

R-Trees for tracking reconstruction

Albert Pernía Vázquez

La Salle - Universitat Ramon Llull

October 10, 2018

Previous works

- ▶ A reconstruction algorithm that expands the tracks by checking the closest hits to them.
- ▶ Problem: hits were stored in a linear data structure.
 - ▶ All the hits belongs to a 3D coordinate system. How to organize them?
 - ▶ We had to traverse all the structure to find the K nearest hits.

Previous works

- ▶ A reconstruction algorithm that expands the tracks by checking the closest hits to them.
- ▶ Problem: hits were stored in a linear data structure.
 - ▶ All the hits belongs to a 3D coordinate system. How to organize them?
 - ▶ We had to traverse all the structure to find the K nearest hits.

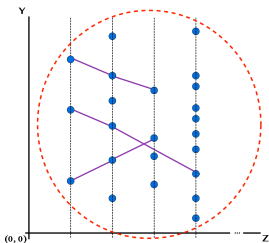


Figure 1: Using the linear data structure

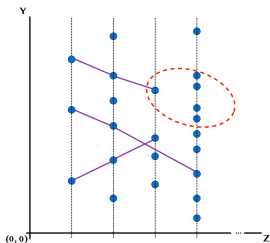


Figure 2: We want to check the closest hits for a line track

First solution

- ▶ Use an existing API for spatial data indexing that implements a **R-Tree**.
- ▶ The API is implemented in C++ but uses Cython to be usable with Python with the power of the C++ language.

R-Tree

- ▶ Based on B-Trees.
- ▶ Inner nodes contains M entries with M regions.
- ▶ Leaf nodes contains M hits.
- ▶ KNN searches.

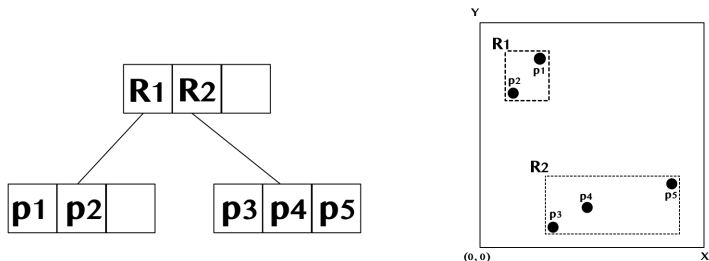


Figure 3: Example of a R-Tree with $M = 3$

R-Tree: Insertion

- ▶ Regions increase to cover all the hits.
- ▶ The region implying the minimum increase is chosen to cover the new hit.

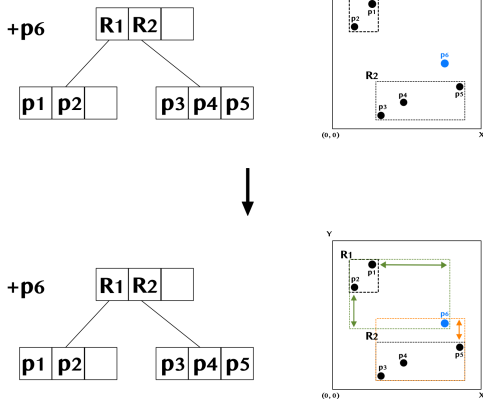


Figure 4: Insertion example

R-Tree: *Overflow*

- ▶ The node where the hit should be stored is full.
- ▶ It is needed to perform a node **split** (figure 5).

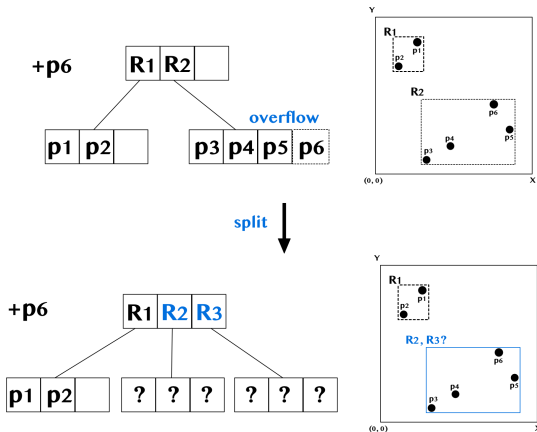


Figure 5: Hits of the node split must be distributed over the new two resulting nodes

R-Tree: *Split*

- ▶ Hits distribution must be as optimal as possible (figure 6).
- ▶ **Bad distribution** (figure 7) will imply **slow searches**.

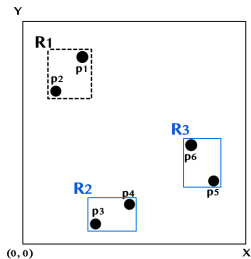


Figure 6: Perfect distribution

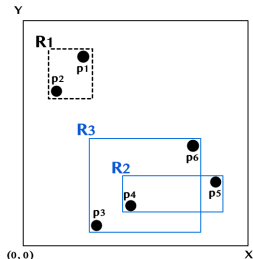


Figure 7: Bad distribution

R-Tree: Split algorithms

- ▶ There are many split algorithms:
 - ▶ Exponential
 - ▶ Quadratic
 - ▶ Linear

- ▶ The R-Tree API uses quadratic split.

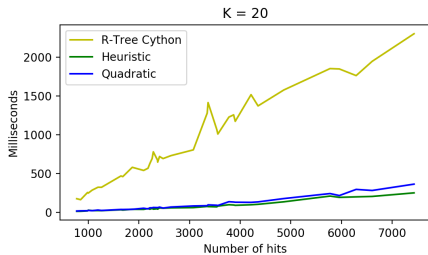
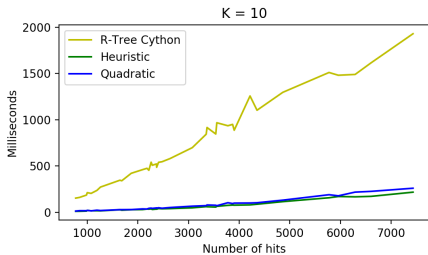
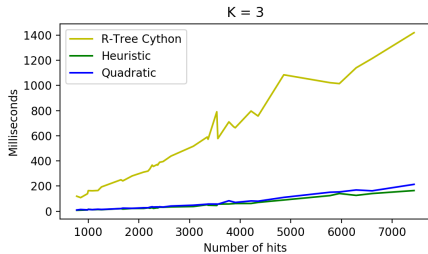
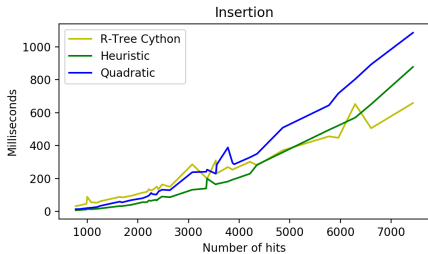
Second solution

- ▶ R-Tree API presents some **problems**
 - ▶ We cannot add/change the split algorithm being used.
 - ▶ Generic structure.
 - ▶ Last update was in 2011.

Second solution

- ▶ R-Tree API presents some **problems**
 - ▶ We cannot add/change the split algorithm being used.
 - ▶ Generic structure.
 - ▶ Last update was in 2011.
- ▶ **Next step:** create a R-Tree project in C with a flexible implementation.
 - ▶ Add custom split algorithms.
 - ▶ Change other properties.
 - ▶ We tried it with VELO datasets using the quadratic split and a custom heuristic algorithm.

Results



Conclusions

- ▶ Insertion is not always faster than the Python API.
- ▶ Significant decrease of KNN execution time compared to the Python API.
- ▶ Easy adaptable project: add new split algorithms with no need to understand all the implementation.

Questions?