

Indexicon: A Spatial Indexing Library

Panagiotis Simatis¹ Panagiotis Bouros² Nikos Mamoulis¹

¹University of Ioannina, Greece ²Johannes Gutenberg University Mainz, Germany

POSTER

S-194

ACM SIGSPATIAL '26 · Riverside, CA

The problem

Spatial indices power GIS and data analysis, yet open-source options are **fragmented**: single-index libraries, heavy dependencies, missing operations.

Library	Indices	Bulk	Insert / Delete	Code
Indexicon	R-tree, Quad-tree, Oct-tree, KD-tree	✓	✓ / ✓	Low
Boost Geometry	R-tree	✓	✓ / ✓	High
CGAL	Orth-tree, KD-tree	✓	rebuild / lazy	High
GEOS	R-tree, Quad-tree	R-tree only	Quad-tree only	High
PCL	Oct-tree, KD-tree	KD-tree only	Oct-tree insert only	High
Nanoflann	KD-tree	✓	✓ / lazy	Medium
Vigier	Quad-tree	-	✓ / ✓	Low

No existing library combines a **complete feature set**, **low code complexity** and **state-of-the-art performance**.

Indexicon

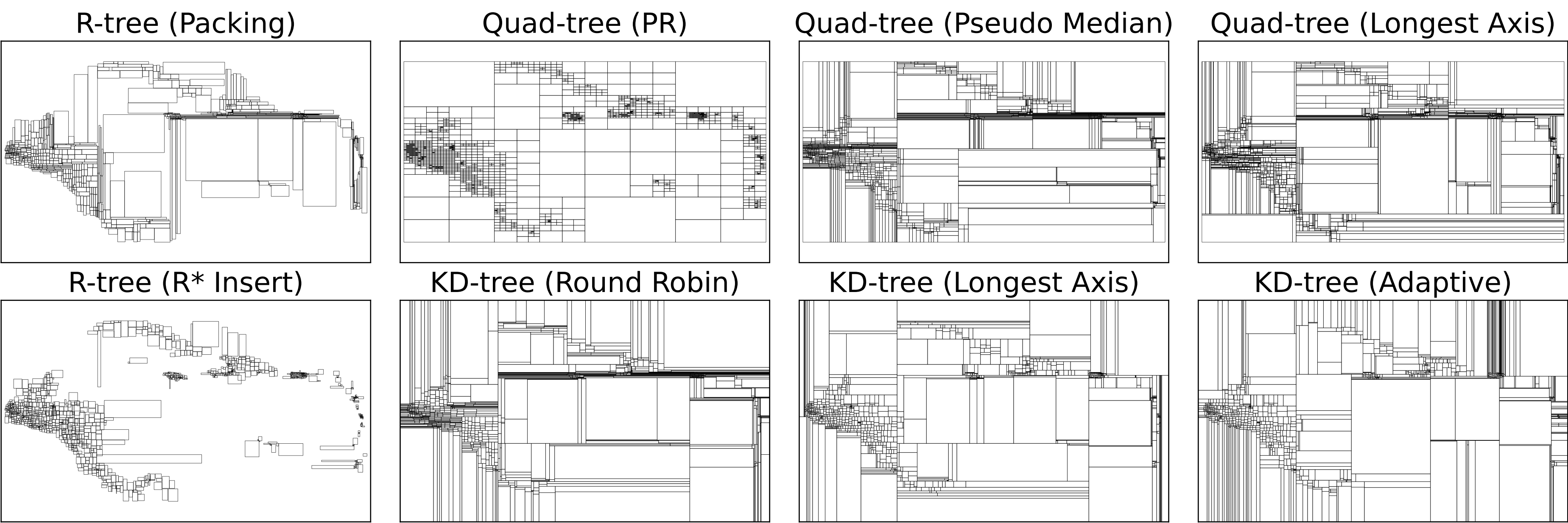
- **In-memory, one file per index**: header-only C++ templates, no dependencies beyond the standard library.
- **One API for all**: bulk-load, insert, remove, range and kNN queries.
- **Points and boxes** in 2D, 3D and any dimension.
- **Python too**: `pip install indexicon`, NumPy in, NumPy out.

```
#include "rtree_point.hpp"
using Tree = RTree<size_t, double, 2, 128>;

Tree tree(data.begin(), data.end()); // bulk-load
tree.insert(p, id);
tree.remove(p, id);
tree.query(box, back_inserter(hits));
tree.knnQuery(q, 10, back_inserter(nn));
```

Indexes and partitioning strategies

- **R-tree**: top-down packing, R*-tree insertion with forced re-insertion, deletions re-insert orphaned entries.
- **Quad-trees**: PR Quad-tree, pseudo-median Quad-tree, longest-axis Quad-tree, MX-CIF Quad-tree for boxes, Oct-tree in 3D.
- **KD-tree**: round-robin, adaptive, longest-axis splits, true deletions with leaf merging (no tombstones).
- **Queries**: subtrees fully inside the query are reported without comparisons.



Faster queries on six real datasets

Mixed workload: bulk-load 80%, insert 20%, delete 5%, then 1,000 range queries covering 0.01%, 0.1% and 1% of the space, on datasets of up to 113M points. Query time is on a log scale, lower is better.

R-tree vs. Boost

1.35×

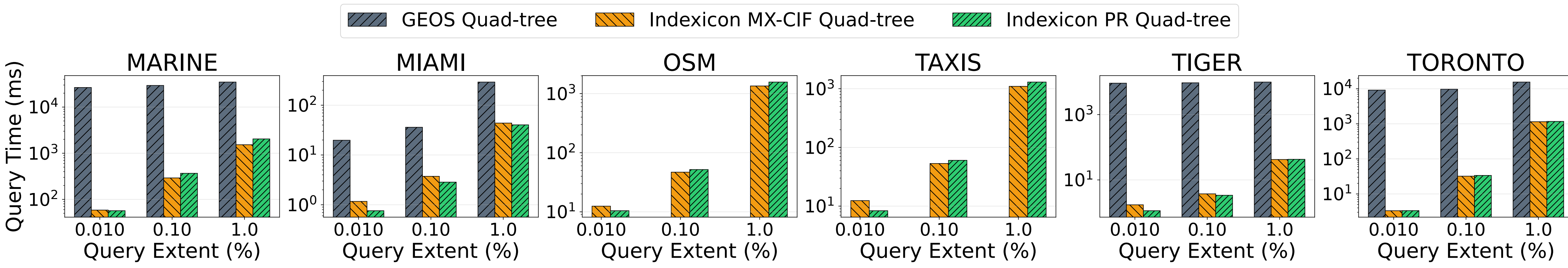
faster on OSM 1% queries, on average 1.28× faster inserts



Quad-tree vs. GEOS

17×

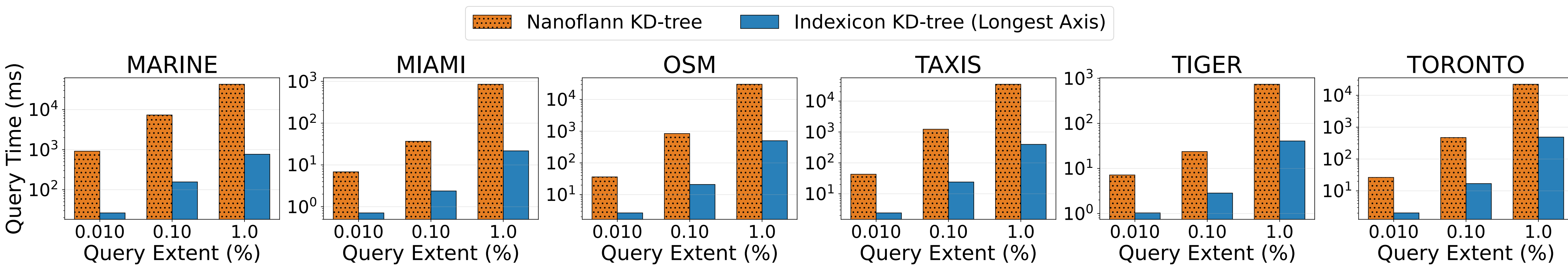
faster on MARINE 1% queries, on average 99× faster deletes



KD-tree vs. Nanoflann

60×

faster on OSM 1% queries, on average 1.35× faster construction, no tombstones left behind



Get the code

`pip install indexicon` or copy a single .hpp file

Next: More indexes, spatial joins, concurrency, and disk support. Contributions welcome!



Code (GitHub)



Full paper (arXiv)