# The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles

Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger

Abhishek Srivastava
Student ID: 861307778

February 2, 2017
**CS 236**, Winter 2017

---

**The problem:**
This paper presents problems with the existing R-Tree spatial indexing method such as high disc access, bad storage utilization, not generalized for different data distributions which lead to less efficient queries responses.

**The contribution:**
The authors propose an 'R*-Tree' index structure that can represent multi-dimensional point and spatial data, supports dynamic operations, needs no periodic reorganization and outperforms all R-Tree variants. The paper discusses difference in the implementation with respect to the existing R-Tree and perform comparisons with different data distributions and different type of queries.

**The method:**
The R*-Tree is extension of R-tree but differ in some respects. R*-tree focus on these heuristics to improve overall efficiency:

- Area covered by rectangles should be minimized.
- Overlap between rectangles should be minimized.
- Margin of rectangles should be minimized.
- Storage utilization should be improved.

Intuition behind these heuristics are that it will lead to less dead space, fewer traversing paths possible, reduction in bounding area of non-leaf nodes and decrease in query cost which are improvement factors. Due to these implementations of insertion and deletion operation changes in some regards. For searching subtree parameters such as merging and overlap are also taken into consideration. Splitting a node also changes and data is sorted along each axis and split is done based on the goodness of each distribution for different set of entries, expected runtime for this is O(MlogM).

The main difference from R-tree is Forced Reinsertion operation. Splitting nodes can leave under-filled data, so these nodes are deleted and orphaned nodes are re-inserted distributing entries into different nodes. This resulted improvement in storage utilization. This is done during **Overflowtreatment** on the nodes and propagates upward.This results in higher CPU cost during insertion but it help improve search and deletion queries.

**Comments:**
The paper presents a novel robust indexing structure that outperforms all existing R-tree variants in each comparison test. It also provide high gain in point access queries and better spatial join queries. However, it also gives rise to a few problems:

- The primary focus of R*-tree was high dimensional points and 2-D spatial data but there is no evidence given how it will perform in high dimensional spatial data. Insertion can provide high overhead because of Forced-Reinsertion.
- Same issue as R-tree like spatial structures are represented in the rectangle boundaries form so it wont be able to uniquely present objects of different shapes who have same rectangle boundaries.