Visual Search System

EQ2425, Project 2

Author 1
remich@kth.se

Author 2
sebcas@kth.se

September 29, 2021

Summary

This project is about the building and the evaluation of a hierarchical tree structure. The purpose of this report is to show our work and our considerations about the implemented system.

Furthermore, we will deal with a weighting method called TF-IDF that allowed us to better consider each outcome.

Moreover, to choose the accuracy of the solution, both top-1 and top-5 recall will be implemented. We will then show how, according to the logic, the top-5 recall gives us better results.

Not all the outcomes we obtained are trivial to analyze, due to the intrinsic properties of image feature extraction methods. For example, we faced the issue of different numbers of features extracted from different pictures with the same threshold values. This led to a non-homogeneous descriptors representation in our database. As a result, some query images have been associated with the images in the database with more features instead of the correct ones. As a possible solution, without manipulating the database, we extracted a high number of features. In this way the issue is not completely solved, because every image will have proportionally an augment of descriptors, but associated with a deeper tree, we were able to better represent each image in the leaf nodes.

1 Introduction

The goal of this project is to build a visual search system and evaluate it. To do so, SIFT descriptors are used to create a Vocabulary Tree as a database structure. This database is composed by 3 images of the same building taken from different perspectives. Some of the buildings have only 2 images, while one has 4. This create a non homogeneous database.

Once created the tree, it is fed with query images to evaluate the performances of the structure. It outputs how likely the current image is represented by certain objects in the database descriptors. To evaluate so, TF-IDF (term frequency inverse document frequency) score is used.

SIFT stands for Scale Invariant Feature Transform, meaning that it is robust against manipulations such illumination, scale, perspective, and occlusions. These features are discriminated for location, orientation, and scale, based on a comparison of Gaussian filtered and scaled images. Moreover, it is known to be a real-time technique. More information can be found in the official article [4].

A Vocabulary Tree is an hierarchical structure that solves the computational complexity of a single search command in a big database. To discriminate the decisions a concept for object recognition is taken into account, the clustering.

Given a distribution in the features space, clusters represent the "centers" of a subset of features. Given the number of branches in the tree, the clusters are generated with a MATLAB function called *kmean*. Later in the report we will face the problem of an unbalanced tree, due to the low amount of feature concerning the nodes and leafs generated [2].

To "weight" these decisions, a score called TF-IDF is used. We followed the general hints provided during lectures and guidelines in the paper called "Scalable Recognition with a Vocabulary Tree" [3]. Approximating probabilistic and statistical concepts, given the features space, it is possible to express a weighting function. To do such it has been considered the number of occurrences of a cluster in a query object, the total number of clusters in a query object, the number of query objects that contain the cluster, and the total number of query objects.

2 Problem Description & Results

The project is divided into 3 main sections.

The first part is about extracting the SIFT features from each database and query image. Following the hints provided by the professors, we set *edge* and *peak_threshold* in order to get an average of features per building of around 3,470 in the database descriptors. Given a total amount of around 517,000 features, while the query descriptors are around 167,000 with an average of 3,340 per query building (1)(2). Indeed, as show in the images below, the 2 histograms are similar given that the query images are taken from the database.

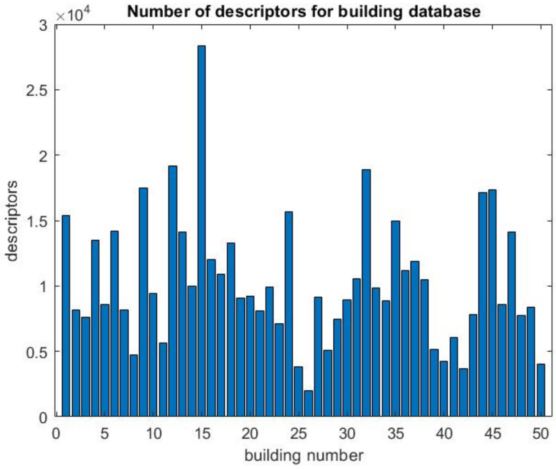


Figure 1: Number of SIFT features per building of the database images.

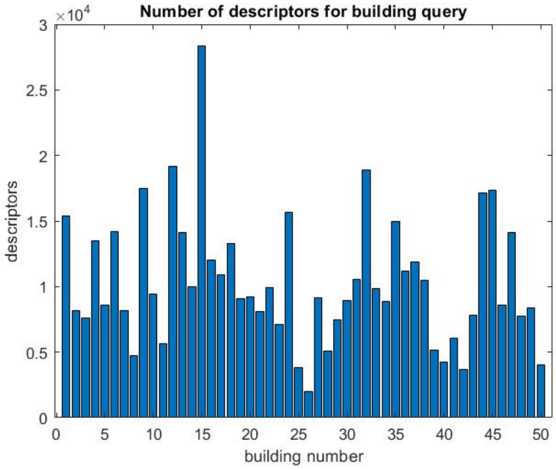


Figure 2: Number of SIFT features per building of the query images.

The second part of the project is about Vocabulary Tree Construction. As said before, it is based on a hierarchical k-means algorithm. To control the structure, "branch" and "depth" values are given. The function responsible to do such is called "hi_kmeans". It is implemented to be recursive and the structure is saved in nested cells. Depending on the number of features given to create the tree and depending on the structure of the tree we can obtain an unbalanced configuration. In this case, reasonable considerations have to be done in order to have enough features to generate the next clusters.

In the evaluation step, when a query image is fed into the tree, it is important and necessary to evaluate the centers of these clusters. Then the decision is taken through the euclidean distance between a single feature of the query and the current cluster center. Then the chosen node is followed to obtain in the end the optimal path.

In the final node, the score TF-IDF is evaluated for each building image to retrieve the best value. Indeed, it will tell us if there is a match.

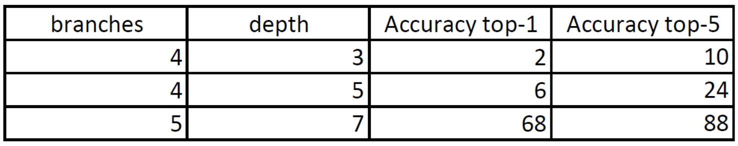
The third section is about Querying. As said before, for each building query image, the descriptors extracted are sent one at a time. Reaching the final node, the score vector is saved and further summed with all the other score vectors of the other descriptors. In the end, the resulted score vector is ranked according to its best value. Later on, the evaluation will be done over the 5 best values, and we will see how the accuracy will increase.

As required in the project, we generated 3 different tree structures.

The first one is composed of 4 branches and 3 depth levels. We obtained 2% on top-1 recall and 10% on top-5 recall.

The second one is composed of 4 branches and 5 depth levels. We obtained 6% of accuracy on the top-1 recall, and 24% on top-5.

The last tree structure composed of 5 branches and 7 depth levels, gave us the best values as expected. With a 68% of accuracy on top-1 recall, and 88% on top-5 recall.

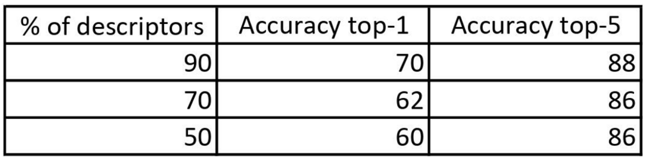


branches	depth	Accuracy top-1	Accuracy top-5
4	3	2	10
4	5	6	24
5	7	68	88

Figure 3: Levels of accuracy in the different tree structures.

As we can observe in figure (3) the accuracy depends mainly on the depth of the tree. The higher the levels the better the score. We kept the same number of features in all the 3 cases, resulting to burden on the smaller trees given that for each leaf node the number of the building represented were more.

The last requirement is about verifying the robustness of the architecture feeding the tree with respectively 90, 70, and 50% of the query features. Our outcomes resulted to be as we expected, but with the exception of the 90% case, in which the top-1 recall gives a better score.



% of descriptors	Accuracy top-1	Accuracy top-5
90	70	88
70	62	86
50	60	86

Figure 4: Levels of accuracy with respect to different amount of query descriptors.

Hierarchical clustering allows for an increment in the speed of computational time. At each time I will perform an evaluation using the euclidean distance with just the total number of nodes, or clusters, present in the current level, for each level of the tree. While instead, without a hierarchical tree, the comparisons necessary using the euclidean distance have to be performed concerning all the clusters. To give a quantitative example, with a tree with 2 branches and 2 levels of depth, in the first case we will compute the comparison 4 times, while in the second case 6 times. This behavior is exponentially increasing as the tree structure grows.

3 Bonus

To further improve the solution, a method called RANSAC can be implemented, as shown in this paper [1].

Otherwise, as we implemented, improving the number of features extracted and the depth of the tree, we were able to reach an accuracy of 86% on top-1 recall, and of 98% on top-5 recall. With this configuration, the buildings represented in the leaf nodes are never more than 4 each.

Anyway, with this method, we can encounter a very tricky issue, the over-fitting. If we force the generation of too many features in one single image, as a result, we can obtain features that in reality don't exist, performing a wrong evaluation.

4 Conclusions

In conclusion, analyzing the vocabulary tree algorithm and applying the TF-IDF weight method we noticed the different characteristics of the methodology. In primis, the depth of the tree is very representative of the final accuracy. In particular with an extreme number of nodes the accuracy increases, but the different clusters go to represent a very low number of features. In secundis, when reducing the features of the query the accuracy of the algorithm goes to decrease, but in the case of 90% a better accuracy of the query is noticed, and this is going to represent a problem of over-fitting. We also noticed that each different image of both the database and the query varies a lot in the number of features while keeping the threshold values constant.

Appendix

Who Did What

The work was divided equally based on the previous knowledge of the components and the work was organized and developed in parallel.

Code Explanation

In this section, we indicate which function the various files of the code have.

averageDescriptorsCalculation: in this file we calculate the average number of descriptors.

hi_kmeans: in this file we create the function for create the tree.

main: in this file we compute the tree.

Making_query: in this file we compute all the function for make the queries.

query: this function make the query for a single descriptor (not used).

query_node: this function make the query for a single descriptor and return the best score building.

*query_node_top*5: this function make the query for a single descriptor and return the associated 5 best building.

SIFT_Database: compute all the descriptors of the database.

SIFT_Query: compute all the descriptors of the query images.

References

[1] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: 24.6 (1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[2] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

[3] D. Nister and H. Stewenius. "Scalable Recognition with a Vocabulary Tree". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 2161–2168. DOI: 10.1109/CVPR.2006.264.

[4] Sivic and Zisserman. "Video Google: a text retrieval approach to object matching in videos". In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1470–1477 vol.2. DOI: 10.1109/ICCV.2003.1238663.