EQ2425 Analysis and Search of Visual Data

EQ2425, Project 2

Yuzhou Liu yuzhoul@kth.se

Yage Hao yage@kth.se

October 1, 2021

Summary

In this project, our first task is to o extract a few thousands SIFT features from each database image, based on the 50 building figures. Our second task is to build the vocabulary tree with the SIFT keypoint descriptors obtained above. Client database of 50 images is used for querying. Our scoring system for object retrieval is TF-IDF (term frequency inverse document frequency) scoring.

1 Introduction

1.1 Obejective

In this project, we will build a visual search system and evaluate it. Use SIFT descriptors as image features and use vocabulary trees as the database structures. Using query images, the goal is to recognize buildings. That is, the visual search system retrieves the objects that are most similar to the query object as represented by the query image. TF-IDF (term frequency inverse document frequency) score is used for object retrieval.

1.2 Image Feature Extraction

The first part of the project is to extract a few thousands SIFT features from each database image. Scale-invariant feature transform is a computer vision algorithm used to detect and describe local features in images. It looks for extreme points in the spatial scale, and extracts its position, scale, and rotation invariants. This algorithm was published by David Lowe in 1999 and summarized in 2004.

1.3 Vocabulary Tree Construction

Vocabulary tree is a compact method for image retrieval. The realization of this algorithm is basically divided into three steps, which highly depends on other computer vision technologies, such as SIFT features. The first step is to build a k-means tree using filter descriptors. The leaf nodes of this tree contain a "packet" of filter descriptors. The second step is to build an image database using the vocabulary tree we built in the first step. We can think of this process as quantizing the image into a vector space. Then the third step is to query the image based on the image database.

2 Problem Description

2.1 Image Feature Extraction

Inside the data2 compressed file, there are two folders. The images in the server folder are the database images. Each building object appears in three images. There are 50 building objects in total. We need to build a vocabulary tree with scoring information using the database images. The images in the client folder are used as query images. Each building object appears in one query image. The 50 images are taken from the same building objects in the database.

We can notice that some images in the database folder are missing or added. According to the guidance, there should be three similar images of the same building. Therefore, we need to process these database images, three for each building. According to requirements, thousands of SIFT features must be extracted from each image. These features are extracted through professional OpenCV functions, and each image retains only 1000 strongest features. This number is determined by trial and error. Because it will affect the calculation time and normal operation of our Vocabulary Tree Construction.

2.2 Vocabulary Tree Construction

After we extract SIFT features from both server objects and client objects. We want to build the visual search system by constructing a vocabulary tree using hierarchical k-means algorithm.

The basic idea of hierarchical k-mean algorithm is: first run k-means algorithm on the original data, then for each resulting clusters, run k-means algorithm on them recursively. For example: assume we are doing a two-level tree k-means, the first level is a 20-class k-means, and for each second level branch, a 10-class k-means is done. Then, we have done a k-means with totally 20*10=200 classes. When quantizing a data point, we first quantizing at first level, that is, projecting the data point to one of the 20 classes. After that, quantizing at the second level at the nearest branch.

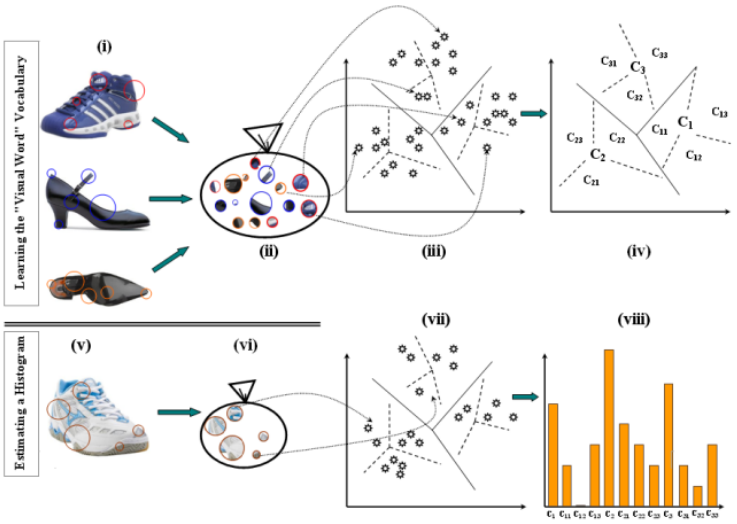

Figure 1: vocabulary tree

Figure 1 shows how vocabulary tree works for object retrieval. There are basically three steps to implement this algorithm. The first step is to build a k-means tree using sift descriptors. The leaf nodes of this tree contain a "bag" of sift descriptors. The second step is to build a image database using the vocabulary tree, that is quantizing an image into a vector space. Then the third step is to query the image against the image database.

3 Results

3.1 Image Feature Extraction

We found that the fewer the number of key points to be extracted, the more likely it is to throw a "zero score" error in a deeper tree. For example, extracting 500 SIFT key points from each image is not enough to build a tree with a depth of 10 levels and 5 branches per level, but our algorithm can use exception handling methods (when empty clusters are found) to overcome this The question will store an empty TF-IDF score list).
All of our features are extracted in the sift folders. We combine the features of the same building and remove similar descriptors. This removal is performed after the SIFT matching algorithm is executed based on the nearest neighbor distance between the first image descriptor of the same document and the other two corresponding images. The selection threshold of the ratio distance is 0.8, this threshold will be more suitable for our algorithm. Once the same features of the building are grouped together, they are indexed to create the tree later. Using these parameters, the average number of features extracted for each building (document) is 1073.39. With the name 'obj_number.npy', all of the features are saved in folder.
In the same way, 1000 SIFT key points are detected for each of the 50 images in the query database and saved separately. Intuitively, the average number of features extracted from each database image should be 1000, but in our example it is 1001.86. This strange value may be due to some errors in the OpenCV SIFT algorithm when detecting a fixed number of key points.

3.2 Vocabulary Tree Construction

(a) In each node in the vocabulary tree, we stored the cluster centroid vectors as well as all new clusters' information, including all sift-feature descriptor vectors in the children cluster, the object where the feature extracted from.

(b) We stored the TF-IDF scores for every node in the tree, including the leaf nodes, which will be used during querying. To compute the TF-IDF score for a node, we need to know the number of occurrence of a building in the corresponding cluster, the number of descriptors in the cluster, total object numbers and the number of objects that appeared at least once in the cluster.

(c) We applied the KMeans function in sklearn.cluster to build our vocabulary tree hierarchically. It is mainly realized through function 'recursive' and 'hi_kmeans'. For detail implementation, please check 'main.py'.

3.3 Querying

(a) The recall rate results with different 'b' and 'depth' values are shown below:

- **b=4, depth=3:** Top 1 recall rate is 0.02; top 5 recall rate is 0.1.

- **b=4, depth=5:** Top 1 recall rate is 0.12; top 5 recall rate is 0.24.
- **b=5, depth=7:** Top 1 recall rate is 0.6; top 5 recall rate is 0.86.

(b) With b=5 and depth=7 set, the results of querying on 50%, 70% and 90% features are shown below:

- **50% of query features:** Top 1 recall rate is 0.66; top 5 recall rate is 0.84.
- **70% of query features:** Top 1 recall rate is 0.64; top 5 recall rate is 0.84.
- **90% of query features:** Top 1 recall rate is 0.6; top 5 recall rate is 0.88.

(c) When consider calculating the number of Euclidean distances without hierarchical clustering, we need to compute between all descriptors for every server images. With a hierarchical clustering, each cluster contains similar descriptors and we can only compute the Euclidean distance from the cluster centroid.

3.4 Bonus

We believe that weights can be added to improve the performance of the visual bag of words, and the general TF-IDF formula can be improved. Different combinations of initial node and b d parameters should generate different trees. At the same time, TF-IDF scores are accumulated for different possible paths in the tree. According to the conclusion, we believe that the more features are extracted, the larger the tree, the finer the classification, and the better the result.

4 Conclusions

According to the comparison of data, extract more features, the larger the tree, the finer the classification, the better the result. Moreover, through SIFT and word tree, our classification effect is more satisfactory.

Appendix

Who Did What

Yuzhou Liu is responsible for the "Image Feature Extraction". In detail, he implemented question 2's code and wrote the report's 'Summary', 'Introduction', '2.1', '3.1' and 'Conclusion' parts.
Yage Hao is responsible for the "Vocabulary Tree Construction" and "Querying". In detail, she implemented question 3's and 4's code and wrote the report's '2.2', '3.2', '3.3' and '3.4' parts.

References

[1] Markus Flierl, *EQ2425 Analysis and Search of Visual Data*, Lecture Slides, 2021

[2] Wikipedia, *Bag-of-words model in computer vision*, Available: https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision, Last edited: August 21, 2021