Testing k-d tree implementations

Jure Slak, E6 Lab
November 19, 2018

1 Premise

We are currently using ANN [1] and `libssrckdtree` [2] in Medusa for static and dynamic k-d trees, respectively. This choice can be reevaluated for multiple reasons:

- ANN is rather old and some unnecessary bookkeeping is needed to make it work
- ANN only supports `double`
- `libssrckdtree` depends on Boost and is the only part of Medusa that requires boost. Removing this library would cut a rather heavy (and the last unneeded!) dependency.
- It would be nice to only use a single library

Libraries `nanoflann` [3] and `spatial` [4] provide both static and dynamic trees (to some extent) and are more modern. `nanoflann` is a fork of FLANN [5] that optimizes execution time and memory usage. It builds an index over an existing data structure, and supports search for k nearest neighbors and radius search. For dynamic clouds adding points is supported normally (provided your underlying structure supports it) while removing is done lazily (and does not decrease memory usage). It also allows the user to specify how many points are kept in the leaf nodes. A comparison of all 4 libraries was done to determine a possible swap.

All testing was done on a laptop computer with `Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz` core using `g++ (GCC) 8.1.1` compiler and `-O3 -DNDEBUG` flags.

2 Static k-d tree

Main usage of a static k-d tree data structure in Medusa is to find support domains. Therefore finding support for domain with N nodes was chosen as the test. Three domains were considered: a 2D and 3D rectangle $[0,1]^d$ with uniform fill and a 2D rectangle $[0,1]^2$ filled with density $\delta r(p) = \delta x + \|p\|/\sqrt{2} \cdot (\Delta x - \delta x)$, $\delta x = \frac{1}{1000n}$, $\Delta x = \frac{10}{n}$. Finding 9 support nodes in 2D and 27 support nodes in 3D was tested. Results are shown in Figure 1.

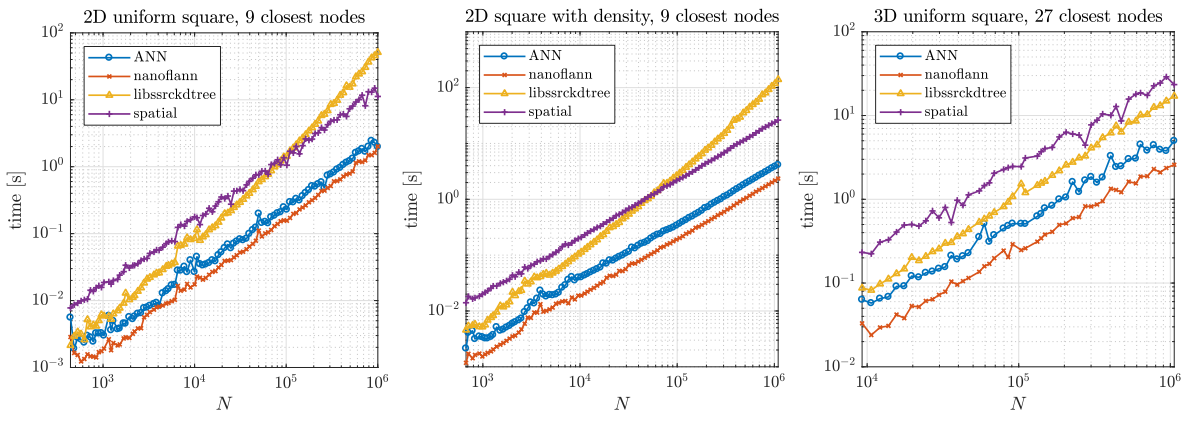

Figure 1: Execution time of support search using different spatial libraries.

3 Dynamic *k*-d tree

Main usage of dynamic k-d tree allowing point insertion is for Poisson Disk Sampling based discretization algorithm. ANN was not included in this test as it does not allow dynamic insertions. Same three domains as above were used except that the fills were timed.

The results are shown in Figure 2.

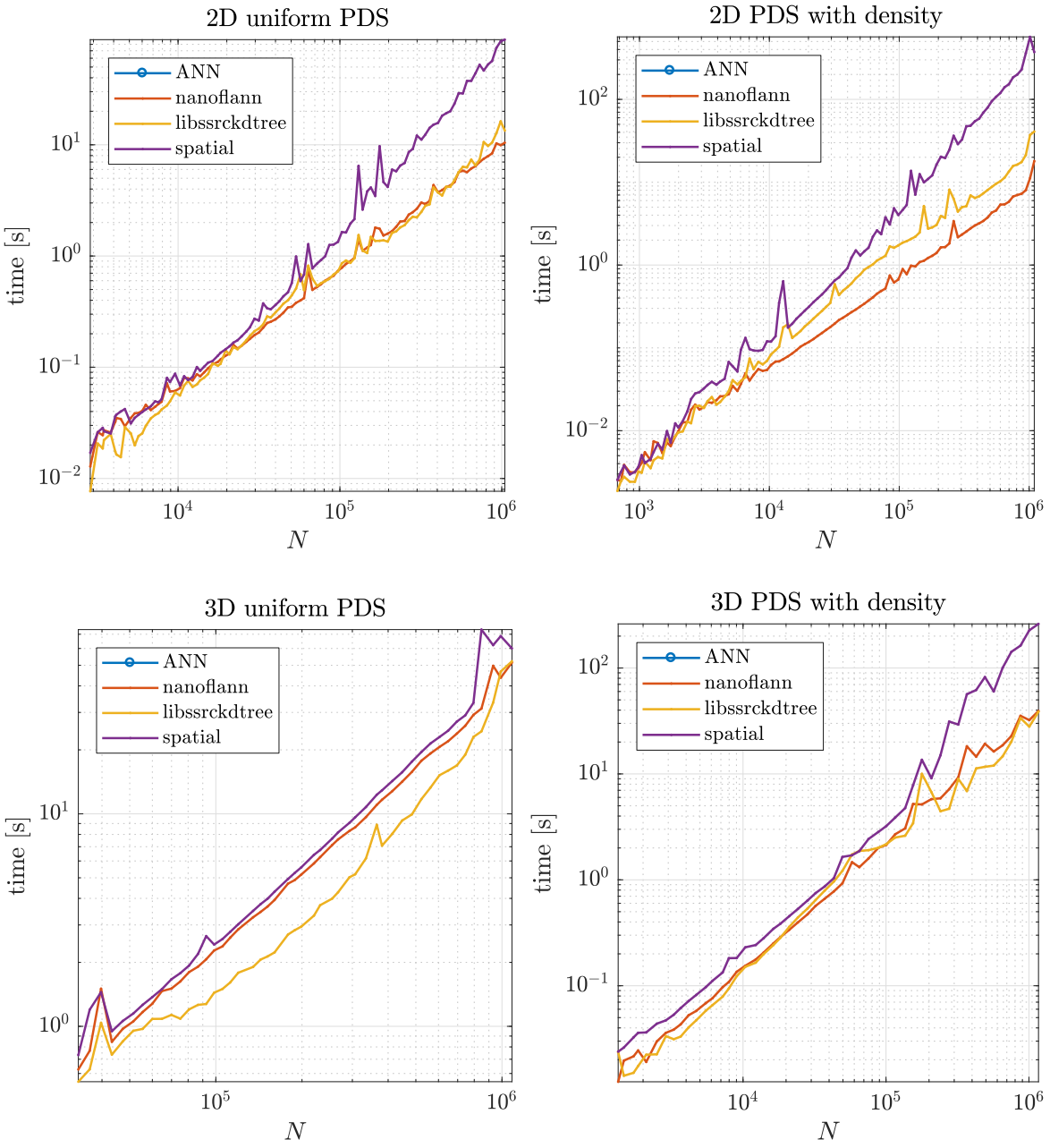

Figure 2: Execution time of Poisson Disk Sampling fill using different spatial libraries.

4 Conclusions

We have chosen `nanoflann` as our library of choice both for static and dynamic spatial search structures. The choice for static structures was easy as it performed best and offered greater

generality than the current ANN library. For dynamic data structures, the performance was similar to `libsrckdtree` and sometimes worse, however `nanoflann`'s lack of dependencies, simplicity and the fact that we only wish to include one library prevailed. Thus, we have decided to use `nanoflann` as the only spatial search library in Medusa [6].

References

[1] `http://www.cs.umd.edu/~mount/ANN/`

[2] `https://www.savarese.com/software/libssrckdtree/`

[3] `https://github.com/jlblancoc/nanoflann/`

[4] `http://spatial.sourceforge.net/`

[5] `https://www.cs.ubc.ca/research/flann/`

[6] `http://e6.ijs.si/medusa`