53

Fast Intersection Detection on Serial and Parallel Machines using the Uniform Grid Technique

Wm. Randolph Franklin

Narayanaswami Chandrasekhar

Mohan Kankanhalli

Electrical, Computer and Systems Engineering Dept.,
Rensselaer Polytechnic Institute,
Troy, NY, 12180, USA,
(518) 276-6077,
Internet: franklin@turing.cs.rpi.edu,
Telex: 6716050 RPI TROU

Varol Akman

Dept. of Information Sciences and Computer Eng.,
Bilkent University,
POB 8, 06572 Maltepe, Ankara, Turkey.

Please send the proofs to the first address.

537

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 1 million 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. One million edges were processed to find 2.01 million intersections in 294 seconds on a Sun 4/280 workstation. This data structure is also ideally suited for implementation on a parallel machine. When executing on a 16 processor Sequent Balance 21000, the 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.

Key Words

Graphics, Computer-aided Design, Computational Geometry, Intersection Detection, Parallel Geometric Algorithms.

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 (1981), Eastman and Yessios(1972), Levin (1979), Maruyama (1972), Ottman, Widmayer, and Wood (1985), Nievergelt and Preparata (1982), Six and Wood (1980), Tilove (1980), and Bentley and Wood (1980).

We are given thousands or 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 (1983). 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 and Edelsbrunner(1988) have 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 (1984). 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 (1986).

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 it 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(1978) and was later expanded by Franklin, Akman, and Wu (1980, 1981, 1982, 1983a, 1983b, 1985, 1987). 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 the theoretical development of the uniform grid, see the databases used for 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. If the edges are *a priori* known to be either *vertical* or

horizontal, the vertical edges are compared with the horizontal edges only. To determine if a pair of edges intersect, 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. The determination of $N_{c/e}$ is similar to the Buffon's Needle Problem (McCord 1964). A simple analysis shows that,

$$N_{c/e} = (1 + \frac{4}{\pi} LG)$$

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

$$N_p = N(1 + \frac{4}{\pi} LG)$$

The average number of edges per cell is

$$N_{e/c} = \frac{N_p}{G^2}$$

$$= \frac{N}{G^2}(1 + \frac{4}{\pi} LG)$$

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

$$T_1 = \alpha N_p$$

where α is a constant. The time to test the edges for intersections is about

$$T_2 = \beta\ G^2 N_{e/c}(N_{c/e} - 1)$$

where β is a constant. The overhead for processing the cells is

$$T_3 = \gamma G^2$$

where γ is a constant. and the total time is

$$T = T_1 + T_2 + T_3$$

$$= N\left[(\alpha - \beta) + \frac{4}{\pi}(\alpha - \beta) L G\right] + \beta N^2 \left[\frac{1}{G^2} + \frac{8}{\pi}\frac{L}{G} + \frac{16}{\pi^2} L^2\right] + \gamma G^2$$

This is minimized if the 2 fastest terms in the sum grow at the same speed, which occurs when $G = \min\left(\delta\sqrt{N}, \frac{\pi}{4L}\right)$ for some δ.

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 (Chandrasekhar 1987).

1. The Resch Ukranian Easter egg(Blinn 1988) , 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,
   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 4/280,

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 end-point,

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 116896 edges in all the 4 overlays of the Chikamauga DLG (Fig 1c). There are 144,666 intersections in all, and the best time is 37 seconds with a 325×325 grid. The time is within 50% of this for grids from 175×175 up to 1000×1000, 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 the number of comparisons between pairs of edges needed to isolate the intersections is about twice the number of the edges when using the optimal grid resolution. 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 (Fig 1d) consisted of 1,000,000 edges from the densest region of a 2,500,000 edge chip database with an average length of 0.0012. It took only 294 seconds on a 600×600 grid to compute the 2.01 million intersections when we used the fact that all the edges are either horizontal or vertical. If the above fact was not used, it took 784 seconds on 2000×2000 grid. The histogram of edge lengths (Fig 3) shows the bad distribution in this example. It can be seen that a few very long edges increase the mean value and that the length of most of the edges is less than the mean length. This example proves that the uniform grid method is suitable for badly distributed edges too. The size of the grid, 1,710,208 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.

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 (Sequent 1986, Kallstrom 1988), and compared the elapsed time when up to 15 processors were used to the time for only one processor (Kankanhalli 1988). We used the 'data partioning' paradigm of parallel programming which involves creating multiple, identical processes and assigning a portion of the data to each process. The edges are distributed among the processors to determine the grid cells to which each edge belongs and then the cells are distributed among the processors to compute the intersections. Since the Sequent Balance 21000 is a shared memory parallel computer, shared data structures is the communication mechanism for the processors. The synchronization of the processors is achieved by using atomic locks. Basically, the concept of 'local processing' has been adopted in this algorithm to achieve parallelism.

There were several different ways of implementing the uniform grid data structure. First, we had a G^2MP array of cells, where G is the grid size, M is the maximum number of edges per cell per processor and P is the number of processors. However this implementation took up a lot of memory space though it obviated the use of locks. Then, G^2 array of linked lists was used. This also did not require locking but it was slow because of the dynamic allocation of shared global memory. Then it was implemented using a linked list of $(cell, edge)$ pairs but this also was slow because of dynamic memory allocation. Finally a G^2M array implementation was made which used atomic locks. This implementation gave the best results.

Table 3 shows the results of the parallel implementation of the algorithm. The speedup ratios range from 8 to 13. Figure 4 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 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.

The speedup achieved for any parallel algorithm is dependent on the amount of inherently sequential computation in the algorithm, the hardware contention imposed by the competing processors, the overhead in creating multiple processes and the overhead in synchronization & communication among the multiple processes. We believe that the first factor is not dominant when using the uniform grid technique. The large speedups achieved show that the other three factors also do not affect the performance significantly. 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. CCR-8351942. The CIF data was supplied by Jim Guilford and Edwin Rogers, and the Resch Easter Egg by Julian Gomez. Their assistance is appreciated especially since it is surprisingly difficult to obtain large real graphic databases.

References

Bentley, J.L. and Wood, D. (1980) "An Optimal Worst Case Algorithm for Reporting Intersections of Rectangles," *IEEE Trans. Comput.* **C-29**(7), pp. 571-576 (July 1980).

Blinn, James F. (Blinn 1988) "The World's Largest Easter Egg and what came out of it," *IEEE Computer Graphics and Applications* **8**(2), pp. 16-23 (March 1988).

Brown, K.Q. (1981) "Comments on 'Algorithms for Reporting and Counting Geometric Intersections'," *IEEE Trans. Comput.* **C-30**(2), pp. 147-148 (February 1981).

53-15

Chandrasekhar, Narayanaswami and Seshan, Manoj "The Efficiency of the Uniform Grid for Computing Intersections," (Chandrasekhar 1987) M.S. thesis, Electrical, Computer, and Systems Engineering Dept., Rensselaer Polytechnic Institute (December 1987).

Chazelle, Bernard and Edelsbrunner, Herbert (1988) "An Optimal Algorithm for Intersecting Line Segments in the Plane," *Foundations of Computer Science - 29th Annual Symposium*, White Plains (October 1988).

Chazelle, B.M. (1983) *Reporting and Counting Arbitrary Planar Intersections,* Dept. of Computer Science, Brown University, Providence, RI (1983).

Eastman, C.M. and Yessios, C.I. (1972) *An Efficient Algorithm for Finding the Union, Intersection, and Differences of Spatial Domains,* Carnegie-Mellon University, Dept. of Computer Science (Sept. 1972).

Franklin, W. Randolph (1978) *Combinatorics of Hidden Surface Algorithms,* Center for Research in Computing Technology, Harvard University (June 1978). Ph.D. thesis

Franklin, Wm. Randolph (1980) "A Linear Time Exact Hidden Surface Algorithm," *ACM Computer Graphics* **14**(3), pp. 117-123, Proceedings of SIGGRAPH'80 (July 1980).

Franklin, Wm. Randolph (1981) "An Exact Hidden Sphere Algorithm That Operates In Linear Time," *Computer Graphics and Image Processing* **15**(4), pp. 364-379 (April 1981).

Franklin, Wm. Randolph (1982) ''Efficient Polyhedron Intersection and Union,'' *Proc. Graphics Interface'82*, Toronto, pp. 73-80 (19-21 May 1982).

Franklin, Wm. Randolph (1983a) ''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.

Franklin, Wm. Randolph (1983b) ''Adaptive Grids For Geometric Operations,'' *Proc. Sixth International Symposium on Automated Cartography (Auto-Carto Six)*, Ottawa, Canada **2**, pp. 230-239 (16-21 October 1983).

Franklin, Wm. Randolph and Akman, Varol (1985) *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

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

Kallstrom, Marta and Thakkar, Shreekant (Kallstrom 1988) ''Programming three Parallel Computers,'' *IEEE Software* **5**(1), pp. 11-22 (January 1988).

Kankanhalli, Mohan ''Uniform Grids for Lines Intersection in Parallel,'' (Kankanhalli 1988) M.S. Thesis, Electrical,Computer &

Systems Engineering Dept., Rensselaer Polytechnic Institute, Troy, NY (February 1988).

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

Maruyama, K. (1972) "A Procedure to Determine Intersections Between Polyhedral Objects," *Int. J. Comput. Infor. Sc.* **1**(3), pp. 255-266 (1972).

McCord, J. and Moroney, R (McCord 1964) *Probability Theory,* The Macmillan Company, New York (1964), pp. 4-8.

Nievergelt, J. and Preparata, F.P. (1982) "Plane-Sweep Algorithms for Intersecting Geometric Figures," *Comm. ACM* **25**(10), pp. 739-747 (October 1982).

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

Samet, H. (1984) "The Quadtree and Related Hierarchical Data Structures," *ACM Computing Surveys* **16**(2), pp. 187-260 (June 1984).

Sequent, Computer Systems (Sequent 1986) *Balance Technical Summary,* Sequent Computer Systems Inc. (1986).

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

Tilove, R.B. (1980) "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," *IEEE Trans. Comput.* **C-29**(10), pp. 874-883 (October 1980).

Waugh, T.C. (1986) "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*, Seattle, Wash. USA, pp. 33-37 (5-10 July 1986).

53-19

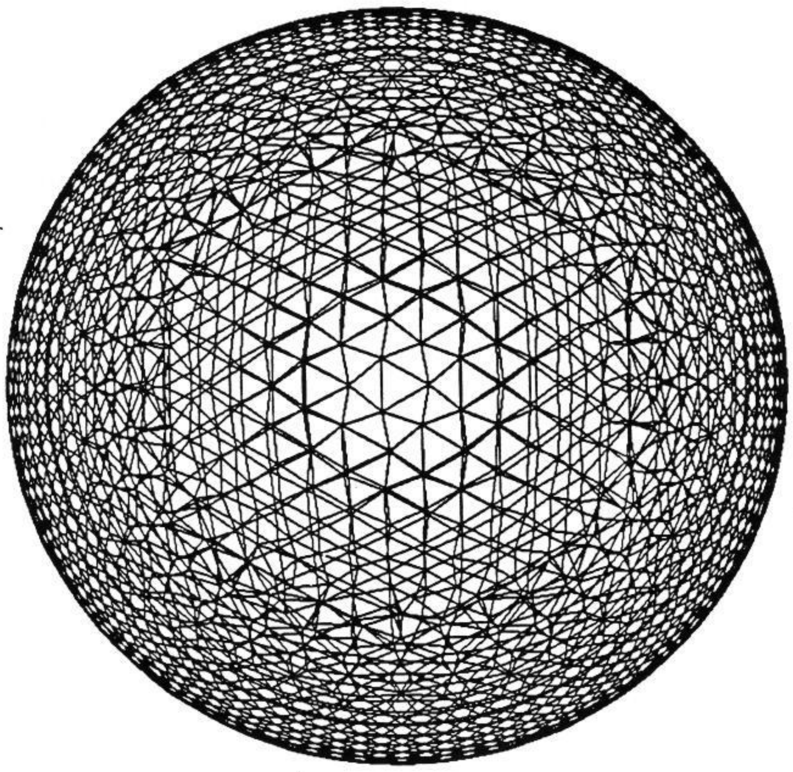


Fig 1(a). X-Z Plane Projection of the Resch Easter Egg

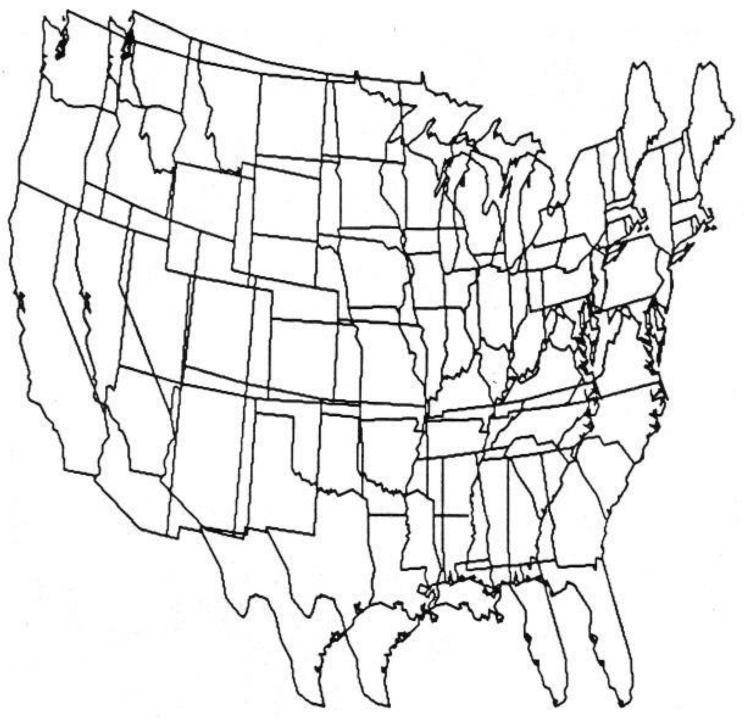


Fig 1(b). USA Map - Shifted and Overlaid on Itself

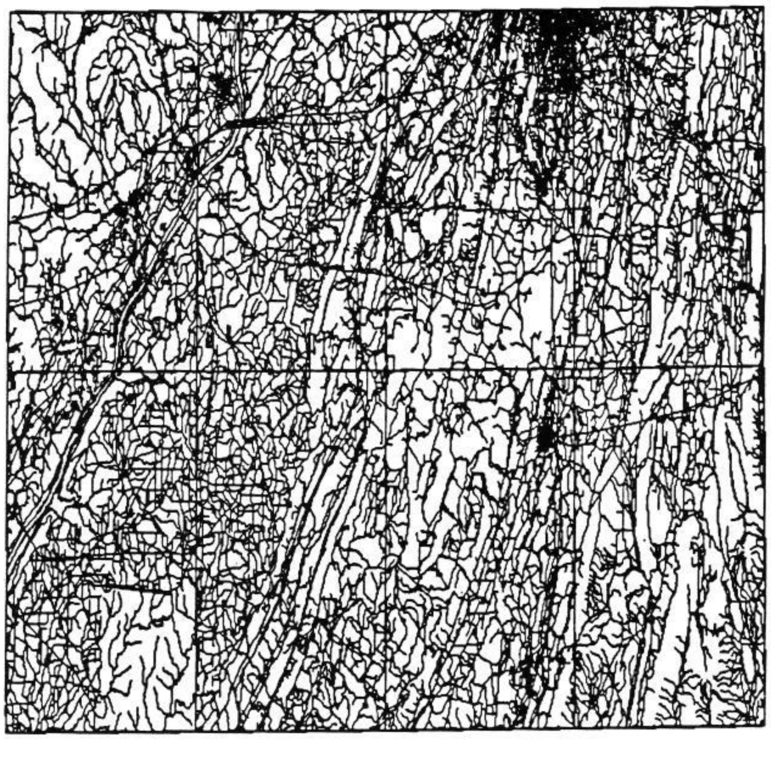


Fig 1(c). Chikamauga Area - all 4 overlays

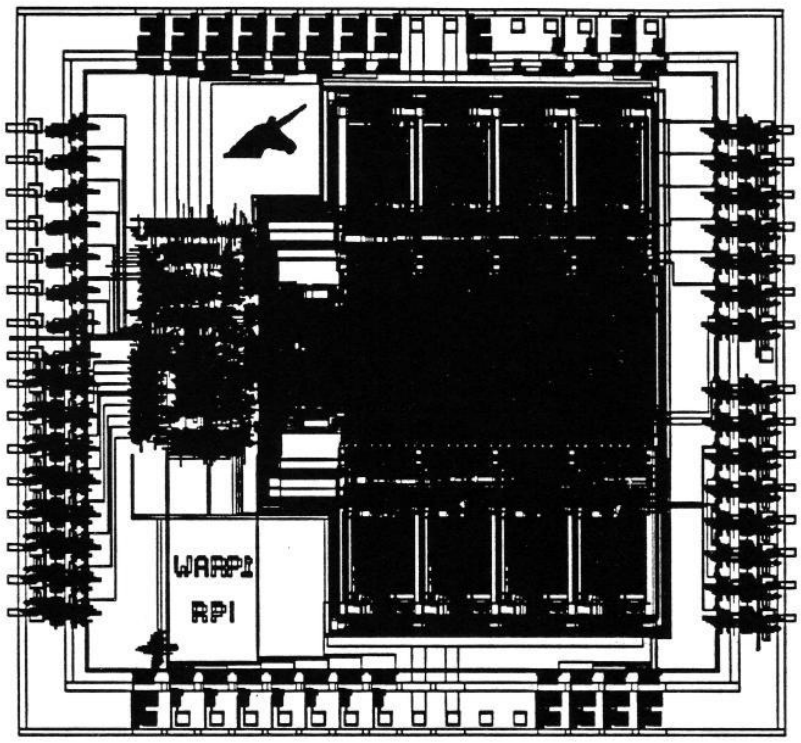


Fig 1(d). CIF Data - 2218116 edges

U.S.A. MAP

No. of edges	=	915
Avg. edge length	=	0.0186
Total intersections	=	1078

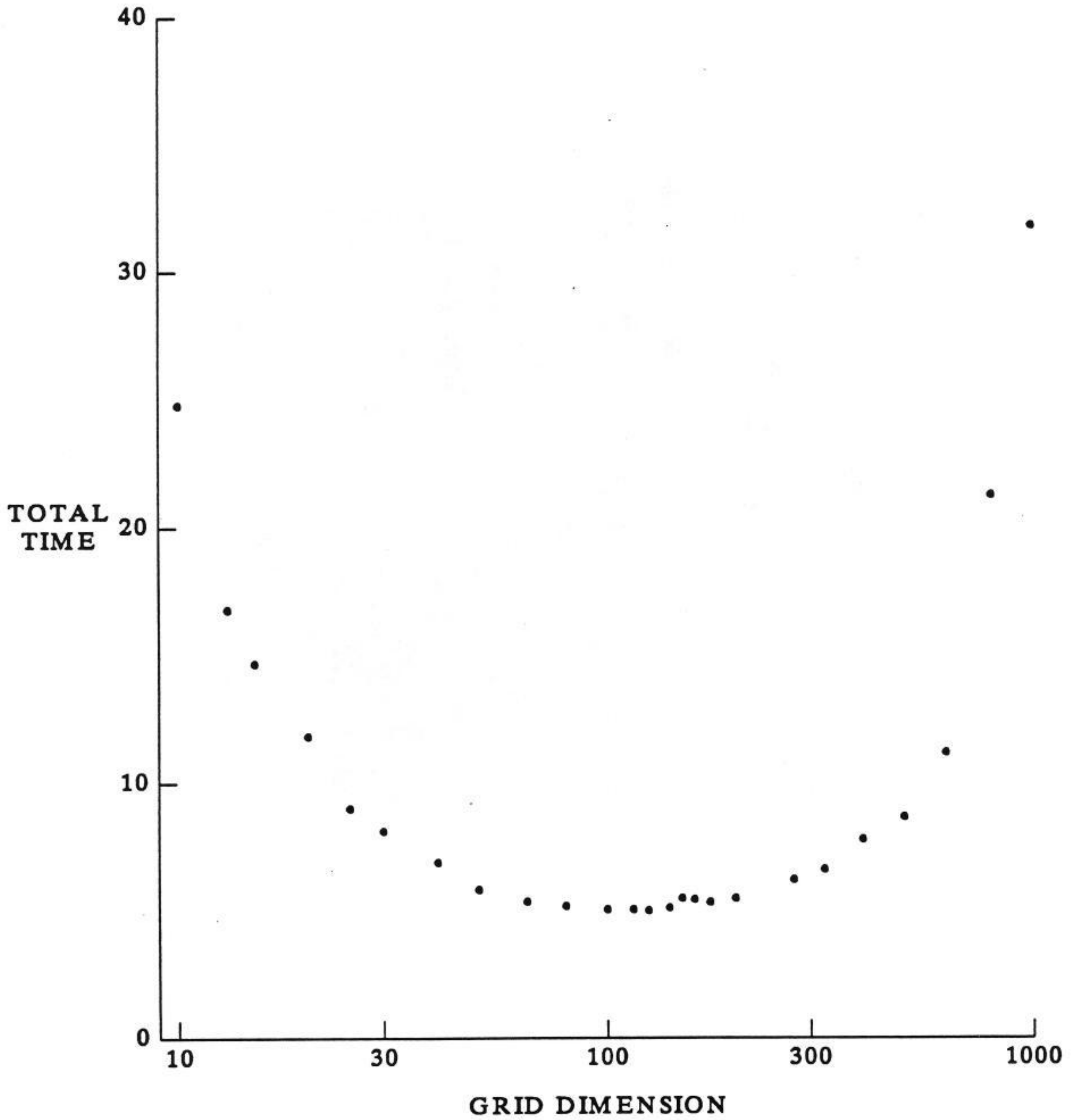
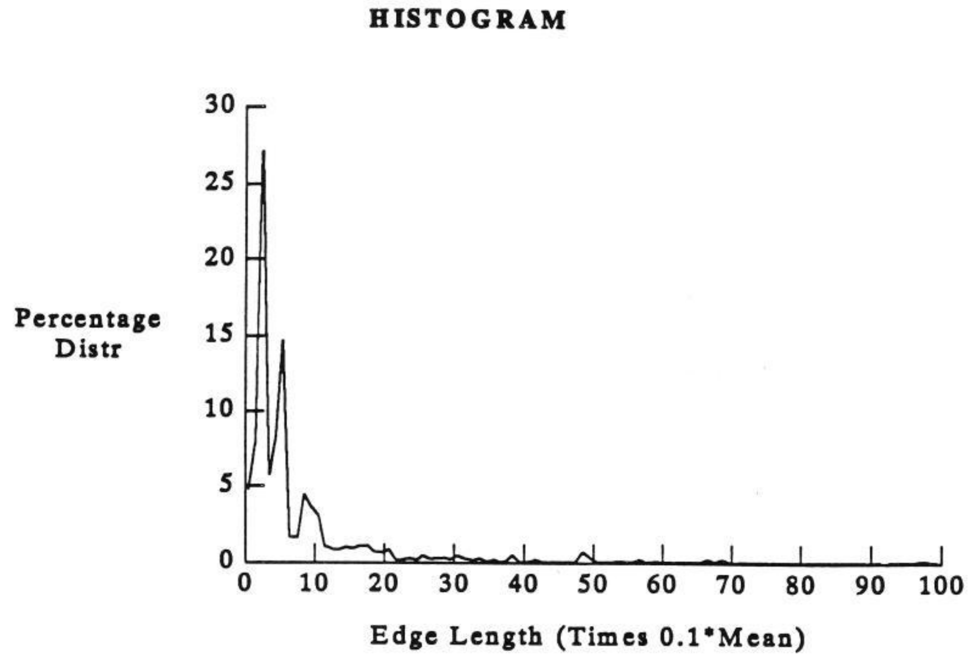


Fig 2. Graph of Time vs Grid Resolution for the USA Overlaid on Itself

VLSI file (windowed between 0.2-0.8 in x and y dir)



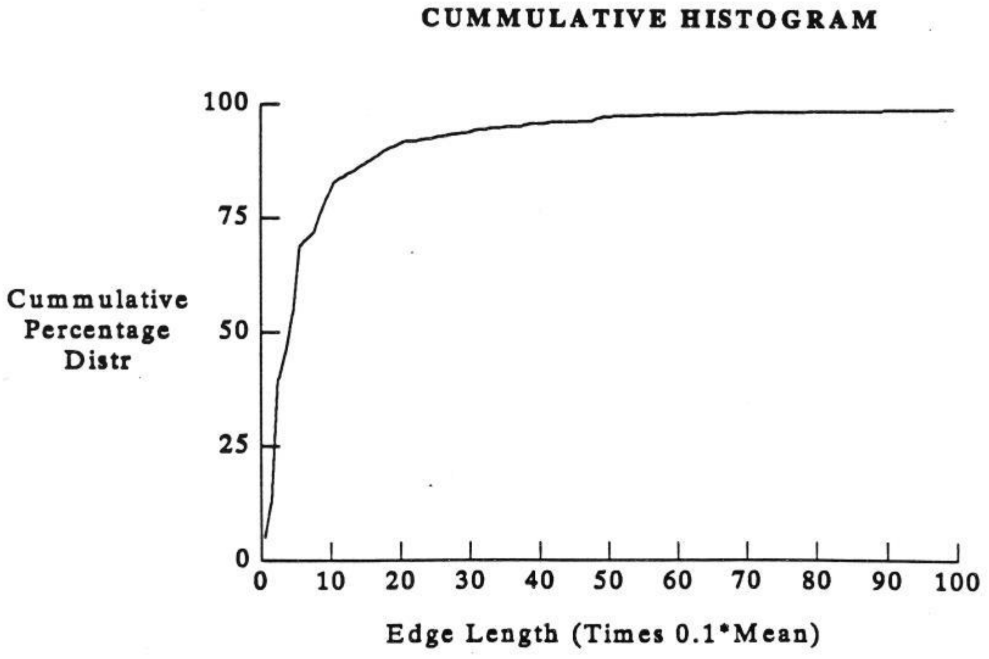
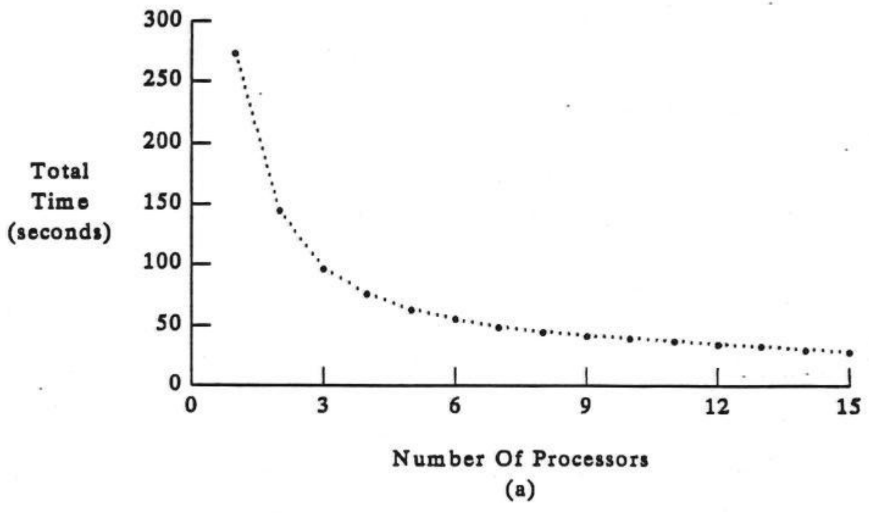


Fig 3. Histogram of edge lengths of database shown in Fig. 1(d)



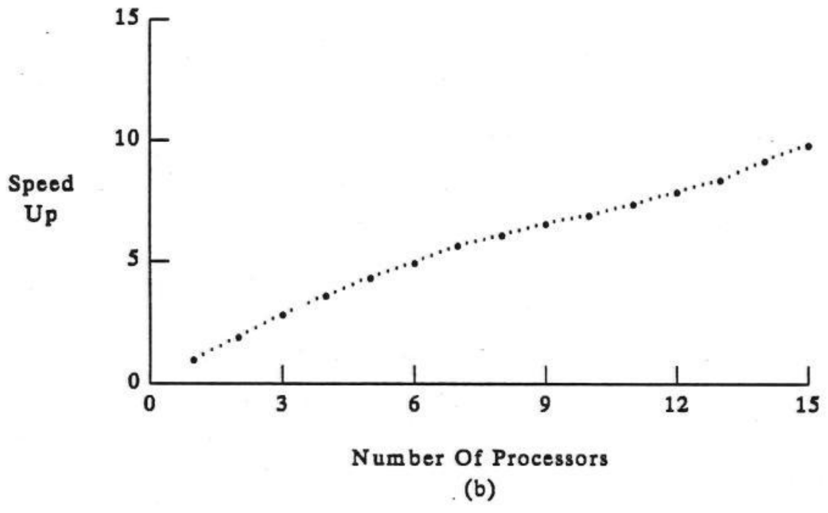


Fig 4. Time and Speedup When Intersecting the 62045 Edges in the Roads & Trails, Railroads, and Pipes and Transmission Lines Overlays of the Chikamauga DLG in Parallel on 1 to 15 Processors. Grid Size = 250. 81,373 Intersections Found.

Database	Edges	Length	Std Dev	Xsects	Grid Size	Time
Resch Egg - YZ Projection	5897	0.0355	0.0124	39666	40	7.27
XZ Projection	5897	0.0391	0.0132	37415	50	7.65
XY Projection	5897	0.0352	0.0131	40177	40	7.02
USA Map	915	0.0186	0.0245	1078	50	0.25
Shifted by 2% and overlaid on itself	1830	0.0184	0.0243	2430	65	0.65
Shifted by 10% & overlaid	1830	0.0180	0.0237	2348	50	0.60
Chikamauga Area 1 - Hydrography, Roads & Trails	14006	0.0045	0.0087	15513	150	3.50
Area 2, HR&T	14145	0.0049	0.0080	16595	125	3.47
Area 3, HR&T	18092	0.0044	0.0061	23586	150	4.83
Area 4, HR&T	16425	0.0048	0.0076	20335	150	4.17
Area 5, HR&T	12869	0.0053	0.0103	14978	140	3.28
Area 6, HR&T	13871	0.0050	0.0080	16072	140	3.42
Area 7, HR&T	13578	0.0048	0.0083	16066	140	3.28
Area 8, HR&T	11937	0.0048	0.0098	13283	125	2.80
All sections - Railroads	1122	0.0159	0.0543	1316	65	0.33
Pipe & Trans. Lines	850	0.0277	0.0523	1211	65	0.33
Railroads, Pipe & Trans. Lines	1972	0.0206	0.0533	2745	65	0.85
Railroads, Pipe & Trans. Lines Overlaid on itself	3944	0.0206	0.0533	13268	65	2.87
Hydrography, Railroads, Pipe & Trans. Lines	56823	0.0022	0.0122	62108	400	15.90
Roads & Trails, Railroads, Pipe & Trans. Lines	62045	0.0026	0.0106	81373	325	19.18
Hydrography, Roads & Trails, Railroads, Pipe & Trans. Lines	116896	0.0022	0.0115	144666	325	37.15
VLSI Data - xfacell.mag	1960	0.0467	0.0852	6488	65	0.82
VLSI Data - Windowed between 0.2-0.8	1,000,000	0.0012	0.0038	2,010,564	600	293.75

Table 2. Summary of Results for Serial Computation

No. of edges	116896
Avg. edge length	0.00231
Standard deviation	0.0081
Xsects. by end pt. coincidence	135875
Xsects. by actual equation soln	8791
Total intersections	144666

Grid Size	Pairs	P/Cell	P/Edge	Grid Time	Sort Time	Xsect Time	Total Time
50	131462	52.585	1.125	4.33	3.67	182.04	190.04
80	140407	21.939	1.201	4.50	3.93	90.75	99.18
100	146389	14.639	1.252	4.72	4.22	67.31	76.25
125	153492	9.823	1.313	4.88	4.32	51.36	60.56
175	168341	5.497	1.440	5.43	4.82	36.22	46.46
200	175791	4.395	1.504	6.70	6.13	35.18	48.01
275	197815	2.616	1.692	8.45	7.68	31.78	47.91
325	212282	2.010	1.816	7.37	6.18	23.60	37.15
400	234372	1.465	2.005	8.37	7.15	21.82	37.33
500	263646	1.055	2.255	10.18	7.78	20.62	38.58
625	300413	0.769	2.570	11.72	8.92	20.22	40.85
800	351891	0.550	3.010	14.52	10.77	21.37	46.65
1000	410589	0.411	3.512	17.72	12.93	23.05	53.70
2000	704147	0.176	6.024	31.05	22.92	29.57	83.53

Table 1. Results from Intersecting the 116896 Edges in the Roads & Trails, Railroads, and Pipes and Transmission Lines Overlays of the Chikamauga DLG

Database	Edges	Xsects	Grid	Time Taken For			
			Size	1 Proc	5 Procs	10 Procs	15 Procs
Resch 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
Hydrography, Railroads, Pipe & Transmission Lines	56823	62108	250	255.80	54.62	29.43	21.01
Roads & Trails, Railroads, Pipe & Transmission Lines	62045	81373	250	273.11	62.98	39.42	27.77
Hydrography, Roads & Trails, Railroads, Pipe & Trans. Lines	116896	144666	100	1292.06	302.46	160.69	120.04
Random Edges of Size 0.01	50000	45719	100	521.06	108.90	57.88	40.15

Table 3. Summary of Results for Parallel Computation