BUREAU OF THE CENSUS
STATISTICAL RESEARCH DIVISION REPORT SERIES
SRD Research Report Number: Census/SRD/RR-87/21

It Doesn't Make Me Nearly as Cross
Some Advantages of the Point-Vector Representation of Line Segments in Automated Cartography

by

Alan Saalfeld
Statistical Research Division
Bureau of the Census
Washington, D.C. 20233

Recommended by: Tom O'Reagan

Report completed: September, 1986

Report issued: October 1, 1987

It Doesn't Make Me Nearly as CROSS
Some Advantages of the Point-Vector Representation of Line Segments in Automated Cartography

Alan Saalfeld

Statistical Research Division

Bureau of the Census

Washington, DC 20233

Abstract:

The point-vector representation for line segments offers several advantages over other more familiar representations for lines and line segments, such as the point-slope form, the slope-intercept form and the two-point form. With the point-vector form, line segment intersection routines and related cartographic computations, such as point-in-polygon routines and detection of near-intersections, can be streamlined, simplified, and more easily understood geometrically. The point-vector segment representation retains information on the line segment and not just the line. Special case handling for vertical lines is not necessary as with some other representations; and several computational shortcuts can be derived directly from the segment end-point coordinates.

Introduction:

In 1974, the Harvard University Laboratory for Computer Graphics and Spatial Analysis published a note by David H. Douglas entitled, "It Makes Me So CROSS," which presented a light-hearted discussion of some of the difficulties involved in developing a computer algorithm for finding line-segment intersections. The author stressed the relatively straightforward nature of the mathematical problem, then noted idiosyncracies of spatial position and computer behavior which tended to frustrate any one unified approach to solving the problem by computer.

This note recommends the use of a particular mathematical representation for line segments which facilitates many important computations made frequently in automated cartography, including tests for intersection and near-intersection. The approach resolves a number of the issues raised by David Douglas and provides shortcuts to related problems. Separate case treatment depending on spatial position becomes unnecessary. Some, but not all, of the computer behavior problems related to rounding, truncating, and overflow and underflow may be circumvented using the point-vector representation described here. Integer arithmetic suffices when line segments themselves can be describe with integer coordinates. This last feature resolves most of the computational problems and provides significant improvement in speed of computation when many intersections must be found, such as in map overlay routines.

The geometric significance of some of the intermediate computations is also illustrated here; and a point-in-polygon application is outlined as an algorithm.

Mathematical Preliminaries.

There are many mathematical representations of straight lines in a plane. Some expressions for the locus of a line are the following:

Point-Slope Form. Given a point (x_0,y_0) and slope m:

$$L = \{(x,y)|(y-y_0)=m(x-x_0)\}$$

Slope-Intercept Form. Given slope m and y-intercept b:

$$L = \{(x,y)|y=mx+b\}$$

Two-Point Form. Given points (x_0,y_0) and (x_1,y_1):

$$L = \{(x,y)|(y-y_0)/(x-x_0)=(y_1-y_0)/(x_1-x_0)\}$$

Better Two-Point Form (no possible division by zero):

$$L = \{(x,y)|(y-y_0)(x_1-x_0)=(x-x_0)(y_1-y_0)\}$$

Linear Equation Form. Given real numbers A, B, and C:

$$L = \{(x,y)|\ Ax + By + C = 0\}$$

Point-Vector Form. Given point (x_0,y_0) and vector (v_1,v_2):

$$L = \{(x_0,y_0)+(rv_1,rv_2)|\ r \text{ is a real number}\}$$

Notice that the point-vector form is not defined by an equation, but instead is a parametrized expression for the line. Every real number r corresponds to a point on the line in a one-to-one fashion.

If the user is given the two end points of a line segment (x_0,y_0) and (x_1,y_1), then a vector which will satisfy the point-vector form is:

$$(v_1,v_2) = (x_1-x_0,y_1-y_0).$$

For the above choice of vector, the point-vector form has the following important property:

The point corresponding to the parameter r_0:

$(x_0,y_0)+(r_0v_1,r_0v_2)$ is on the segment joining (x_0,y_0) and (x_1,y_1) if and only if r_0 lies between 0 and 1.

This important property permits one to use the same parametrized expression for the line and the line segment. Hereafter, it will be assumed that the point-vector form for two given points is:

$$L = \{(x_0,y_0)+(r[x_1-x_0],r[y_1-y_0]) \mid r \text{ is a real number}\}$$

Line Segment Intersection Detection.

Algorithms for detecting line segment intersection are well-known and generally straightforward. Professor Douglas has pointed out, however, that many of the straightforward approaches have computational drawbacks. Most algorithms use a two-stage approach: first, they determine the point in which the extended lines meet, then they check to see if that meeting point is actually on both segments. Occasionally, when the lines are nearly parallel (or vertical with some approaches), or when the coordinate values are large in absolute value, intermediate computations lead to underflow or overflow or arithmetic errors due to rounding.

The following description of computations of the line segment intersection algorithm for the point-vector representation illustrates the ease of application of the algorithm, the one-step computation of the intersection point, and simple estimates of magnitudes of intermediate computations (to aid in avoiding roundoff, overflow, and underflow difficulties.)

The following mathematical notation will be used to describe the problem and its solution. Two line segments, L and L′, are given in the plane:

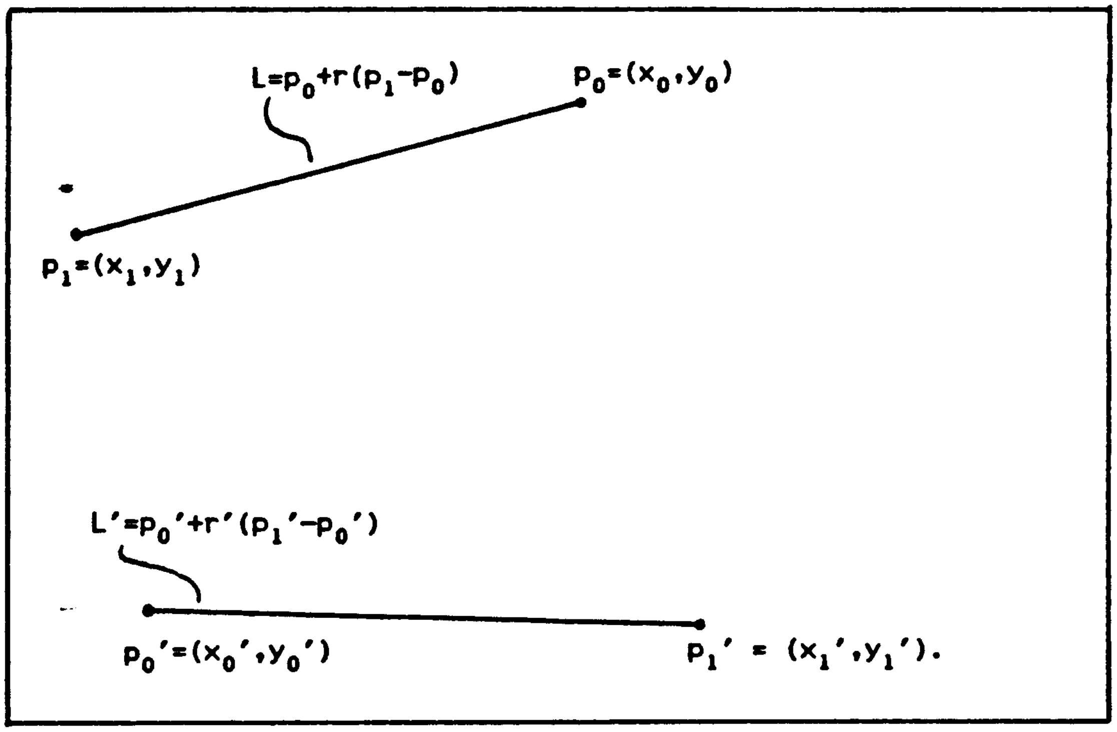

Figure 1. Line segments and their representation.

The values of r and r′ between 0 and 1 define the line segments between p_0 and p_1 and between p_0' and p_1', respectively. All other values of r and r′ define points on the infinite lines extending L and L′ which do not lie on the finite length segments.

For the typical intersection problem, the values of p_0, p_1, p_0', and p_1' are given. The values r and r′ are computed to satisfy simultaneous equations in the coordinate functions. An intersection occurs when r and r′ are such that:

$$p_0 + r(p_1 - p_0) = p_0' + r'(p_1' - p_0')$$

In terms of coordinates, the above equality becomes two equations:

$$x_0 + r(x_1 - x_0) = x_0' + r'(x_1' - x_0')$$
$$y_0 + r(y_1 - y_0) = y_0' + r'(y_1' - y_0')$$

If both r and r′ are between 0 and 1, inclusive, then there is an intersection of the segments and not just the lines. Moreover, if both r and r′ are in the real interval $[0-\varepsilon, 1+\varepsilon]$, then there is nearly (within ε) an intersection. This last check can be very useful for finding gaps in raster-scanned vectorized files and "spaghetti-coded" non-topological map files.

Solving and Simplifying the Simultaneous Equations.

Unless the two lines are parallel (and that includes the case where they are collinear), the two equations have a unique solution for the pair r and r′. The line segments intersect if both r and r′ lie in the closed interval [0,1]. The line segments nearly intersect if both values r and r′ lie near to the interval [0,1], but not inside it; or if one of the two values lies in the interval and the other lies near but not inside the interval [0,1]. The values of r and r′ may be found by computing the following two-by-two determinants:

$$D = \begin{vmatrix} (x_1-x_0) & -(x_1'-x_0') \\ (y_1-y_0) & -(y_1'-y_0') \end{vmatrix}$$

$$D_1 = \begin{vmatrix} (x_0'-x_0) & -(x_1'-x_0') \\ (y_0'-y_0) & -(y_1'-y_0') \end{vmatrix}$$

$$D_2 = \begin{vmatrix} (x_1-x_0) & (x_0'-x_0) \\ (y_1-y_0) & (y_0'-y_0) \end{vmatrix}$$

The determinant D will be zero if and only if the lines are parallel. If D is not zero, then $r = D_1/D$ and $r' = D_2/D$. Note that if the x's and y's are in integer coordinates (for instance screen pixel coordinates), then the D, D_1, and D_2 computations are all integer additions and multiplications, i.e. fast.

If the determinant D is zero, but not both D_1 and D_2 are zero also, then the parallel lines do not intersect. If D, D_1, and D_2 are all zero, then the lines are collinear; and the segment end-point coordinates may be compared to see if the collinear segments overlap or touch at a single end-point.

If the determinant D is not zero, then the following result will further simplify checking for intersections by eliminating division:

Lemma. A real number r lies in the interval [0,1] if and only if the product r(1-r) is non-negative.

The proof of the lemma is seen easily from the graph of the function: y=x(1-x), which is a concave downward parabola:

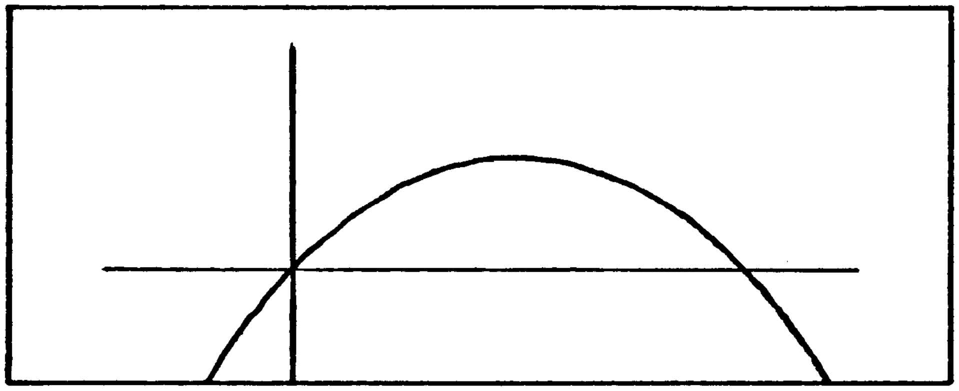

Figure 2. The graph of the parabola y = x(1-x)

Moreover, since the parabola is symmetric about x=1/2, a test that y be slightly smaller than zero is equivalent to a test that x be near to the interval [0,1], where nearness is measured uniformly at either end of the interval.

By the lemma, a ratio, such as $r=D_1/D$, will lie in the interval [0,1] if and only if:

$(D_1D-D_1D_1)/(DD)$ is greater than or equal to zero.

Since the denominator, DD or D^2, is always positive, the above expression will be greater than zero if and only if the numerator is greater than zero. The numerator may be also written as:

$$D_1(D - D_1)$$

Applying similar arguments to $r' = D_2/D$, one shows that the line segments intersect if and only if both:

$D_1(D - D_1)$ and $D_2(D - D_2)$ are non-negative.

In order to determine if an infinite line intersects a line segment, only one of the products $D_i(D - D_i)$ needs to be tested.

Such a one-sided test can be the principal component of a fast point-in-polygon algorithm such as the example outlined in the final section of this exposition.

It is worth emphasizing that the above computation of $D_i(D - D_i)$ may be done without division; and thus it is not only faster, but is less subject to computer idiosyncracies associated with real arithmetic.

The Geometric Meaning of D, D_1, and D_2.

The determinant D is sometimes called the discriminant of the linear equations that were solved for r and r'. The value of D is also recognized as the norm of the cross-product of the two vectors:

$\{(x_1-x_0,y_1-y_0),(x_1'-x_0',y_1'-y_0')\}.$

The value of $|D|$ is also equal to the area of the parallelogram formed by placing the vectors at the origin or at a common starting point, as in the figure below:

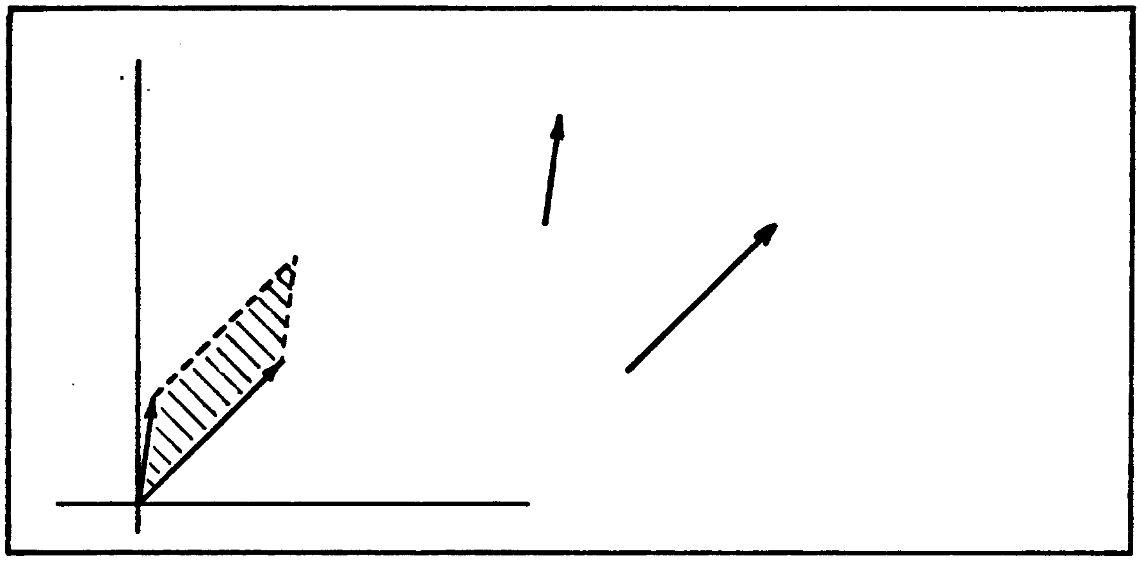

Figure 3. The discriminant D measures area of a parallelogram.

The sign of D depends on the orientation of the two vectors. The values of D_1 and D_2 may also be regarded geometrically as areas of parallelograms. D_1 corresponds to the parallelogram subtended by the vectors formed by the vertices (x_0,y_0), (x_0',y_0'), and (x_1',y_1'). This parallelogram will have area less than the area of the parallelogram for D, and D_1 will have the same orientation sign if and only if the vector $[(x_0,y_0),(x_1,y_1)]$ intersects the extended line through the segment $[(x_0',y_0'),(x_1',y_1')]$.

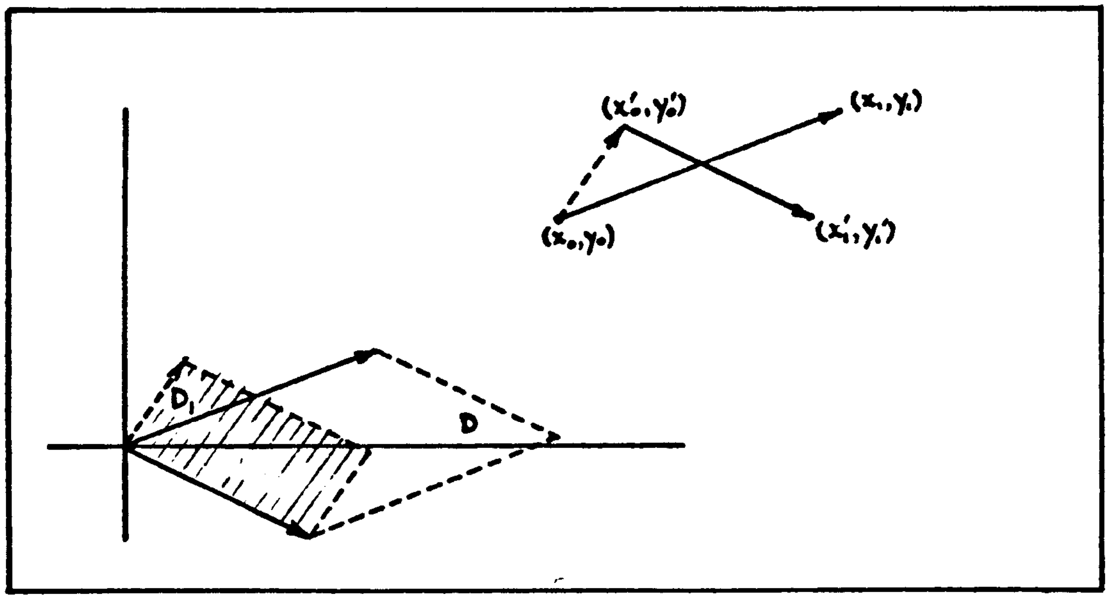

Figure 4. Parallelograms for D and D_1.

A Point-in-Polygon Algorithm Based on Point-Vector Representation.

The Jordan Curve Theorem provides an elegant means of determining if a point lies within a closed curve. By drawing any ray from the point in question and counting the number of times the ray crosses the curve, one may know if the point is inside or outside the curve. An odd number of crossings means the point is inside; an even number signifies the point is outside. Of course, implementation of the rule is not always easy for any curve. However, for polygons (closed polygonal curves) the rule is fairly easy to implement. There will be some need to clarify the meaning of "crossing" the curve. One may guarantee only legitimate full crossing--not merely touching-- of the interior segments of the polygonal line by choosing the ray judiciously, as shown below.

Suppose (x_0,y_0) is the given point to be tested to see if it is inside the polygon $\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$.

Consider the minimum non-zero $\{\|y_0-y_i\| \mid i=1,2,\ldots,n\}$ (call it m_y), and also the maximum of $\{\|x_0-x_i\| \mid i=1,2,\ldots,n\}$ (call it M_x).

Consider the ray emanating from (x_0,y_0) with slope $m_y/2M_x$. This slope is chosen so that the ray rises very slightly, but not fast enough to hit any vertices strictly above (x_0,y_0). This ray cannot intersect any of the vertices (x_i,y_i) of the polygon because of the way in which m_y and M_x were chosen. This ray may be described parametrically as:

$$R = \{(x_0,y_0) + r(2M_x,m_y) \mid r > 0\}$$

By letting r=1 a second point on the ray is:

$$(x',y') = (x_0+2M_x, y_0+m_y).$$

The point-in-polygon test can then be reduced to counting the intersections of the ray R from (x_0,y_0) through (x',y') with all of the sides of the polygon $[(x_i,y_i),(x_{i+1},y_{i+1})]$, for $i=1,2,\ldots,n$. (Here vertex n+1 is the same as vertex 1.)

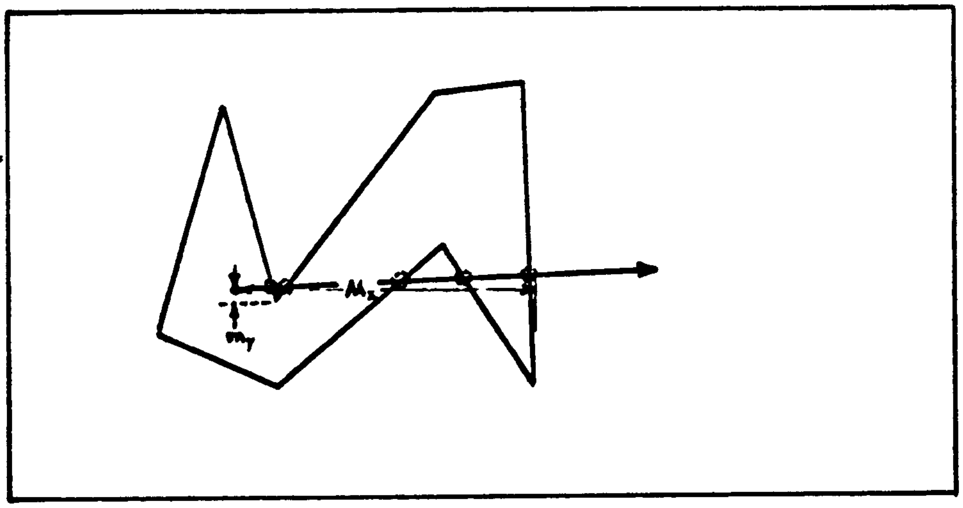

Figure 5. Ray intersections for point-in-polygon test

Counting intersections amounts to computing products of determinants $D'_i(D^*_i - D'_i)$ and counting non-negative results, where:

$$D^*_i = \begin{vmatrix} (2M_x) & (x_i - x_{i+1}) \\ (m_y) & (y_i - y_{i+1}) \end{vmatrix}$$

$$D'_i = \begin{vmatrix} (2M_x) & (x_i - x_0) \\ (m_y) & (y_i - y_0) \end{vmatrix}$$

or one may simply multiply all the products together (using the Π notation: $\Pi[D'_i(D^*_i-D'_i)]$). Then $\Pi[D'_i(D^*_i-D'_i)]$ will be negative if and only if an odd number of factors are negative.

Conclusions.

The geometric intuition provided by the point-vector form for line segment representation, along with the collection of algebraic manipulations presented herein to simplify line intersection computations, more than justify the use of that form to represent lines and line segments. Many of the problems identified by Proffessor Douglas in his paper have been addressed and resolved by this particular representation. The representation, moreover, conforms to current computer graphics wisdom and practice which calls for parametric representation of all curves; for, indeed, this form gives such a representation of the straight line or line segment.

References.

DOUGLAS, D. H., 1974, "It Makes Me So CROSS," Harvard University Laboratory for Computer Graphics and Spatial Analysis.

FOLEY, J.D., and A. VAN DAM, 1984, *Fundamentals of Interactive Computer Graphics*, (Massachusetts: Addison-Wesley).