CS676A: Introduction to Computer Vision

Assignment #3

Instructor: Prof. Vinay P. Namboodiri

Mohit Singh Solanki(12419)
Deepak Kumar(12228)

1 Assignment

Scalable Vocabulary Recognition Tree In this assignment you have to implement the scalable vocabulary recognition tree as described in the paper by Nister and Strewenius, CVPR 2006. You are allowed to use a library for k-means and SIFT feature. You have to evaluate the method with different branch factors and depth factors and evaluate the matching on the UKY dataset .

2 Scalable Vocabulary Recognition Tree Method

2.1 Training the tree

The vocabulary tree defines a hierarchical quantization that is built by hierarchical k-means clustering. Instead of k defining the final number of clusters or quantization cells, k defines the branch factor (number of children of each node) of the tree. First, an initial k- means process is run on the training data, defining k cluster centers. The training data is then partitioned into k groups, where each group consists of the descriptor vectors closest to a particular cluster center.

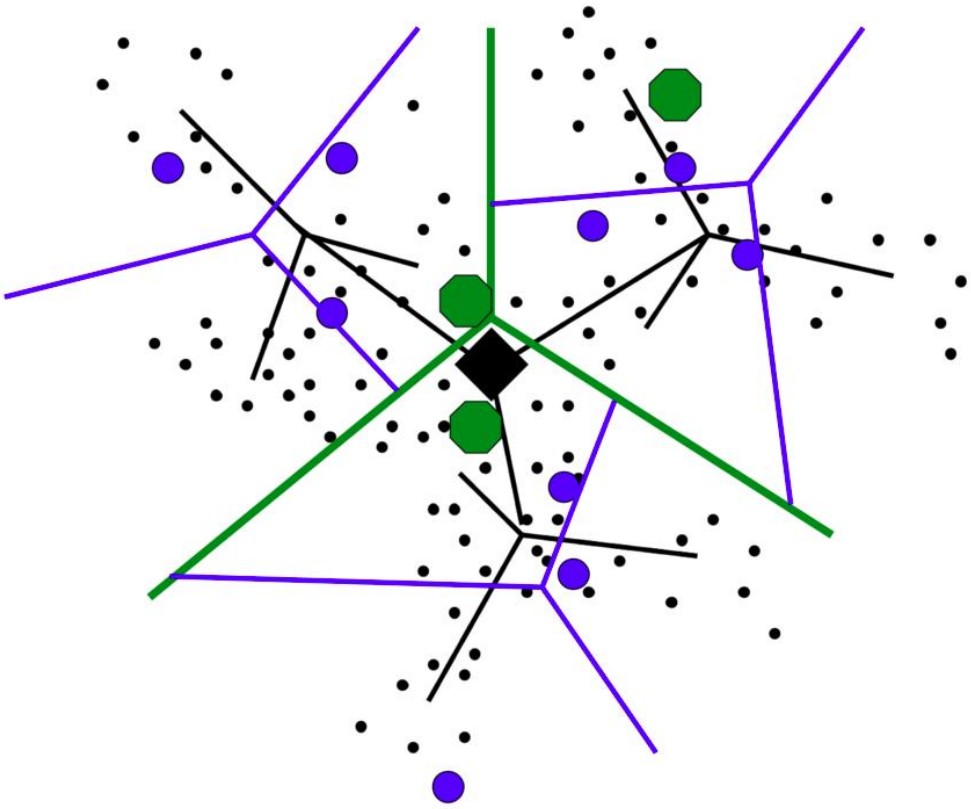

FIGURE 1: Heirachical k-means

The same process is then recursively applied to each group of descriptor vectors, recursively defining quantization cells by splitting each quantization cell into k new parts. Then the tree is determined by these levels upto a maximum no. of level L.

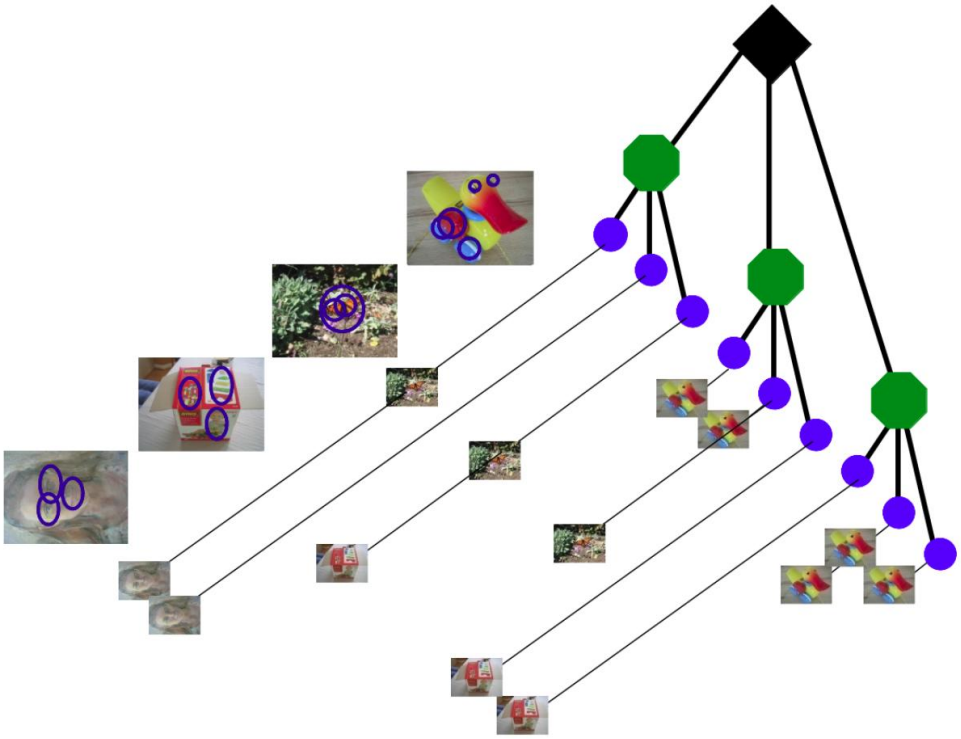

FIGURE 2: Vocabulary Tree

2.2 Scoring

Once the tree is constructed we have to define relevance of database image for given query image based on the similarity in path of two images. So we are going to assign a weight w_i to each node, based on entropy and then defining both query and database vectors q_i and d_i according to following equations :

$$q_i = n_i w_i \tag{2.1}$$

$$d_i = m_i w_i \tag{2.2}$$

where n i and m i are the number of descriptor vectors of the query and database image, respectively, with a path through node i. A database image is then given a relevance score s based on the normalized difference between the query and database vectors :

$$s(q,d) = \| \frac{q}{\|q\|} - \frac{d}{\|d\|} \| \tag{2.3}$$

This normalization can be by any possible method but as described in given paper we are considering L_1 norm (which should give us best results). For calculating weights of nodes we are using idf(inverse document frequency) in which retrieval performance is typically improved by an entropy weighting :

$$w_i = ln \frac{N}{N_i} \tag{2.4}$$

where N is the number of images in the database, and N i is the number of images in the database with at least one descriptor vector path through node i. After calculating weights for each node query and database vectors are normalized such that $\sum_{allimages} q_i = 1$ and $\sum_{allimages} d_i = 1$ After this step normalized difference is calculated and score is assigned to images for a given image. After this, these images are sorted in ascending order and images with least score are the best matches for the given image.

3 Implementation and Results

In our implementation due to memory limitations we have taken 700 images only. On these images we have performed the above described method for different values of k(branching factor) and depth of tree.

Sl No	**Height(H)**	**Branching Factor(K)**	**Accuracy(in %)**	**InTop4 (in %)**
1	6	3	31.43	33.57
2	6	4	55	66.28
3	5	5	50.42	60.57
4	4	10	56.57	68

4 Image Results

FIGURE 3: Test Image : All Matches

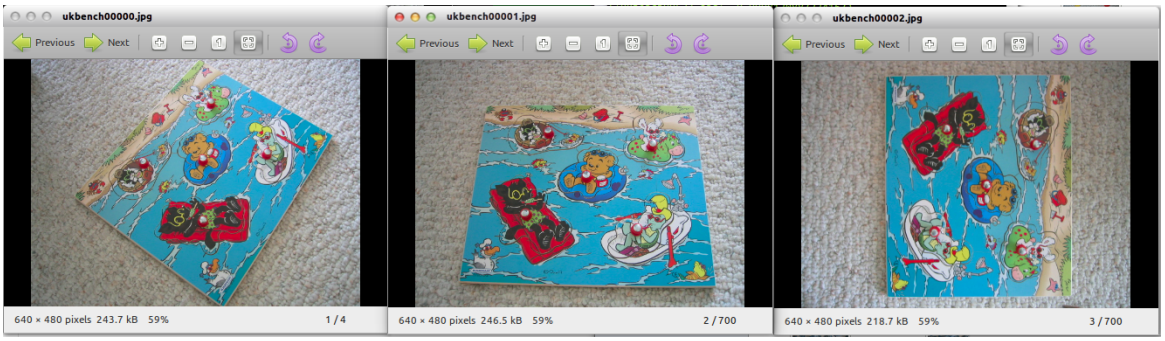

FIGURE 4: Matched Images

FIGURE 5: Test Image : 3 Matches

FIGURE 6: Matched Images

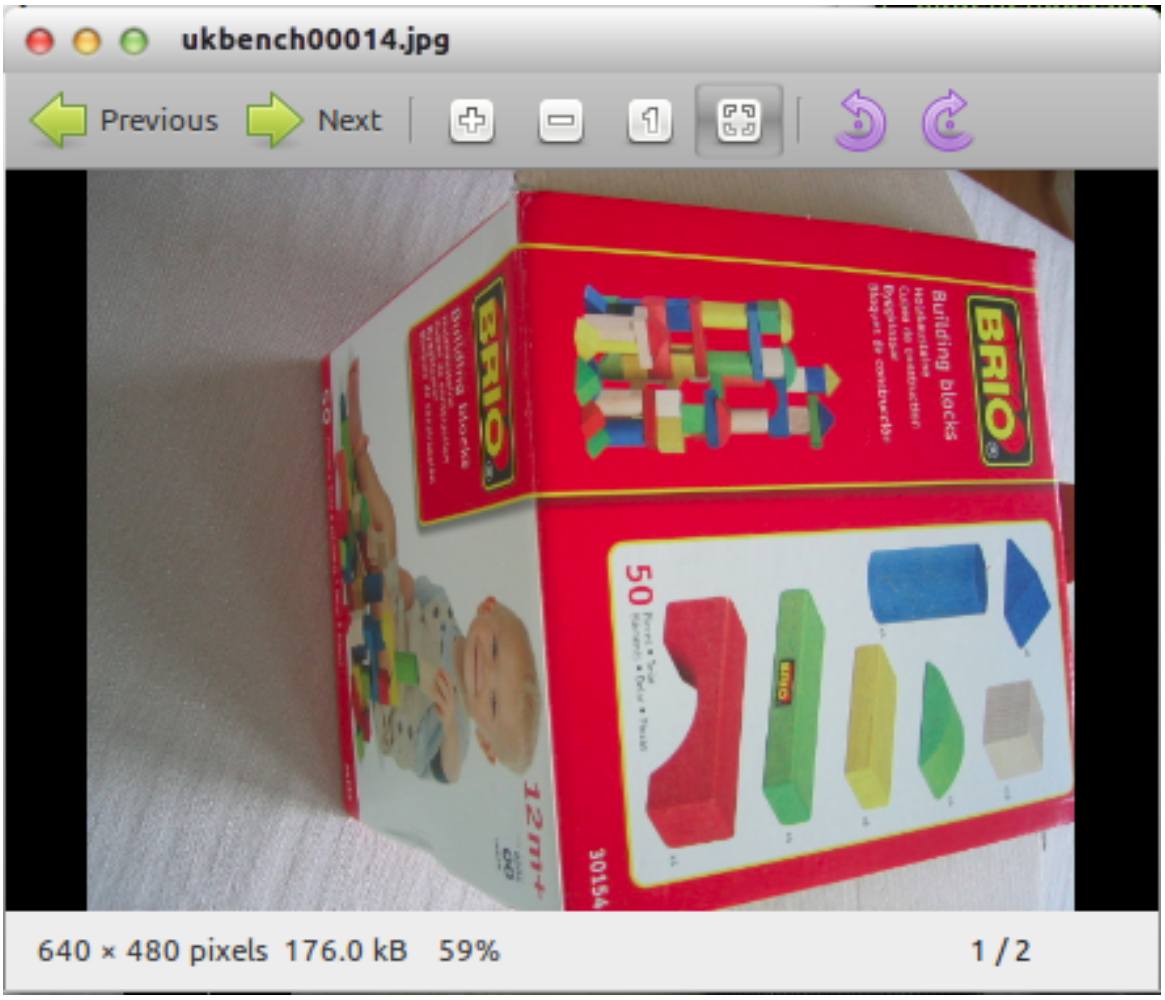

FIGURE 7: Test Image : 2 Matches

FIGURE 8: Matched Images