# Stability of Map Topology and Robustness of Map Geometry

Alan Saalfeld
Statistical Research Division
Bureau of the Census
Washington, DC 20233
(301) 763-4506

## ABSTRACT

A map's topology and its geometry are grounded in the mathematical theory of cellular structures on continuous surfaces. We say that the map's geometry, the actual physical positioning of the features on a surface, is a realization of the map's topology, which refers to the relative positioning of the features. A single topology has many geometric realizations, any two of which are related by some "rubber-sheeting transformation." Any computerized implementation of the geometry of a map, however, requires a discrete approximation of the point data of the map and a rounding of geometric positioning of those points. The implicit or explicit rounding required will re-position the points, which may in turn change the topology and give rise to topological inconsistencies or topological uncertainties.

In this paper we review the mathematical model of a map as a continuous surface with cell decomposition, and we examine a discrete-location/linear submodel which correctly models the computer's approximation to the continuous surface model. We describe the submodel, its relation to the larger model, special limitations of the submodel, and special useful properties of the submodel. In particular, we derive useful measures of stability of realizations of the submodel.

Because linear features in the submodel are straight line segments, the stability of the topology (or the robustness of the geometry) of a particular discrete-location/linear map realization turns out to be simply the upper bound distance that any line-segment endpoint may move in any direction and still not change the topological structure of the map. Looked at in the more general context, robustness is a geometric measure of the closeness of features on a map. Robustness is also a measure of the closeness of other maps having a one-to-one correspondence of point features, but having different topologies. This paper describes how to compute the geometric robustness of a particular geometric realization of a map and how to improve the topological stability. It also examines changes in stability that occur under various map update routines and transformation procedures. It proposes means of modifying or restricting those routines and procedures to preserve stability or to recover stability when it is diminished by those procedures.

The Mathematical Model.

A standard or usual mathematical model for a map is the two-dimensional manifold or surface with a finite cellular decomposition. A two-dimensional manifold is a continuous, infinitely subdividable space such that every non-boundary point has a neighborhood that looks like a small disk in the plane, and every boundary point has a neighborhood that resembles a half disk. A cellular decomposition of a manifold is a partition of the space into mutually disjoint subsets, each of which is topologically equivalent to a point, an open interval, or an open disk (possibly with holes*). The partitioning subsets are called cells; and those cells which consist of a single point are called 0-cells; those which are topologically equivalent to an open interval are 1-cells; and the two-dimensional disks are called 2-cells. Partitioning means that every point in the map sheet belongs to exactly one of the cells in the finite collection. In other words, the union of the cells exhausts the space; and the cells themselves are pairwise disjoint. In order to guarantee that the cells do not overlap and that they fit together properly, the cells are defined in such a way that their boundaries do not belong to the cells themselves, but instead are made up of cells of lesser dimension. The cells must fit together with a plane-like smoothness and fill up the space. The rules for fitting together constitute the basis for the topological edits. Those rules are (1) combinatorial (i.e. describe finite relations among finite sets), and hence are machine-verifiable and (2) form a complete set of axioms for the theory of cellular structures on surfaces.

The Submodel.

A submodel of a mathematical model places additonal constraints on the model components, in our case the cells; and thereby, it reduces the number of legitimate instances of the model that must be considered.

The 1-cells in the usual topological manifold model are arcs or smooth curves. In practice, 1-cells are stored and displayed as polygonal lines, or polylines. The practice is based upon mechanical and mathematical constraints. Machines draw straight lines more easily than curved lines; and piecewise linear approximations are satisfactory approximations from a visual as well as a theoretical viewpoint. "Piecewise linear" is more suitable computationally for algorithm development; and piecewise linear can be as close as desired, certainly within machine precision constraints. For our submodel, we allow only polylines for our 1-cells.

* *Some authors require that the 2-cells be simply-connected (no holes). This exposition does not. In fact, the structure of non-simply-connected surfaces with well-behaved singularities at the boundary is well known and is the basis for our understanding of our elementary 2-dimensional building blocks, the punctured 2-cells.*

The 0-cells in the usual topological model may take any real coordinate values on a surface that has infinite divisibility. In any implementation, however, machine precision will force the values into some finite grid. For our submodel, the 0-cells and the polyline interior vertices must have coordinates in some finite grid.

Our submodel puts considerable restrictions on the 0-cells and 1-cells; and one might ask if our submodel is as good as the general topological manifold for representing maps. In a very important sense, it is better for representing digital maps: Every computer implementation of a digital map is an instance or realization of our submodel; and many of the difficulties arising from machine precision constraints, such as topological uncertainty under transformation, can be better understood in the context of our finite-grid/ polyline (or discrete-location/linear) submodel.

While our 0-cells can come from only a finite set (in any particular instance, where the grid is given explicitly or implicitly), the points on our 1-cells are infinite in number. We keep track only of the vector ends of the segments making up the polylines; but our mathematical model requires that all of the points on a line segment be locatable, even though they cannot be explicitly stored.

Every instance of our submodel is also an instance of the more general topological manifold model; hence, we may use special properties of the submodel structure or use general properties of the larger model as needs arise. We examine topological stability in both contexts.

## STABILITY

Continuous Deformation.

The mathematical notion of continuous deformation is just a formal representation of the intuitive concept. For a surface or manifold, S, in a space, K, a continuous deformation over time T is simply a continuous map:

$$\Phi: S \times [0,T] \dashrightarrow K$$

satisfying $\Phi(s,0) = s$, which says intuitively that, at time zero, every point is in its original position.

For each intermediate value of t in [0,T], we have the image of the ongoing deformation of S at time t given by:

$$\Phi(S \times \{t\}).$$

Continuous deformations need not preserve topological properties of S at each stage t. In other words, the intermediate image, $\Phi(S \times \{t\})$, may be topologically different from S. (It may even shrink to a single point if $\Phi$ is a contraction!) If S has a cell structure, then that cell structure may induce the same, a different, or no cell structure on the intermediate image, $\Phi(S \times \{t\})$.

We want to examine deformations that "almost always" preserve some cell structure on S (for all but finitely many values of t in [0,T]). Then we will be able to recognize when a deformation has changed the cell structure.

We also want to be able to distinguish small deformations from large deformations by looking at the distances through which the deformations move points. This is accomplished by limiting the maximum path length allowed in our deformations, where path length for each point s in S is the length of the arc:

$$\Phi(\{s\}\times[0,T]).$$

The class of all continuous deformations of our manifold is much too large to use for our study of stability. Moreover, this large class contains many exotic maps under which our cell structures become immediately unstable for all t > 0. In order to study stability, we examine families of deformations which do not move points too far and which move neighboring points in similar directions across similar distances. These deformations will be defined by their action on a finite set of points and extended in a piecewise linear manner to the whole space.

Our goal in this short paper is to study stability, not to develop a theory of interesting deformations. So without further elaborating on the theory behind the class of deformations described above, we simply point out that the deformations are defined for all instances of the larger continuous model and hence for all instances of the submodel. However, the intermediate image, $\Phi(S\times\{t\})$, of the deformation of an instance of the finite-grid/polyline submodel will not always be an instance of that submodel. Nevertheless, this intermediate image will always be an instance of the polyline submodel because of the piecewise-linear nature of the allowable deformations!

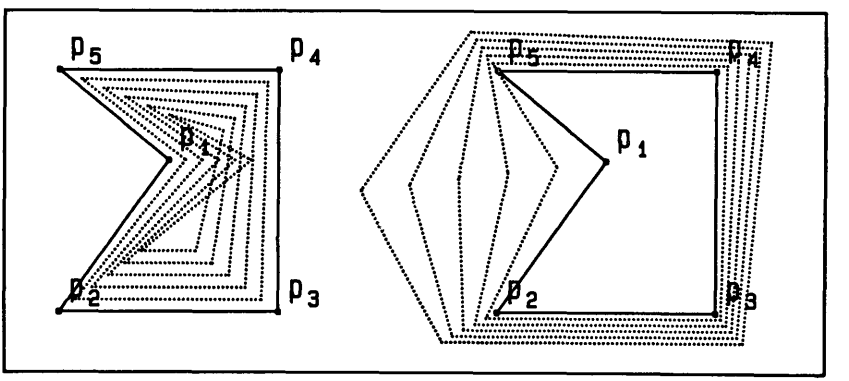


Figure 1. Illustrations of five intermediate deformations of polyline map portions

Notice in Figure 1 that the intermediate deformations on the left eventually change the cell structure when the lines double over on themselves. As the point $p_1$ moves to the right it gets closer to the linear feature $p_3p_4$, which itself is simultaneously moving to the left.

The initial polygon, however, becomes more stable if it is deformed as shown on the right. In the right-hand deformation, the features move toward an equilibrium position in which they are in some sense "as far from one another as possible." The "best" shape that they could attain in this simple example is a regular pentagon. The "good" deformation on the right is achieved by sending the vertices in just the opposite directions as in the "bad" deformation on the left.

The two deformations depicted in Figure 1 in some sense embody the basic ideas concerning stability:

(1) Stability is threatened when point features move toward nearby or nearest non-adjacent line segment features (and may then possibly cross over them!)

(2) Stability is improved when point features move away from nearby or nearest non-adjacent line segment features.

In the general situation of the 0-cells, 1-cells, and 2-cells of a map, however, the features are surrounded by other features; and movement is constrained in all directions:

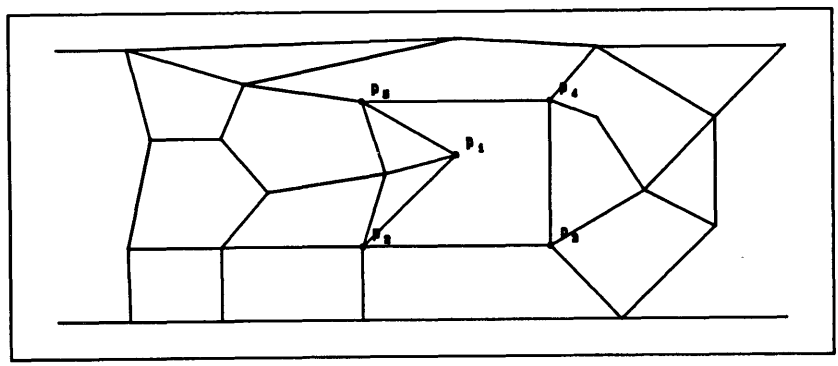


Figure 2. Cells in more complex map example.

In the example in Figure 2, the point $p_1$ is now further constrained by the additional features around it. That point is no longer free to move to the left to fill out the pentagon unless the points on the left of it move further to the left. There are two approaches that one may take with the general situation; and they correspond merely to assessing how good or bad the situation is, or to describing how to improve the situation. The easier first approach we will call "measuring robustness."

## ROBUSTNESS

Robustness of a statistical estimator is a quality of permanence and reliability under varying conditions. We borrow the notion from the area of statistics, and we apply it to our geometric realizations of our map data. In statistics, an estimator is robust if it can withstand relatively large perturbations in the statistical data. For our application, a geometric realization of a specific cell configuration will be said to be a robust realization if it can tolerate considerable perturbation of feature positions without changing the cell structure.

If we ask how "bad" is the particular configuration, and where is it "worst," we may want to find one feature and the minimum distance we may perturb that feature (toward the nearest non-adjacent feature) to change cell structure. (Equivalently, we can ask for the least upper bound of distances that we can move all of the features simultaneously and still not change the cell structure.) If we ask how "good" can we make the map, we are asking the more difficult question of how to move all of the features simultaneously to a "best" or in some sense "most stable" position. That second problem appears to be much more formidable than the first, and rather like the classic unsolved n-body problem. We can, indeed, treat the problem as a force problem, and achieve interesting stability results. First, we will examine the easier problem of determining how unstable a feature configuration is and where the instability is worst by locating nearest non-adjacent feature pairs.

## ROBUSTNESS AND INSTABILITY MEASURES

The following result regarding line segments is the key attribute that makes the finite-grid/linear submodel superior for studying instability.

(1) The minimum distance between two non-interesecting closed line segments is always attained by a pair of points, at least one of which is an end-point of one of the line segments.

This fact is easily seen and easily proved; however, the very important ramification of the fact is that computing distances between features in a polyline submodel boils down to computing point-to-line-segment distances, which are easy to compute.

If our topological data is stored in a TIGER-like file that "builds neighborhoods" in $O(N_f)$ time, where $N_f$ is the number of cells in the neighborhood of a feature $f$, then the following algorithm will detect the nearest pair of features in $O(\Sigma(m^2))$ time where $m$ is the polyline vertex count in each 2-cell, and the sum is over all 2-cells. The algorithm will also find the nearest segment to every point feature (0-cell or polyline vertex), and may be modified to yield nearest segment-to-segment distances using the fact (1) stated above.

Algorithm for computing robustness measures.
The minimum distance between a pair of features and the minimum distance from each vertex to a neighboring non-adjacent segment may be computed as follows:

INPUT: 0-cells, 1-cells, 2-cells, polyline vertex and polyline segment identifiers, and coordinates for 0-cells and polyline vertices.

OUTPUT: Closest-pair(a,b,c); where
a is a 0-cell or a polyline vertex identifier;
b is a polyline segment identifier; and
c is the distance between them.

Nearest-segment-to-_x_=(b,d); where
x assumes every 0-cell or a polyline vertex identifier;
b is the nearest polyline segment's identifier; and
d is the distance between them.

```
PROCEDURE NEAREST

Initialize Closest-pair(a,b,c) to any polyline vertex,
any polyline segment, and their distance.

FOR every 0-cell or 1-cell f DO

   Collect in a buffer all of the features that lie in
   the smallest closed neighborhood N_f of f

   IF f is a 0-cell, THEN DO

      Initialize Nearest-segment-to-f to any
      non-adjacent segment and compute distance

      FOR each non-adjacent polyline segment in N_f DO

         Compute distance to segment and update
         Closest-pair(a,b,c) and Nearest-segment-to-f,
         if necessary.

   ELSE DO

      FOR each interior polyline vertex v of f DO

        Initialize Nearest-segment-to-v to any
        non-adjacent segment and compute distance

        FOR each non-adjcnt. polyline segment in N_f DO

           Compute distance to segment and update
           Closest-pair(a,b,c) and Nearest-segment-to-v,
           if necessary.

END PROCEDURE NEAREST
```

For most map inputs with polygons having relatively few components, the buffering step of collecting all features of $N_f$ may be done in an array for faster processing.

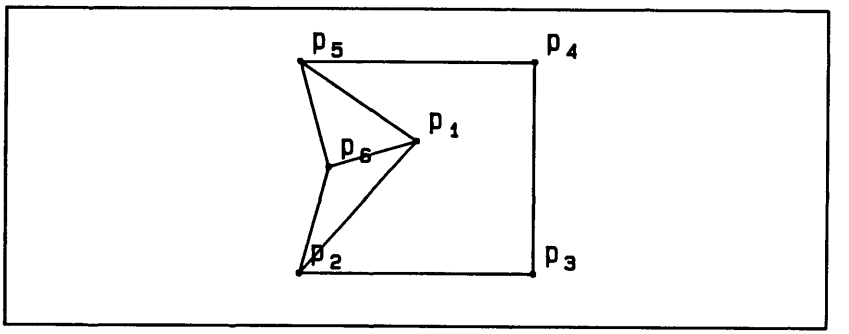


Figure 3. Smallest closed neighborhood of $p_1$.

## RECOVERING STABILITY

Because the above algorithm examines every non-adjacent segment in the smallest closed neighborhood of every point feature (0-cell or interior polyline vertex), we may modify the algorithm to have it compute a net "force" of all of those non-adjacent segments on each point feature instead of having it merely locate the nearest segment, by making the following change:

Replace:

> Compute distance to segment and update
> Closest-pair(a,b,c) and Nearest-segment-to-f (or v),
> if necessary.

By:

> Compute the force on point feature due to segment
> and add to net-force-on-f (or v).

Since the non-adjacent segments of the smallest closed neighborhood surround the point feature and in some sense isolate the point from effects of other segments, it makes sense to use this force model. As with the n-body problem, we can compute a force on each of our vertices in our initial configuration. We may model the forces on a vertex to be inversely proportional to the distance of points on neighboring non-adjacent segments.

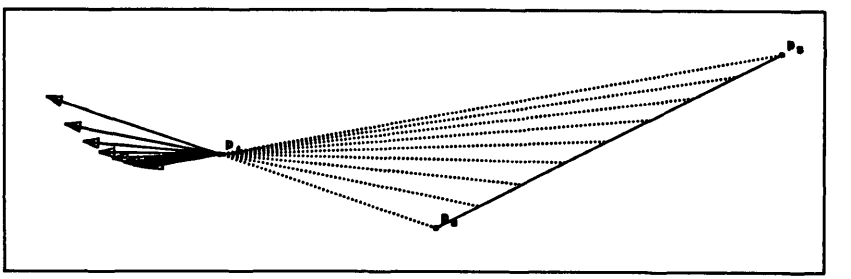

Figure 4. Forces exerted on a point by a line segment.

We may sum the forces by a straightforward vector integration. The result will be a vector whose magnitude and direction provide a best initial direction and speed to move our vertex in order to improve stability. As with the n-body problem, computing initial forces is not difficult. The hard part is determining the movement of the system, and, in our case, finding the eventual equilibrium position. We may simulate the movement by iterative linear approximations; and, perhaps surprisingly, that approach looks promising.

## FUTURE DIRECTIONS

The usual drawbacks to iterative methods are cost and convergence. Some experimentation is required to learn more about convergence, but our finite grid submodel promises to be extremely useful in establishing a bound for tolerances to replace "exact or total stability."

Cost also remains managable. Because the force computation is local, depending only on the smallest closed neighborhood, $N_f$, we can achieve, for all size maps having approximately the same local neighborhood configurations, a linear (in the number of point features) force computation algorithm. This possibility makes an iterative approach to stability improvement seem reasonable for large maps as well as for small maps. We plan to do more experimenting with iterative approaches to stability improvement.

## REFERENCES


Aho, A. V., J. Hopcroft, and J. Ullman, 1983, Data Structures and Algorithms, Addison-Wesley, Reading, MA

Corbett, James, 1979, Topological Principles in Cartography, Bureau of the Census Technical Paper 48.

Dugundji, James, 1966, Topology, Allyn and Bacon, Inc., Boston.

Guibas, Leonidas and Jorge Stolfi, 1985, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," ACM Transactions on Graphics, Vol. 4, No. 1 (April), pp 74-123.

Pavlidis, Theo, 1982, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, MD.

White, Marvin, and Patricia Griffin, 1979, "Coordinate Free Cartography," AutoCarto IV, Vol II, Proceedings of the Fourth International Symposium on Cartography and Computing, pp. 236-245.

White, Marvin, 1984, "Technical Requirements and Standards for a Multipurpose Geographic Data System, The American Cartographer, Vol. 11, No. 1, pp. 15-26.