Scalable Recognition with a Vocabulary Tree (Nister & Stewenius)

A recognition scheme that scales efficiently to a large number of objects, using vocabulary trees and hierarchical scoring

Project GENERAL OVERVIEW

- Algorithm scales well with the size of the database, and can select one out of a large number of objects in acceptable time
- Builds a Vocabulary tree from database image features
- In my implementation, I do pre-computing of the database vectors and weights. Speeds up scoring...
- Uses hierarchical scoring as oppose to TF-IDF scoring
- Obtain top 10 images from database, perform RANSAC to find image with most inliers compared with the query images

Vocabulary Tree

- Vocabulary Tree directly defines the visual vocabulary and an efficient search procedure in an integrated manner
- Builds a Vocabulary tree through K-clustering
  - The vocabulary tree defines a hierarchical quantization that is built by hierarchical k-means clustering.
  - K is the branch factor, L is the level of the tree
  - Thus there are K^L many words "defined" for a given set of database image
  - Used SIFT's hikmeans API function to build the tree
    - K^L minimum leaf nodes generated in last level

Hierarchical scoring

- Determine the relevance of a database image to the query image based on how similar the paths down the vocabulary tree are for the descriptors from the database image and the query image
  - di = mi * wi
  - qi = ni * wi;
  - ni and mi are the number of descriptor vectors of the query and database image, respectively, with a path through node i
  - wi = ln N / Ni , Ni is the number of images in the database with at least one descriptor vector path through node I
  - Take the L1 Norm; 2 + Sum of q and d(|qi − di |^p − |qi |^p − |di |^p), p = 1 for L1 norm

Runtime

- the vocabulary tree directly defines the visual vocabulary and an efficient search procedure (hierarchical scoring) in an integrated manner
- Compared to traditional means of defining a visual vocabulary non-hierarchically, and then devising an approximate nearest neighbor search in order to find visual words efficiently
- Cost of increasing the size of the vocabulary in a non-hierarchical manner would be very high, the computational cost in the hierarchical approach is logarithmic in the number of leaf nodes

Now to present a demo

- Using K=9, L=6
- Precomputed vocabulary tree, database vectors, node weights