Lab: Sampling spatial data using Quadtrees

Auke-Dirk Pietersma

November 30, 2018

1 The Problem

GIS databases can consist out of thousands of geometries, and even though the database itself has no problems in fetching these, sending them to the client often leads to bandwidth problems. The latter is the case since more and more queries are performed through mobile devices. One solution, on which we will focus during this lab, is to subsample the result of the query. Our goal is to improve on naive sampling (first n elements) as seen in figure(5), such that it better resembles our total set (figure(1)).

2 Lab

The student needs to improve on the provided code such that sampling through Quadtrees is possible. In the final program there should be two variables controlling the final sampling:

- (Quadtree) quadtree : add a QuadTree with certain depth
- (Plotter) quadlevel: at which level the points should be sampled.

In order to achieve the above you will first need to implement the following:

KDTree	def closest(self, point, sidx = si.StorageIndex())
QuadTree	def recurse(self,bbox, depth)

3 The role of each class

3.1 Documentation

The classes Database, QuadTree, KDTree, BoundingBox come with full documentation and tests. The documentation can be viewed in code, or through ipython: All the classes contain a main entry point such that can tested/viewed independently.

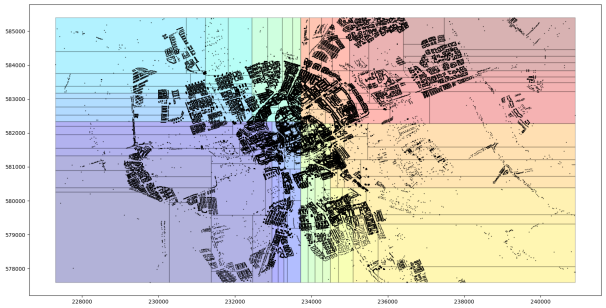

Figure 1: Original

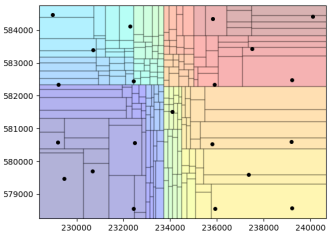

Figure 2: depth 4 and level 3

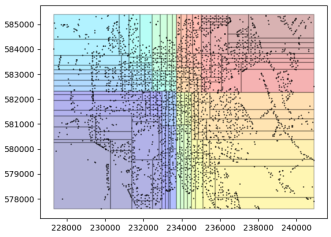

Figure 3: depth 8 and level 7

Figure 4: Sampling using Quadtree

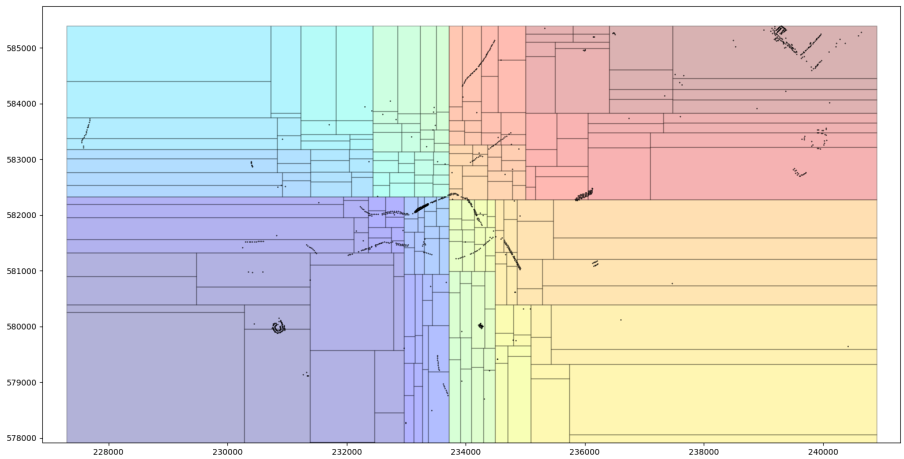

Figure 5: Naive sampling selecting the first N items

```
python kdtree.py
{'index': 1, 'depth': 0, 'partition': 5, 'axis': 0}
{'index': 2, 'depth': 1, 'partition': 4, 'axis': 1}
{'index': 3, 'depth': 1, 'partition': 2, 'axis': 1}
{'index': 4, 'depth': 2, 'elements': array([1, 2]), 'axis': 0}
..
```

4 Explaining the steps

4.1 Read all the classes documentation

Before starting to implement the missing functions take some time to read through the classes: Database, QuadTree, KDTree, BoundingBox.

4.2 Implementing the QuadTree

The QuadTree is a recursive datastructure, where in each new iteration, the current BoundingBox needs to be split into 4 equal sized BoundingBoxes. These newly formed boxes should fill the exact space as their predecessor.
To test your QuadTree:

```
python3 plot_kdtree.py --quadtree 4 --quadshow True
```

and you should see an 8x8 chessboard pattern.

4.3 Implementing the KDTree

The KDTree is used to partition the datapoints into several BoundingBoxes, after which the tree can be queried to obtain a collection of these datapoints. The range-query, find all BoundingBoxes/Leaf-nodes that intersect with a given BoundingBox and return their datapoint collection, is already implemented. However the closest function is still left for the student. The closest function will traverse through the tree similar to that of the rquery instead it operates on a single point $[x, y]$, and can thus only traverse left or right. After reaching the leafnode, closest should return all the datapoints within the leafnode.

4.4 Using the QuadTree depth to subsample the KDTree

In the final section we will need to add/update the field 'quad' for every record inside our database. This field represents whether or not a record should be sampled. For example: if we wish to sample all records up to quad-level 4, then we simply check whether or not a record his 'quad' field is lower or equal to 4 (The higher the level the more data).
The student needs to add code in 'plot_kdtree.py' at:

#To be implemented by the student

The following steps are needed:

- initially set the quad field for all records to the max quad-level
- iterate through all levels of the quadtree in reverse order. (maxdepth to 0)
- for every box inside that level obtain its centroid.
- use this centroid to obtain the list of closest points inside the KDTree
- find the closest point inside that list
- update the 'quad' field corresponding to that point to the current quad-level

5 Final result

Executing the final program with different values of 'quadtree' and 'quadlevel':

```
python3 plot_kdtree.py
        --filename ../data/addressen_groningen/adressen_groningen.shp
        --bbox-depth 8
        --max-depth 9
        --plot kdtree-bb
        --quadtree 8
        --quadlevel 4
```

should result in sampling as in figures(2,3).