 Technical Write-up

For the part 2 of the assignment I experimented with SURF (speeded up robust features). To obtain SURF features I used Petter Strandmark's implementation SURFmex [3]. I also experimented with SIFT features by reducing their dimensionality of each feature vector from 128 to 20.

Salient features of improved classification framework

- **SURF (Speeded up robust features)**
  SURF features are intrinsically lesser in dimension, in this case each feature descriptor is of dimension 1x 64 which is twice less than a typical SIFT descriptor. And usage of integral images makes computation so faster. So for the given 25 class data computing the features and clustering the features will be more efficient if the features are SURF features.
- **Kmeans ++[4]**
  Kmeans++ is almost consistently optimal and very fast compared to ordinary kmeans. Running time of kmeans++ is O(log K) whereas ordinary kmeans has a running time of $O(n^{dk+1} \log n)$ where 'n' is the number of entities to be clustered and k is the number of clusters.

Method Comparison:

My method:

Features: Speeded Up Robust Features [5]

I used SURF implementation by Petter Strandmark with hessian threshold of 100. Conceptually the implementation is completely similar to [5] except the threshold value I used which I have chosen empirically.

Clustering: Kmeans ++ [1]

Classification: Bag of words and histogram matching.

Method and results

Classification using SURF features:

Step1: Training Feature matrix: (surfpoints)

Training feature matrix is generated by appending all the SURF feature vectors obtained for each image in all the classes in the training data set.

Step2: Clustering and code book generation (Kmeans++ and knnsearch)

Once the training feature matrix is generated it is clustered using Kmeans++. In this case I clustered the data into 5200 clusters. Once the data is clustered the centroid of each cluster represents a unique word. So I have a dictionary of 5200 unique words.

Step3: Representative histogram generation for each class (knnsearch)

Once the dictionary is created representative histogram for each class is created by counting the number of occurrences of those words in their respective feature matrices. Each representative histogram is normalized by dividing the bin count with total number of features in each class under consideration.

I used correlation as the distance metric, as it gave better accuracy.

Step4: Testing (knnsearch)

SURF features of each test Image are obtained and they are clustered with dictionary generated during the training phase. Histogram for a given test image is obtained by counting the occurrences of different visual words present in the dictionary. Once the histogram for each test is computed it is normalized by dividing with the number of features in each test image under consideration. Normalized histograms are then compared to the representative histograms of each class and are associated to any one of these representative histograms by computing any distance/similarity metric.

I used Euclidean distance to find nearest histogram to the representative histogram.

Confusion matrix:

As the confusion matrix is very large I saved it and uploaded it along with the code.99 out of 292 images are correctly classified.

Classification accuracy of 25 class Image classification using SURF features = 33.9%

The accuracy I got when I used SIFT Features + PCA is very low.

Handling large datasets

I made sure that I declared lesser variables, which in turn occupy lesser workspace of my MATLAB, thereby reducing the usage of my RAM.

References:

1. http://www.vlfeat.org/overview/sift.html
2. http://statinfer.wordpress.com/2011/12/12/efficient-matlab-ii-kmeans-clustering-algorithm/
3. http://archive.is/3BXK
4. http://www.slideshare.net/renaud.richardet/kmeans-plusplus
5. http://www.vision.ee.ethz.ch/~surf/