International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:5, No:5, 2011

Computing a Time Based Effective Radius-of-Curvature for Roadways

Gary D. Cantrell and E. Alex Baylot

***Abstract*—**The radius-of-curvature (ROC) defines the degree of curvature along the centerline of a roadway whereby a travelling vehicle must follow. Roadway designs must encompass ROC in mitigating the cost of earthwork associated with construction while also allowing vehicles to travel at maximum allowable design speeds. Thus, a road will tend to follow natural topography where possible, but curvature must also be optimized to permit fast, but safe vehicle speeds. The more severe the curvature of the road, the slower the permissible vehicle speed. For route planning, whether for urban settings, emergency operations, or even parcel delivery, ROC is a necessary attribute of road arcs for computing travel time.

It is extremely rare for a geo-spatial database to contain ROC. This paper will present a procedure and mathematical algorithm to calculate and assign ROC to a segment pair and/or polyline.

***Keywords*—**linear features, radius-of-curvature, roads, routing, traffic, turning

I. Introduction

WHEN performing route planning or computing speed predictions on a specific route, one of the most important considerations is the roadway radius-of-curvature (ROC). This measurement can have a significant effect on vehicle speed and travel time. The maximum safe speed is a function of the angular acceleration induced by the curvilinear motion, the traction between the road surface and the wheel tread, and the super-elevation/bank of the roadway [1]. Some roads in pre-industrial-era cities or over mountainous areas have such small ROCs on particular sections that some large vehicles cannot use these roads. In this type of situation, the ROC renders the curve impassable, and the speed is effectively reduced to zero.

Assigning an ROC to every possible turn (segment pair) in large data sets can be computationally intensive. Consider the road vector data displayed in Fig. 1. It shows part of a digitized road network covering an area of approximately 6 km × 4 km in Baltimore.

This area has a total of 4,955 polylines. Polylines are actually multi-segment lines. If all the various segments were included in this count as well, this would result in an even greater number of ROC computations.

Finally if one were to consider all the multi-directional turns (left, right, or straight) at each intersection of lines, then the number of computations would be considerably larger.

Within the database, each polyline has a single record containing all of its attribute information such as road width, road type, and any other attribute assigned to that specific polyline. A polyline encompasses one or multiple straight-line segments grouped together with a single set of attributes. As the ROC is computed using the angle formed by two intersecting segments, assigning an ROC to a polyline must take one of two forms. A user must either divide all polylines up such that each polyline contains only two segments forming a single common vertex duplicated for each possible turn or devise a method for calculating an effective ROC for the entire polyline.

Dividing a polyline up into a polyline with only two segments would cause a drastic increase in the number of records in the database, as each new polyline would require a new database record for its attributes. These new records would contain duplicate information because the only attribute that would be different is the ROC. This leads to a great deal of data storage waste. This solution also results in more intersections between polylines. This issue will be discussed in more detail later in this work. In the file used to create Fig. 1, this method would cause at least a 50% increase in the number of polylines, from 4,955 to 7,432. This creates a data storage and processing time cost that can be unmanageable with large sets. However, creating a single ROC can be a non-trivial problem as well. Consider Fig. 2 where a road polyline (highlighted) with two intersections is curving in opposite directions: thus the polyline has two curves, each with a different ROC.

Considerations for assigning an effective ROC are (a) average, (b) worse case, or (c) some form of weighted average. The average value would not be suitable, as the effects of ROC on speed are non-linear; and, in fact, there is a threshold at which ROC no longer governs speed. Also for polylines of a great distance, the worse case would be too limiting. Thus, the third option of creating a weighted average is described in this paper.

This paper describes a method for assigning an effective ROC to a single polyline with one curve or with multiple curves in any direction. The assignment is optimized for route planning.

G. D. Cantrell is with the Dixie State College, University Plaza Bldg, 225 South 700 East, St George, UT 84770 USA (phone: 228-342-0110).

E. A. Baylot is with the U.S. Army Engineer Research and Development Center, CEERD-GM-M, 3909 Halls Ferry Road, Vicksburg, MS 39180-6199 USA (phone: 601-634-3474; fax: 601-634-3068; e-mail: alex.baylot@usace.army.mil).

Fig. 1 Road vector map of digitized 6-km × 4-km section of Baltimore, Maryland

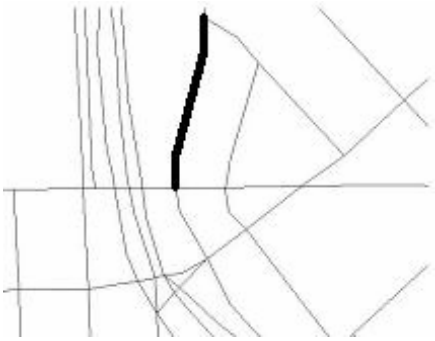

Fig. 2 Digitized road polyline with multiple curves

II. Database File Format Requirements

The method described herein could be used on any file format that utilizes geodetic points to represent linear features like roads or vehicle trails. However, this method written as software was created for and tested on the Environmental Systems Research Institute shapefile format [2]. Therefore, a high level of understanding of shapefile format is useful in understanding the process.

The shapefile is actually a file set of multiple files. This set contains three key files that are always present: a main file (shp), an index file (shx), and a database file (dbf). Other supporting files can be present as well. The main file with extension shp contains the geographic coordinates for all data structures. The shx file is an indexing file containing offsets used to locate specific records in the main file. This file is necessary only if a program or user is searching for a specific record. The process described in this paper utilizes all records, so it ignores this file completely. The last file is the database file (dbf). This file contains the attributes for each shapefile record found in the shp file. The database record 0 matches shapefile record 0; database record 1 matches shapefile record 1; and so on. This is the targeted file for writing the ROC values if the user wants to store the values with the shapefile.

The shp file is divided into two parts: a header and a set of shapefile records. The header contains information about the file contents, including the shape type stored within the file. Each record following the header has the same shape type specified by the header. However, the only shape types of concern here are the lines types used to describe roadways—namely polylines, polylineMs, and polylineZs. Because the differences between these three types are irrelevant for the described procedure, they will not be explained. All three of these types will be referred to in this paper as polylines. Fig. 3 is a graphical representation of the shp file structure.

A polyline is a group of one or more segments, each with a list of two vertices. The list of points for each polyline represents a series of connecting line segments [2].

Thus, a polyline, restated in other terms, is a series of line segments representing a single structure sharing a group of attributes and identical attribute values. The number of segments depends on how the polyline was created and how many segments can share the same attribute values. There are basically two types of polylines: single-part and multi-part. A single-part polyline consists of multiple line segments that are spatially together and are organized to share a common record in the database. Fig. 4 shows a single-part polyline. Multi-part polylines are spatially apart and like the single-part polylines are organized to share a common record in the database. Fig. 5 illustrates a multi-part polyline.

Sharing a common database record is efficient, but when an attribute value of one polyline is changed, the record must be broken from the set of polylines and given a separate record. Assigning a single ROC in this type of situation would not be correct. Simple software tools are available to convert a shapefile from a “multi-part” to a “single-part” and should be used prior to use of this algorithm.

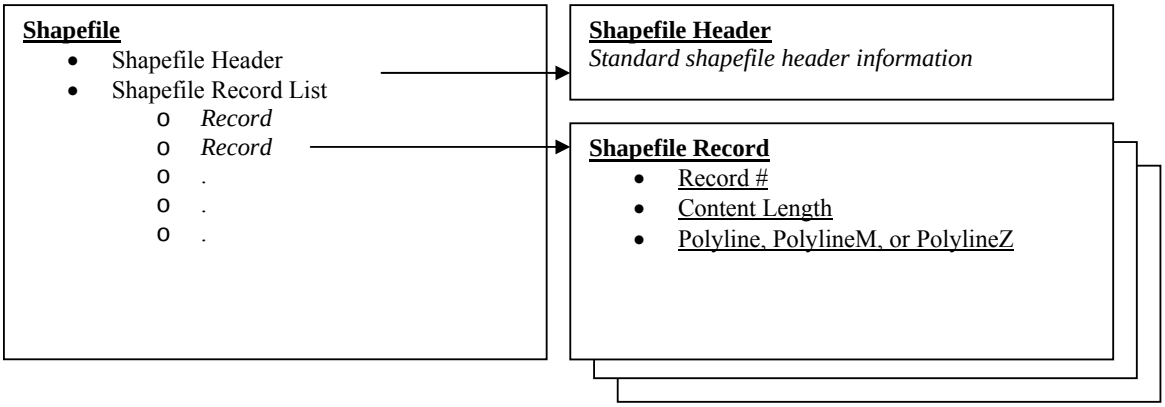

Fig. 3 Structure of a shapefile

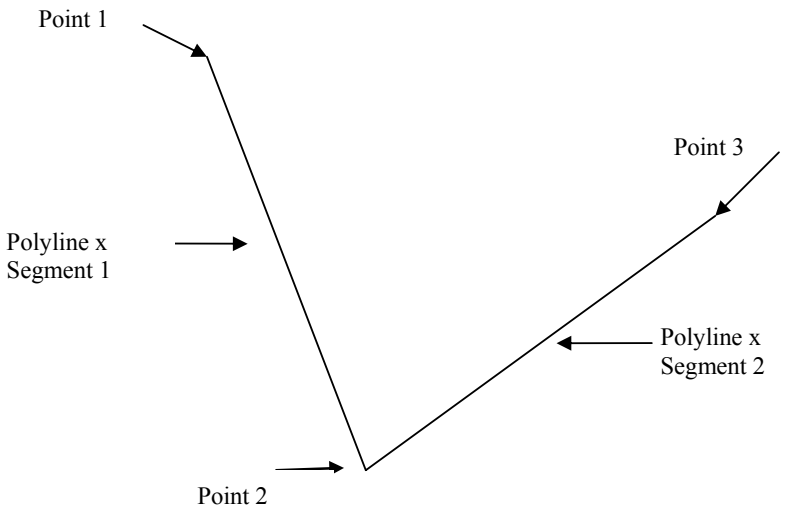

Fig. 4 Single part polyline segments

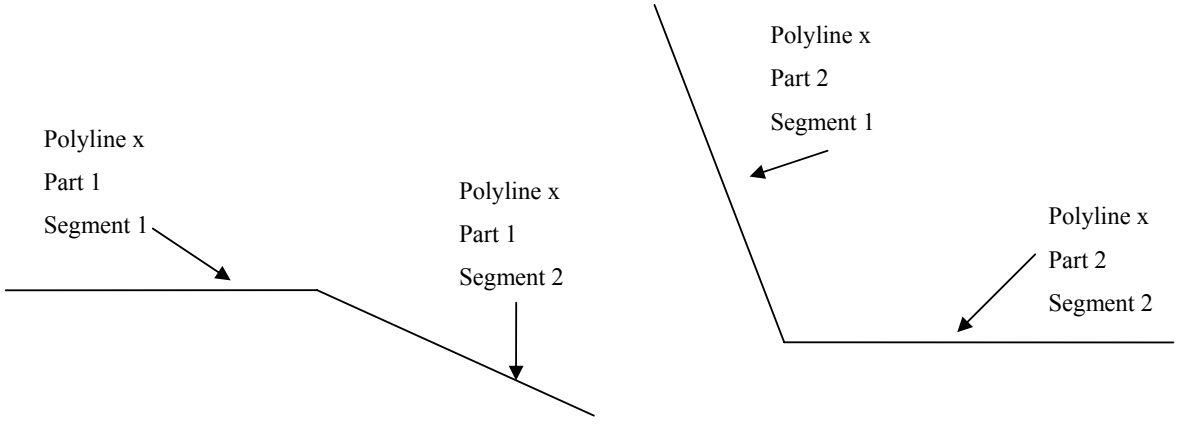

Fig. 5 Multi-part polyline

As seen in Fig. 6, the digitization process can have some error. The dashed line shows the digitized road superimposed on the photograph of the actual road identifiable in Fig. 6 as a paler shade of tan. The distance from the center of the road to the digitized road ranges from approximately 13-23 m. This error can be affected by image resolution, user or computer error during digitization, or other factors. This error was not necessarily accounted for in the computation but was addressed during validation.

III. Basis for Radius-of-Curvature

In road design, the ROC is obtained by laying down surveyed points using a 30.5-m (100 ft) chord of a specified curvature. That curvature is described as a degree-of-curve and is the central angle subtended by a 30.5-m chord [1]. This chord length varies depending on the organization conducting the development. For example, according to the Transportation Research Board for developing countries, the angle is subtended by a 20-m chord [3]. Additionally, the Asian highways design standards call for a minimum 30-m ROC for Class III roads, the minimum design standard [4]. Space and topography limitations may necessitate the relaxation of these standards. The relationship of ROC (m) to degree-of-curve is illustrated in Fig. 7. The relationship of ROC is given as the following equation (the 30.5 in the numerator will change depending on the chord length used):

$$ROC = \frac{30.5}{2\sin D/2} \quad (1)$$

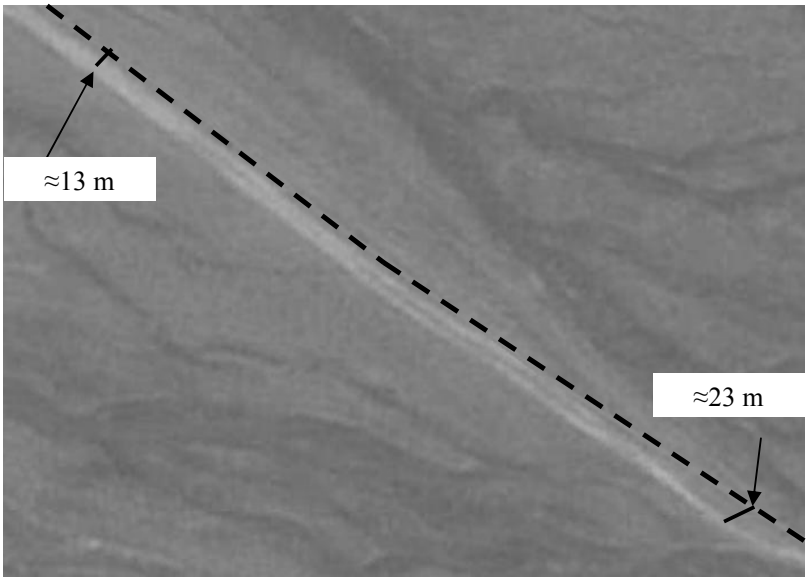

Fig. 6 Polyline superimposed on a satellite photo

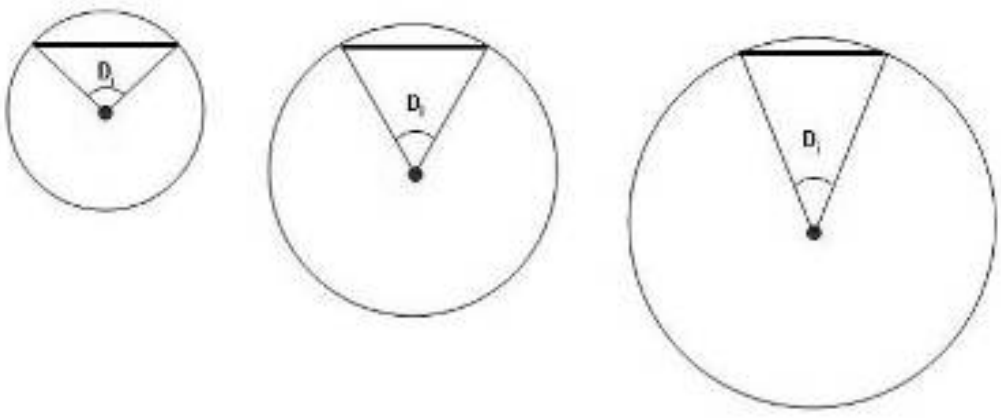

Fig. 7 Illustration of how a fixed length chord will yield a smaller degree-of-curvature as the radius increases as governed by (1)

The degree-of-curve is described as a function of velocity, radius, coefficient of side friction, and super-elevation. Degree-of-curve is given in (2). This definition and method of calculating an ROC was created for road construction to ensure that a curve of a road did not exceed a specified safety tolerance.

$$D_{max} = \frac{17{,}190(e+f)}{V^2} \quad (2)$$

where:

D = permissible degree-of-curve (degree (deg))
e = the super-elevation (decimal percent slope)
f = the coefficient of friction
V = the design speed (meters per second)

Equations (1) and (2) form a basis for the effective ROC calculation. These equations or slight variations of them are found in various highway engineering references.

IV. Procedure for Calculating an Effective Radius-of-Curvature

An effective ROC is defined here as an ROC that will yield the same travel time as would the series of ROCs and straightaways found on the polyline. The radius is a representative number intended to help calculate the travel time and provide a means to create ground vehicle mobility predictions and route analysis. However, as previously stated, if the polyline consists of only two adjacent segments, the effective ROC will also be the correct geometric approximation of ROC. The computation of an effective ROC is calculated with the following steps:

(1) Examine the polyline segment by segment and determine viable points for fitting a curve to each segment pair within the polyline.

(2) Fit a circle at each intersection using the projected points (the radius of each circle represents the ROC for that intersection).

(3) Compute segment lengths for the segment area within each circle and total length outside the circles.

(4) Compute an effective ROC using the data created from steps 2 and 3.

Each one of these steps will now be described in more detail in its own subsection.

Step 1: Examine the polyline segment by segment and determine viable points for fitting a curve to each segment pair within the polyline.

An ROC is first assigned to each segment pair on a polyline by fitting a circle to that vertex through three computed points. The middle point is the shared vertex. The other two points will lie somewhere on the adjacent segments or a projection of those segments. A projection method was needed for extremely short segments arising from digitizing error. Fig. 8 shows two circles where two segment pairs of different lengths serving as chords form the same angle. The resulting radii are of different lengths as well. Thus, the length of the segment chords is considered as well as the angle formed.

A road network is the expected linear feature input. Therefore, there is no real predictability on segment length. The length is dependent on how it was digitized. Lines will often be uneven as shown in Fig. 9. As demonstrated in the figure, not considering the lengths of segments may lead to poor approximations of an ROC. To better approximate the actual ROC, the algorithm calculates a more suitable chord endpoint along a given or projected segment.

This algorithm uses an oscillating circle fit with a central angle subtended by a fixed-size chord to calculate an ROC. This is a user provided length or defaults to 30.5 m. All the imagery and validation tests showed that using a chord length of 30.5 m (100 ft), the standard chord distance for roadway design, yielded good results. The start and end points used for generating the ROC circles are created by projecting the intersection point toward the previous and next points along the polyline (Fig. 8). The amount of projection or shift is determined by calculating the distance required to create a right triangle along the radius of the circle with half the specified chord length (Fig. 10). A point on either side of the intersection is projected along the existing polyline. Equation (3) shows the formula used to project the point.

$$Shift = \frac{Chord}{2\sin\left(\Theta/2\right)} \quad (3)$$

Creating new points in this manner will sometimes lead to situations in which points do not fall on existing lines (Fig. 11). There is no way to avoid this and still have comparable curves. The final calculations do take this into consideration by reducing the segment lengths back to the actual length of the segment when computing travel time.

Step 2: Fit a circle at each vertex using the projected points (the radius of each circle approximates the ROC for that segment pair).

This step actually runs in conjunction with step 1 per segment pair. It should be noted that two filters that can cause alternate computations are applied in this step. One of these filters is applied to very large angles and the other to very small ROCs. As the angle Θ approaches 180 deg (absolute value), the turn becomes more and more viable for a vehicle to circumvent. Thus, for angles approaching 180 deg, no slow down will be required, and a segment pair should be considered straight for vehicle speed. At this point no curve fitting is necessary, and the ROC for that intersection should be assigned a number large enough to have no effect on speed. In the implementation used for testing, this number was 1,000 m. Road curvature designed at a 1,000-m radius will allow safe travel speeds up to 35.8 mps [1]. The angle at which a segment pair should be considered straight enough to be assigned this large ROC is dependent on road type, vehicle

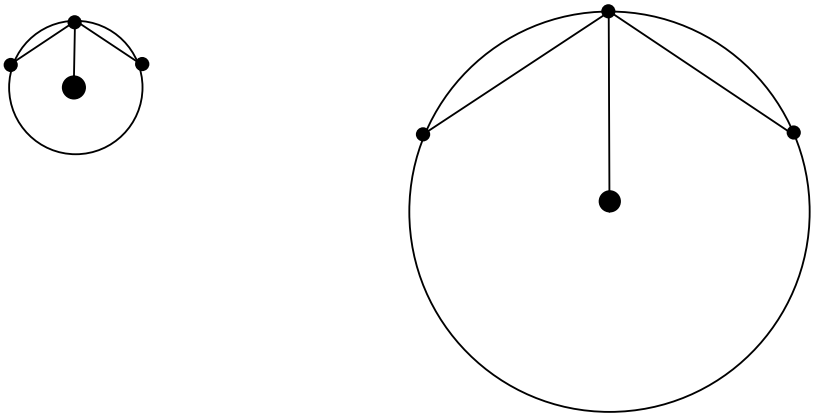

Fig. 8 Two curves with the same angle but different chord lengths

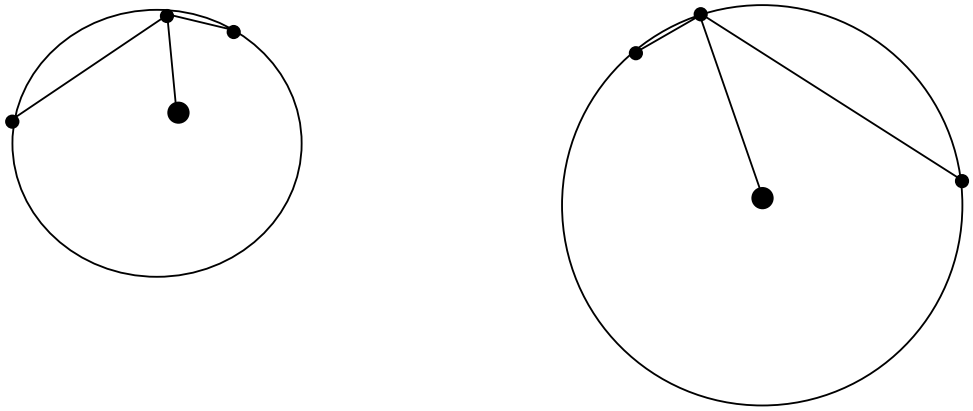

Fig. 9 Circles fit to uneven roadways

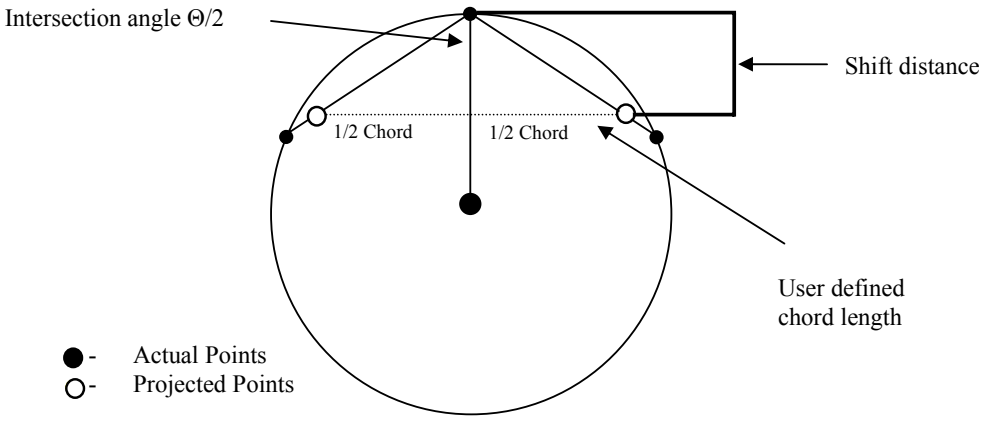

Fig. 10 Actual and projected points

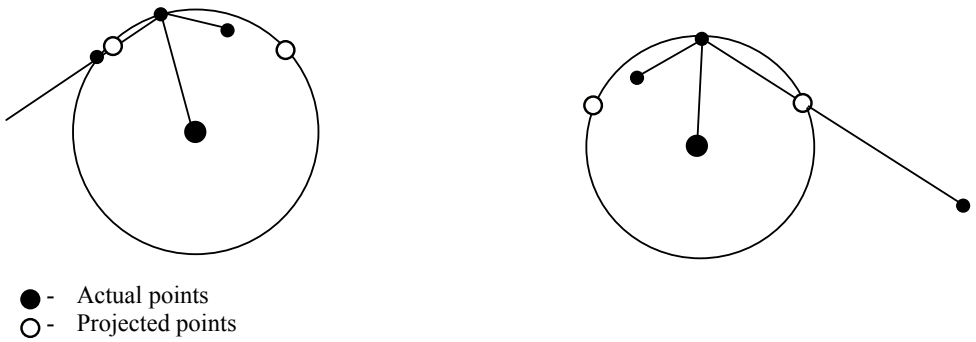

Fig. 11 Projected points based on Segment Length Constant

type, and other environmental factors. Therefore, this number is left as user defined input.

Once step 1 has been completed for a segment pair and the segment pair has not been filtered, then a circle is fit to the middle-point vertex based on the following computational process with start (X_1, Y_1), middle (X_2, Y_2), and end (X_3, Y_3) representing the three selected points from Step 1. Point (X_4, Y_4) is the center-point of the circle.

$$A = X_2 - X_1$$

$$B = Y_2 - Y_1$$

$$C = -\tfrac{1}{2}(B \cdot (Y_1 + Y_2) + A \cdot (X_1 + X_2))$$

$$D = X_3 - X_2$$

$$E = Y_3 - Y_2$$

$$F = -\tfrac{1}{2}(E \cdot (Y_2 + Y_3) + D \cdot (X_2 + X_3))$$

$$G = B \cdot D - A \cdot E$$

If G is equal to zero, then points represent a straight line. Go to the next segment pair, else:

$$X_4 = -((B \cdot F - C \cdot E) / G$$

$$Y_4 = ((A \cdot F - C \cdot D) / G$$

$$R = (X_4 - X_1)^2 + (Y_4 - Y_1)^2$$

The second filter is applied after the ROC, R, is calculated. If an ROC is too small, the vehicle cannot make the turn and the road will be impassable. This is more for large tractor-trailer trucks than for cars. Given this aggregation method for summing travel times down a polyline, it would be possible for a small impassable ROC for a segment pair to be minimized in the effective ROC calculations, given larger passable ROCs on the same polyline. Thus, if any ROC of a polyline is judged too small by the user set criteria, the effective ROC for the entire polyline is set to this threshold. This ensures that the polyline will be impassable for vehicle routing.

Step 3: Compute line lengths within each circle and total length outside the circles.

During the calculation of the effective ROC, a chord length has to be calculated for the segment pair on the circle and for those not falling on the circle. The cases are given thus:

1) Projected point falls outside the actual line.

In this case the actual segment was shorter than the projected point (Fig. 12). For this situation, the segment length is reduced back to the actual length during computation prior to time across computation but after the ROC has been stored. This adjustment has to be made before the next adjustment.

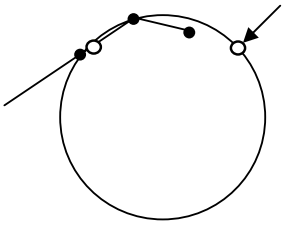

Fig. 12 Projected points

2) Two circles share a segment.

This is the case of an S type curve where the two computed circles share a section (Fig. 13). At this point, the degree of segment sharing is determined. If the shared segment is a 100% overlap as shown in Fig. 13, half the length is assigned to each circle. If there is no overlap, nothing needs be done. If there is a partial overlap, then once again the segment is halved and each half assigned to a circle.

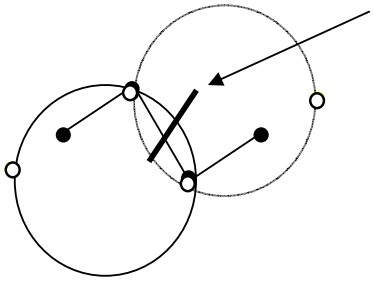

Fig. 13 Shared segment

Combinations of these two situations, such as when two circles share a segment that is less than the projected point length, also have to be considered. Each vertex is then assigned a length based on these cases; or if the case does not apply, the length is set to the actual distance from the vertex to the projected points. The remaining segment lengths (straightaway lengths) can then be calculated by summing all segment lengths and subtracting that value from the summation of all the segment pair lengths used for ROC calculations.

Step 4: Compute an effective ROC using the data created from Steps 2 and 3.

At this point in the process there will be a list of vertices for segment pairs, each with a designated ROC. In addition, there will be a total length for the areas inside and outside the ROC radius. This information will be used to compute an effective ROC.

The first step is to convert all the existing data to travel time based on a maximum safe velocity. Given (1) and (2), then solving for V and eliminating D, the velocity as a function of ROC is given in (4).

$$V = \sqrt{\frac{17190(e+f)}{2\sin^{-1}\left(15.25/ROC\right)}} \tag{4}$$

Note: The value 15.25 is based on half the 30.5-m chord.

Assuming a mid-level value of 0.06 for *e* and 0.15 for *f*, the equation is further simplified to (5) [1].

$$V = 42.5 \cdot \sqrt[-2]{\sin^{-1}\left(15.25/ROC\right)} \tag{5}$$

This now allows the segment pair lengths contained within a radius to be converted to travel time, (6). For the set of segments forming a radius (number of segments in a set is *k*, number of sets is *m*), given each length *l* (m) the total transit time is t_j (sec):

$$t_1 = \sum_{j=1}^{m}\left(\frac{\sum_{i=1}^{k} l_i}{V_j}\right) \tag{6}$$

For the set of segments within the polyline not used for computing a radius (number of segments is *k*, number of sets is *m*), given each length *l* (m), the total transit time is t_2 (sec):

$$t_2 = \sum_{j=1}^{m}\sum_{i=1}^{k}\frac{l_{ij}}{V_{ij}} \tag{7}$$

The straightaway velocity for (7) must be provided. If particular speed limits for each segment are unknown, then assume the speed limits given in Table I [1].

TABLE I
SPEED LIMITS FOR STRAIGHTAWAYS

Description	Feature Code	RST*	V_{max} (mph)	V_{max} (mps)
Vehicle Trail	AP010	N/A	30	13.4
Secondary Road	AP030	≠1	40	17.9
Primary Road	AP030	1	60	26.8

Note: Feature Codes and Attribute definition are found in International Standards Organization 19110:2005 [5]. *Road Surface Type (RST).

Then the total transit time for the set of segments within the polyline not used for computing a radius is simplified to (8), as V is now a constant.

$$t_2 = \frac{\sum_{j=1}^{m}\sum_{i=1}^{k} l_{ij}}{V_{max}} \tag{8}$$

The total transit time, t_3, is simply the sum of the results from (6) and (7) or (8).

$$t_3 = t_1 + t_2 \tag{9}$$

For all segments of the polyline, the effective velocity for the combined length (number of polyline segments *k*) is given:

$$V_e = \frac{\sum_{i=1}^{k} l_i}{t_3} \tag{10}$$

Thus, after solving (5) for ROC, the effective ROC (m), with V (mps) for a polyline is given:

$$ROC_e = \frac{15.25}{\sin\left(1805/V_e^2\right)} \tag{11}$$

Fig. 14 is sample output from the software program running the algorithm described herein. This visualization is an example of alternate output that was produced for validation purposes. It is included here to provide a good visualization of how the computation is performed on a line with multiple curves. The circles are fitted to each segment pair that form enough of an angle to cause a reduction in speed. As can be seen in Fig 14, the bigger the circle, the straighter the curve. The cross-hatched points indicate where the curves start and stop as computed by the algorithm.

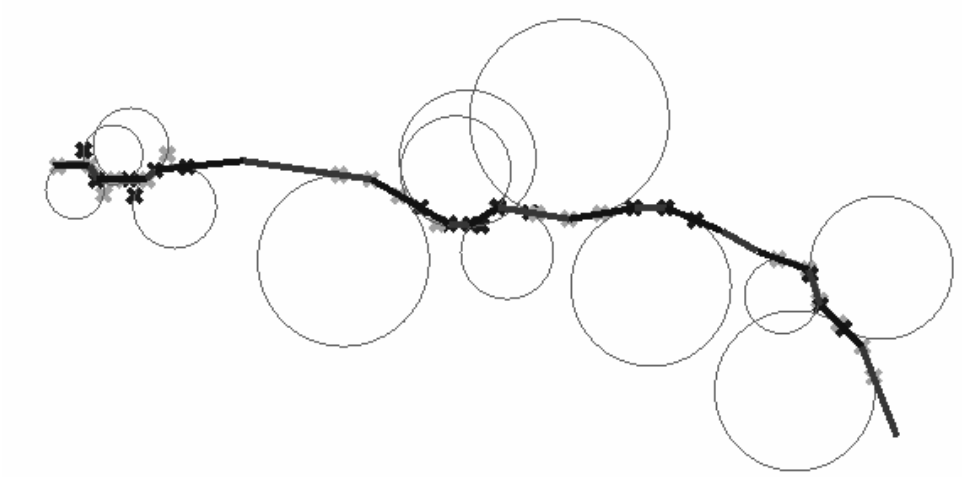

Fig. 14 Example of actual polyline with multiple ROCs

This depiction shows all geometries described thus far. The radius of each circle is the ROC, the areas outside the circles are the straightaways, and the intersections that have no circle drawn on them are considered not drastic enough to cause a slowdown.

V. Effective Radius-of-Curvature Limitations and Validation

The method described herein is for calculating the ROC of only a single polyline, not the intersections between polylines. For example, in Fig. 15 there are actually two polylines represented. The end point of Polyline 1 overlaps the start point of Polyline 2 connecting two linear features. The circles indicate the fitted ROCs to the polyline. Notice that the point where the two lines meet is not assigned an ROC, even though it appears sharp enough to need one. The method is aware of only each separate polyline structure, not of the whole road network.

Calculating an ROC at the intersection of polylines is beyond the scope of this work. The original problem statement revolved around assigning an ROC per polyline, not per intersection. Also, polylines that share end points but are distinct usually have different attributes such as Road- With, Surface Type, Use, etc. This would indicate that it is probably a different road, and thus would not be designed to flow as a continuous road. Therefore calculating an ROC for two distinct but intersecting roads might not be applicable. A lot of this depends on how the road was digitized.

For vehicle routing problems along such a road network, these intersections of polylines are best solved by considering the geometrics and kinematics of a vehicle's turning from one road onto another road. This problem becomes even more complicated when more than two polylines share a common vertex. Sarker and Baylot (2008) handled a per-intersection turning trajectory calculation that incorporates lane widths and turning trajectories [6]. All these situations are part of the bigger problem of route planning.

With these limitations in mind, validation testing was performed on a per polyline basis. Test cases were chosen, and

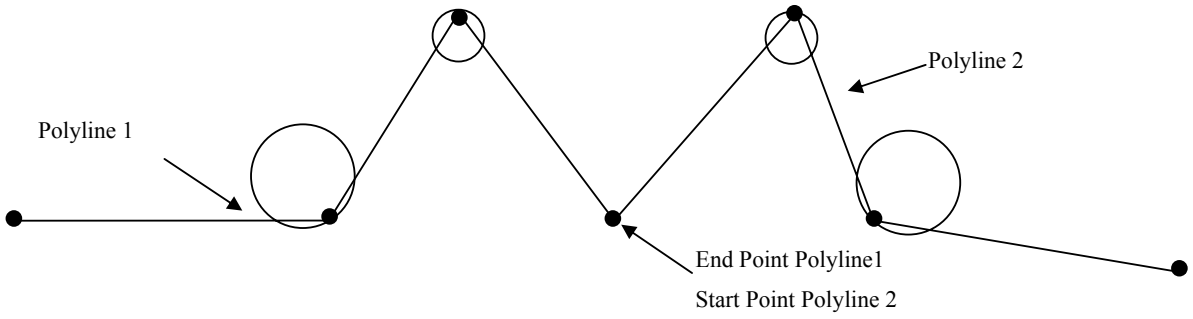

Fig. 15 Two connecting polylines with one or more differing attribute values

the resulting coordinates were overlaid onto existing multi-spectral satellite imagery of 2.5-m and sub-meter resolution where available. The findings indicate that the computed effective ROCs are a good fit on high resolution imagery of roadways. Fig. 16 shows a section of a winding mountainous road with circles overlaid on the centerlines of the curves. Their radii (RAD) are provided inside each circle. The digitized lines show where two adjacent polylines fall on the actual roadway (dashed line). The calculated effective ROC is noted along each polyline. The digitized polylines are not angled as the image shows; thus, it follows that the effective ROC derived from the computed ROC for each curve is greater than the measured ROC. The polylines are not as angled because the imagery used for the digitization was of a 2.5-m pixel resolution rather than the sub-meter shown in the figure. Nevertheless it is a reasonable approximation of reality when given only the digitized polylines.

Fig. 17 shows a case in which the effective ROC is less than the measured ROC. Again there is a digitization error, although more pronounced for this case. Using this scheme of an effective ROC of the entire polyline actually gives a more reasonable estimation of the actual ROC than using just a segment pair. For this case, had only segment pairs been considered for the ROC, the sharpest curve would have been impassable.

VI. Further Work

This work was only an initial step in incorporating ROC assignment to polylines without altering the number of polylines in the database. During the design of this process, intersections between polylines were ignored, as intersections were considered a vehicle routing decision and not part of the contiguous roadway. To minimize the omission of computing the ROC formed by adjacent polylines, the connected end-segments of two polylines would have to be examined to determine if a significant angle is formed. If so, the segments of the polylines would have to be sliced and/or reassembled such that the formed angles are insignificant. This procedure would be extensive and is beyond the scope of this paper, but it would greatly mitigate the effects of omitting the curvature formed between connecting polylines. Another solution to considering the intersection effects would be creating a different process that produces point data at the common vertex of polylines. Each point would be assigned a curvature that represents the underlying curvature for an intersection turn(s). A method of identifying which curvature to use for the turning direction would have to be established.

The accuracy of the calculations is dependent on the accuracy of the data. The data accuracy is dependent on several factors, including the digitization process and the resolution of the imagery containing the roads serving as the background. Other work supporting the expansion of this research is adjusting the standard chord length as a function of the resolution of the imagery. This idea is based on the concept that since the degree-of-curve is derived from the chord length, the degree-of-curve will actually appear less severe when lower resolution imagery is used as background for digitization. Adjustments could be made through standard chord length adjustment factors.

VII. Summary and Conclusion

This research addressed the issue of how to assign a single ROC to a polyline with multiple curves in multiple directions. The issue was successfully addressed for vehicle speed calculations, and the problems associated with such an assignment were explored. The problems identified included:

- Single-part and multi-part polylines: The lines have to be dissolved into single part lines before any meaningful ROC calculation can be carried out.
- Intersections between polylines and true road intersections: These were deemed to be route planning issues for turning off or on to another road rather than the vehicle's following the curve of the road.
- Standard comparable ROCs: Different chord lengths used for curve fitting can cause different ROCs independent of the road angle. A user-defined chord was included in the method to standardize the lengths and make all ROCs comparable.
- Single ROC assignment to a polyline with curves in multiple directions: This was handled by converting the segment lengths to time based on maximum safe velocity within each ROC and predefined speed limits for the remaining length, then converting the overall time back to a constant velocity so an effective ROC could be calculated for the entire polyline.

After the problems were identified and the solutions proposed, an algorithm was designed that addressed them. The algorithm was implemented using the computer programming language, C#. This algorithm was tested using a digitized road network in mountainous terrain. Verification of the accuracy was done by overlaying the digitized road network and the calculated circles (curvature) onto satellite imagery. Some variation was noted, but shifting the digitized road network over to align with the road networks showed that the fitted curves were still accurate. Accuracy of the mathematical process was verified by performing the same calculations in a spreadsheet and comparing the results. Tests showed that the curves created and the mathematical process were representative. Across multiple runs, the effective ROC for each polyline was also compared to the individual ROCs produced along the polyline to determine if it fell within a reasonable range. Out of approximately 175 curve calculations, and after discarding the extremes of straight lines and impassable turns using a chord of 30.5 m, the average difference between the average ROC value and the effective ROC was 3.1 m. This demonstrated that the effective ROC is a reasonable distance within the range of actual ROCs.

This indicates that the procedure creates a good estimation of an effective ROC for a polyline in multiple directions. The software algorithm has been implemented into a developmental build of the Battlespace Terrain Reasoning

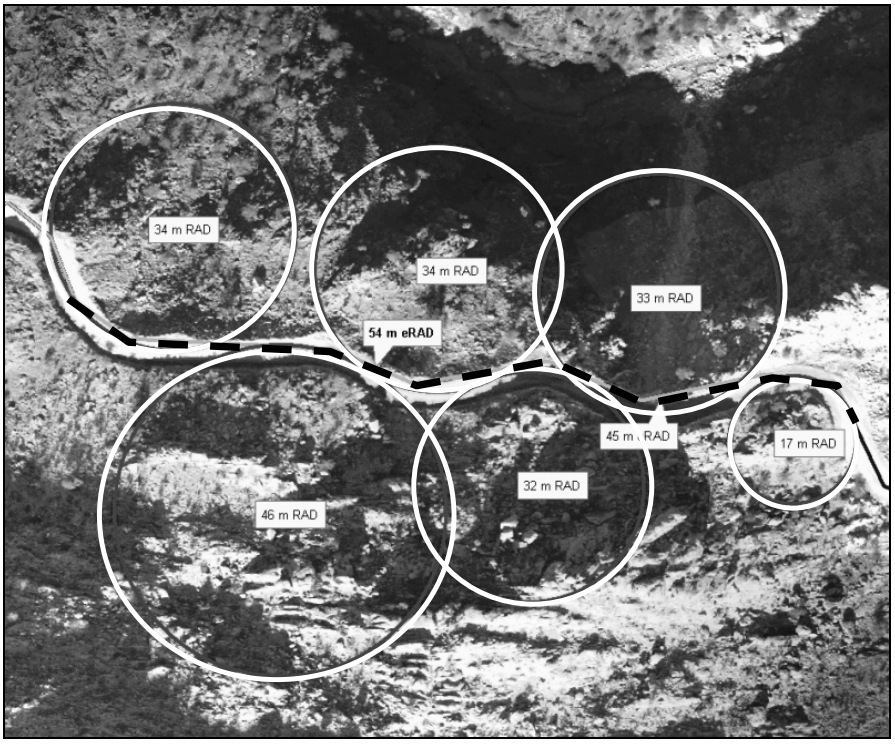

Fig. 16 Comparison of measured ROC (RAD) and the calculated effective ROC (eRAD) for a mountain roadway

Fig. 17 Comparison of measured ROC (RAD) and the calculated effective ROC (eRAD) for a sharp curve

REFERENCES

[1] C. H. Oglesby, and G. R. Hicks, Highway Design, 4th ed., New York, NY: Wiley, 1982.
[2] Environmental Systems Research Institute, "ESRI Shapefile Technical Description: An ESRI White Paper—July 1998," ESRI Online Library, Redlands, CA, 1998.
[3] Transportation Research Board, "Geometric Design Standards for Low-Volume Roads," Compendium 1, Transportation Technology Support for Developing Countries, Washington, DC, 1978.
[4] United Nations, Transport Division, Asian Highway – The Road Networks Connecting China, Kazakhstan, Mongolia, the Russian Federation, and the Korean Peninsula, UNESCAP Ref No. ST/SCAP/ 173, Blue Ridge Summit, PA: United Nations Publications, 2001.
[5] International Standards Organization, "Methodology for Feature Cataloguing," Standard 19110:2005, Geneva, Switzerland, 2005.
[6] B.R. Sarker, and E. A. Baylot, "Estimation of Throughput Capacity for Convoy Movement," in Proceedings of the 13th International Conference of Hong Kong Society for Transportation Studies, Hong Kong, China, 2008.