

Preliminary ANN/Objectivity tests

This test shows the amount of CPU time per object needed to build a tree and search through it versus the number of objects in the database. The data storage is managed by objectivity.

ANN can build two kinds of trees: kd^1 and bd^2 (box decomposition) tree. The former divides the space recursively splitting the sample along a single dimension; the latter, in addition to this, divides the space in concentric boxes: the bd -tree is optimal when highly clustered distributions of points occur. During this building phase, the recursive subdivision of the space stops when the bucket size i.e. the maximum allowed number of points in a box, is reached; this box is called a *leaf-node*. For this test the bucket size is 10 points.

From figure 1, building a bd tree takes approximately twice as much as building a kd -tree, because of the larger amount of data analysis involved (see *kd-no* and *bd-no* trends; *no* stands for “no search performed”). The two data structures are time-wise equivalent during the search phase, instead (see *kd std* and *bd std* trends; *std* stands for “standard search performed”).

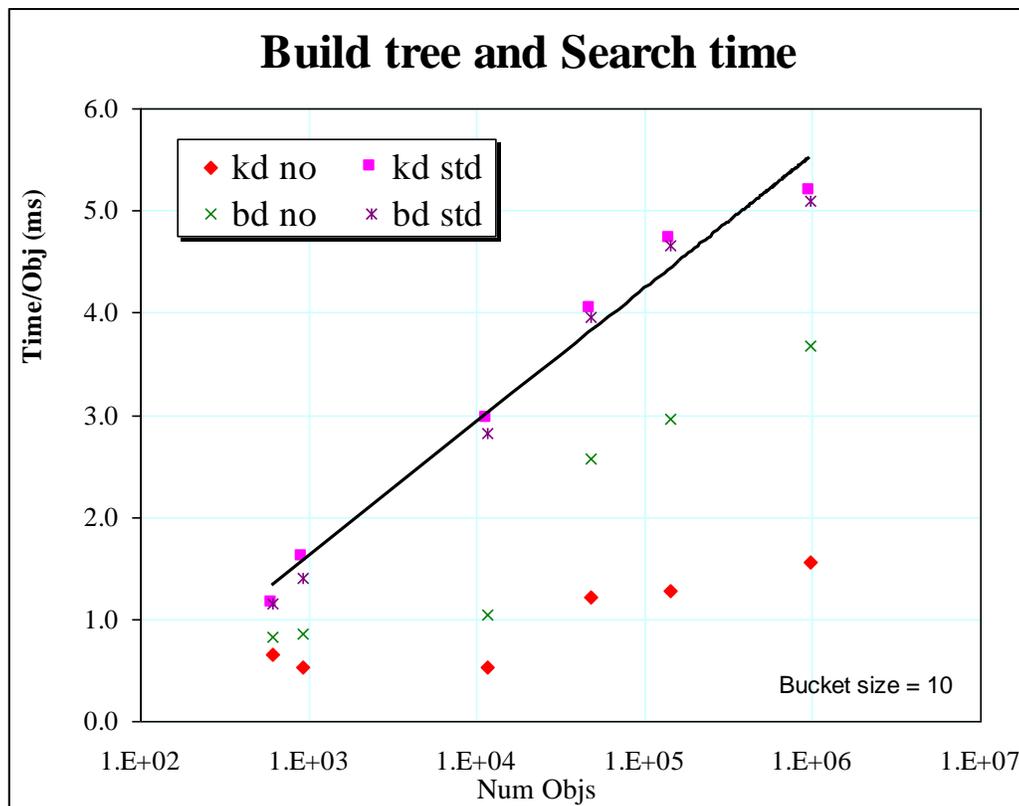


Figure 1

I expect the search time to be $O(N \text{ Log} N)$ i.e. Time/Obj to be $O(\text{Log} N)$. This trend corresponds to a straight line in the plot of Figure 1, having the X axis a logarithmic scale. This hypothesis is true with a squared correlation coefficient (R^2) of 0.98. For

larger database size I expect a slow down caused by Objectivity swapping data in memory.

Figure 2 shows a comparison between the 3 different kinds of search algorithms available: standard³ (*std*), priority⁴ (*pri*) and brute force (*bf*) search. The first is a recursive search of the tree; the second searches the tree following a priority queue of the leaf-nodes sorted by increasing distance from the query point; the third is the classical $O(N^2)$ search algorithm.

Standard and priority searches are essentially equivalent. It is immediate the advantage of using a tree to perform the search with respect to a brute force approach.

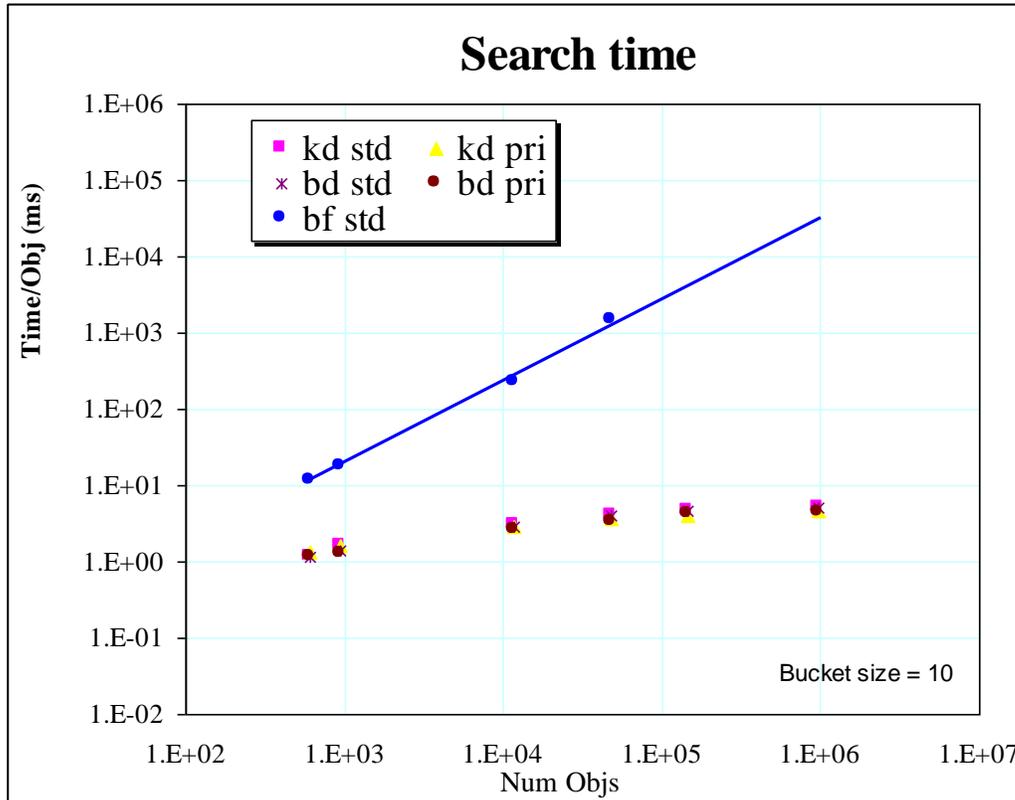


Figure 2

The system used for the test is a Silicon Graphics Origin 2000, MIPS R 12000/12010, with 1GB of Ram, 32KB of cache and with a 138 GB internal disk xfs mounted.

References

¹ J.L. Bentley. K-d trees for semidynamic point sets. In *Proc. 6th Ann. ACM Sympos. Somput. Geom.*, pages 187-197, 1990

² S. Arya, D.M. Mount, N. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithm*, pages 573-582, 1994.

³ J.H. Friedman, J.L. Bentley, and R.A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209-226, 1977

⁴ S. Arya and D.M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithm*, pages 271-280, 1993.