Miguel Alcón Doganoc
Advanced Data Structures
June 21, 2019

Vantage Point Trees

Introduction

In this document I am going to compare two implementations of the vantage point trees (*VPT*), one written in *Python* [1] and the other one in *C++* [2]. With this, we can compare which of the programming languages perform better with this kind of data structure. The objective of this work is to compare the times that each implementation lasts to build the tree and to search the point neighbors of a queried point.

Experiment

In order to compare both implementations, I designed the following experiment. First, in *C++* I generated randomly a vector of 3-dimensional points of size k, with real values between 100 and -100. Then, I created the *VPT* with the vector. Finally, I searched n neighbors of a random point, 100 times. I saved the vector of points, the searched points, the results, and the execution times in a file, to reproduce the same experiment with the other implementation and compare the results.

With the *Python* implementation I did the same, but instead of generating the data, I read from the saved files.

Results

I executed the experiment with different values for the parameters, and here you have the results:

Parameters		Python Time (s)		C++ Time (s)		Python Results	
k	n	Build	Search (avg.)	Build	Search (avg.)	Similarity (%)	Lost points
100	10	0.00954	0.00077	0.00086	0.00003	89	61
1000	100	0.11866	0.00609	0.01239	0.00022	89	353
10000	1000	1.56576	0.13238	0.18959	0.00207	51	3558
100000	100	18.72097	0.01371	2.38338	0.00039	97	0
100000	10000	20.09716	7.28623	2.56153	0.02612	0	84459

Table 1: Results of the different experiments

As you can see in table 1, the *C++* implementation performs a lot better than the *Python* one, in both time and quality of the found neighbors. The *Similarity* field shows the percentage of neighbors that are equal comparing the results of the search in *Python* and *C++*. While k is small or the difference between k and n is big, the *Python* algorithm founds good results. But when it does not happen, *Python* can differ totally with *C++*. Furthermore, sometimes, the cause of the inequality of the neighbors is that the *Python* implementation returns fewer neighbors than the requested ones. This is seen reflected in the *Lost point* field, which shows the total number of neighbors the *Python* implementation lost when searching. Because of that, and because with a small k both implementations behave equally, I assume that the *C++* results are the correct ones, so *Similarity* can be seen as accuracy of the *Python* implementation. Regarding the time, numbers speaks for themselves.

References

[1] Rickard Sjögren. vptree. https://pypi.org/project/vptree/, 2017.

[2] (GitHub) pderkowski. vptree. https://github.com/pderkowski/vptree, 2017.