LINEAR-TIME SLEEVE-FITTING POLYLINE SIMPLIFICATION ALGORITHMS

Zhiyuan Zhao, Alan Saalfeld
Department of Civil and Environmental Engineering and Geodetic Science
The Ohio State University, Columbus, OH 43210, USA
zhao.29@osu.edu, saalfeld.1@osu.edu

ABSTRACT

We present three variants of a polyline simplification algorithm. The basic algorithm uses a variable angle tolerance measure to find maximal subsequences of vertices of the polyline that may be replaced by a single segment in a simplified approximation. In our key theoretical development, we prove that an easily implemented angle-testing procedure is locally equivalent to ε-buffering; then we demonstrate that we may iterate the angle-testing procedure to find a maximum sleeve (rectangular strip in 2-D) of width 2ε that starts at any vertex $\mathbf{p_i}$ and contains successive vertices $\mathbf{p_{i+1}}, \ldots, \mathbf{p_{j-1}}, \mathbf{p_j}$. The sleeve is maximum in the sense that it is the rectangular strip of width 2ε that covers the largest number **(j-i+1)** of consecutive vertices starting with $\mathbf{p_i}$. We proceed to build the longest possible sleeve from $\mathbf{p_0}$ to some $\mathbf{p_i}$, then from $\mathbf{p_i}$ to some $\mathbf{p_j}$, and so on, until we have covered the entire polyline with "long sleeves". The center-line (or a near-center-line) of each sleeve offers a one-segment approximation to the sub-polyline of the original polyline linking of all of the consecutive vertices inside the sleeve. The three variants of our basic algorithm are the result of using different criteria to "trim the sleeve".

BACKGROUND

Our approach to polyline simplification applies unconstrained local processes to the polyline to produce a simplified polyline that lies within a given prescribed distance ε of the original polyline. We process the vertices of the polyline in order; and at any stage in our processing, all vertices are partitioned into three subsequences: the first **i+1** vertices $\{\mathbf{p_0}, \mathbf{p_1}, \ldots, \mathbf{p_i}\}$, that have already been completely processed (i.e., each vertex has received its final classification of "in" or "not in" the simplified polyline), the next **k** vertices $\{\mathbf{p_{i+1}}, \mathbf{p_{i+2}}, \ldots, \mathbf{p_{i+k}}\}$, that belong to an active pool of vertices currently undergoing processing, and the final **n-i-k** vertices $\{\mathbf{p_{i+k+1}}, \ldots, \mathbf{p_n}\}$, that have yet to be processed. Our selection algorithms are local because they only operate on the pool of vertices $\{\mathbf{p_{i+1}}, \mathbf{p_{i+2}}, \ldots, \mathbf{p_{i+k}}\}$, adding one candidate vertex $\mathbf{p_{i+k+1}}$, at a time on the right and removing a variable number of vertices $\mathbf{p_{i+1}}, \mathbf{p_{i+2}}, \ldots$, from the left hand side of the pool (as soon as they have been classified as "in" or "not in" the simplified line). Because the pool size **k** is not bounded, our local algorithm is called "unconstrained" (McMaster, 1992). Instead of using a distance tolerance directly as our nearness threshold, we convert each distance tolerance into a variable angle tolerance for testing the vertices of our polyline. Our sequential process places each successive polyline vertex into a pool of candidates for

possible deletion or inclusion. As the pool of candidates grows, the angle tolerance (angle range) in which we search to find a candidate vertex that will allow us to delete the entire pool grows smaller. When the angle range becomes empty, then we must include in our simplified polyline at least one vertex from the candidate pool. Our methods behave like a local filter that throws away unnecessary vertices and retains necessary vertices for the simplified polyline.

In this paper, we first present a greedy algorithm that deletes polyline vertices as they are found to lie within locally computed angle tolerances (and keeps them in the simplified polyline when they are not). The greedy-deletion algorithm is simple and fast (linear running time), but may fail to delete some clearly unnecessary vertices from the approximating polyline. To overcome this drawback, we designed a second algorithm that postpones decisions about certain ambiguously situated vertices in the candidate pool. The second algorithm removes more vertices than the first algorithm, but the polyline with fewer vertices still successfully approximates the original polyline. The second variant, however, requires more complex processing of the pool values; and the worst-case complexity of the second algorithm is no longer linear. We finally offer a third variant that allows vertices to be perturbed slightly to obtain even simpler polyline representations within a prespecified threshold. This relaxation of the constraint on location of vertices for the simplifying polyline not only further reduces the number of output vertices, but also actually recovers the linear time complexity of the first variant.

We analyzed our basic and modified algorithms mathematically and compared them to the Douglas-Peucker and other algorithms. We also tested our methods empirically on real cartographic data and on special short vector data produced by vectorizing raster images into Freeman-code vectors. We found our method especially suitable for large, dynamic data sets and real-time processing because we do not need to store all original data at one time; and we do not need to preprocess data before simplification. Our sequential local processes produce a satisfying overall appearance of the output. Downloadable C++ code, a more extensive write-up of empirical test results of our algorithms, and an interactive Java demonstration have been set up on a web page (Zhao, 1996b)

MATHEMATICAL PRELIMINARIES

Distances and angles are two related geometric measures for determining point selection or rejection in polyline simplification. We present a ***sector bound*** as another geometric measurement useful in polyline simplification. Geometrically, a sector bound is a swept angle emanating from a distinguished point. Mathematically, it may be used to constrain segments emanating from that distinguished point to pass within a threshold distance of all points in a set of points. A sector bound is represented by two angles. It is easily computed and updated by local processes. With sector bounds we build polyline simplification

algorithms that process a polyline's vertices in order (locally) and yet produce a guaranteed satisfying overall approximation of the polyline.

Suppose that **p** and **q** are two points on the plane, which we write $\mathbf{p}, \mathbf{q} \in \mathbf{R}^2$. We will use the following notation:

$\mathbf{L(p,q)}$ denotes the directed line segment from point **p** to point **q**. (This means $\mathbf{L(p,q)} \neq \mathbf{L(q,p)}$.)

$\alpha(\mathbf{p,q})$ signifies the angle measured counter-clockwise from the positive X-axis to the directed line corresponding to the line segment $\mathbf{L(p,q)}$.

$\mathbf{d(p,q)}$ will represent the distance from a point **p** to the point **q.**

We define $\mathbf{d(p_1,q,p_2)}$ to be the perpendicular distance from point **q** to the line passing through points $\mathbf{p_1}$ and $\mathbf{p_2}$ ($\mathbf{p_1}, \mathbf{p_2} \in \mathbf{R^2}$). Notice that we write the vertices in an unusual order. The reason for this unusual choice is that in our polyline vertex sequence, the vertices will actually appear in this order; and the intermediate vertex **q** will be tested for significant displacement from the approximating segment joining neighbor points $\mathbf{p_1}$ and $\mathbf{p_2}$. We also have the following definition: The *Sector Bound* (or *swept sector*) $\mathbf{A(p, \alpha_1, \alpha_2)}$ of point **p** and angles α_1 and α_2 is a point set given by:

$$\{\mathbf{q} \in \mathbf{R}^2 \mid \alpha_1 \leq \alpha(\mathbf{p}, \mathbf{q}) \leq \alpha_2\}$$

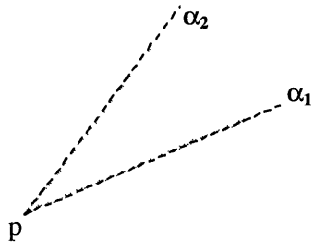

Figure 1. Sector bound

The sector bound $\mathbf{A(p, \alpha_1, \alpha_2)}$ of point **p** is described by the two angles: start angle α_1 and finish angle α_2 . Figure 1 shows a sector bound. We may always assume that α_1 is less than or equal to α_2, and the angle from α_1 to α_2 in the counter-clockwise direction is positive and less than 360°. For example, if α_1=350° and α_2=15°, then we express α_2 as 375° (=15°+360°) and α_1 as 350° to make the counter-clockwise difference positive to correspond to the usual order of the real numbers.

Lemma 2.1. The intersection of two sector bounds $\mathbf{A(p,\alpha_{11},\alpha_{12})}$ and $\mathbf{A(p,\alpha_{21},\alpha_{22})}$ which have the same initial point **p** is again a sector bound with same initial point **p**. $\mathbf{A(p, \alpha', \alpha'')} = \mathbf{A(p, \alpha_{11}, \alpha_{12})} \cap \mathbf{A(p, \alpha_{21}, \alpha_{22})}$, where $\alpha' = \boldsymbol{max}\{\alpha_{11},\alpha_{21}\}$, the larger of the start angles; and $\alpha'' = \boldsymbol{min}\{\alpha_{22},\alpha_{12}\}$, the smaller of the finish angles.

Proof: It is clear from the geometry in Figure 2. The intersection is $\mathbf{A}(\mathbf{p}, \alpha', \alpha'')$ with $\alpha' = \alpha_{21}$ and $\alpha'' = \alpha_{12}$. ■
If $\alpha' > \alpha''$, then we have that the intersection $\mathbf{A}(\mathbf{p},\alpha',\alpha'')$ is empty.

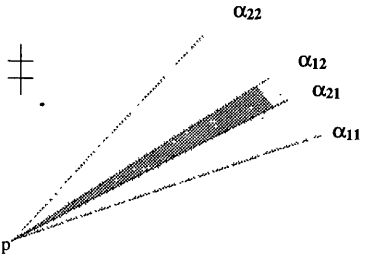

Figure 2. Intersection of sector bounds

Next we give the following definition: The *Epsilon Sector* $\mathbf{Q}(\mathbf{p}_1,\mathbf{q},\varepsilon)$ of point $\mathbf{q}$ from point $\mathbf{p}_1$ with threshold ε is a point set given by $\{\mathbf{p}_2 \in \mathbf{R}^2 | \mathbf{d}(\mathbf{p}_1,\mathbf{q},\mathbf{p}_2) \leq \varepsilon\}$.

Notice that $\mathbf{p}_2$ does not need to be near to $\mathbf{q}$. $\mathbf{p}_2$ only needs to determine a direction together with $\mathbf{p}_1$ that passes near to $\mathbf{q}$. A line through point $\mathbf{p}_1$ and a point $\mathbf{p}_2$ in $\mathbf{Q}(\mathbf{p}_1,\mathbf{q},\varepsilon)$ has the perpendicular distance from $\mathbf{q}$ no larger than ε. We can use the epsilon sector for polyline simplification. The segment from $\mathbf{p}_1$ to $\mathbf{p}_2$ is the simplified line segment and $\mathbf{q}$ is an original point. The point $\mathbf{q}$ has perpendicular distance $\leq \varepsilon$ to line $(\mathbf{p}_1, \mathbf{p}_2)$, hence point $\mathbf{q}$ can be deleted if $\mathbf{p}_1$ and $\mathbf{p}_2$ are selected. In our line simplification process, $\mathbf{p}_1$ is both the end point of the last accepted line segment in the sequential polyline building process and the starting point of the next simplified line segment that will be added to the already processed initial sequence of points. The three points are consecutive points of the original line. We can delete the point $\mathbf{q}$ if and only if the point $\mathbf{p}_2$ is in $\mathbf{Q}(\mathbf{p}_1,\mathbf{q},\varepsilon)$. Next, we will examine the relationship between the epsilon sector and the sector bound.

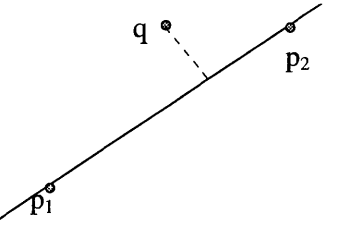

Figure 3. Epsilon sector

Theorem 2.1: When $\mathbf{d}(\mathbf{p},\mathbf{q}) \geq \varepsilon$, the Epsilon Sector $\mathbf{Q}(\mathbf{p},\mathbf{q},\varepsilon)$ is equivalent to the sector bound $\mathbf{A}(\mathbf{p}, \alpha_1, \alpha_2)$ with

$$\alpha_1 = \alpha(\mathbf{p},\mathbf{q}) - \delta,\ \alpha_2 = \alpha(\mathbf{p},\mathbf{q}) + \delta, \text{ where } \delta = \mathbf{sin}^{-1}(\varepsilon/\mathbf{d}(\mathbf{p},\mathbf{q})).$$

Proof: Suppose $\mathbf{v} \in \mathbf{Q}(\mathbf{p},\mathbf{q},\varepsilon)$. Then according to the definition of epsilon sector, we have $\mathbf{d}(\mathbf{p},\mathbf{q},\mathbf{v}) \leq \varepsilon$. Denote $\phi = \| \alpha(\mathbf{p},\mathbf{q}) - \alpha(\mathbf{p},\mathbf{v}) \|$, since $\mathbf{sin}\phi = \mathbf{d}(\mathbf{p},\mathbf{q},\mathbf{v})/\mathbf{d}(\mathbf{p},\mathbf{q})$, $\mathbf{sin}\delta = \varepsilon/\mathbf{d}(\mathbf{p},\mathbf{q})$, and $\phi, \delta \in [0°, 90°]$, we get $\phi \leq \delta$. That is: $\alpha(\mathbf{p},\mathbf{q}) - \delta \leq \alpha(\mathbf{p},\mathbf{v}) \leq \alpha(\mathbf{p},\mathbf{q}) + \delta$. According to the definition of sector bound, $\mathbf{v} \in \mathbf{A}(\mathbf{p},\alpha_1,\alpha_2)$, where $\alpha_1 = \alpha(\mathbf{p},\mathbf{q}) - \delta$, $\alpha_2 = \alpha(\mathbf{p},\mathbf{q}) + \delta$, $\delta = \mathbf{sin}^{-1}(\varepsilon/\mathbf{d}(\mathbf{p},\mathbf{q}))$.

Conversely, if $\mathbf{v} \in \mathbf{A}(\mathbf{p},\alpha_1,\alpha_2)$ with $\alpha_1 = \alpha(\mathbf{p},\mathbf{q}) - \delta$, $\alpha_2 = \alpha(\mathbf{p},\mathbf{q}) + \delta$, where $\delta = \mathbf{sin}^{-1}(\varepsilon/\mathbf{d}(\mathbf{p},\mathbf{q}))$, then $\alpha(\mathbf{p},\mathbf{q}) - \delta \leq \alpha(\mathbf{p},\mathbf{v}) \leq \alpha(\mathbf{p},\mathbf{q}) + \delta$. Thus, we see that $\phi = \| \alpha(\mathbf{p},\mathbf{q}) - \alpha(\mathbf{p},\mathbf{v}) \| \leq \delta$, since $\mathbf{sin}\phi = \mathbf{d}(\mathbf{p},\mathbf{q},\mathbf{v})/\mathbf{d}(\mathbf{p},\mathbf{q})$, $\mathbf{sin}\delta = \varepsilon/\mathbf{d}(\mathbf{p},\mathbf{q})$, and $\phi, \delta \in [0°, 90°]$, we get $\mathbf{d}(\mathbf{p},\mathbf{q},\mathbf{v}) \leq \varepsilon$. According to the definition of epsilon sector, we have $\mathbf{v} \in \mathbf{Q}(\mathbf{p},\mathbf{q},\varepsilon)$. ■

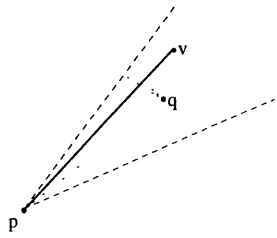

Figure 4. Sector bound and epsilon sector

This theorem tells us that an epsilon sector is geometrically equivalent to a sector bound. Since a sector bound is far easier to calculate and maintain, we will use the sector bound for polyline simplification to find new points of the simplified line segments. This means we may convert a global problem of epsilon sector determination into a local process of determining sector bounds. To determine if a point **q** is inside a given band of tolerance ε of a line segment from point **p** to point **v**, we determine if a point **v** is inside the sector bound $\mathbf{A}(\mathbf{p},\alpha_1,\alpha_2)$ which only necessitates calculation of the angle $\alpha(\mathbf{p},\mathbf{v})$.

Let consider a polyline, a sequence of line segments. Here we will use the symbol "$\cap$" (intersection) to represent the intersection set of epsilon sectors.

Theorem 2.2: Suppose a polyline has vertices $\{\mathbf{p_i} \mid \mathbf{i=0,1,\ldots k}\}$. Then there exists a point **q** such that all points $\mathbf{p_i}$ **(i=1, ... k)** have perpendicular distance to line $\mathbf{L(p_0,q)}$ within a given tolerance ε if and only if

$$\cap \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon) \neq \varnothing.$$

Proof: If such a point **q** exists, then all points $\mathbf{p_i}$ **(i=1, ..., k)** have perpendicular distance to line $\mathbf{L(p_0,q)}$ less than the tolerance ε, $\mathbf{d(p_0, p_i, q)} \leq \varepsilon$ **(i=1, ... k)**, that is $\mathbf{q} \in \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon)$ **(i=1, ... k)**. So $\cap \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon) \neq \varnothing$.

If $\cap \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon) \neq \phi$, suppose $\mathbf{q} \in \cap \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon) \neq \phi$, then $\mathbf{q} \in \mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon)$ **(i=1,... k)**, that is $\mathbf{d(p_0, p_i, q)} \leq \varepsilon$ **(i=1, ... k)**, So all points $\mathbf{p_i}$ **(i=1, ... k)** have perpendicular distance to line $\mathbf{L(p_0,q)}$ within the given tolerance ε. ■

Corollary 2.1: Suppose a polyline has vertices $\{\mathbf{p_i} \mid \mathbf{i=0,1,\ldots k}\}$. Then there exists a point **q** such that all points $\mathbf{p_i}$ **(i=1, ... k)** have perpendicular distance to line $\mathbf{L(p_0,q)}$ within a given tolerance ε if and only if

$\alpha' \leq \alpha''$, where $\alpha' = \mathit{max}\ \{\ \alpha_{1i} \mid \mathbf{i=1,...,k}\ \}$, and $\alpha'' = \mathit{min}\ \{\ \alpha_{2i} \mid \mathbf{i=1,...,k}\ \}$.

Here α_{1i} and α_{2i} are the two angles of sector bounds $\mathbf{A}(\mathbf{p},\alpha_{1i},\alpha_{2i})$ equivalent to $\mathbf{Q}(\mathbf{p_0}, \mathbf{p_i}, \varepsilon)$ **(i=1, ... k)**.

Proof: Combine Lemma 2.1 and Theorem 2.1 to get the conclusion. ■

This theorem obviously gives us a new opportunity for line simplification. For a polyline, the sector bound intersection is the only possible location for a subsequent point that can simplify the pending chain of points with a single segment. The second point of the simplified segment must lie inside this intersection. If this intersection is empty, there will exist no simplified segment that meets the distance tolerance condition. Figure 5 illustrates such a polyline.

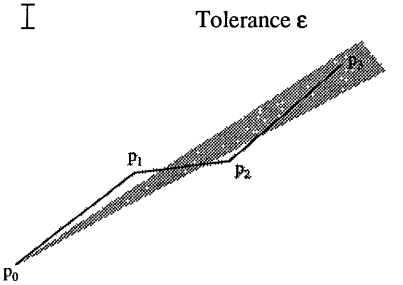

Figure 5. Epsilon bounds intersection of a polyline

In Figure 5, we have a polyline of points $\{\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}\}$. The simplified segment will start from point $\mathbf{p_0}$. For the given distance tolerance ε, the left-upper sector (shown as a triangle) is the epsilon bound $\mathbf{Q}(\mathbf{p_0},\mathbf{p_1},\varepsilon)$; the right-down sector (also shown as a triangle) is the epsilon bound $\mathbf{Q}(\mathbf{p_0},\mathbf{p_2},\varepsilon)$. They have a intersection shown as a darker area. Since this intersection is not empty, we can delete points $\mathbf{p_1}$ and $\mathbf{p_2}$ and use a point in the intersection to form a new line segment to approximate the original lines. We see point $\mathbf{p_3}$ is in the intersection, so line $\mathbf{p_0}$ to $\mathbf{p_3}$ is the simplified line of the original polyline.

Theorem 2.1 tells us that the sector bound and the epsilon sector are geometrically equivalent. We will see, however, that it is much easier to work with sector bounds than with epsilon sectors.

We are now ready to describe and examine our three algorithm variants.

A GEOMETRIC DESCRIPTION OF OUR ALGORITHMS

Imagine trying to fit a sleeve of width 2ε to the first $\mathbf{k}$ points in our polyline. If $\mathbf{k=2}$, then this is easy. Suppose that the sleeve fits the first $\mathbf{k}$ points; and we want to adjust it (if necessary) to fit the $(\mathbf{k+1})^{st}$ point as well. It is clear that there will only be a limited amount of "play" in the sleeve to realign it. It is also clear that if we keep the first point $\mathbf{p_0}$ in the center of the sleeve that the "play" in the sleeve will correspond to a swept angle, our sector bound above. Our mathematical preliminaries have guaranteed that we can slide the sleeve forward as far as possible with a very fast and efficient sector bound update computation.

The missing step in our polyline simplification routine handles

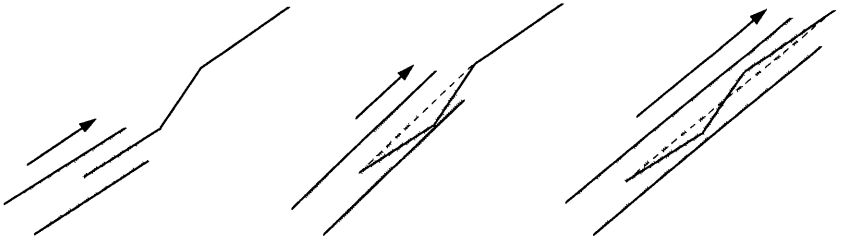

Figure 6. A sleeve moves along the polyline covering consecutive vertices.

the case for which the next point $\mathbf{p_k}$ cannot be included in the sleeve. We must advance and reset the starting point for our sleeve algorithm and also decide what to do with the points that had been intermediate vertices inside the sleeve. All of the points inside the sleeve can certainly be approximated by the sleeve centerline to within ε; and choosing the sleeve centerline as a segment in an approximating simplified polyline is one of the variants that performs well. If we choose the centerline, then we may throw away all of the intermediate points inside the sleeve. Choosing the centerline end points may force us to choose an approximating vertex that is not one of the original polyline vertices. Nonetheless the centerline end point will be within ε of the final $\mathbf{k}^{th}$ vertex $\mathbf{p_{k-1}}$.

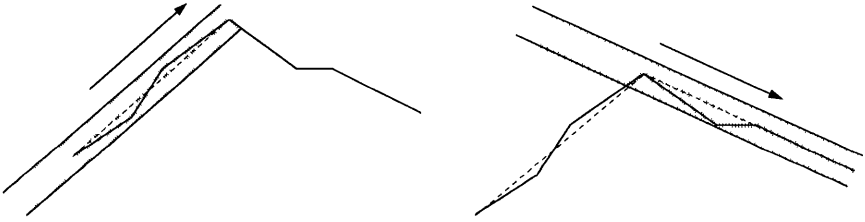

Figure 7. When a vertex cannot fit in the sleeve, a new sleeve is begun

If we simply choose the last vertex $\mathbf{p_{k-1}}$ as the approximating segment end point, then we will only guarantee that our approximating segment is within 2ε of the original polyline. We may turn this logic around and use it to our advantage by building a sleeve of width ε instead of 2ε. In that case, we may simply accept the final vertex that fits in the sleeve as our current approximating segment's end point and our next approximating segment's starting point. If we employ the straightforward narrower ε-sleeve strategy, however, we may wind up choosing many more vertices than necessary for our simplifying polyline.

A middle-of-the-road option that uses a 2ε sleeve, but only subsamples vertices of the original polyline (i.e., does not create new approximating vertices), requires a special subroutine to handle vertices in the sleeve after the sleeve's vertex set has reached a limit. The special subroutine will advance the starting vertex and decide whether to keep intermediate vertices as vertices of the simplifying polyline. There are several options for this special subroutine; and

no single option appears to guarantee the fewest number of vertices in the approximating polyline for all input polylines. If the sleeve contains vertices $\{p_{i+1}, p_{i+2}, \ldots, p_{i+k}\}$, where p_i is the current starting vertex, then we choose the vertex p_{i+j} with the largest subindex in the sleeve such that $L(p_i,p_{i+j})$ approximates the subpolyline chain $p_i, p_{i+1}, p_{i+2}, \ldots, p_{i+j}$ to within ε. Then we make p_{i+j} the new starting vertex. There will always be such an approximating vertex, although in the worst case that vertex might be p_{i+1}. Even though the new sleeve from the new starting vertex p_{i+j} will clearly contain the subsequent vertices $p_{i+j+1}, p_{i+j+2}, \ldots, p_{i+k}$ that had been in the old sleeve, we must, nevertheless, recompute all of the sector bounds since the starting vertex has changed, hence the swept angle is different in every case. This requirement to recompute the sector bounds can become expensive; and it may render our algorithm complexity quadratic instead of linear as it had been earlier.

GENERAL ALGORITHM DESCRIPTION AND PSEUDOCODE

We summarize the common components of our three algorithms below:

Initialization:

```
(1)   Set starting vertex to p0;
(2)   Set vertex set of simplifying line to {p0};
(3)   Set sleeve set to ∅;
(4)   Set sector bound to full 360° range: ([0°,360°]).
```

Iterative introduction of vertices $\{p_1, p_2, \ldots, p_n\}$ to processing:

```
(5)    For i = 1 to n {
(6)      While pi is not in the current sector bound {
(7)        Update the starting vertex to some pj* in the sleeve set;
(8)        Update the sleeve set by removing all points through pj;
(9)        Update the vertex set of simplifying line by adding pj*;
(10)       Recompute sector bound for new start vertex, sleeve set;
(11)     }
(12)     Add pi to the sleeve set;
(13)     Recompute the sector bound;
(14)   }                                        // End of For loop.
(15)   Add pn* to the vertex set of the simplifying line.
```

The variants to the basic algorithm given above require further specification of the three update statements (7) through (9) and statements (10) and (15). Here are the details:

a) For the algorithm that uses the narrower sleeve (of width ε), the value of p_j^* in steps (7) and (9) is p_{i-1}; and the value of p_j in step (8) is also p_{i-1}. The sector bound for the empty sleeve set in (10) is once again always the full 360° range. The value for p_n^* in step (15) is p_n.

b) For the algorithm that uses the wider sleeve (of width 2ε) and allows perturbation of vertices, the value of p_j^* in steps (7) and (9) is the point on the sleeve centerline that is closest to p_{i-1}; and the value of p_j in step (8) is

the original polyline's vertex $\mathbf{p_{i-1}}$. The sector bound for the empty sleeve set in (10) is once again always the full 360° range. The value for $\mathbf{p_n}^*$ in step (15) is the point on the sleeve centerline that is closest to $\mathbf{p_n}$.

c) For our middle-of-the-road variant with a 2ε width sleeve, the update procedures, including recomputing the sector bound in step (10), may require several steps. As we are adding vertices to the growing sleeve, we may easily keep track of those vertices that are capable of forming, along with the current starting point, a single line segment that adequately approximates all of the intermediate vertices between the currently added vertex and the current starting vertex. A vertex will provide the best kind of single segment approximator if and only if that vertex actually falls within the current sector bound. We will choose $\mathbf{p_j}^*$ in steps (7) and (9) and $\mathbf{p_j}$ in steps (8) to be the one such simplifying vertex $\mathbf{p_j}$ with the largest index $\mathbf{j \leq i-1}$.

In both variants a) and b), we clearly have linear time performance because each sleeve is augmented one vertex at a time until it cannot be augmented further. At that point, the entire subsequence of vertices within the sleeve is "retired"; and a new sleeve is begun from where the last one ended. There is no backtracking; and each vertex of the original polyline appears once in exactly one sleeve building operation.

CONCLUSION

We showed how to use a sector bound calculation to assign maximum consecutive sequences of polyline vertices to buffer "sleeves". Because the sequences are as large as possible subject to constraints on approximation threshold settings, we produce a rather reduced number of segments in our simplifying polylines. We have conducted empirical tests to compare the performance of the three variants to each other and to the classic Douglas-Peucker algorithm for polyline simplification. The results of those tests were very favorable for our techniques, both in appearance and in quantitative measurements of vertices used in the simplification. The results of those experiments and more information on the algorithms themselves, including complete working code, are available to the interested reader at the World Wide Web site ***http://ra.cfm.ohio-state.edu/grad/zhao/algorithms/linesimp.html***.

Our key practical result is the use of the sector bound as a new measurement of line simplification; and our key theoretical result is the proof that the sector bound, an easily maintained measurement, is locally geometrically equivalent to an ε-buffer strip of the type used in the classic Douglas-Peucker algorithm.

Finally we mention that future work is suggested by our choice of the descriptor "sleeve" and its extended meaning in 3-D. Our "sector bound" in 3-D is not just a single cone, but an intersection of cones. A sleeve, however, is nothing more nor less than a right circular cylinder having radius ε.

ACKNOWLEDGEMENTS

The authors would like to thank Paula Stevenson for providing test data. Thanks also go to Dr. Raul Ramirez and Dr. John Bossler of the Ohio State University Center for Mapping for their support in this research.

REFERENCES

Chrisman, Nicholas R., 1983, Epsilon Filtering: A Technique for Automated Scale Changing, ***Technical Paper of 43rd Annual ACSM Meetings***, Washington D.C., 322-331.

Cromley, Robert G., 1991, Hierarchical Methods of Line Simplification, ***Cartography and Geographic Information System***, 18(2):125-131.

Douglas, David H. and Thomas K. Peucker, 1973, Algorithms for Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature, ***The Canadian Cartographer***, 10(2):112-123.

Imai, Hiroshi, and Masao Iri, 1986, Computational Geometric Methods for Polygonal Approximations of a Curve, ***Computer Vision, Graphics, and Image Processing***, 36:31-41.

McMaster, Robert B., and K. Stuart Shea, 1992, ***Generalization in Digital Cartography***, Washington, DC: Association of American Geographers.

Williams, C.M., 1978, An Efficient Algorithm for the Piecewise Linear Approximation of a Polygonal Curves in the Plane, ***Computer Vision, Graphics, and Image Processing***, 8:286-293.

Zhao, Z., 1996, Java Gallery of Geometry Algorithm, ***http://ra.cfm.ohio-state.edu/grad/zhao/algorithms/linesimp.html***

Zhao, Z. and Alan Saalfeld, and Raul Ramirez, 1996, A General Line-Following Algorithm for Raster Maps, ***Proc. of GIS/LIS'96 Conference***, Denver, CO, 85-93.