Thalmann, N. Magnenat, and Thalmann, D. (Eds.), "New Trends in Computer Graphics," Proceedings of CG International '88, Springer-Verlag.

Efficiency of Uniform Grids for Intersection Detection on Serial and Parallel Machines

Wm. R. Franklin, N. Chandrasekhar, M. Kankanhalli, M. Seshan, and V. Akman (USA)

Abstract

The uniform grid data structure is a flat (non-hierarchical) grid whose resolution adapts to the data. An exhaustive analysis of the uniform grid data structure for determining intersections in a set of many small line segments is presented. Databases from cartography, VLSI, and graphics with up to 115,973 edges are used. For each data set the intersection time, the ratio of edge pairs tested to pairs found to intersect, and size of intermediate data structures was measured as a function of grid resolution. The execution time was relatively insensitive to the grid size over a range of up to a factor of 10. 115,973 edges were processed to find 135,050 intersections in 683 seconds on a Sun 3/50 workstation. This data structure is also ideally suited for implementation on a parallel machine. When executing on a 16 processor Sequent Balance 21000, total times averaged ten times faster than when using only one processor. Finding all 81,373 intersections in a 62,045 edge database took only 28 seconds elapsed time. This research shows that more complicated, hierarchical data structures, such as quadtrees, are not necessary for this problem.

Introduction

In diverse disciplines such as graphics, cartography, and VLSI design there are problems, such as hidden surface detection, map overlaying, and interference detection, respectively, where the fundamental, low level, operation that consumes most of the time is edge intersection. Some applications are described in Brown [3], Eastman and Yessios[6], Levin [17], Maruyama [18], Ottman, Widmayer, and Wood [20], Nievergelt and Preparata [19], Six and Wood [22], Tilove [23], and Bentley and Wood [2].

We are given from thousands to millions of small edges, very few of which intersect, and must determine the pairs of them that do intersect. Clearly, a quadratic algorithm comparing all $\binom{N}{2}$ pairs is not acceptable. A worst case solution that finds all K intersections of N edges in time $T=\theta\left(K+\frac{N\log^2 N}{\log\log N}\right)$ is presented in Chazelle [5]. However, this method has some limitations. First, it cannot find all the red-blue intersections in a set of red and blue edges without finding (or already knowing) all the red-red and blue-blue intersections. Second, it is inherently sequential, and is more difficult to parallelize. Chazelle has recently improved the time to $T=\theta(K+N\log N)$.

Alternative data structures, based on hierarchical methods such as quadtrees, have also been used extensively, Samet [21]. They are intuitively reasonable data structures to use since they subdivide to spend more time on the complicated regions of the scene. A criticism of their overuse in Geographic Information Systems in given in Waugh [24].

This paper concentrates on an alternative data structure, *the uniform grid*. Here, a flat, non-hierarchical grid is superimposed on the data. The grid adapts to the data since the number of grid cells, or resolution, is a function of some statistic of the input data, such as average edge length. Each edge is entered into a list for each cell that is passes through. Then, in each cell, the edges in that cell are tested against each other for intersection. The grid is completely regular and is not finer in the denser regions of the data.

The uniform grid (in our use) was first presented in Franklin[7] and was later expanded by Franklin, Akman, and Wu [8, 9, 10, 11, 12, 13, 14]. In these papers the uniform grid was called an *adaptive* grid. However, there is another, independent and unrelated, use of the term adaptive grid in numerical analysis in the iterative solution of partial differential equations. Our papers present an expected linear time object space hidden surface algorithm that processed 10,000 random spheres packed ten deep in 383 seconds on a Prime 500. The idea was extended to a fast haloed line algorithm that was tested on 11,000 edges. The concept was applied to other problems such as point containment in polygon testing. Finally it was used, in Prolog and with multiple precision rational numbers in the map overlay problem in cartography.

However, an objection has been repeatedly raised to the uniform grid. Although it is proven theoretically and demonstrated experimentally that the grid is fast for random data, real world data appears much worse than random. Frequently some parts of a real scene are much denser than other parts so that an evenly spaced grid would appear not to work. A hierarchical technique, such as a quadtree, appears to be necessary.

However, even a quadtree cannot efficiently process all data sets. If we have N parallel edges separated by distances of N^{-p} for $p>1$, then it will take more than quadratic time to build either a uniform grid or a quadtree with cells fine enough to distinguish the edges. The plane sweep algorithm would work well in this case. However, the plane sweep cannot handle the red-blue intersection case mentioned above.

There are good reasons for assuming that data sets with one region exponentially denser than another are not common. If there are relatively sparse regions in the data, people then tend to put anything at all in to fill the vacuum. We could also define such data sets out of existence as numerical analysts do with partial differential equations. Just as they consider only equations that satisfy a Lipschitz condition where the greatest slope of a curve is bounded, we might restrict ourselves to sequences of data sets where the densest region's density, relative to the average density, remains bounded as $N \rightarrow \infty$.

This present paper presents experimental evidence that the uniform grid is an efficient means of finding intersections between edges in real world data also. The uniform grid is similar to a quadtree is the same sense that a relational database schema is similar to a hierarchical schema. The power of relational databases, derived from their simplicity and regularity, is also becoming apparent.

The uniform grid data structure is also ideally suited to execution on a parallel machine because of the simpler data structures. Also, it is more numerically robust than sweepline algorithms that have problems

In the following sections, we will review a theoretical development of the uniform grid, see the databases used to testing, and learn the test results.

Intersection Algorithm

Assume that we have N edges of length L independently and identically distributed (i.i.d.) in a 1×1 screen. We place a $G \times G$ grid over the screen. Thus each grid cell is of size $\frac{1}{G} \times \frac{1}{G}$. The grid cells partition the screen without any overlaps or omissions. The intersection algorithm proceeds as follows.

1. For each edge, determine which cells it passes through and write ordered pairs *(cell number, edge number)*.
2. Sort the list of ordered pairs by the cell number and collect the numbers of all the edges that pass through each cell.
3. For each cell, compare all the edges in it, pair by pair, to test for intersections. To determine if a pair of edges intersects, we test each edge's endpoints against the equation of the other edge. We ignore calculated intersections that fall outside the current cell. This handles the case of some pair of edges occurring together in more than one cell.

Theoretical Analysis

Let $N_{c/e}$ be the number of cells that an average edge passes through. Then, approximately,

$$N_{c/e} = (1+2LG)$$

Then N_p, the total number of (cell, edge) pairs is

$$N_p = N(1+2LG)$$

The average number of edges per cell is

$$N_{e/c} = \frac{N_p}{G^2}$$
$$= \frac{N}{G^2}(1+2LG)$$

The time to calculate the (cell, edge) pairs is

$$T_1 = N_p$$

The time to test the edges for intersections is about

$$T_2 = G^2 N_{e/c}{}^2$$
$$= \frac{N^2}{G^2}(1+2LG)^2$$

and the total time is

$$T = T_1 + T_2$$
$$= N + 2LGN + \frac{N^2}{G^2} + \frac{2LN^2}{G} + 4L^2N^2$$

This is minimized if the 2 fastest terms in the sum grow at the same speed, which occurs throughout the range from $N_{c/e} = 2$ to $N_{e/c} = 4$, i.e.

$$\frac{1}{2L} < G < \frac{2NL + \sqrt{4N^2L^2 + 16N}}{8}$$

The flatness of the time curve from about $N_{c/e} = 2$ to $N_{e/c} = 4$ is also observed experimentally. If the average is used for N then the grid is quite insensitive to nonuniform data.

What about some cells being denser since the edges are randomly distributed? Since the time to process a cell depends on the square of the number of edges in that cell, an uneven distribution might increase the total time. However, since the edges are assumed independent, the number of edges per cell is Poisson distributed, and the expected value of the square of the number of edges equals the square of the expected number of edges. Therefore the expected time doesn't increase.

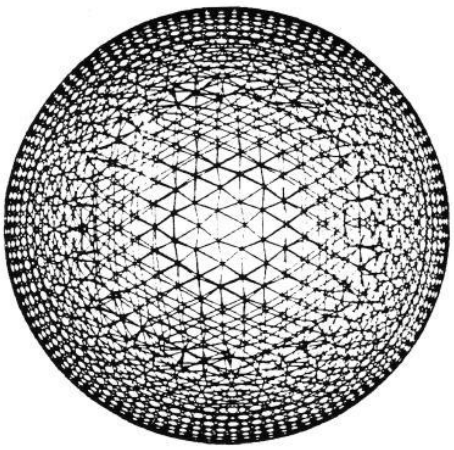
Test Data

We used four different types of data sets, as follows [4].

1. The Risch Ukranian Easter egg, projected onto a plane. The multiple coincidences make this a hard case, especially for a sweep-line algorithm that must keep all the active edges ordered.
2. The state boundaries of the coterminous USA, shifted and overlaid on themselves. The multiple near correlations make this a bad case also.
3. The USGS (United States Geological Survey) DLG (Digital Line Graph) sampler tape. This represents a quadrangle around Chikamauga Tennessee that is split into 8 rectangles. Each rectangle has 4 overlays, for a total of 32 files. The overlays are

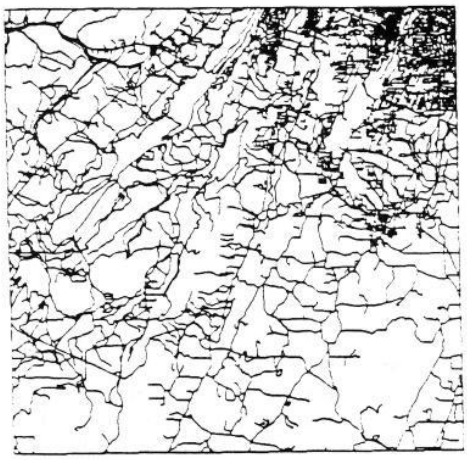
   a) hydrography,

50-3

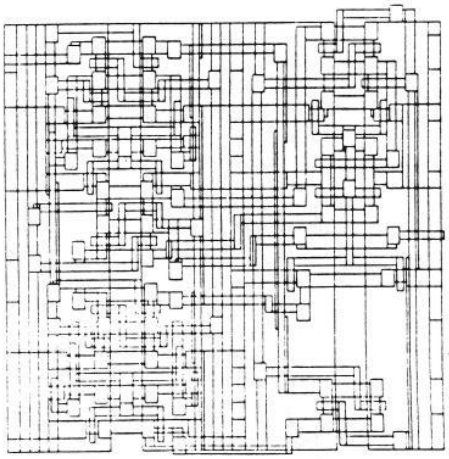


(a) X-Z Plane Projection of the Risch Easter Egg

(b) USA Map - Shifted and Overlaid on Itself



(c) Chikamauga Area 3/8 - Hydrography vs Roads & Trails



(d) CIF Data - XFACE11.MAG

Figure 1: Test Data Sets

50-4

b) roads and trails,

c) railroads, and

d) pipes and transmission lines.

Each file overlay was divided into 8 sections by USGS. These data files were sometimes processed separately and sometimes combined.

4. Some CIF (Caltech Intermediate form) VLSI data.

Sample plots of this data is shown in figure 1.

Experimental Results

For each data set, we tried different grid sizes to find the optimum. For each experiment, we measured

a) the standard deviation of edge length,

b) the number of (edge, cell) pairs,

c) the number of pairs per cell,

d) the number of pairs per edge,

e) the time in seconds to determine the pairs on the Sun 3/50,

f) the time to sort the pairs by cell number,

g) the time to pair up the edges in each cell and calculate intersections,

h) the total time,

i) the number of intersections where the two edges shared an endpoint,

j) the number of intersections where they didn't,

k) the total number of intersections,

l) the expected number of intersections, calculated from $.2\ N^2L^2$,

m) the observed number of comparisons between pairs of edges,

n) the expected number of comparisons, assuming that the edges were independently and uniformly randomly distributed, and

o) the ratio of observed and expected comparisons; large values indicating nonuniform or correlated data.

For each data set we tried many values of G to learn the variation of time with G. Table 1 shows the results from intersecting the 18,092 edges in the roads & trails and hydrography overlays of the Chikamauga DLG. There are 23,586 intersections in all, and the best time is 93 seconds with a 275 $\times$275 grid. The time is within 50% of this for grids from 115 $\times$115 up to 800 $\times$800, which shows the extreme insensitivity of the time to the grid size. This is why real scenes with dense and sparse areas can be accommodated efficiently.

The economy of the grid structure is shown by the fact that only 40,031 comparisons of pairs of edges were needed to isolate the 23,586 intersections. This behavior was also observed in hidden surface algorithm described in earlier publications. There is not much room for further improvement by a hierarchical method.

Figure 2 graphs the time versus G for the USA state boundaries shifted and overlaid on themselves. The execution time is within 20% of the optimum from about $G = 40$ to $G = 400$ and is within a factor of two of the optimum from about $G = 20$ to $G = 700$. Outside these limits, the execution time starts to rise quickly.

Table 2 shows the results from processing each data set. Our biggest example overlaid all four parts of the DLG, totaling 115,973 edges of average length 0.0022. It found the 135,050 intersections in 683 seconds with a 650 $\times$650 grid.

The size of the grid, 422,500 cells, may appear inefficient. However, most cells are empty and, unlike in a tree data structure, an empty cell does not occupy even one word of storage, not even for a nil pointer.

50-5

Table 1: Chikamauga Area 3 - Hydrography, Roads & Trails

No. of edges	18092
Avg. edge length	0.0044
Standard deviation	0.0061
Xsects. by end pt. coincidence	23007
Xsects. by actual equation soln	579
Total intersections	23586

Grids	Pairs	P/Cell	P/Edge	Grid Time	Sort Time	Xsect Time	Total Time
10	18988	189.880	1.050	15.45	4.60	3060.15	3080.20
13	19235	113.817	1.063	15.43	4.62	2486.20	2506.25
15	19421	86.316	1.073	17.15	7.55	2101.47	2126.17
20	19959	49.898	1.103	15.58	4.75	1370.98	1391.31
25	20420	32.672	1.129	16.17	5.17	927.71	949.05
30	20888	23.209	1.155	15.83	4.92	689.41	710.15
40	21931	13.707	1.212	15.78	4.92	421.88	442.58
50	22862	9.145	1.264	15.88	5.10	308.15	329.14
65	24378	5.770	1.347	16.18	5.50	217.57	239.26
80	25841	4.038	1.428	16.50	5.80	168.63	190.93
100	27713	2.771	1.532	16.95	6.27	131.89	155.11
115	29187	2.207	1.613	17.47	6.53	114.10	138.09
125	30131	1.928	1.665	17.72	6.70	105.30	129.71
140	31572	1.611	1.745	18.22	7.15	95.23	120.60
150	32496	1.444	1.796	18.47	7.20	89.38	115.05
160	33514	1.309	1.852	18.77	7.47	84.50	110.73
175	35005	1.143	1.935	19.33	8.07	79.40	106.80
200	37340	0.933	2.064	20.15	8.38	72.06	100.60
275	44483	0.588	2.459	22.63	10.03	60.61	93.28
325	49373	0.467	2.729	24.68	11.42	57.48	93.58
400	56617	0.354	3.129	28.72	13.37	55.01	97.10
500	66222	0.265	3.660	30.92	16.03	56.05	103.00
625	78304	0.200	4.328	36.22	19.25	56.70	112.16
800	95143	0.149	5.259	45.91	24.13	61.85	131.89
1000	114419	0.114	6.324	61.35	30.20	69.01	160.56

Execution in Parallel

The uniform grid method is ideally suited to execution on a parallel machine since it mostly consists of two types of operations that run well in parallel: applying a function independently to each element of a set to generate a new set, and sorting. Determining which cells each edge passes through is an example of the former operation.

We implemented several versions of the algorithm on a Sequent Balance 21000 computer, which contains 16 National Semiconductor 32000 processors [1, 15], and compared the elapsed time when up to 15 processors were used to the time for only one processor [16]. The speedup ratios ranged from 8 to 13. Figure 3 shows the results from processing 3 overlays of the United State Geological Survey Digital Line Graph, totaling 62,045 edges. 81,373 intersections were found. The

50-6

Table 2: Summary of Results from Processing All the Data Sets

Serial Computation						
Database	Edges	Length	Std Dev	Xsects	Grid Size	Time
Risch Egg - YZ Projection	5897	0.0355	0.0124	39666	100	194.24
XZ Projection	5897	0.0391	0.0132	37415	115	193.18
XY Projection	5897	0.0352	0.0131	40177	80	183.83
USA Map	915	0.0186	0.0245	1078	125	4.97
Shifted by 2% and overlaid on itself	1830	0.0184	0.0243	2430	140	14.38
Shifted by 10% & overlaid	1830	0.0180	0.0237	2348	125	12.57
Chikamauga Area 1 - Hydrography, Roads & Trails	13712	0.0044	0.0084	15039	275	68.50
Area 2, HR&T	14145	0.0049	0.0080	16595	275	71.11
Area 3, HR&T	18092	0.0044	0.0061	23586	275	93.28
Area 4, HR&T	16425	0.0048	0.0076	20335	200	88.58
Area 5, HR&T	12869	0.0053	0.0103	14978	275	62.93
Area 6, HR&T	13871	0.0050	0.0080	16072	275	69.40
Area 7, HR&T	13579	0.0134	0.0518	16640	160	188.76
Area 8, HR&T	11937	0.0048	0.0098	13283	275	58.86
All sections - Railroads	1122	0.0159	0.0543	1316	150	8.10
Pipe & Transmission Lines	850	0.0277	0.0523	1211	115	7.95
Railroads, Pipe & Transmission Lines	1972	0.0206	0.0533	2745	115	22.28
Railroads, Pipe & Transmission Lines Overlaid on itself	3944	0.0206	0.0533	13268	115	84.15
Hydrography, Railroads, Pipe & Transmission Lines	55973	0.0023	0.0162	53426	500	323.09
Roads & Trails, Railroads, Pipe & Transmission Lines	62045	0.0026	0.0106	81373	500	436.35
Hydrography, Roads & Trails, Railroads, Pipe & Trans. Lines	115973	0.0022	0.0115	135050	650	682.51
VLSI Data - XFACEA.MAG	436	0.0314	0.0908	1403	150	5.22
VLSI Data - XFACELL.MAG	1960	0.0467	0.0852	6488	65	16.87
VLSI Data - XFACELL.MAG - Rotated by 30 deg.	1960	0.0352	0.0643	6488	125	32.48
VLSI Data - XFACELL.MAG - Rotated by 90 deg.	1960	0.0467	0.0852	6488	65	18.67

Parallel Computation							
Database	Edges	Xsects	Grid Size	Time Taken For 1 Proc	5 Procs	10 Procs	15 Procs
Risch Egg - YZ Projection	5897	39666	100	98.91	24.02	14.19	11.96
XZ Projection	5897	37415	115	97.88	23.55	14.83	11.81
XY Projection	5897	40177	80	92.33	20.33	12.36	10.40
Roads & Trails, Railroads, Pipe & Transmission Lines	62045	81373	250	273.11	62.98	39.42	27.77
Random Edges of Size 0.01	50000	45719	100	521.06	108.90	57.88	40.15

50-7

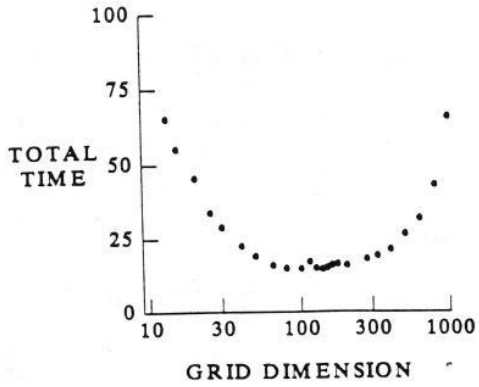


Figure 2: Graph of Time vs Grid Resolution for the USA Overlaid on Itself

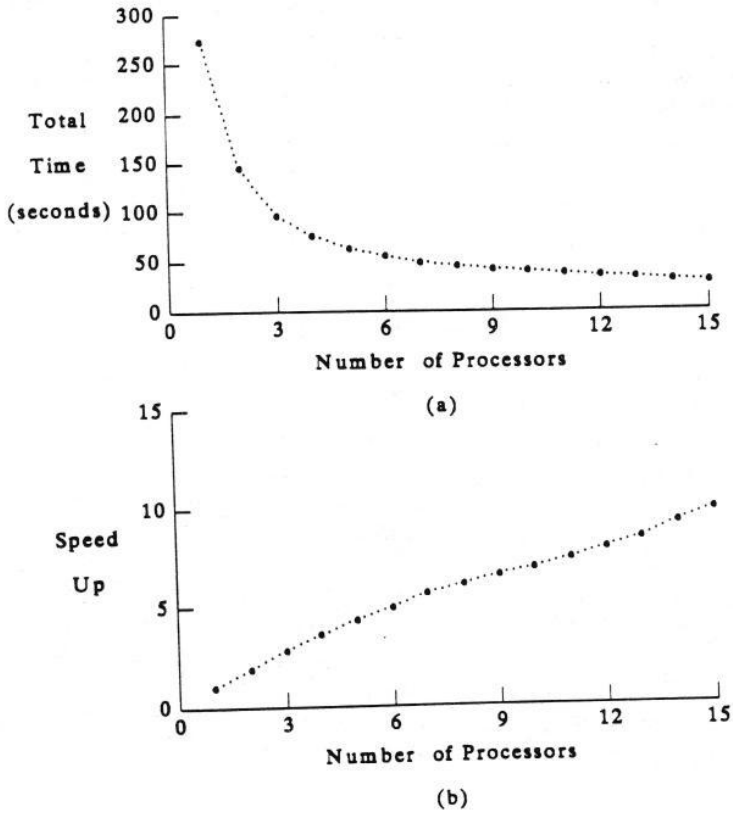


Figure 3: Time and Speedup When Intersecting the 62045 Edges in the Roads & Trails, Railroads, and Pipes and Transmission Lines Overlays of the Chikamauga DLG in Parallel on 1 to 15 Processors. Grid Size = 250. 81,373 Intersections Found.

50-8

time for one processor was 273 seconds, and for 15 processors was 28 seconds, for a speedup of about 10. This is a rate of 7.9 million edges and 10.5 million intersections per hour. For other data sets, these extrapolated times would depend on those data sets' number of intersections per edge. Times for intersecting other data sets in parallel are given in table 2.

Finally, the speedup, as a function of the number of processors, was still rising smoothly at 15 processors. This means that we should achieve an even bigger speedup on a more parallel machine.

Conclusion

We have answered the objection that the uniform grid is suitable for only evenly spaced data by showing experimentally that it is just as efficient on unevenly spaced, real data. Since it is also very easy to implement, and executes well on parallel machines, there is now no need for more complicated methods such as quadtrees and plane-sweep algorithms.

Acknowledgements

This work was supported by the National Science Foundation under PYI grant no. DMC-8351942. Don Porter of RPI's Information and Technology Services allowed us to drag down his Suns running the time tests. The CIF data was supplied by Jim Guilford, and the Risch Easter Egg by Julian Gomez. Their assistance is appreciated especially since it is surprisingly difficult to obtain large real graphic databases.

References

1. Sequent Computer Systems Inc., *Balance Technical Summary*, 1986.
2. J.L. Bentley and D. Wood, "An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles," *IEEE Trans. Comput.*, vol. C-29, no. 7, pp. 571-576, July 1980.
3. K.Q. Brown, "Comments on 'Algorithms for Reporting and Counting Geometric Intersections'," *IEEE Trans. Comput.*, vol. C-30, no. 2, pp. 147-148, February 1981.
4. Narayanaswami Chandrasekhar and Manoj Seshan, "The Efficiency of the Uniform Grid for Computing Intersections," M.S. thesis, Electrical, Computer, and Systems Engineering Dept., Rensselaer Polytechnic Institute, December 1987.
5. B.M. Chazelle, *Reporting and Counting Arbitrary Planar Intersections*, CS-83-16, Dept. of Computer Science, Brown University, Providence, RI, 1983.
6. C.M. Eastman and C.I. Yessios, *An Efficient Algorithm for Finding the Union, Intersection, and*, Sept. 1972. Carnegie-Mellon University, Dept. of Computer Science
7. W. Randolph Franklin, *Combinatorics of Hidden Surface Algorithms*, Center for Research in Computing Technology, Harvard University, June 1978. Ph.D. thesis
8. Wm. Randolph Franklin, "A Linear Time Exact Hidden Surface Algorithm," *ACM Computer Graphics*, vol. 14, no. 3, pp. 117-123, July 1980. Proceedings of SIGGRAPH'80
9. Wm. Randolph Franklin, "An Exact Hidden Sphere Algorithm That Operates In Linear Time," *Computer Graphics and Image Processing*, vol. 15, no. 4, pp. 364-379, April 1981.
10. Wm. Randolph Franklin, "Efficient Polyhedron Intersection and Union," *Proc. Graphics Interface'82*, pp. 73-80, Toronto, 19-21 May 1982.
11. Wm. Randolph Franklin, "A Simplified Map Overlay Algorithm," *Harvard Computer Graphics Conference*, Cambridge, MA, 31 July - 4 August 1983. sponsored by the Lab for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University.
12. Wm. Randolph Franklin, "Adaptive Grids For Geometric Operations," *Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six)*, vol. 2, pp. 230-239, Ottawa, Canada, 16-21 October 1983.

30-9

13. Wm. Randolph Franklin and Varol Akman, *A Simple and Efficient Haloed Line Algorithm for Hidden Line Elimination*, Univ. of Utrecht, CS Dept., Utrecht, October 1985. report number RUU-CS-85-28

14. Wm. Randolph Franklin and Peter Y.F. Wu, *A Polygon Overlay System in Prolog*, Autocarto 8 Conference, Baltimore, March-April 1987.

15. Marta Kallstrom and Shreekant Thakkar, "Programming three Parallel Computers," *IEEE Software*, vol. 5, no. 1, pp. 11-22, January 1988.

16. Mohan Kankanhalli, "Uniform Grids for Lines Intersection in Parallel," M.S. Thesis, Electrical,Computer & Systems Engineering Dept., Rensselaer Polytechnic Institute, Troy, NY, February 1988.

17. J.Z. Levin, "Mathematical Models For Determining the Intersections of Quadric Surfaces," *Computer Graphics and Image Processing*, vol. 11, pp. 73-87, 1979.

18. K. Maruyama, "A Procedure to Determine Intersections Between Polyhedral Objects," *Int. J. Comput. Infor. Sc.*, vol. 1, no. 3, pp. 255-266, 1972.

19. J. Nievergelt and F.P. Preparata, "Plane-Sweep Algorithms for Intersecting Geometric Figures," *Comm. ACM*, vol. 25, no. 10, pp. 739-747, October 1982.

20. Thomas Ottmann, Peter Widmayer, and Derick Wood, "A Fast Algorithm for the Boolean Masking Problem," *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 249-268, Academic Press, 1985.

21. H. Samet, "The Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys*, vol. 16, no. 2, pp. 187-260, June 1984.

22. H.-W. Six and D. Wood, "The Rectangle Intersection Problem Revisited," *BIT*, vol. 20, pp. 426-433, 1980.

23. R.B. Tilove, "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 874-883, October 1980.

24. T.C. Waugh, "A Response to Recent Papers and Articles on the Use of Quadtrees for Geographic Information Systems," *Proceedings of the Second International Symposium on Geographic Information Systems*, pp. 33-37, Seattle, Wash. USA, 5-10 July 1986.

50-10

Addendum to

EFFICIENCY OF UNIFORM GRIDS FOR INTERSECTION DETECTION ON SERIAL AND PARALLEL MACHINES

Wm. Randolph Franklin[1], Narayanaswami Chandrasekhar[1], Mohan Kankanhalli[1], Manoj Seshan[1], Varol Akman[2]

April 26, 1988

We have now transported the program to the Sun 4/280, with the following results.

1. On the United State Geological Survey Digital Line Graph sampler tape with 116,896 edges (Figure 1), we found 144,666 intersections with a 325x325 grid in 37 seconds.

2. We tested a 1,000,000 (one million) edge database from the middle 60% x 60% (i.e. densest region) of a 2,500,000 edge chip design (Figure 2). The edges were badly distributed; the average length was 0.001 while the standard deviation was 0.004 (Figure 3).

   Using a 600x600 grid, 2,010,564 intersections were found (of the 500,000,000,000 possible intersections) in 295 seconds. In this case we used the fact that all the edges are either horizontal or vertical; otherwise the time would 784 seconds with a 2000x2000 grid.

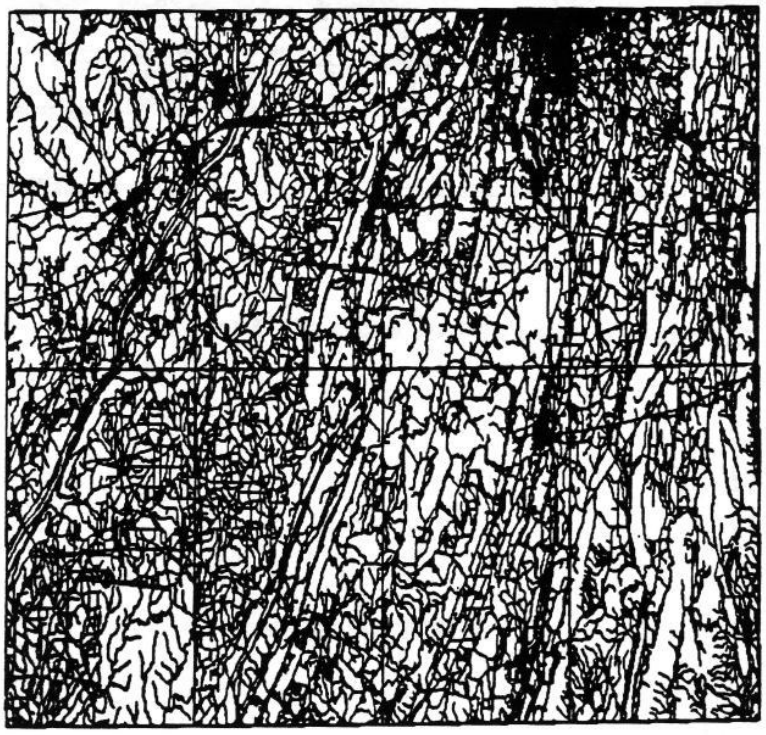


Figure 1: USGS DLG Sampler Tape with 116,896 Edges

---

1 Electrical, Computer, and Systems Engineering Dept., Rensselaer Polytechnic Institute, Troy, NY, 12180, USA, (518) 276-6077, Internet: Franklin@CS.RPI.EDU, Telex: 6716050 RPI TROU

2 Centrum voor Wiskunde en Informatica, Amsterdam, THE NETHERLANDS, varol@cwi.nl

CHIPS DATA FILE - CONTAINS 554629 BOXES I.E. 2218116 EDGES

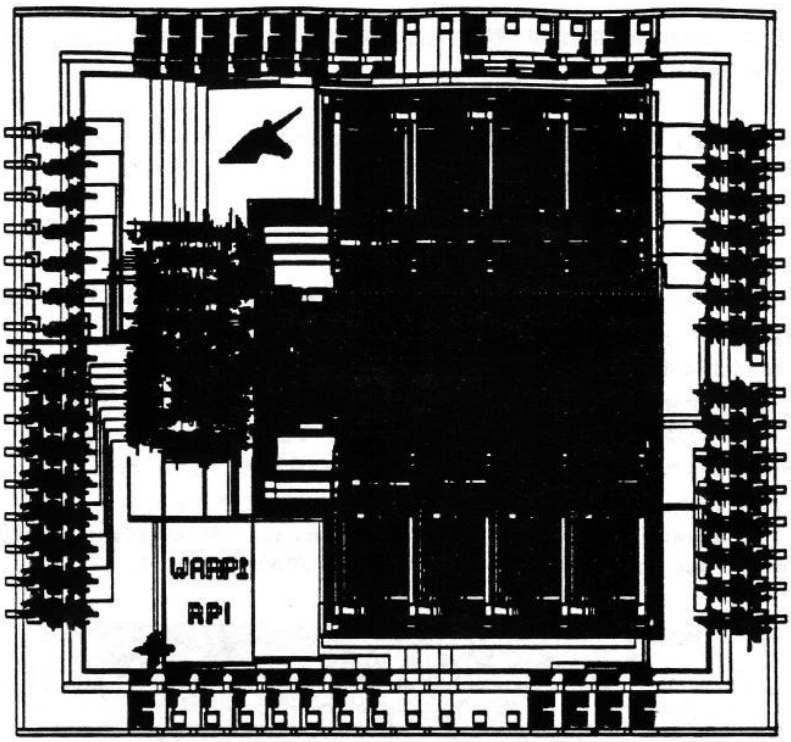


Figure 2: Sample Chip, Of Which The Middle 60% With 1,000,000 Edges Was Used.

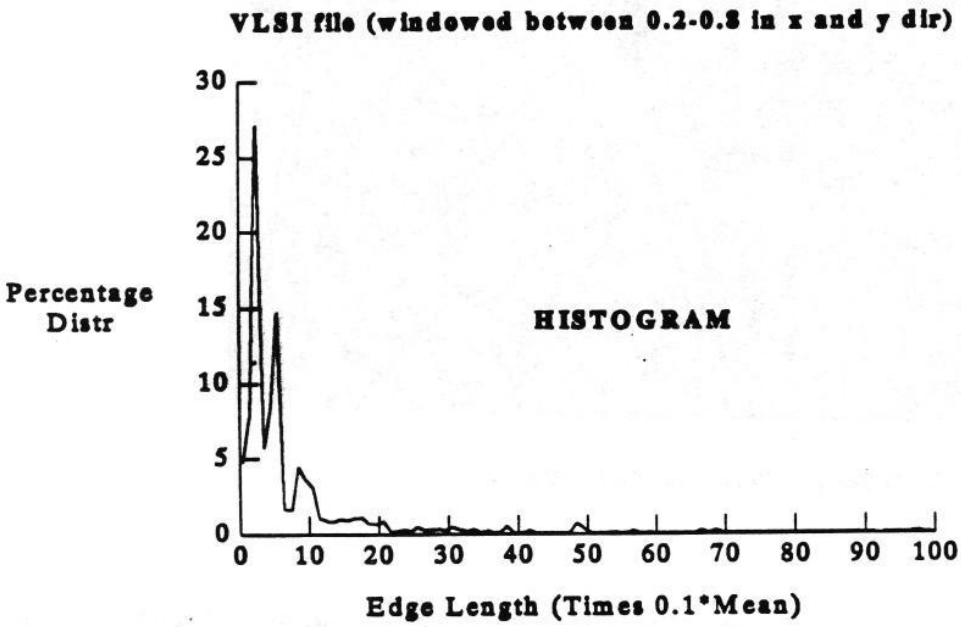


Figure 3: Distribution of Lengths of the 1,000,000 Edges