

CHAPTER VIII

IMPLEMENTATION

8.1 Overview

In this chapter we discuss a variety of details concerning the implementation of our cartogram method. Code was developed in the MicroStation Development Language, a standard C programming language that compiles and executes within the MicroStation CAD program (Bentley Systems, Incorporated).

8.2 Angle Calculation Between Invertible Edges

The angle between two adjacent edges is given by

$$\varphi = \cos^{-1} \left(\frac{\mathbf{e}_1 \cdot \mathbf{e}_2}{\|\mathbf{e}_1\| \|\mathbf{e}_2\|} \right), \quad (8.1)$$

where \mathbf{e}_1 and \mathbf{e}_2 are edge direction vectors outward from a common vertex. This formulation only computes the *minimum* angle between the two vectors, thus effectively limiting the range to $[0^\circ, 180^\circ]$. However, it is not unusual in our method to encounter complex situations where a hinge inverts by closing below 0° or opening beyond 360° .

We can compute the “true” angle between a pair of adjacent edges by considering both the cross product of edge direction vectors and the angle at the previous time step. We define two edge direction vectors, \mathbf{d}_1 and \mathbf{d}_2 , that point from each edge’s origin to its endpoint, as depicted in Figure 8.1 (recall that the region vertices are ordered clockwise).

Using φ from equation 8.1 we compute the angle θ_t at time t as

$$\theta_0 = \begin{cases} \varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \otimes \\ 360^\circ - \varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \oplus, \end{cases} \quad (8.2)$$

$$\theta_t = \begin{cases} -\varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \oplus, \theta_{t-1} \leq 90^\circ \\ \varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \otimes, \theta_{t-1} \leq 270^\circ \\ 360^\circ - \varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \oplus, \theta_{t-1} > 90^\circ \\ 360^\circ + \varphi, & \mathbf{d}_1 \times \mathbf{d}_2 = \otimes, \theta_{t-1} > 270^\circ, \end{cases} \quad (8.3)$$

where \otimes and \oplus are directions into and out of the page, respectively.

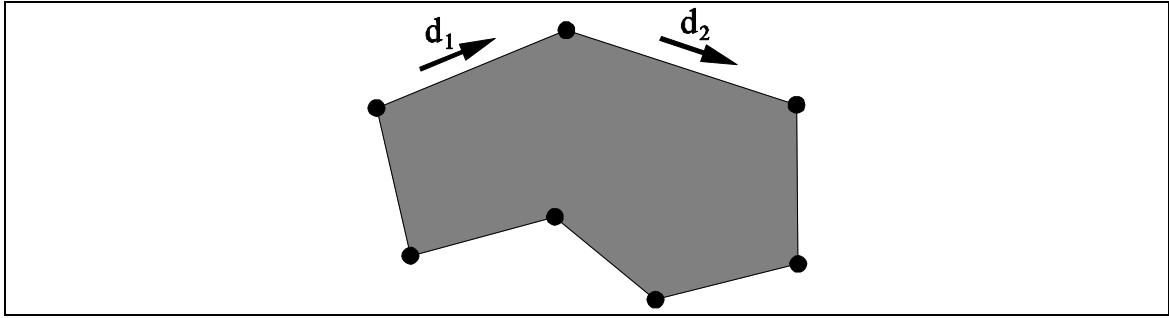


Figure 8.1: Direction vectors along two adjacent edges.

We assume that the pair of edges is initially non-inverted, and calculate the initial angle using equations 8.2 by using the cross product to determine whether the angle falls in the range 0 to 180° , or 180 to 360° . Equations 8.3 are used for subsequent time steps. The first case detects edges that have closed below 0° . The inversion is evident because of the change of direction of the cross product. The next two cases differentiate the 0 to 180° or 180 to 360° ranges. The final case detects edges that have opened beyond 360° , since *valid* angles above 270° will always result in a \oplus direction. This formulation correctly computes angles within the range $(-180^\circ, 540^\circ)$, as long as edges are restrained from jumping quadrants during successive time periods.

8.3 “Windowed” Adaptive Stepsize

Our computational method utilizes a “windowed” adaptive stepsize version of the Euler discrete timestep ordinary differential equation (ODE) solver. The user initially specifies a window of the minimum and maximum threshold forces to be allowed upon a vertex during a single timestep. During the cartogram construction, the timestep is adjusted *down* if the force is above the threshold and *up* if the force is below the threshold in order to keep the total impulse applied to a vertex in a single timestep within reasonable bounds. This prevents instability while also making the most out of each iteration.

8.4 Scale Factor Between Angle and Unit Length

All area, shape, and constraint forces within our system are applied relative to a deviation in either *angle* or *unit length*. Finding the correct balance in magnitude between these two types of forces can be difficult. Once a good balance has been found, however, any resizing of the initial map will tilt the balance. The problem lies in the fact that the magnitudes of length-based forces are directly related to the map size, whereas angle-based forces are not.

Our method balances the two types of forces using a scale factor K_{map} as a percentage of the diagonal length of the initial map's bounding box. This scale factor is multiplied by force in every length-based force calculation to ensure the force magnitude is scaled properly relative to the angle-based forces.

8.5 Hierarchical Resolution Level of Detail

Simplification of a region is based on a coarseness level, also computed as a percentage of the diagonal of the region's bounding box. Large regions will allow greater offsets from the simplified lines than smaller ones. Since adjacent regions share common edges and vertices, the order of region traversal during simplification can affect the level of detail of the map.

Prior to the initial simplification, we sort the regions by ascending area so that smaller regions have a chance to be simplified at their level of coarseness. This means, however, that larger regions will be conglomerates of varying levels of detail. Conversely, the regions could be put in descending area order so as to reduce the number of simplified edges, but we have not experimented with this approach.

Another factor affecting the simplification level of detail is the order of vertex traversal. During a simplification between two key points, as documented in section 5.2, the simplified edge endpoint is stepped back one vertex until all inner vertices are within the allowed offset distance. The result from a clockwise simplification can be quite different from that obtained with a counter-clockwise ordering. In implementation, the

simplification routine could analyze both results and choose the result that minimizes the number of edges, for example. Again, we have not experimented with this approach.

8.6 Ensuring Clockwise Vertex Traversal Order

In order to maintain the mathematical integrity of our method, directional and constraint calculations require that the list of vertices within each region be ordered in a clockwise fashion. Counter-clockwise regions are easily identified since they produce a negative result from the area calculation in equation 4.1. In our implementation, upon building the map we simply reverse the list of vertices for all regions with a negative area.

8.7 Stagnation

In our method, the goal of achieving desired region areas continues as long as the average area error continues to decrease. Once the average error increases, the system is considered “stagnated” and the focus is shifted to the second goal of restoring shape. Similarly, we use a stagnation indicator to signal the end of shape restoration and another indicator to advance the hierarchical resolution to a more detailed level of coarseness.

In practice, however, average error can fluctuate throughout the cartogram process. For example, area error can rise suddenly due to topological constraints upon an oversized region. Therefore, we define stagnation as a steady increase over a small period of time. For the area and shape goals in our implementation, seven consecutive iterations of increasing error signals stagnation. In hierarchical resolution, we reconstruct and resimplify the map when the resulting map from the area-adjusting process contains more shape error than the previous result. In our current implementation we simply reconstruct and resimplify following the second increase in shape error.

8.8 “Smoothing” Area Forces

A undesired artifact can appear while resizing a region containing a concave section. When area forces are enlarging a region, such as in Figure 8.2a, a concave bay area will actually *constrict* while the region enlarges. Conversely, a bay area will *expand* while area forces drive the region smaller, as shown in Figure 8.2b. While hierarchical resolution usually simplifies away relatively small concave areas, large concave regions are still a possibility.

Our implementation contains a “smoothing” routine to adjust for concave regions that at the first level of coarseness are not simplified away. At the user’s request, area forces upon a vertex can be recalculated as a weighted average of nearby vertices. The weight of a vertex Q_i upon a chosen vertex P can be found as

$$w_i = \begin{cases} 0 & \|P - Q_i\| > r, \\ \frac{r - \|P - Q_i\|}{r \sum w} & \|P - Q_i\| \leq r, \end{cases} \quad (8.4)$$

where r is a user-specified radius from the point P . The vector sum of all weights provides the final area force direction and magnitude. This process is repeated for each vertex on the map.

The smoothing effect is depicted in Figure 8.2c, with the resulting area forces on the concave bay area correctly moving the bay in unison toward the west. Note, however, that the main coastline points will move westward at a faster rate than the bay.

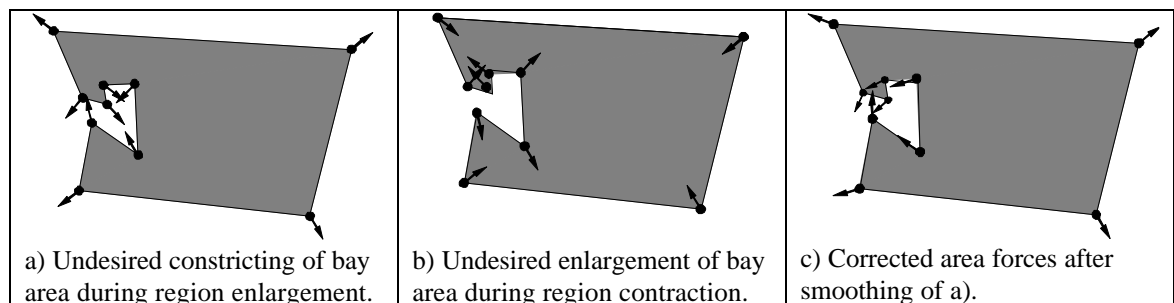


Figure 8.2: Smoothing area forces.

8.9 Data Structures and Input Files

Data structures used in our implementation of this method are included in Appendix A. Each region stores basic information about itself, as well as a list of pointers to the original and simplified vertices, an array of edges, a dynamic list of current intersections among its edges, and pointers to the next and previous region. All information required by the area springs and constraints are contained in the region data structure.

Each region edge stores basic information about itself, as well as links to its two vertices, its parent region, and the next and previous edge. All required information for the orientation springs and edge length proportionality springs is contained within the edge data structure. By setting the “previous” link of each region’s first edge to point to the last edge, the complexity of several algorithms was reduced.

Each vertex in the global point array stores essential information about itself, including a force accumulator to store applied forces. We have a separate list of key point pairs, with each pair containing a list of simplified edges between the key points. All essential reconstruction information for inner vertices that have been simplified is stored within each simplified edge node. We have a separate structure for the hinge constraints that provides links to their adjacent edge pairs.

Appendix A also includes the data structures for intersections, as well as the sparse matrix data structure used in the dynamic area constraints. A sample parameter file and map data file utilized in our implementation is included in Appendices B and C, respectively. Map data can also be converted directly from grouped MicroStation CAD elements, modeling an embedded geographical information system.