CS783 : Visual Recognition

Rohin Garg
160583

Aaditya Singh
160002

Assignment 1: Multiple Instance Recognition

February 5, 2019

Method 1: Text retrieval based approach

This is our final method that we have used. The outline of the algorithm used is as follows:

1. Find the 'visual words' in the image: We have used the descriptors given by the Scale Invariant Feature Transform (SIFT) of the images in the database as their representatives. Each key point descriptor is a 128 length vector with integral values. The number of key points vary for each image.
2. 'Stem' the visual words: We clustered the descriptors using K-means clustering, and each descriptor is assigned the closest cluster centroid. These centroids represent the stemmed visual words. The number of cluster centroids is 12500.
3. Creating an inverted list for 'Visual Indexing': An inverted list for each image is created using Term frequency - Inverse document frequency weights. For each image, there is a weight vector that has weights for every visual word.
4. Procedure for Querying: We extract the descriptor from the query and assign the nearest cluster centroid to each descriptor. Then we use the inverted list for finding the best matches for the query.
   A similarity vector was calculated using **cosine similarity** between the query and all image tf-idf vectors.

Improvements

1. Non-uniform weights: The training images had all the objects located in the center. So, while calculating Tf-Idf weights, we gave more weights to the key-points in the center of the image, thus reducing the dominance of background objects such as chess-board key-points.

Method 2: Scalable recognition with a vocabulary tree

The outline of the algorithm used is as follows:

1. Find the 'visual words' in the image: We have used the descriptors given by the Scale Invariant Feature Transform (SIFT) of the images in the database as their representatives. We have used the descriptor dimensions of (500,128) for every image.
2. Create the 'vocabulary tree': We recursively used k-means clustering for finding the cluster centroids at each level. We chose 6 levels and 5 descendant cluster centroids for every centroid.
3. Creating an inverted list: We again created a dictionary of dictionaries, but with the leaf cluster centroid indices as the keys. The dictionary for the leaf cluster centroids still has the same key-value pair as before.
4. Procedure for Querying: We extract the descriptor from the query and assign the nearest leaf cluster centroid to each descriptor. Then we use the inverted list for finding the best matches for the query.

Preferred Method

We prefer **Text retrieval based approach** over the scalable recognition with a vocabulary tree, mostly due to the fact that our second method suffers from the problem of many examples being assigned to a few leaf clusters, and a few examples being assigned to a large number of leaf clusters. This resulted in poor results with the 2nd method. But this method takes considerably lesser time than the first one for training. If we could resolve its problems of clustering, then we would have preferred the second method.

Challenges Faced

1. **Background Objects**.
   The presence of unwanted background objects such as a chessboard in the training images results in a majority (> 350 out of 500) of descriptors which are not required for identifying the required objects.

   We solved this challenge by giving more weight to the key-points in the center of the training images, and less to the surrounding ones. This works because the required object was always present in the center if the image.

2. **Background Scenery**.
   The presence of background scenes in the query images such as forests and mountains in the test image results in a lot of descriptors which adversely affect the matching. We can resolve this issue by using a deep learning based object detection algorithm which can extract the required objects from the image.

   We tried to solve this by only taking the key-points whose euclidean distance from the cluster assigned to it was less than some threshold, computed by experimenting on hard test-cases.

3. **Lack of Computational Power**.
   In order to obtain better matches for instance recognition, a large number of clusters were required. This requires high computational power, something which neither of us have.

Results

Text retrieval based approach:

The results on sample_test are provided in a folder ”sample_test_results”.

1. **Training images:**
   All the images with the chess board not in the center (in front or in the back) of the image/object were correctly identified. This happened because of the way we have accentuated

2. **Easy tests:**
   The results are nice, with only some confusion among similar looking objects. The result improves with taking more finer key-points by decreasing sigma (scale) at octave = 0.

3. **Hard tests:**
   Our algorithm does not perform good on hard_single 2 and 3, because of a large number of key-points extracted from the background scene. Hard_single 1 was identified, with a high value of sigma (scale) at octave = 0 for sift-features by blurring the image a little.
   But this was not done as increasing sigma results in more confusion among objects in easy cases.

MD5 Hashes

Hashes of the KMeans cluster and Tf-Idf weight matrix obtained after training and the code are provided in ”hashes” in the zip file.