

Recent Developments in NEARPT3 — Nearest Point Query in E^3 with a Uniform Grid

W. Randolph Franklin
ECSE Dept, 6026 JEC, RPI, Troy NY 12180
geom@wrfranklin.org, <http://wrfranklin.org>

1. INTRODUCTION

We present developments, including an extended discussion, concerning NEARPT3 since the 2004 Fall Workshop in Computational Geometry. NEARPT3 is an algorithm and implementation to preprocess more than 10^8 fixed points in E^3 and then perform nearest point queries against them. With fixed and query points drawn from the same distribution, NEARPT3's expected preprocessing and query time are $\theta(1)$ per point, with a very small constant factor. The data structure is a uniform grid in E^3 , typically with the same number of grid cells as points. The storage budget, in addition to the space to store the points themselves, is 4 bytes per grid cell plus 4 bytes per point. Since last year, the program has been refined and larger tests have been conducted.

Currently, NEARPT3 has been tested on the UNC complete powerplant, on the largest ply datasets in the Georgia Tech Large Geometric Models Archive, and the Stanford Digital Michelangelo Project Archive. The examples with up to 30,000,000 points can be processed on a laptop computer, and the others, the largest of which is St Matthew with 184,088,599 points, on a Xeon.

For lack of space and in order to present new material, this extended abstract omits the following: prior art, algorithm description, comparison to ANN, tests on varying the grid size, table of times on an IBM T30 laptop. They are covered in the older papers[1, 2], and a new comprehensive paper, [3].

2. TESTS

We implemented NEARPT3 in C++ under SuSE Linux. We report on every large dataset that we tested; there are no bad cases omitted.

Our test data has four sources: • **uniXXX**: Our uniformly and independently distributed sets of 10^5 to 10^8 random points. • **blade**, **bone6**, **dragon**, **hand**: The Georgia Institute of Technology's Large Geometric Models Archive, [6]. • **powerplant**: The complete powerplant from the University of North Carolina's UNC Chapel Hill Walkthru Project, [7]. • **bunny**: Stanford University Computer Graphics Laboratory, [5]. • **david**, **stmatthew**: The Stanford Digital Michelangelo Project Archive, [4]. We used the ply files. Figure 1 shows one dataset, illustrating how nonuniform the data is.

In many cases, we sorted the data and deleted duplicate points because some PLY files store a separate copy of a point for each incident triangle. Deleting duplicates made

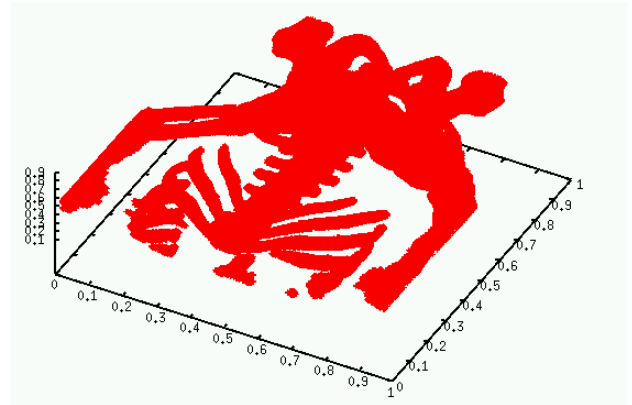


Figure 1: bone6

the problem harder since it meant that the closest fixed point could not be the same as the query point. Then, in each case, we uniformly scaled the data into a $[0, 1]^3$ cube, selected 10,000 points with indices equally spaced throughout the data as query points, and used the rest as fixed points. Then we stored the data as 2-byte binary unsigned short ints, which is sufficient resolution for a laser scanner. (If the points had been stored in ASCII, then reading them would have taken much more time than the nearest point determination.) The grid size used was $1.6\sqrt[3]{N_f}$ for the real datasets and $1.0\sqrt[3]{N_f}$ for the synthetic uniform random distribution.

For each test on the dual 2.2GHz Xeon, the statistics described shown in Table 1 were recorded. T_{pp} and T_{qq} are preprocessing time per fixed point and query time per query point, each in microseconds. We also recorded histograms of the number of points per cell and number of cells checked per query.

3. DISCUSSION

- The absolute times can vary 20% when the same tests are rerun. The relative times of two different tests can vary a factor of two depending on the platform and compiler options. That said:
- The time to preprocess the fixed points is basically constant.
- If we ignore *bunny*, which ran too fast to measure accurately, the time per query varied by a factor of 100 between datasets. The time did not depend on the size of the dataset, but on how evenly distributed the points were. Uniform data sets are good. The powerplant was particularly bad for us because there are a few outlying points, which force the vast majority of the points into a

data	N_f	G	total time (sec)	init time (sec)	preproc time (sec)	query time (sec)	T_{pp}	T_{qq}
bunny	25947	46	0.08	0.	0.010	0.070	0.385	7.
hand	317323	108	0.50	0.050	0.150	0.300	0.473	30.
dragon	427645	120	0.53	0.060	0.220	0.250	0.514	25.
bone6	559636	131	0.53	0.070	0.250	0.210	0.447	21.
blade	872954	152	0.68	0.110	0.400	0.170	0.458	17.
unilm	1000000	160	1.46	0.120	0.940	0.400	0.940	40.
powerplant	5413053	280	63.19	0.980	2.810	59.400	0.519	5940.
uni10m	10000000	344	12.83	1.270	11.130	0.430	1.113	43.
david	28158109	486	20.41	3.400	13.100	3.910	0.465	391.
uni30m	30000000	496	41.69	3.880	37.350	0.460	1.245	46.
uni100m	100000000	742	150.50	13.010	137.020	0.470	1.370	47.
stmatthew	184088599	568	160.08	23.750	128.710	7.620	0.699	762.

Table 1: Tests Run on the Dual 2.2 GHz Xeon with 4MB

small part of the grid. That demonstrates the limitation of this method.

- Even on the uniform random data, the query time rose slowly with N_f . This is puzzling and needs study; our current guess is that accessing large amounts of memory is slightly less efficient.
- NEARPT3 would fail on query points that were from a very different distribution from the fixed points, so that the distance to the nearest point was large, and most of the cells had to be searched. However, many competing methods would also have difficulties.

NEARPT3's cost is affected by the grid resolution, however values within a factor of two of the optimum typically change the time less than a factor of 2.

- Why do the David and St Matthew datasets have such large query times? Their points' local topology is two dimensional. Therefore the distribution of points around each query is very nonuniform. Most of the cells around each query are empty, while a few contain many points. We have collected some statistics on this. For such datasets, a more sophisticated, probably hierarchical, data structure would allow faster queries, if it didn't bloat up the execution size so much that the data set couldn't be processed at all.

4. EXTENSIONS

- NEARPT3 could return approximate nearest matches in much less time than exact nearest matches, since then the spiral search could stop sooner.
- NEARPT3 could be extended to E^d for other d ; the cost of searching would be exponential in d , as for any search procedure.
- How might the fixed point storage budget be reduced? If the user's program doesn't need a separate copy of the points, then we can store the points' coordinates in the grid, instead of storing the points' indexes. Also, knowing which cell contains point p tells us the high order bits of p 's coordinates. For a $512 \times 512 \times 512$ grid, that would save 27 of the 48 bits.
- How might the storage budget of 4 bytes per cell be reduced? Using a hash table keyed on the cell location would reduce that to 0 bytes per empty cell plus perhaps 16 bytes per occupied cell (for the cell location, pointer to its contents, and number of points in it, all times 2 for a conservative hash table load factor). That would be a win for our nonuniform examples. Also, that would allow larger grids to be run on our laptop. However, the execution time might be slower, tho that's not clear.

5. SUMMARY

The general lesson of NEARPT3 is that simple data structures like the uniform grid can be quite efficient in both time and space, especially in E^3 .

6. ACKNOWLEDGEMENTS

This research was supported by NSF grant CCR-0306502. We are grateful to be able to use datasets from the Stanford University Computer Graphics Laboratory, including the Stanford Digital Michelangelo Project Archive, Georgia Institute of Technology's Large Geometric Models Archive, and the University of North Carolina's UNC Chapel Hill Walkthru Project.

7. REFERENCES

- [1] Franklin, W. R. Nearpt3 — nearest point query in E^3 with a uniform grid. In *14th Annual Fall Workshop on Computational Geometry*. MIT.
- [2] Franklin, W. R. Nearpt3 — nearest point query on 184M points in E^3 with a uniform grid. In *17th Canadian Conference on Computational Geometry*. University of Windsor, Ontario.
- [3] Franklin, W. R. Nearpt3 — nearest point query on 184M points in E^3 with a uniform grid. <http://wrfranklin.org/Research/nearpt3/>, 2005.
- [4] Levoy, M. The digital Michelangelo project archive of 3D models. <http://www-graphics.stanford.edu/data/mich/>, 2003.
- [5] Levoy, M. The Stanford 3D scanning repository. <http://www-graphics.stanford.edu/data/3Dscanrep/>, 2005.
- [6] Turk, G. and B. Mullins. Large geometric models archive. http://www.cc.gatech.edu/projects/large_models/, 2003.
- [7] UNC Chapel Hill Walkthru Project. Complete power plant model. <http://www.cs.unc.edu/~geom/Powerplant/>, 1997.